استادی در DAX:تحلیل داده‌های کسب‌وکار به سبک ایرانی

عناوین

پیشگفتار

در دنیای امروز که داده‌ها به‌عنوان ارزشمندترین دارایی سازمان‌ها شناخته می‌شوند، توانایی تبدیل این داده‌ها به بینش عملی، یک مزیت رقابتی تعیین‌کننده است. ابزارهای هوش تجاری (Business Intelligence) و در رأس آن‌ها مایکروسافت Power BI، این امکان را برای کسب‌وکارها فراهم کرده‌اند تا با سرعت و دقت بی‌سابقه‌ای به تحلیل داده‌های خود بپردازند. در قلب این تحلیل‌های قدرتمند، زبان فرمول‌نویسی DAX (Data Analysis Expressions) قرار دارد؛ زبانی که تسلط بر آن، مرز بین یک تحلیلگر معمولی و یک متخصص حرفه‌ای را مشخص می‌کند.

کتاب‌های متعددی در سطح بین‌المللی به آموزش DAX پرداخته‌اند، اما جای خالی مرجعی که دانش عمیق این زبان را با چالش‌ها و نیازهای بومی کسب‌وکارهای ایرانی پیوند دهد، همواره احساس می‌شد. این کتاب با همین هدف نگاشته شده است: ارائه مفاهیم پیشرفته DAX در قالب سناریوهای واقعی و ملموس برای بازار ایران.

در این کتاب، ما تنها به ترجمه مفاهیم اکتفا نکرده‌ایم. تلاش ما بر این بوده تا با ترکیب دانش جهانی و سال‌ها تجربه عملی در پروژه‌های هوش تجاری داخل کشور، راهکارهایی ارائه دهیم که مستقیماً به کار شما بیایند. از پیاده‌سازی صحیح تقویم شمسی (جلالی) و محاسبات مرتبط با آن گرفته تا طراحی مدل‌های امنیتی پیشرفته در محیط‌های سازمانی که از Power BI Report Server و SQL Server استفاده می‌کنند، تمام مثال‌ها با نگاهی کاربردی انتخاب شده‌اند.

این کتاب برای کسانی نوشته شده که گام‌های اولیه را در دنیای DAX برداشته‌اند و اکنون به دنبال آن هستند تا مهارت‌های خود را به سطح بالاتری ارتقا دهند و از یک کاربر ابزار، به یک معمار راهکارهای هوش تجاری تبدیل شوند. هدف ما این است که پس از مطالعه این کتاب، شما نه‌تنها بتوانید پیچیده‌ترین محاسبات را در DAX پیاده‌سازی کنید، بلکه بیاموزید که "مانند یک متخصص DAX فکر کنید" و برای هر چالشی، بهینه‌ترین و کارآمدترین راه‌حل را طراحی نمایید.

امیدواریم این مجموعه، چراغ راهی برای شما در مسیر تسلط بر این زبان قدرتمند و خلق ارزش روزافزون برای سازمانتان باشد.

درباره گردآورنده کتاب

صالح دسین یک متخصص برجسته در زمینه هوش تجاری و تحلیل داده است که با سال‌ها تجربه در پیاده‌سازی راهکارهای داده‌محور برای کسب‌وکارهای ایرانی، به سازمان‌ها کمک کرده است تا از پتانسیل کامل داده‌های خود بهره‌برداری کنند. تخصص او در Power BI، DAX و مدل‌سازی داده، همراه با درک عمیق از چالش‌ها و نیازهای بازار ایران، او را به یکی از پیشروان این حوزه تبدیل کرده است.

او این کتاب را توسط هوش مصنوعی گردآوری کرده تا به زبانی ساده و کاربردی به اشتراک گذاشته تا علاقه‌مندان و متخصصان داده در ایران بتوانند به‌راحتی مفاهیم پیچیده DAX را فراگرفته و آن‌ها را در پروژه‌های واقعی خود به کار گیرند. هدف اصلی او، توانمندسازی جامعه تحلیلگران داده ایرانی برای ایجاد بینش‌های قدرتمند و تصمیم‌گیری‌های آگاهانه است.

https://www.linkedin.com/in/saleh-dasin/

مقدمه: چرا این کتاب برای شماست؟

در دنیای پرشتاب کسب‌وکار امروز، داده‌ها به‌عنوان طلای جدید شناخته می‌شوند. توانایی جمع‌آوری، تحلیل و استخراج بینش از این داده‌ها، دیگر یک مزیت رقابتی نیست، بلکه یک ضرورت برای بقا و رشد است. در ایران نیز، باوجود پتانسیل‌های فراوان اقتصادی و تنوع کسب‌وکارها، نیاز به ابزارهای قدرتمند تحلیل داده بیش‌ازپیش احساس می‌شود. این کتاب، با تمرکز بر زبان DAX (Data Analysis Expressions)و پلتفرم Power BI، به شما کمک می‌کند تا از این ابزارها به بهترین شکل ممکن برای تحلیل داده‌های کسب‌وکار خود، با رویکردی کاملاً ایرانیزه، بهره‌برداری کنید.

هدف این کتاب

هدف اصلی این کتاب، ارائه یک راهنمای جامع و کاربردی برای تسلط بر زبان DAX است. ما تنها به ترجمه مفاهیم نمی‌پردازیم، بلکه با درنظرگرفتن چالش‌ها و فرصت‌های خاص بازار ایران، مثال‌ها، سناریوها و بهترین روش‌ها را به‌گونه‌ای ارائه می‌دهیم که برای مخاطب ایرانی کاملاً ملموس و قابل استفاده باشد. این کتاب، پلی است بین دانش جهانی تحلیل داده و نیازهای بومی کسب‌وکارهای ایرانی.

مخاطبان این کتاب

این کتاب برای طیف وسیعی از افراد طراحی شده است، از جمله:

- تحلیلگران داده و هوش تجاری **:**کسانی که می‌خواهند مهارت‌های DAX خود را به سطح پیشرفته‌ای برسانند و راه‌حل‌های تحلیلی پیچیده‌تری را پیاده‌سازی کنند.
- کاربران **Power BI:**افرادی که با Power BI کار می‌کنند و می‌خواهند از قابلیت‌های کامل DAX برای ایجاد گزارش‌ها و داشبوردهای قدرتمندتر بهره‌مند شوند.
- مدیران و تصمیم‌گیرندگان **:**کسانی که می‌خواهند درک عمیق‌تری از نحوه عملکرد تحلیل‌های داده‌ای داشته باشند و بتوانند سؤالات تحلیلی دقیق‌تری را مطرح کنند.
- دانشجویان و علاقه‌مندان **:**هر کسی که به دنیای تحلیل داده و هوش تجاری علاقه‌مند است و می‌خواهد یک مسیر یادگیری ساختاریافته و بومی‌شده را دنبال کند.

رویکرد ایرانیزه

یکی از نقاط قوت اصلی این کتاب، رویکرد ایرانیزه و بومی‌شده آن است. ما در این کتاب، صرفاً به ترجمه مفاهیم نمی‌پردازیم، بلکه تلاش می‌کنیم تا با ارائه مثال‌های عملی از کسب‌وکارهای ایرانی، استفاده از تقویم شمسی در تحلیل‌های زمانی، و پرداختن به چالش‌های خاص داده‌ای در ایران، این کتاب را به یک منبع کاملاً بومی و کاربردی تبدیل کنیم. این رویکرد شامل موارد زیر خواهد بود:

- مثال‌های عملی و سناریوهای بومی **:**به‌جای استفاده از داده‌ها و سناریوهای عمومی، از مثال‌هایی استفاده می‌کنیم که برای کسب‌وکارهای ایرانی ملموس و قابل‌درک باشند. این شامل تحلیل فروش محصولات خاص بازار ایران، بررسی رفتار مشتریان ایرانی، و تحلیل داده‌های مالی بر اساس استانداردهای حسابداری ایران است.
- تقویم شمسی و هوش زمانی **:**یکی از بزرگ‌ترین چالش‌ها در تحلیل داده در ایران، عدم تطابق تقویم میلادی با تقویم رسمی کشور است. ما در این کتاب به طور مفصل به نحوه پیاده‌سازی هوش زمانی (Time Intelligence) با استفاده از تقویم شمسی در DAX خواهیم پرداخت و راه‌حل‌های عملی برای این موضوع ارائه خواهیم داد.
- واژه‌نامه و اصطلاحات فارسی **:**تلاش می‌کنیم تا در کنار اصطلاحات تخصصی انگلیسی، معادل‌های فارسی رایج و مناسب را نیز ارائه دهیم تا درک مطالب برای خواننده فارسی‌زبان آسان‌تر شود.
- چالش‌های خاص داده‌ای در ایران **:**به مسائلی مانند کیفیت داده‌ها، فرمت‌های داده‌ای رایج در ایران، و نحوه پاک‌سازی و آماده‌سازی داده‌ها برای تحلیل در محیط‌های ایرانی می‌پردازیم.

ساختار کتاب

این کتاب به‌گونه‌ای ساختاریافته است که شما را از مفاهیم پایه هوش تجاری و Power BI به سمت تسلط بر DAX و سناریوهای تحلیلی پیشرفته هدایت کند. هر فصل بر روی یک جنبه خاص تمرکز دارد و با مثال‌های عملی و توضیحات گام‌به‌گام همراه است. ساختار کلی کتاب به شرح زیر است:

- فصل ۱: هوش تجاری و **Power BI** برای کسب‌وکارهای ایرانی **:**معرفی مفاهیم پایه هوش تجاری، اهمیت آن در بازار ایران، و آشنایی با Power BI به‌عنوان ابزار اصلی.
- فصل ۲: طراحی مدل داده بهینه **:**اصول طراحی مدل داده ستاره‌ای ،(Star Schema) روابط بین جداول، و اهمیت یک مدل داده قوی برای تحلیل‌های.DAX
- فصل ۳: زبان **- DAX** شروعی قدرتمند **:**معرفی سینتکس ،DAX انواع توابع، و اولین فرمول‌های عملی برای محاسبه شاخص‌های کلیدی عملکرد.(KPIs)
- فصل ۴: توابع تکرارکننده **(Iterators)** و توابع جدول **:(Table Functions)** درک عمیق‌تر توابع DAX که بر روی جداول یا ردیف‌ها عمل می‌کنند و کاربرد آن‌ها در سناریوهای پیچیده.

- فصل ۵: توابع فیلتر **(Filter Functions)** و توابع رابطه **:(Relationship Functions)** کنترل زمینه فیلتر (Filter Context)و استفاده از روابط برای ایجاد تحلیل‌های پویا.
- فصل ۶: هوش زمانی **(Time Intelligence)** با تقویم شمسی **:**پیاده‌سازی تحلیل‌های زمانی مانند مقایسه سال‌به‌سال، ماه‌به‌ماه، و محاسبه رشد با استفاده از تقویم شمسی.
- فصل ۷: سناریوهای پیشرفته تحلیلی **:**بررسی سناریوهای پیچیده‌تر مانند تحلیل سبد خرید (Market Basket Analysis)،تحلیل ABC، و تخصیص مقادیر.(Allocation)
- فصل ۸: بهینه‌سازی عملکرد **:(DAX Performance Tuning)** شناسایی گلوگاه‌های عملکردی در فرمول‌های DAXو روش‌های بهینه‌سازی برای گزارش‌های سریع‌تر.
- فصل ۹: مدیریت خطا و دیباگینگ در **:DAX** تکنیک‌های شناسایی و رفع خطا در فرمول‌های DAX برای اطمینان از صحت محاسبات.
- فصل **DAX :۱۰** و **- Power Query** هم‌افزایی برای تحلیل بهتر **:**نحوه استفاده از Power Query برای آماده‌سازی داده‌ها و نقش آن در تکمیل قابلیت‌های.DAX
- فصل ۱۱: سناریوهای پیشرفته هوش زمانی و مالی **:**عمیق‌تر شدن در تحلیل‌های زمانی پیچیده و کاربرد DAX در تحلیل‌های مالی پیشرفته.
- فصل ۱۲: بهترین روش‌ها و الگوهای طراحی در **:DAX** جمع‌بندی و ارائه بهترین روش‌ها و الگوهای طراحی برای نوشتن فرمول‌های DAX کارآمد و قابل نگهداری.

چگونه از این کتاب استفاده کنید؟

این کتاب به‌گونه‌ای طراحی شده است که هم برای مطالعه پیوسته و هم به‌عنوان یک مرجع قابل‌استفاده باشد. توصیه می‌کنیم که فصول را به ترتیب مطالعه کنید تا مفاهیم به‌تدریج و به‌صورت منطقی در ذهن شما جای بگیرند. هر فصل شامل مثال‌های عملی است که می‌توانید آن‌ها را در Power BI پیاده‌سازی کنید. فایل‌های داده و نمونه‌های Power BI برای این کتاب در دسترس خواهند بود تا بتوانید همراه با مطالعه، تمرین کنید.

سخن پایانی

امیدواریم این کتاب نه‌تنها دانش شما را در زمینه DAX و Power BI افزایش دهد، بلکه به شما کمک کند تا با دیدی عمیق‌تر و بومی‌تر، داده‌های کسب‌وکار خود را تحلیل کرده و بینش‌های ارزشمندی را برای تصمیم‌گیری‌های بهتر استخراج کنید. سفر شما در دنیای تحلیل داده با DAX، از همین‌جا آغاز می‌شود.

فصل ۱: هوش تجاری و Power BI برای کسب‌وکارهای ایرانی

در عصر حاضر، داده‌ها به‌عنوان شریان حیاتی هر کسب‌وکاری شناخته می‌شوند. از کوچک‌ترین استارتاپ‌ها تا بزرگ‌ترین شرکت‌های چندملیتی، همگی به دنبال راهی برای جمع‌آوری، سازماندهی، تحلیل و در نهایت، استخراج بینش از حجم عظیمی از داده‌هایی هستند که روزانه تولید می‌شوند. اینجاست که مفهوم هوش تجاری **(Business Intelligence - BI)** وارد میدان می‌شود. هوش تجاری صرفاً یک ابزار یا یک فناوری نیست، بلکه یک رویکرد جامع برای تبدیل داده‌های خام به اطلاعات معنی‌دار و دانش قابل اقدام است که به تصمیم‌گیرندگان کمک می‌کند تا تصمیمات آگاهانه‌تر و بهتری بگیرند.

۱-۱هوش تجاری چیست و چرا برای کسب‌وکارهای ایرانی حیاتی است؟:

هوش تجاری مجموعه‌ای از فرایندها، فناوری‌ها و ابزارهاست که برای جمع‌آوری، ذخیره‌سازی، تحلیل و دسترسی به داده‌ها باهدف کمک به تصمیم‌گیری‌های بهتر در کسب‌وکار طراحی شده است. این فرایند شامل مراحل کلیدی زیر است:

1. جمع‌آوری داده‌ها **:**از منابع مختلف مانند سیستم‌های ERP، CRM، پایگاه‌های داده، فایل‌های اکسل، وب‌سایت‌ها و شبکه‌های اجتماعی.
2. پاک‌سازی و تبدیل داده‌ها **:(ETL)** داده‌های خام اغلب دارای خطا، ناسازگاری یا فرمت‌های نامناسب هستند. در این مرحله، داده‌ها پاک‌سازی، تبدیل و برای تحلیل آماده می‌شوند.
3. ذخیره‌سازی داده‌ها **:**داده‌های آماده شده در انبارهای داده (Data Warehouses) یا دیتا مارت‌ها (Data Marts) ذخیره می‌شوند که برای تحلیل بهینه شده‌اند.
4. تحلیل داده‌ها **:**با استفاده از ابزارهای تحلیلی و زبان‌های پرس‌وجو مانند DAX، الگوها، روندها و بینش‌های پنهان در داده‌ها کشف می‌شوند.
5. گزارش‌دهی و بصری‌سازی **:**نتایج تحلیل‌ها در قالب گزارش‌ها، داشبوردها و نمودارهای بصری جذاب ارائه می‌شوند تا تصمیم‌گیرندگان بتوانند به‌راحتی آن‌ها را درک کنند.

چرا هوش تجاری برای کسب‌وکارهای ایرانی حیاتی است؟

بازار ایران دارای ویژگی‌ها و چالش‌های منحصربه‌فردی است که اهمیت هوش تجاری را دوچندان می‌کند. برخی از این دلایل عبارت‌اند از:

- نوسانات اقتصادی و عدم قطعیت **:**اقتصاد ایران همواره با نوسانات و عدم قطعیت‌های زیادی همراه است. هوش تجاری به کسب‌وکارها کمک می‌کند تا با تحلیل داده‌های لحظه‌ای، تغییرات بازار را سریع‌تر شناسایی کرده و تصمیمات چابک‌تری برای مقابله با ریسک‌ها یا بهره‌برداری از فرصت‌ها بگیرند.

- رقابت فزاینده :در بسیاری از صنایع، رقابت بسیار شدید است. کسب‌وکارهایی که بتوانند با استفاده از داده‌ها، رفتار مشتریان خود را بهتر درک کنند، فرایندهای داخلی خود را بهینه سازند و مزیت‌های رقابتی جدیدی ایجاد کنند، موفق‌تر خواهند بود.
- پیچیدگی داده‌ها و منابع متعدد :کسب‌وکارهای ایرانی نیز مانند سایر نقاط جهان، با حجم زیادی از داده‌ها از منابع مختلف روبرو هستند. بدون یک سیستم هوش تجاری منسجم، استخراج بینش از این داده‌های پراکنده تقریباً غیرممکن است.
- نیاز به بومی‌سازی تحلیل‌ها :بسیاری از ابزارهای تحلیلی جهانی، به طور پیش‌فرض برای تقویم میلادی، واحدهای پولی بین‌المللی و فرهنگ کسب‌وکار غربی طراحی شده‌اند. هوش تجاری به کسب‌وکارها این امکان را می‌دهد که این ابزارها را با نیازهای بومی خود تطبیق دهند، به‌عنوان‌مثال با استفاده از تقویم شمسی در تحلیل‌های زمانی.
- بهبود فرایندهای داخلی :هوش تجاری نه‌تنها به تصمیم‌گیری‌های استراتژیک کمک می‌کند، بلکه می‌تواند با شناسایی ناکارآمدی‌ها در فرایندهای عملیاتی، به بهبود بهره‌وری و کاهش هزینه‌ها نیز منجر شود.
- درک عمیق‌تر مشتری :با تحلیل داده‌های مشتریان، کسب‌وکارها می‌توانند الگوهای خرید، ترجیحات و نیازهای آن‌ها را بهتر درک کنند و بر اساس آن، استراتژی‌های بازاریابی و فروش خود را شخصی‌سازی کنند.

در نهایت، هوش تجاری به کسب‌وکارهای ایرانی این قدرت را می‌دهد که از حالت واکنشی به حالت پیش‌بینی‌کننده و فعال تغییر کنند. به‌جای اینکه صرفاً به رویدادها واکنش نشان دهند، می‌توانند با پیش‌بینی روندها و شناسایی فرصت‌ها، آینده خود را شکل دهند.

۱-۲:معرفی :Power BI ابزار پیشرو در هوش تجاری

در میان ابزارهای متعدد هوش تجاری موجود در بازار **Microsoft Power BI** ،به‌سرعت به یکی از محبوب‌ترین و قدرتمندترین گزینه‌ها تبدیل شده است Power BI .مجموعه‌ای از سرویس‌های نرم‌افزاری، برنامه‌ها و کانکتورهاست که داده‌های نامرتبط شما را به بینش‌های بصری و تعاملی تبدیل می‌کند. این ابزار به کاربران امکان می‌دهد تا به‌راحتی به منابع داده مختلف متصل شوند، داده‌ها را مدل‌سازی و تبدیل کنند، گزارش‌ها و داشبوردهای جذاب ایجاد کنند و آن‌ها را با دیگران به اشتراک بگذارند.

چرا **Power BI**؟

- سهولت استفاده و رابط کاربری بصری Power BI :با رابط کاربری شبیه به سایر محصولات مایکروسافت (مانند اکسل)، برای بسیاری از کاربران آشنا و قابل‌دسترس است. این امر منحنی یادگیری را کاهش می‌دهد و به کاربران کسب‌وکار اجازه می‌دهد تا بدون نیاز به دانش برنامه‌نویسی عمیق، تحلیل‌های قدرتمندی انجام دهند.
- قابلیت اتصال به منابع داده متنوع Power BI :می‌تواند به صدها منبع داده مختلف، از پایگاه‌های داده محلی (SQL Server، Oracle)، سرویس‌های ابری (مانند Azure SQL Database، Google Analytics)، فایل‌های اکسل و CSV، تا سرویس‌های آنلاین (مانند Dynamics 365، Salesforce) متصل شود.

- قدرت مدل‌سازی داده Power BI **:**دارای یک موتور مدل‌سازی داده بسیار قدرتمند به نام **VertiPaq** است که امکان فشرده‌سازی و ذخیره‌سازی حجم عظیمی از داده‌ها را با عملکرد بالا فراهم می‌کند. این موتور، پایه و اساس زبان DAX است.
- زبان **DAX** برای محاسبات پیشرفته (DAX (Data Analysis Expressions **:**زبان فرمول‌نویسی Power BIاست که به کاربران امکان می‌دهد تا محاسبات پیچیده، شاخص‌های سفارشی و تحلیل‌های پیشرفته را ایجاد کنند. این کتاب به طور کامل به این زبان اختصاص دارد.
- بصری‌سازی‌های تعاملی و جذاب Power BI **:**طیف وسیعی از نمودارها، گراف‌ها و ابزارهای بصری‌سازی را ارائه می‌دهد که به کاربران کمک می‌کند تا داستان داده‌های خود را به شکلی جذاب و قابل‌فهم روایت کنند. این بصری‌سازی‌ها کاملاً تعاملی هستند و به کاربران اجازه می‌دهند تا با داده‌ها بازی کنند و بینش‌های جدیدی کشف کنند.
- قابلیت اشتراک‌گذاری و همکاری **:**گزارش‌ها و داشبوردهای ایجاد شده در Power BI Desktop را می‌توان به سرویس Power BI (Power BI Service)منتشر کرد و با دیگران به اشتراک گذاشت. این امر همکاری تیمی را تسهیل می‌کند و اطمینان می‌دهد که همه تصمیم‌گیرندگان به یک منبع واحد از حقیقت دسترسی دارند.
- جامعه کاربری بزرگ و پشتیبانی قوی Power BI **:**دارای یک جامعه کاربری بسیار فعال و بزرگ در سراسر جهان است که منابع آموزشی فراوان، انجمن‌های پشتیبانی و مثال‌های عملی را ارائه می‌دهد. مایکروسافت نیز به طور مداوم این ابزار را با ویژگی‌های جدید و بهبودها به‌روزرسانی می‌کند.
- قیمت‌گذاری منعطف Power BI **:**در نسخه‌های مختلفی از جمله نسخه رایگان ،(Desktop) نسخه Pro و نسخه Premiumارائه می‌شود که آن را برای کسب‌وکارهای با اندازه‌ها و بودجه‌های مختلف قابل‌دسترس می‌کند.

۱-۳:اجزای اصلی Power BI

Power BIاز چندین جزء اصلی تشکیل شده است که هر یک نقش خاصی در فرایند هوش تجاری ایفا می‌کنند:

1. **Power BI Desktop:**این برنامه رایگان ویندوزی، قلب Power BI است. بیشتر کارهای توسعه، شامل اتصال به داده‌ها، تبدیل و مدل‌سازی داده‌ها (با Power Query و ،(DAX و ایجاد گزارش‌ها و داشبوردها در این محیط انجام می‌شود Power BI Desktop .شامل سه نمای اصلی است:

- نمای گزارش **:(Report View)** جایی که بصری‌سازی‌ها را ایجاد می‌کنید و گزارش‌های خود را طراحی می‌کنید.
- نمای داده **:(Data View)** جایی که می‌توانید داده‌های خود را در قالب جدول مشاهده کنید و تغییرات اعمال شده توسط Power Query را بررسی کنید.

- نمای مدل **:(Model View)** جایی که می‌توانید جداول خود را سازماندهی کنید، روابط بین آن‌ها را ایجاد و مدیریت کنید، و سلسله‌مراتب (Hierarchies) و نقش‌ها (Roles) را تعریف کنید. این نما برای مدل‌سازی داده بسیار حیاتی است.

2. **Power BI Service (**سرویس **:Power BI** این یک سرویس مبتنی بر ابر (SaaS - Software as a Service)است که برای انتشار، اشتراک‌گذاری و مصرف گزارش‌ها و داشبوردها استفاده می‌شود. کاربران می‌توانند از طریق مرورگر وب به این سرویس دسترسی پیدا کنند. ویژگی‌های کلیدی آن عبارت‌اند از:

- داشبوردها **:**مجموعه‌ای از بصری‌سازی‌ها از یک یا چند گزارش که دیدی جامع از شاخص‌های کلیدی عملکرد ارائه می‌دهند.
- فضاهای کاری **:(Workspaces)** محیط‌هایی برای همکاری تیمی که در آن گزارش‌ها، داشبوردها و مجموعه‌های داده (Datasets) را می‌توان به اشتراک گذاشت.
- دروازه‌های داده **:(Data Gateways)** برای اتصال امن به منابع داده محلی (On-premises) از سرویس Power BI استفاده می‌شوند.
- تازه‌سازی برنامه‌ریزی شده **:(Scheduled Refresh)** امکان به‌روزرسانی خودکار داده‌ها در فواصل زمانی مشخص.

3. **Power BI Mobile Apps:**برنامه‌های موبایل Power BI برای دستگاه‌های iOS، Android و Windows، به کاربران امکان می‌دهند تا گزارش‌ها و داشبوردهای خود را در هر زمان و مکانی مشاهده و با آن‌ها تعامل داشته باشند.

4. **Power BI Report Server:**برای سازمان‌هایی که نیاز به میزبانی گزارش‌های Power BI در محیط داخلی خود (On-premises) دارند Power BI Report Server ،یک گزینه مناسب است. این ابزار به آن‌ها اجازه می‌دهد تا گزارش‌ها را بدون نیاز به فضای ابری مایکروسافت منتشر و مدیریت کنند.

5. **Power BI Embedded:**این سرویس به توسعه‌دهندگان امکان می‌دهد تا گزارش‌ها و داشبوردهای Power BIرا در برنامه‌های کاربردی خود (مانند وب‌سایت‌ها یا نرم‌افزارهای سفارشی) جاسازی (Embed) کنند.

Power Query :۴-۱و :DAX دوروی یک سکه

در اکوسیستم Power BI، دو زبان قدرتمند وجود دارند که هر یک نقش مکملی را ایفا می‌کنند**) Power Query** :زبان **M**و .**(DAX (Data Analysis Expressions** درک تفاوت و کاربرد هر یک از این زبان‌ها برای ساخت مدل‌های داده‌ای کارآمد و تحلیل‌های دقیق ضروری است.

Power Query (زبان **:(M** آماده‌سازی و تبدیل داده‌ها

Power Queryیک ابزار (Extract, Transform, Load) ETL قدرتمند است که در) Power BI Desktop و همچنین اکسل) تعبیه شده است. وظیفه اصلی ،Power Query اتصال به منابع داده، استخراج داده‌ها، پاک‌سازی، تبدیل و بارگذاری آن‌ها در مدل داده Power BI است. زبان پشت ،Power Query زبان M نامیده می‌شود. کارهایی که معمولاً با Power Queryانجام می‌دهند عبارت‌اند از:

- اتصال به منابع داده **:**از پایگاه‌های داده، فایل‌ها، وب‌سایت‌ها و سرویس‌های آنلاین.
- پاک‌سازی داده‌ها **:**حذف ردیف‌های تکراری، پر کردن مقادیر خالی، اصلاح خطاهای تایپی.
- تبدیل داده‌ها **:**تغییر نوع داده‌ها، تقسیم ستون‌ها، ادغام ستون‌ها Pivot ،و Unpivot کردن جداول.
- ادغام داده‌ها **:**ترکیب داده‌ها از چندین منبع (Merge Queries) یا افزودن ردیف‌ها از جداول مختلف (Append Queries).
- ایجاد ستون‌های سفارشی **:**با استفاده از زبان ،M می‌توانید ستون‌های جدیدی بر اساس منطق خاصی ایجاد کنید.

Power Queryبر روی داده‌ها قبل از بارگذاری آن‌ها در مدل داده عمل می‌کند. به‌عبارت‌دیگر، تغییراتی که در Power Queryاعمال می‌کنید، دائمی هستند و بر روی داده‌های اصلی شما در مدل داده Power BI تأثیر می‌گذارند. این مرحله برای اطمینان از کیفیت و ساختار صحیح داده‌ها قبل از تحلیل بسیار حیاتی است.

DAX (Data Analysis Expressions):تحلیل و محاسبه داده‌ها

DAXیک‌زبان فرمول‌نویسی است که برای ایجاد محاسبات و تحلیل‌های پیشرفته بر روی داده‌های موجود در مدل داده Power BIو همچنین SQL Server Analysis Services و (Excel Power Pivot) استفاده می‌شود DAX .برای کارهایی مانند:

- ایجاد **Measures (**معیارها **:(**محاسباتی که نتایج آن‌ها بر اساس زمینه فیلتر (Filter Context) تغییر می‌کند. مثال: مجموع فروش، میانگین سود، تعداد مشتریان فعال.
- ایجاد **Calculated Columns (**ستون‌های محاسباتی **:(**ستون‌های جدیدی که بر اساس فرمول‌های DAX ایجاد می‌شوند و به مدل داده اضافه می‌شوند. این ستون‌ها مانند ستون‌های عادی عمل می‌کنند و می‌توانند در بصری‌سازی‌ها و فیلترها استفاده شوند. مثال: سود ناخالص، سن مشتری.
- ایجاد **Calculated Tables (**جداول محاسباتی **:(**جداول جدیدی که با استفاده از فرمول‌های DAX ایجاد می‌شوند. این جداول می‌توانند برای سناریوهای پیشرفته مدل‌سازی داده مفید باشند. مثال: جدول تاریخ (Date Table) سفارشی.

تفاوت کلیدی بین Power Query و DAX در این است که Power Query بر روی داده‌ها عمل می‌کند و آن‌ها را برای تحلیل آماده می‌سازد، درحالی‌که DAX بر روی مدل داده عمل می‌کند و محاسبات و تحلیل‌ها را بر روی داده‌های آماده شده انجام می‌دهدPower Query .

داده‌ها را تبدیل می‌کند، درحالی‌که DAX داده‌ها را تحلیل می‌کند. Power Query قبل از بارگذاری داده‌ها اجرا می‌شود (زمان طراحی مدل)، درحالی‌که DAX در زمان اجرای گزارش‌ها و تعامل کاربر با آن‌ها (زمان تحلیل) اجرا می‌شود.

برای مثال، اگر نیاز به پاک‌سازی و استانداردسازی ستون‌های تاریخ دارید (مثلاً تبدیل فرمت‌های مختلف تاریخ به یک فرمت واحد)، این کار را باید در Power Query انجام دهید. اما اگر می‌خواهید فروش سال‌به‌سال را مقایسه کنید یا میانگین فروش ماهانه را محاسبه کنید، این کار را باید با DAX انجام دهید.

درک این تمایز بسیار مهم است، زیرا استفاده صحیح از هر یک از این ابزارها به شما کمک می‌کند تا مدل‌های داده‌ای کارآمدتر و گزارش‌های دقیق‌تری بسازید. تلاش برای انجام کارهای Power Query در) DAX یا بالعکس) معمولاً منجر به فرمول‌های پیچیده، ناکارآمد و دشوار برای نگهداری می‌شود.

۱-۵:نصب و راه‌اندازیPower BI Desktop

برای شروع کار با Power BI و ،DAX اولین قدم نصب Power BI Desktop است. این نرم‌افزار رایگان است و به‌راحتی می‌توانید آن را از وب‌سایت مایکروسافت دانلود و نصب کنید.

مراحل نصب:

1. دانلود **Power BI Desktop:** به وب‌سایت رسمی مایکروسافت برای Power BI مراجعه کنید. می‌توانید عبارت "Power BI Desktop download"را در گوگل جستجو کنید یا مستقیماً به آدرس https://powerbi.microsoft.com/desktop/بروید. در صفحه دانلود، گزینه دانلود رایگان را انتخاب کنید. توصیه می‌شود نسخه ۶۴ بیتی را دانلود کنید. 2. اجرای فایل نصب **:**پس از دانلود، فایل PBIDesktopSetup.exe را اجرا کنید. مراحل نصب را دنبال کنید. معمولاً می‌توانید تنظیمات پیش‌فرض را قبول کنید. 3. اولین اجرا **:**پس از اتمام نصب ، Power BI Desktopرا اجرا کنید. ممکن است از شما خواسته شود با حساب مایکروسافت خود وارد شوید. این کار برای استفاده از برخی قابلیت‌های آنلاین و اشتراک‌گذاری گزارش‌ها ضروری است، اما برای شروع کار با DAX و مدل‌سازی داده، اجباری نیست.

اولین نگاه به محیط کار**:Power BI Desktop**

پس از باز کردن ،Power BI Desktop با یک رابط کاربری آشنا مواجه خواهید شد که شامل بخش‌های اصلی زیر است:

- نوار ریبون **:(Ribbon)** در بالای صفحه قرار دارد و شامل تب‌های مختلفی مانند ،Home ،Insert ،Modeling View و Help است. هر تب شامل گروه‌هایی از ابزارها و دستورات مرتبط است.
- نمای گزارش **:(Report View)** ناحیه اصلی در مرکز صفحه که برای طراحی گزارش‌ها و بصری‌سازی‌ها استفاده می‌شود. در این نما، می‌توانید نمودارها، جداول و سایر عناصر بصری را اضافه و تنظیم کنید.
- نمای داده **:(Data View)** با کلیک بر روی آیکون جدول در سمت چپ صفحه، به این نما منتقل می‌شوید. در اینجا می‌توانید داده‌های بارگذاری شده در مدل خود را در قالب جدول مشاهده کنید. این نما برای بررسی داده‌ها، مرتب‌سازی و فیلترکردن آن‌ها مفید است.
- نمای مدل **:(Model View)** با کلیک بر روی آیکون مدل (سه جدول متصل‌به‌هم) در سمت چپ صفحه، به این نما منتقل می‌شوید. این نما برای طراحی و مدیریت مدل داده شما، شامل ایجاد و مدیریت روابط بین جداول، تعریف سلسله‌مراتب و نقش‌ها، و مشاهده ساختار کلی مدل بسیار حیاتی است.
- پنل فیلدها **:(Fields Pane)** در سمت راست صفحه قرار دارد و لیست تمام جداول و ستون‌های موجود در مدل داده شما را نمایش می‌دهد. از اینجا می‌توانید فیلدها را به بصری‌سازی‌ها بکشید و رها کنید.
- پنل بصری‌سازی‌ها **:(Visualizations Pane)** در سمت راست صفحه، زیر پنل فیلدها قرار دارد و شامل انواع مختلفی از بصری‌سازی‌ها (نمودارها، جداول، کارت‌ها و غیره) است که می‌توانید برای نمایش داده‌های خود استفاده کنید.
- پنل فیلترها **:(Filters Pane)** در سمت راست صفحه، زیر پنل بصری‌سازی‌ها قرار دارد و برای اعمال فیلترها بر روی گزارش‌ها، صفحات یا بصری‌سازی‌های خاص استفاده می‌شود.

در طول این کتاب، ما به طور مفصل با هر یک از این بخش‌ها کار خواهیم کرد و نحوه استفاده از آن‌ها را برای ساخت مدل‌های داده‌ای قدرتمند و گزارش‌های تحلیلی جذاب خواهیم آموخت.

۱-۶: اتصال به داده‌ها و اولین بارگذاری

قبل از اینکه بتوانیم با DAX کار کنیم، نیاز داریم که داده‌هایی را به Power BI Desktop وارد کنیم Power BI .از منابع داده بسیار متنوعی پشتیبانی می‌کند. در این بخش، ما با یک مثال ساده، نحوه اتصال به یک فایل اکسل و بارگذاری داده‌ها را بررسی می‌کنیم.

مراحل اتصال به فایل اکسل:

1. دریافت داده **:(Get Data)** در تب Home نوار ریبون، بر روی دکمه `Get Data` کلیک کنید. یک منوی کشویی باز می‌شود که شامل رایج‌ترین منابع داده است. گزینه `Excel workbook` را انتخاب کنید.

2. انتخاب فایل **:**یک پنجره باز می‌شود که از شما می‌خواهد فایل اکسل موردنظر را انتخاب کنید. فایل اکسل نمونه‌ای را که برای این فصل آماده کرده‌اید (مثلا `SalesData.xlsx`) انتخاب کرده و بر روی `Open` کلیک کنید.
3. **Navigator:** پنجره `Navigator` باز می‌شود. در این پنجره، می‌توانید شیت‌ها یا جداول موجود در فایل اکسل خود را مشاهده کنید. شیت یا جدولی را که حاوی داده‌های موردنظر شماست (مثلا `Sales`) انتخاب کنید. در سمت راست، پیش‌نمایشی از داده‌ها را مشاهده خواهید کرد.
4. **Load** یا **Transform Data:** در پایین پنجره `Navigator`، دو گزینه اصلی وجود دارد:

- **Load:** اگر داده‌ها تمیز و آماده تحلیل هستند و نیازی به تغییرات اولیه ندارند، بر روی `Load` کلیک کنید. داده‌ها مستقیماً به مدل داده Power BI بارگذاری می‌شوند.
- **Transform Data:** اگر داده‌ها نیاز به پاک‌سازی، تبدیل یا تغییر شکل دارند، بر روی `Transform Data` کلیک کنید. این کار `Power Query Editor` را باز می‌کند که در آن می‌توانید تمام عملیات ETL را انجام دهید. (ما در فصل‌های بعدی به طور مفصل به Power Query خواهیم پرداخت.)

برای مثال فعلی، فرض کنید داده‌های شما نسبتاً تمیز هستند. بر روی `Load` کلیک کنید. پس از چند لحظه، داده‌ها در Power BI Desktop بارگذاری می‌شوند. می‌توانید در پنل `Fields` در سمت راست، جدول `Sales` و ستون‌های آن را مشاهده کنید.

بررسی داده‌های بارگذاری شده**:**

پس از بارگذاری داده‌ها، به `Data View` (نمای داده) بروید. در اینجا می‌توانید داده‌های جدول `Sales` را مشاهده کنید. مطمئن شوید که انواع داده‌ها (Data Types) برای هر ستون به‌درستی شناسایی شده‌اند (مثلاً ستون تاریخ به‌عنوان تاریخ، ستون عدد به‌عنوان عدد). اگر نوع داده‌ای اشتباه بود، می‌توانید با کلیک راست بر روی سربرگ ستون و انتخاب `Change Type`، آن را اصلاح کنید.

این اولین گام شما در مسیر تحلیل داده با Power BI است. با داشتن داده‌ها در مدل، اکنون آماده‌ایم تا به سراغ مفاهیم مدل‌سازی داده و سپس زبان قدرتمند DAX برویم.

فصل ۲: طراحی مدل داده بهینه

مدل داده، ستون فقرات هر راهکار هوش تجاری موفق است. یک مدل داده خوب طراحی شده، نه‌تنها عملکرد گزارش‌ها و داشبوردها را بهبود می‌بخشد، بلکه درک و استفاده از داده‌ها را برای کاربران آسان‌تر می‌کند و امکان ایجاد تحلیل‌های پیچیده با DAX را

فراهم می‌آورد. در مقابل، یک مدل داده ضعیف می‌تواند منجر به محاسبات نادرست، عملکرد کند و دشواری در نگهداری شود. در این فصل، به اصول طراحی مدل داده بهینه در Power BI، با تمرکز بر مفهوم مدل ستاره‌ای **(Star Schema)** خواهیم پرداخت.

۲-۱: چرا مدل داده مهم است؟

تصور کنید یک کتابخانه بزرگ دارید که کتاب‌ها به‌صورت تصادفی در قفسه‌ها چیده شده‌اند. پیداکردن یک کتاب خاص در چنین کتابخانه‌ای بسیار دشوار و زمان‌بر خواهد بود. اما اگر کتاب‌ها بر اساس موضوع، نویسنده و سال انتشار مرتب شده باشند، پیداکردن هر کتابی آسان می‌شود. مدل داده در Power BI نیز دقیقاً همین نقش را ایفا می‌کند؛ داده‌های خام و پراکنده را به شکلی سازماندهی می‌کند که به‌راحتی قابل‌دسترسی، تحلیل و درک باشند.

اهمیت مدل داده از جنبه‌های مختلفی قابل‌بررسی است:

- صحت محاسبات **:**مدل داده صحیح، تضمین می‌کند که محاسبات DAX شما نتایج دقیق و معتبری را ارائه می‌دهند. روابط نادرست یا ساختار نامناسب جداول می‌تواند منجر به نتایج اشتباه شود.
- عملکرد **:(Performance)** یک مدل داده بهینه، به موتور VertiPaq Power BI اجازه می‌دهد تا داده‌ها را به‌صورت کارآمد فشرده‌سازی و پردازش کند. این امر منجر به بارگذاری سریع‌تر گزارش‌ها و پاسخگویی بهتر بصری‌سازی‌ها می‌شود، حتی با حجم عظیمی از داده‌ها.
- سهولت استفاده **:**مدل داده‌ای که به‌خوبی طراحی شده باشد، برای کاربران نهایی (حتی کسانی که دانش فنی عمیقی ندارند) قابل‌درک است. نام‌گذاری مناسب جداول و ستون‌ها، و سازماندهی منطقی داده‌ها، به کاربران کمک می‌کند تا به‌راحتی فیلدها را پیدا کرده و گزارش‌های خود را بسازند.
- انعطاف‌پذیری برای تحلیل **:**یک مدل داده قوی، امکان ایجاد انواع تحلیل‌ها را فراهم می‌کند. با داشتن ابعاد (Dimensions)و حقایق (Facts) مناسب، می‌توانید به سؤالات کسب‌وکار مختلفی پاسخ دهید و بینش‌های عمیقی را استخراج کنید.
- قابلیت نگهداری **:**مدل‌های داده‌ای که بر اساس اصول طراحی صحیح ساخته شده‌اند، آسان‌تر نگهداری و توسعه می‌یابند. اضافه‌کردن داده‌های جدید، تغییر ساختار یا افزودن تحلیل‌های جدید، در یک مدل سازماندهی شده بسیار ساده‌تر است.

۲-۲: مفهوم) Star Schema مدل ستاره‌ای(

) **Star Schema**مدل ستاره‌ای) رایج‌ترین و توصیه‌شده‌ترین الگوی طراحی مدل داده برای هوش تجاری و انبار داده است. این الگو به دلیل سادگی، کارایی و قابلیت درک بالا، به‌عنوان بهترین روش در Power BI شناخته می‌شود. در یک مدل ستاره‌ای، داده‌ها به دو نوع اصلی از جداول تقسیم می‌شوند:

1. جداول حقیقت **:(Fact Tables)** این جداول حاوی مقادیر عددی و قابل‌اندازه‌گیری (Metrics) هستند که نشان‌دهنده رویدادها یا تراکنش‌های کسب‌وکار هستند. مثال‌ها:

- فروش **:**شامل مقادیری مانند مقدار فروش، تعداد اقلام فروخته شده، تخفیف.
- سفارش‌ها **:**شامل تعداد سفارش‌ها، مبلغ کل سفارش.
- تراکنش‌های مالی **:**شامل مبلغ بدهکار، مبلغ بستانکار.
- بازدید وب‌سایت **:**شامل تعداد بازدیدها، زمان صرف شده در صفحه.

جداول حقیقت معمولاً بسیار بزرگ هستند و حاوی میلیون‌ها یا میلیاردها ردیف داده هستند. آن‌ها همچنین حاوی کلیدهای خارجی (Foreign Keys) هستند که به جداول ابعاد مرتبط می‌شوند.

2. جداول ابعاد **:(Dimension Tables)** این جداول حاوی ویژگی‌ها و توصیف‌کننده‌هایی هستند که برای فیلترکردن، گروه‌بندی و دسته‌بندی داده‌های جداول حقیقت استفاده می‌شوند. جداول ابعاد،

معمولاً کوچک‌تر از جداول حقیقت هستند و حاوی اطلاعات توصیفی در مورد موجودیت‌های کسب‌وکار هستند. مثال‌ها: ❋ بعد زمان **:(Date Dimension)** شامل سال، ماه، روز، فصل، تعطیلات و سایر ویژگی‌های مربوط به زمان. (بسیار مهم برای هوش زمانی) ❋ بعد محصول **:(Product Dimension)** شامل نام محصول، دسته محصول، برند، رنگ، اندازه. ❋ بعد مشتری **:(Customer Dimension)**شامل نام مشتری، جنسیت، سن، شهر، استان، نوع مشتری. ❋ بعد فروشنده **:(Salesperson Dimension)**شامل نام فروشنده، منطقه فروش، عملکرد. ❋ بعد مکان **(Location :(Dimension**شامل شهر، استان، کشور، منطقه جغرافیایی.

هر جدول بعد دارای یک کلید اصلی (Primary Key) است که به کلید خارجی متناظر در جدول حقیقت مرتبط می‌شود. این روابط، اساس فیلترکردن و تحلیل داده‌ها در Power BI را تشکیل می‌دهند.

مزایای**:Star Schema**

- سادگی و قابلیت درک **:**ساختار ستاره‌ای بسیار ساده و قابل‌درک است. کاربران به‌راحتی می‌توانند جداول حقیقت و ابعاد را شناسایی کرده و نحوه ارتباط آن‌ها را درک کنند.
- عملکرد بالا **:**به دلیل ساختار ساده و روابط مستقیم، موتور VertiPaq Power BI می‌تواند داده‌ها را به‌سرعت فشرده‌سازی و پرس‌وجو کند. این امر منجر به عملکرد عالی در گزارش‌ها و داشبوردها می‌شود.
- انعطاف‌پذیری **:**اضافه‌کردن ابعاد جدید یا ویژگی‌های جدید به ابعاد موجود، بدون تأثیر زیاد بر ساختار کلی مدل، آسان است.

- پشتیبانی قوی از DAX **DAX:** به طور طبیعی با مدل‌های ستاره‌ای کار می‌کند. توابع DAX برای فیلترکردن و تجمیع داده‌ها در این ساختار بسیار کارآمد هستند.
- کاهش افزونگی داده **(Data Redundancy):** اطلاعات توصیفی (ابعاد) تنها یک‌بار ذخیره می‌شوند و از طریق کلیدها به جداول حقیقت مرتبط می‌شوند که این امر باعث کاهش فضای ذخیره‌سازی و بهبود کیفیت داده‌ها می‌شود.

۳-۲:طراحی جداول ابعاد و حقیقت در Power BI

برای طراحی یک مدل ستاره‌ای در Power BI، باید مراحل زیر را دنبال کنید:

1. شناسایی جداول حقیقت **:**ابتدا باید رویدادها یا تراکنش‌های اصلی کسب‌وکار خود را شناسایی کنید که می‌خواهید آن‌ها را تحلیل کنید. هر رویداد معمولاً شامل مقادیر عددی است که می‌خواهید آن‌ها را جمع‌آوری یا محاسبه کنید (مانند فروش، تعداد، سود.)
2. شناسایی جداول ابعاد **:**برای هر جدول حقیقت، باید ابعاد مرتبط را شناسایی کنید. این ابعاد، ویژگی‌هایی هستند که به شما کمک می‌کنند تا داده‌های حقیقت را فیلتر، گروه‌بندی و دسته‌بندی کنید. به‌عنوان‌مثال، برای فروش، ابعاد می‌توانند شامل زمان، محصول، مشتری، فروشنده و مکان باشند.
3. ایجاد روابط **:**پس از بارگذاری جداول حقیقت و ابعاد در Power BI، باید روابط بین آن‌ها را در نمای مدل (Model View) ایجاد کنید. روابط معمولاً از نوع یک به چند **(One-to-Many)** هستند، به این معنی که یک ردیف در جدول بعد می‌تواند به چندین ردیف در جدول حقیقت مرتبط باشد (مثلاً یک محصول می‌تواند در چندین تراکنش فروش ظاهر شود.)

مثال عملی: مدل داده فروش

فرض کنید می‌خواهیم داده‌های فروش یک شرکت را تحلیل کنیم. داده‌های ما شامل موارد زیر است:

- جدول فروش **:(FactSales)** شامل تاریخ فروش، شناسه محصول، شناسه مشتری، شناسه فروشنده، مقدار فروش، تعداد واحد فروخته شده، تخفیف.
- جدول محصولات **:(DimProduct)** شامل شناسه محصول، نام محصول، دسته محصول، برند، رنگ.
- جدول مشتریان **:(DimCustomer)** شامل شناسه مشتری، نام مشتری، شهر، استان، جنسیت.
- جدول فروشندگان **:(DimSalesperson)** شامل شناسه فروشنده، نام فروشنده، منطقه فروش.
- جدول تاریخ **:(DimDate)** شامل تاریخ، سال، ماه، روز، فصل، روز هفته.

در این سناریو `FactSales`، جدول حقیقت ماست و `DimCustomer`، `DimProduct`، `DimSalesperson`و `DimDate` جداول ابعاد ما هستند. روابط بین این جداول به‌صورت زیر خواهد بود:

- (`DimProduct [ProductID]` یک) به `FactSales [ProductID]` چند(
- (`DimCustomer [CustomerID]` یک) به `FactSales [CustomerID]` چند(
- (`DimSalesperson [SalespersonID]` یک) به `FactSales [SalespersonID]` چند(
- (`DimDate [Date]` یک) به `FactSales [OrderDate]` چند(

نکات مهم در طراحی مدل داده**:**

- نام‌گذاری مناسب **:**از نام‌های واضح و توصیفی برای جداول و ستون‌ها استفاده کنید. این کار به‌سهولت استفاده و نگهداری مدل کمک می‌کند.
- حذف ستون‌های غیرضروری **:**فقط ستون‌هایی را در مدل خود نگه دارید که برای تحلیل یا مدل‌سازی ضروری هستند. حذف ستون‌های اضافی باعث کاهش حجم مدل و بهبود عملکرد می‌شود.
- بهینه‌سازی انواع داده **:**مطمئن شوید که انواع داده‌ها برای هر ستون به‌درستی تنظیم شده‌اند (مثلاً عدد برای مقادیر عددی، تاریخ برای تاریخ‌ها). این کار به فشرده‌سازی داده‌ها و عملکرد بهتر کمک می‌کند.
- ایجاد جدول تاریخ **(Date Table):** یک جدول تاریخ مناسب، برای انجام تحلیل‌های هوش زمانی (Time Intelligence)ضروری است. این جدول باید شامل تمام تاریخ‌های ممکن در محدوده داده‌های شما باشد و ستون‌هایی مانند سال، ماه، روز، فصل و روز هفته را داشته باشد. در فصل هوش زمانی به طور مفصل به نحوه ساخت یک جدول تاریخ شمسی خواهیم پرداخت.
- مدیریت روابط **:**روابط بین جداول را بادقت ایجاد کنید. روابط نادرست می‌توانند منجر به نتایج اشتباه در محاسبات DAX شوند. همیشه مطمئن شوید که جهت فیلتر (Filter Direction) و کاردینالیتی (Cardinality) رابطه صحیح است.

۲-۴: مدیریت روابط در Power BI

روابط (Relationships) در Power BI، نحوه ارتباط جداول با یکدیگر را تعریف می‌کنند و به Power BI اجازه می‌دهند تا فیلترها را از یک جدول به جدول دیگر منتقل کند. این امر برای انجام تحلیل‌های متقاطع (Cross-Table Analysis) ضروری است. مدیریت روابط در نمای مدل Power BI Desktop (Model View) انجام می‌شود.

انواع کاردینالیتی**:(Cardinality)**

کاردینالیتی یک رابطه، تعداد ردیف‌های مرتبط بین دو جدول را مشخص می‌کند. چهار نوع کاردینالیتی اصلی وجود دارد:

1. یک به چند **:(One-to-Many)** رایج‌ترین نوع رابطه در مدل‌های ستاره‌ای. یک ردیف در جدول

یک (جدول بعد) می‌تواند به چندین ردیف در جدول چند (جدول حقیقت) مرتبط باشد. مثال: یک محصول می‌تواند در چندین تراکنش فروش ظاهر شود. این نوع رابطه با نماد 1 در یک طرف و * در طرف دیگر نمایش داده می‌شود. 2. چند به یک **(Many-to-One):** برعکس رابطه یک به چند. یک ردیف در جدول چند (جدول حقیقت) می‌تواند به یک ردیف در جدول یک (جدول بعد) مرتبط باشد. این رابطه نیز با نماد * در یک طرف و 1 در طرف دیگر نمایش داده می‌شود و در عمل همانند یک به چند است، فقط جهت رابطه متفاوت است. 3. یک به یک **(One-to-One):** هر ردیف در یک جدول دقیقاً به یک ردیف در جدول دیگر مرتبط است. این نوع رابطه کمتر در مدل‌های ستاره‌ای استفاده می‌شود و معمولاً نشان‌دهنده این است که دو جدول می‌توانند در یک جدول ادغام شوند. 4. چند به چند **(Many-to-Many):** هر ردیف در یک جدول می‌تواند به چندین ردیف در جدول دیگر مرتبط باشد و بالعکس. این نوع رابطه پیچیده‌تر است و می‌تواند منجر به ابهام در محاسبات شود. در Power BI، روابط چند به چند اغلب با استفاده از یک جدول واسط (Bridge Table) یا جدول اتصال (Junction Table) مدیریت می‌شوند تا به روابط یک به چند تبدیل شوند. ما در فصول پیشرفته‌تر به این موضوع خواهیم پرداخت.

جهت فیلتر **(Filter Direction):**

جهت فیلتر مشخص می‌کند که فیلترها چگونه از یک جدول به جدول دیگر در مدل داده اعمال می‌شوند. دو نوع جهت فیلتر اصلی وجود دارد:

1. تک‌جهته **(Single):** فیلترها فقط در یک جهت اعمال می‌شوند. این رایج‌ترین جهت فیلتر در روابط یک به چند است، جایی که فیلتر از جدول یک (بعد) به جدول چند (حقیقت) اعمال می‌شود. به‌عنوان‌مثال، فیلترکردن بر اساس نام محصول در جدول `DimProduct`، ردیف‌های مربوطه را در جدول `FactSales` فیلتر می‌کند.
2. دوجهته **(Both):** فیلترها در هر دو جهت اعمال می‌شوند. این نوع جهت فیلتر باید با احتیاط استفاده شود، زیرا می‌تواند منجر به ابهام در مدل و نتایج غیرمنتظره شود. استفاده از روابط دوجهته می‌تواند در سناریوهای خاصی مفید باشد، اما به طور کلی توصیه می‌شود تا حد امکان از آن‌ها اجتناب شود و در صورت لزوم، با استفاده از توابع DAX مانند `CROSSFILTER`، کنترل دقیق‌تری بر روی فیلترها اعمال شود.

نحوه ایجاد و ویرایش روابط**:**

در نمای مدل Power BI Desktop، (Model View) می‌توانید روابط را به‌صورت بصری ایجاد و مدیریت کنید:

- ایجاد رابطه: می‌توانید یک ستون (کلید اصلی) را از یک جدول بکشید و بر روی ستون متناظر (کلید خارجی) در جدول دیگر رها کنید. **Power BI** به طور خودکار یک رابطه ایجاد می‌کند و کاردینالیتی و جهت فیلتر را حدس می‌زند.
- ویرایش رابطه **:**بر روی خط رابطه بین دو جدول دوبار کلیک کنید تا پنجره `Edit relationship` باز شود. در این پنجره می‌توانید ستون‌های مرتبط، کاردینالیتی و جهت فیلتر را تغییر دهید. همچنین می‌توانید گزینه `Make`

this relationship active را فعال یا غیرفعال کنید. (یک رابطه غیرفعال تنها زمانی استفاده می‌شود که به طور صریح در فرمول DAX با تابع USERELATIONSHIP فراخوانی شود.)

۲-۵: جدول تاریخ (Date Table) و اهمیت آن در تحلیل‌های زمانی

یکی از مهم‌ترین جداول ابعاد در هر مدل داده‌ای، جدول تاریخ **(Date Table)** است. این جدول، ستون فقرات تمام تحلیل‌های هوش زمانی (Time Intelligence) در Power BI و DAX است. بدون یک جدول تاریخ مناسب، انجام محاسباتی مانند فروش سال‌به‌سال، رشد ماهانه، یا مقایسه دوره‌های مختلف زمانی، بسیار دشوار یا غیرممکن خواهد بود.

چرا به یک جدول تاریخ جداگانه نیاز داریم؟

- کامل بودن **:**جدول تاریخ شامل تمام تاریخ‌های ممکن در محدوده زمانی داده‌های شماست، حتی اگر در برخی از روزها یا ماه‌ها هیچ تراکنشی ثبت نشده باشد. این امر به شما امکان می‌دهد تا تحلیل‌های زمانی را بدون شکاف در داده‌ها انجام دهید.
- ویژگی‌های زمانی غنی **:**جدول تاریخ شامل ستون‌های اضافی مانند سال، ماه، نام ماه، روز هفته، شماره هفته، فصل، و حتی پرچم‌هایی برای تعطیلات یا دوره‌های مالی خاص است. این ستون‌ها به شما امکان می‌دهند تا داده‌ها را بر اساس ابعاد زمانی مختلف فیلتر و گروه‌بندی کنید.
- پشتیبانی از توابع هوش زمانی **DAX:** بسیاری از توابع هوش زمانی) DAX مانند ,TOTALYTD SAMEPERIODLASTYEAR, DATEADD)برای کار صحیح، نیاز به یک جدول تاریخ مشخص و علامت‌گذاری شده دارند.
- عملکرد بهتر **:**استفاده از یک جدول تاریخ بهینه، به موتور VertiPaq کمک می‌کند تا محاسبات زمانی را با سرعت و کارایی بیشتری انجام دهد.

ساخت یک جدول تاریخ در **Power BI:**

چندین روش برای ساخت یک جدول تاریخ در Power BI وجود دارد:

1. استفاده از قابلیت خودکار Power BI **Power BI (Auto Date/Time):** به طور پیش‌فرض قابلیت Auto Date/Time را فعال دارد. این قابلیت به طور خودکار برای هر ستون تاریخ در مدل شما، یک سلسله‌مراتب تاریخ (Date Hierarchy)ایجاد می‌کند. این روش برای تحلیل‌های ساده مناسب است، اما برای سناریوهای پیچیده‌تر هوش زمانی و به خصوص برای تقویم شمسی، کافی نیست و توصیه نمی‌شود.

2. استفاده از **Power Query:** می‌توانید یک جدول تاریخ را در Power Query ایجاد کنید. این روش انعطاف‌پذیری خوبی دارد و به شما امکان می‌دهد تا ستون‌های سفارشی زیادی را اضافه کنید. این روش برای تقویم شمسی نیز قابل استفاده است.
3. استفاده از **DAX (**توابع **CALENDAR** و **:(CALENDARAUTO** این روش رایج‌ترین و توصیه‌شده‌ترین روش برای ایجاد یک جدول تاریخ استاندارد در Power BI است. توابع `CALENDAR` و `CALENDARAUTO` به شما امکان می‌دهند تا یک جدول با یک ستون تاریخ ایجاد کنید و سپس با استفاده از توابع DAX دیگر، ستون‌های اضافی (مانند سال، ماه، روز) را به آن اضافه کنید.

مثال: ساخت جدول تاریخ با **DAX (**تقویم میلادی**:(**

برای ساخت یک جدول تاریخ با DAX، به تب `Modeling` در Power BI Desktop بروید و بر روی `New Table` کلیک کنید. سپس فرمول زیر را وارد کنید:

```
DateTable =

VAR MinDate = MIN(FactSales [OrderDate])

VAR MaxDate = MAX(FactSales [OrderDate])

RETURN

ADDCOLUMNS(

CALENDAR(MinDate, MaxDate),



"Year", YEAR([Date]),

"MonthNumber", MONTH([Date]),

"MonthName", FORMAT([Date] , "MMMM") ,

"Quarter", FORMAT([Date] , "Q") ,

"Day", DAY([Date]),
```

```
"DayOfWeek", WEEKDAY([Date] , 2),

"DayName", FORMAT([Date] , "dddd") ,

"YearMonth", FORMAT([Date] , "YYYY-MM") ,

"YearQuarter", FORMAT([Date] , "YYYY-Q") ,

"IsWeekend", IF(WEEKDAY([Date] , 2) IN {6, 7}, TRUE() , FALSE())

)
```

توضیح فرمول:

- `MinDate`و :`MaxDate` این متغیرها حداقل و حداکثر تاریخ موجود در ستون `OrderDate` از جدول `FactSales`را پیدا می‌کنند. این کار تضمین می‌کند که جدول تاریخ شما فقط شامل محدوده تاریخ‌های واقعی موجود در داده‌های شما باشد.
- :`CALENDAR(MinDate, MaxDate)`این تابع یک جدول تک ستونی به نام `Date` ایجاد می‌کند که شامل تمام تاریخ‌ها از `MinDate` تا `MaxDate` است.
- :`ADDCOLUMNS`این تابع به جدول `Date` ستون‌های جدیدی را اضافه می‌کند. هر ستون با یک نام (مثلاً `"Year"`) و یک فرمول DAX (مثلا `YEAR([Date])`) تعریف می‌شود.
- توابعی مانند `FORMAT` ,`WEEKDAY` ,`DAY` ,`MONTH` ,`YEAR` برای استخراج اطلاعات مختلف از ستون `Date` استفاده می‌شوند.
- :`WEEKDAY([Date] , 2)`عدد `2` در تابع `WEEKDAY` به این معنی است که هفته از دوشنبه شروع می‌شود (دوشنبه = ۱، یکشنبه = ۷).
- :`FORMAT([Date] , "MMMM")`نام کامل ماه را برمی‌گرداند (مثلا."January")
- :`FORMAT([Date] , "dddd")`نام کامل روز هفته را برمی‌گرداند (مثلا."Monday")

علامت‌گذاری جدول تاریخ:**(Mark as Date Table)**

پس از ایجاد جدول تاریخ، باید آن را در Power BI Desktop به‌عنوان یک جدول تاریخ علامت‌گذاری کنید. این کار به Power BIکمک می‌کند تا توابع هوش زمانی DAX را به‌درستی اعمال کند.

1. در نمای مدل ،(Model View) جدول `DateTable` را انتخاب کنید.

2. در پنل (Properties سمت راست)، بخش Table tools، بر روی Mark as date table کلیک کنید.
3. در پنجره باز شده، ستون Date را به‌عنوان ستون تاریخ انتخاب کنید و OK را بزنید.

اتصال جدول تاریخ به جدول حقیقت:

در نهایت، باید جدول DateTable را به جدول حقیقت خود (مثلا FactSales) متصل کنید. یک رابطه یک به چند از DateTable [Date] به FactSales [OrderDate] ایجاد کنید. این رابطه باید تک‌جهته (Single) باشد و فیلتر از جدول تاریخ به جدول حقیقت اعمال شود.

۶-۲: بهترین روش‌ها برای مدل‌سازی داده

طراحی یک مدل داده قوی و کارآمد، هنری است که با تجربه و رعایت بهترین روش‌ها بهبود می‌یابد. در اینجا به برخی از مهم‌ترین بهترین روش‌ها برای مدل‌سازی داده در Power BI اشاره می‌کنیم:

1. همیشه از **Star Schema** استفاده کنید: این مهم‌ترین قانون است. مدل ستاره‌ای ساده، کارآمد و قابل نگهداری است و بهترین عملکرد را برای تحلیل‌های DAX فراهم می‌کند. از مدل‌های Snowflake که جداول ابعاد خود نیز دارای ابعاد هستند) تا حد امکان اجتناب کنید، مگر اینکه دلیل بسیار قوی برای استفاده از آن‌ها داشته باشید.

2. جداول ابعاد را از جداول حقیقت جدا کنید: هر جدول باید یک هدف مشخص داشته باشد: یا یک جدول حقیقت (برای مقادیر قابل‌اندازه‌گیری) یا یک جدول بعد (برای توصیف‌کننده‌ها). از ترکیب این دو نوع جدول در یک جدول خودداری کنید.

3. **جداول ابعاد را تا حد امکان

کامل کنید: ** جداول ابعاد باید شامل تمام ویژگی‌های توصیفی مرتبط باشند که ممکن است برای فیلترکردن، گروه‌بندی یا تحلیل داده‌ها نیاز داشته باشید. این کار از نیاز به ایجاد ستون‌های محاسباتی پیچیده در DAX جلوگیری می‌کند.

4. از ستون‌های کلید جایگزین **(Surrogate Keys)** استفاده کنید: به‌جای استفاده از کلیدهای کسب‌وکار (Business Keys)که ممکن است تغییر کنند یا معنی‌دار باشند، از کلیدهای جایگزین عددی (معمولاً اعداد صحیح متوالی) در جداول ابعاد استفاده کنید. این کلیدها پایدارتر هستند و عملکرد بهتری در روابط دارند.

5. از ستون‌های محاسباتی **(Calculated Columns)** در **DAX** با احتیاط استفاده کنید: ستون‌های محاسباتی در DAXفضای حافظه را اشغال می‌کنند و در زمان تازه‌سازی داده‌ها (Refresh) محاسبه می‌شوند. اگر می‌توانید یک

ستون را در Power Query ایجاد کنید، بهتر است این کار را در Power Query انجام دهید. ستون‌های محاسباتی DAXرا برای مواردی نگه دارید که نیاز به دسترسی به زمینه فیلتر (Filter Context) دارند یا برای محاسبات پیچیده‌ای که در Power Query قابل انجام نیستند.

6. نام‌گذاری استاندارد و سازگار **:**از یک استاندارد نام‌گذاری ثابت برای جداول، ستون‌ها و Measures استفاده کنید. این کار به خوانایی و قابلیت نگهداری مدل کمک می‌کند. به‌عنوان‌مثال، می‌توانید جداول ابعاد را با پیشوند `Dim` و جداول حقیقت را با پیشوند `Fact` نام‌گذاری کنید.

7. پنهان کردن ستون‌های غیرضروری **:**ستون‌هایی که برای کاربران نهایی معنی‌دار نیستند یا فقط برای ایجاد روابط استفاده می‌شوند (مانند ستون‌های کلید خارجی در جداول حقیقت)، را پنهان کنید. این کار رابط کاربری را برای کاربران ساده‌تر می‌کند و از سردرگمی آن‌ها جلوگیری می‌کند.

8. استفاده از سلسله‌مراتب **(Hierarchies):** برای ستون‌هایی که دارای یک رابطه سلسله‌مراتبی هستند (مانند سال -> ماه -> روز)، سلسله‌مراتب ایجاد کنید. این کار به کاربران امکان می‌دهد تا به‌راحتی در گزارش‌ها به سطوح مختلف جزئیات (Drill Down)بروند.

9. بهینه‌سازی انواع داده **(Data Types):** مطمئن شوید که انواع داده‌ها برای هر ستون به‌درستی تنظیم شده‌اند. به‌عنوان‌مثال، از نوع `Whole Number` برای اعداد صحیح و `Decimal Number` برای اعداد اعشاری استفاده کنید. از نوع `Text` برای ستون‌هایی که فقط حاوی متن هستند استفاده کنید و از آن برای اعداد یا تاریخ‌ها خودداری کنید.

10. حذف روابط غیرفعال **(Inactive Relationships)** در صورت عدم نیاز **:**اگر یک رابطه غیرفعال دارید که هرگز از آن استفاده نمی‌کنید (حتی با `USERELATIONSHIP` در DAX)، آن را حذف کنید. روابط اضافی می‌توانند پیچیدگی مدل را افزایش دهند.

11. مستندسازی مدل **:**مدل داده خود را مستند کنید. توضیحات (Descriptions) را برای جداول و ستون‌ها اضافه کنید تا هدف آن‌ها را روشن کنید. این کار به سایر توسعه‌دهندگان و کاربران کمک می‌کند تا مدل شما را درک کنند.

۲-۷:سناریوهای پیشرفته مدل‌سازی داده

درحالی‌که مدل ستاره‌ای پایه و اساس اکثر مدل‌های داده‌ای موفق است، برخی سناریوهای پیچیده‌تر ممکن است نیاز به رویکردهای پیشرفته‌تری در مدل‌سازی داشته باشند. در این بخش به چند سناریوی رایج و نحوه مدیریت آن‌ها در Power BI می‌پردازیم.

۲-۷-۱: جداول چند به چند(Many-to-Many Relationships)

همانطور که قبلاً اشاره شد، روابط چند به چند می‌توانند در Power BI چالش‌برانگیز باشند و منجر به نتایج نادرست یا ابهام در محاسبات شوند. یک مثال رایج، رابطه بین دانش‌آموزان و کلاس‌ها است: یک دانش‌آموز می‌تواند در چندین کلاس شرکت کند و یک کلاس می‌تواند چندین دانش‌آموز داشته باشد. برای مدیریت این نوع روابط، بهترین روش استفاده از یک جدول واسط **(Bridge Table)** است.

مثال: دانش‌آموزان و کلاس‌ها

فرض کنید دو جدول داریم:

- `DimStudent`: شامل `StudentID`, `StudentName`
- `DimClass`: شامل `ClassID`, `ClassName`

و یک جدول حقیقت `FactEnrollment` که ثبت‌نام دانش‌آموزان در کلاس‌ها را نشان می‌دهد و شامل `StudentID`, `ClassID`, `EnrollmentDate` است. در این حالت `FactEnrollment`، به‌عنوان جدول واسط عمل می‌کند.

نحوه مدل‌سازی:

1. `DimStudent` به `FactEnrollment` با رابطه یک به چند (1 به *) بر روی `StudentID`.
2. `DimClass` به `FactEnrollment` با رابطه یک به چند (1 به *) بر روی `ClassID`.

با این ساختار، فیلترکردن از `DimStudent` به `FactEnrollment` و سپس به `DimClass` و بالعکس) به‌درستی کار خواهد کرد. این روش به Power BI اجازه می‌دهد تا فیلترها را به‌درستی از طریق جدول واسط منتقل کند و از ابهام جلوگیری کند.

۲-۷-۲: جداول نقش‌آفرین(Role-Playing Dimensions)

گاهی اوقات، یک جدول بعد می‌تواند چندین نقش را در یک مدل داده ایفا کند. به‌عنوان‌مثال، در یک جدول فروش، ممکن است چندین ستون تاریخ داشته باشیم (`OrderDate`: تاریخ سفارش، `ShipDate`: تاریخ ارسال) و `DeliveryDate`: تاریخ تحویل). هر یک از این ستون‌ها نیاز به فیلتر شدن توسط یک جدول تاریخ دارند.

برای مدیریت این سناریو، می‌توانید چندین کپی از جدول تاریخ خود را در مدل داده ایجاد کنید و هر کپی را به‌عنوان یک جدول نقش‌آفرین به یکی از ستون‌های تاریخ در جدول حقیقت متصل کنید. سپس، روابط بین این جداول تاریخ نقش‌آفرین و جدول حقیقت را غیرفعال (Inactive) کنید.

نحوه مدل‌سازی:

1. یک جدول تاریخ اصلی (`DimDate`) ایجاد کنید.
2. دو کپی از `DimDate` ایجاد کنید و نام آن‌ها را به `DimShipDate` و `DimDeliveryDate` تغییر دهید.
3. یک رابطه یک به چند از `DimDate [Date]` به `FactSales [OrderDate]` ایجاد کنید و آن را فعال (Active) نگه دارید.
4. یک رابطه یک به چند از `DimShipDate [Date]` به `FactSales [ShipDate]` ایجاد کنید و آن را غیرفعال (Inactive) کنید.
5. یک رابطه یک به چند از `DimDeliveryDate [Date]` به `FactSales [DeliveryDate]` ایجاد کنید و آن را غیرفعال (Inactive) کنید.

هنگامی که نیاز به تحلیل بر اساس تاریخ ارسال یا تحویل دارید، می‌توانید از تابع `USERELATIONSHIP` در DAX برای فعال کردن موقت رابطه غیرفعال استفاده کنید. این موضوع در فصل توابع رابطه DAX به طور مفصل توضیح داده خواهد شد.

۳-۷-۲جداول پارامتر(Parameter Tables):

جداول پارامتر (که گاهی اوقات جداول What-If نیز نامیده می‌شوند) به کاربران امکان می‌دهند تا مقادیر ورودی را برای سناریوهای What-Ifدر گزارش‌های خود تغییر دهند. این جداول معمولاً شامل یک ستون از مقادیر عددی هستند که می‌توانند توسط یک اسلایسر (Slicer) در گزارش انتخاب شوند. سپس، این مقدار انتخاب شده در فرمول‌های DAX برای انجام محاسبات پویا استفاده می‌شود.

مثال: تحلیل سناریوی تخفیف

فرض کنید می‌خواهید تأثیر درصدهای مختلف تخفیف را بر فروش بررسی کنید. می‌توانید یک جدول پارامتر برای تخفیف ایجاد کنید:

```
Discount Percentage = GENERATESERIES(0, 0.2, 0.01)
```

این فرمول یک جدول تک ستونی به نام `Value` ایجاد می‌کند که شامل مقادیر از ۰ تا ۰.۲ (۲۰٪) با گام‌های ۰.۰۱ است. سپس می‌توانید یک Measure DAX ایجاد کنید که از مقدار انتخاب شده در این جدول پارامتر استفاده کند:

```
Total Sales with Discount =

VAR SelectedDiscount = SELECTEDVALUE('Discount Percentage' [Value])

RETURN

[Total Sales] * (1 - SelectedDiscount)
```

این Measure، کل فروش را بر اساس درصد تخفیف انتخاب شده توسط کاربر محاسبه می‌کند. جداول پارامتر ابزار قدرتمندی برای ایجاد گزارش‌های تعاملی و سناریوهای تحلیلی پویا هستند.

۲-۸: خلاصه فصل ۲

در این فصل، به اهمیت مدل داده در Power BI پرداختیم و مفهوم مدل ستاره‌ای (Star Schema) را به‌عنوان بهترین روش برای طراحی مدل داده معرفی کردیم. آموختیم که چگونه جداول حقیقت و ابعاد را شناسایی کنیم و روابط بین آن‌ها را مدیریت کنیم. همچنین، اهمیت یک جدول تاریخ مناسب برای تحلیل‌های هوش زمانی را درک کردیم و نحوه ساخت آن را با DAX بررسی کردیم. در نهایت، به برخی از بهترین روش‌ها و سناریوهای پیشرفته مدل‌سازی داده پرداختیم.

یک مدل داده قوی و بهینه، پایه و اساس تمام تحلیل‌های DAX شما خواهد بود. با رعایت اصول مطرح شده در این فصل، می‌توانید اطمینان حاصل کنید که مدل شما عملکرد بالا، صحت محاسبات و سهولت استفاده را فراهم می‌کند. در فصل بعدی، وارد دنیای هیجان‌انگیز زبان DAX خواهیم شد و اولین فرمول‌های خود را خواهیم نوشت.

فصل ۳: زبان - DAX شروعی قدرتمند

پس از اینکه در فصل قبل با اهمیت مدل داده و نحوه طراحی آن آشنا شدیم، اکنون زمان آن رسیده که وارد قلب Power BI، یعنی زبان **DAX (Data Analysis Expressions)** شویم .زبانی است که به شما امکان می‌دهد تا محاسبات قدرتمند و پیچیده‌ای را بر روی داده‌های مدل داده خود انجام دهید. این زبان، کلید استخراج بینش‌های عمیق از داده‌ها و ایجاد گزارش‌های پویا و تعاملی است.

۳-۱: DAX چیست و چرا به آن نیاز داریم؟

DAXیک‌زبان فرمول‌نویسی تابعی (Functional Language) است که برای کار با مدل‌های داده‌ای جدولی (Tabular Models)در SQL Server Analysis Services (SSAS)، Power BI و Excel Power Pivot طراحی شده

است. اگر با فرمول‌های اکسل آشنایی دارید، یادگیری DAX برای شما آسان‌تر خواهد بود، اما تفاوت‌های کلیدی و مهمی بین این دو وجود دارد که باید آن‌ها را درک کنید.

چرا به **DAX** نیاز داریم؟

Power BIبه تنهایی می‌تواند داده‌ها را بارگذاری و بصری‌سازی کند، اما برای انجام محاسبات پیچیده و سفارشی، به DAX نیاز داریم DAX .به شما امکان می‌دهد تا:

- شاخص‌های کلیدی عملکرد **(KPIs)** را محاسبه کنید **:**مانند مجموع فروش، میانگین سود، درصد رشد، تعداد مشتریان فعال.
- ستون‌های جدیدی ایجاد کنید **:**که بر اساس منطق خاصی از ستون‌های موجود محاسبه می‌شوند (مانند سود ناخالص، سن مشتری.(
- جداول جدیدی ایجاد کنید **:**که از داده‌های موجود در مدل شما مشتق شده‌اند (مانند جدول تاریخ سفارشی.(
- تحلیل‌های زمانی انجام دهید **:**مانند مقایسه فروش سال‌به‌سال، ماه‌به‌ماه، یا محاسبه فروش تا تاریخ امروز.(YTD)
- سناریوهای **What-If** را پیاده‌سازی کنید **:**با استفاده از پارامترها و محاسبات پویا.
- مدل داده خود را غنی‌تر کنید **:**با افزودن منطق کسب‌وکار که در داده‌های خام وجود ندارد.

DAXبه شما قدرت می‌دهد تا از داده‌های خود فراتر رفته و بینش‌های واقعی را استخراج کنید. این زبان، ابزاری است که تحلیلگران داده را قادر می‌سازد تا به سؤالات پیچیده کسب‌وکار پاسخ دهند و داستان داده‌ها را به شکلی معنی‌دار روایت کنند.

۲-۳مفاهیم کلیدی در :DAX ستون‌های محاسباتی Measures ،و جداول محاسباتی

در ،DAX سه نوع اصلی از محاسبات وجود دارد که هر یک کاربرد خاص خود را دارند:

1. ستون‌های محاسباتی**:(Calculated Columns)**

- تعریف **:**ستون‌های جدیدی هستند که با استفاده از فرمول‌های DAX ایجاد می‌شوند و به مدل داده اضافه می‌شوند. مقدار هر ردیف در این ستون، بر اساس فرمول و مقادیر موجود در همان ردیف یا ردیف‌های مرتبط محاسبه می‌شود.
- زمان محاسبه **:**در زمان تازه‌سازی داده‌ها (Data Refresh) محاسبه می‌شوند و مقادیر آن‌ها در مدل ذخیره می‌شوند. این بدان معناست که فضای حافظه را اشغال می‌کنند.
- کاربرد **:**برای مواردی که نیاز به یک ستون ثابت دارید که بتوانید آن را در بصری‌سازی‌ها، فیلترها یا روابط استفاده کنید. مثال.`Gross Profit = [Sales Amount] - [Cost Amount]` :

- محدودیت **:**نمی‌توانند به زمینه فیلتر (Filter Context) پاسخ دهند. یعنی مقدار آن‌ها در هر ردیف ثابت است و با فیلترکردن گزارش تغییر نمی‌کند.

2. **Measures (** معیارها**:(**

- تعریف: محاسباتی هستند که نتایج آن‌ها بر اساس زمینه فیلتر (**Filter Context**) بصری‌سازی یا گزارش تغییر می‌کند. **Measures** در مدل داده ذخیره نمی‌شوند، بلکه در زمان پرس‌وجو (**Query Time**) و بر اساس فیلترهای اعمال شده توسط کاربر محاسبه می‌شوند.
- زمان محاسبه **:**در زمان اجرای گزارش و تعامل کاربر با آن محاسبه می‌شوند.
- کاربرد **:**برای محاسبه شاخص‌های کلیدی عملکرد (KPIs) و مقادیر تجمیعی (Aggregations) که نیاز به پاسخگویی به فیلترها دارند. مثال : `Total Sales = SUM(FactSales [SalesAmount])`.
- مزیت **:**بسیار انعطاف‌پذیر هستند و به زمینه فیلتر پاسخ می‌دهند، فضای حافظه را اشغال نمی‌کنند و برای تحلیل‌های پویا ایده‌آل هستند.

3. جداول محاسباتی**:(Calculated Tables)**

- تعریف **:**جداول جدیدی هستند که با استفاده از فرمول‌های DAX ایجاد می‌شوند و به مدل داده اضافه می‌شوند. این جداول می‌توانند از جداول موجود یا از ترکیب داده‌ها ایجاد شوند.
- زمان محاسبه **:**در زمان تازه‌سازی داده‌ها محاسبه می‌شوند و مقادیر آن‌ها در مدل ذخیره می‌شوند.
- کاربرد **:**برای ایجاد جداول کمکی (مانند جدول تاریخ سفارشی)، جداول خلاصه (Summary Tables) یا جداول واسط (Bridge Tables) در سناریوهای پیچیده مدل‌سازی.
- مثال: `DateTable = CALENDARAUTO()` :

چه زمانی از کدام استفاده کنیم؟

- ستون محاسباتی **:**زمانی که نیاز به یک ستون ثابت دارید که در هر ردیف مقدار مشخصی دارد و می‌خواهید از آن در فیلترها، گروه‌بندی‌ها یا روابط استفاده کنید. اگر محاسبه می‌تواند در Power Query انجام شود، بهتر است آن را در Power Queryانجام دهید.
- **Measure:**زمانی که نیاز به یک مقدار تجمیعی دارید که باید به فیلترهای اعمال شده در گزارش پاسخ دهد **. Measures**برای اکثر محاسبات تحلیلی شما، به خصوص**KPIs** ، بهترین گزینه هستند **.**
- جدول محاسباتی **:**زمانی که نیاز به ایجاد یک جدول جدید در مدل داده خود دارید که از داده‌های موجود مشتق شده است.

به طور کلی، در Power BI، **Measures** ستون فقرات تحلیل‌های شما هستند و باید اولویت اصلی شما در فرمول‌نویسی DAXباشند. ستون‌های محاسباتی و جداول محاسباتی ابزارهای کمکی هستند که در سناریوهای خاصی مورد استفاده قرار می‌گیرند.

۳-۳سینتکس پایه DAX و عملگرها:

سینتکس DAX شبیه به فرمول‌های اکسل است، اما با تفاوت‌های مهمی در نحوه ارجاع به ستون‌ها و جداول و نحوه عملکرد توابع. در اینجا به برخی از اصول پایه سینتکس DAX و عملگرهای رایج می‌پردازیم.

نام‌گذاری و ارجاع:

- ارجاع به ستون :برای ارجاع به یک ستون، نام جدول و سپس نام ستون را در براکت مربع `[ ]` قرار می‌دهید. مثال : `FactSales [SalesAmount]`.
- ارجاع به **Measure:** برای ارجاع به یک Measure، فقط نام Measure را در براکت مربع `[ ]` قرار می‌دهید. مثال: `[Total Sales]`.
- نام جداول و ستون‌ها :اگر نام جدول یا ستون حاوی فاصله یا کاراکترهای خاص باشد، باید آن را در علامت نقل قول تکی `' '` قرار دهید. مثال: `'Sales Data' [Product Name]`.

عملگرها:

DAXاز عملگرهای ریاضی، مقایسه‌ای، منطقی و متنی مشابه اکسل استفاده می‌کند:

- ریاضی) `+` :جمع) `-` ،(تفریق) `*` ،(ضرب) `/` ،(تقسیم) `^` ،(توان).
- مقایسه‌ای) `=` :مساوی) `>` ،(کوچک‌تر) `<` ،(بزرگ‌تر) `=>` ،(کوچک‌تر یا مساوی) `>=` ،(بزرگ‌تر یا مساوی `<>` ،( )نامساوی).
- منطقی (AND `&&` :منطقی (OR `||` ،(منطقی).
- متنی) `&` :الحاق رشته).

کامنت‌گذاری:

برای افزودن کامنت به فرمول‌های DAX، می‌توانید از دو خط تیره `--` برای کامنت تک خطی یا `/* ... */` برای کامنت چند خطی استفاده کنید.

--این یک Measure برای محاسبه مجموع فروش است

```
Total Sales =

-- SUM(FactSales [SalesAmount])جمع ستونSalesAmount

/*

این یک کامنت چند خطی است.

این Measure مجموع فروش را محاسبه می‌کند

و برای تحلیل‌های مختلف استفاده می‌شود.

*/

Total Quantity = SUM(FactSales [Quantity])
```

۳-۴:اولین توابعDAX: SUM, AVERAGE, COUNT, MIN, MAX

برای شروع کار با DAX، با ساده‌ترین و پرکاربردترین توابع تجمیعی (Aggregation Functions) آشنا می‌شویم. این توابع مقادیر را از یک ستون جمع‌آوری کرده و یک نتیجه واحد را برمی‌گردانند.

۱. SUM ( ):جمع

تابع `SUM` مجموع تمام اعداد در یک ستون را برمی‌گرداند.

```
Total Sales = SUM(FactSales [SalesAmount])
```

توضیح :این Measure مجموع مقادیر ستون `SalesAmount` از جدول `FactSales` را محاسبه می‌کند. این رایج‌ترین Measureبرای محاسبه کل فروش است.

۲. AVERAGE ( ):میانگین

تابع `AVERAGE` میانگین (حساب) تمام اعداد در یک ستون را برمی‌گرداند.

```
Average Order Value = AVERAGE(FactSales [SalesAmount])
```

توضیح: این Measure میانگین مقدار هر تراکنش فروش را محاسبه می‌کند.

۳. COUNT ( ): شمارش

تابع `COUNT` تعداد ردیف‌های حاوی یک مقدار غیرخالی (Non-Blank) در یک ستون را می‌شمارد.

```
Number of Transactions = COUNT(FactSales [TransactionID])
```

توضیح: این Measure تعداد تراکنش‌های فروش را با شمارش ردیف‌های غیرخالی در ستون `TransactionID` محاسبه می‌کند.

۴. MIN ( ): حداقل

تابع `MIN` کوچک‌ترین مقدار در یک ستون را برمی‌گرداند.

```
Min Sales Amount = MIN(FactSales [SalesAmount])
```

توضیح: این Measure حداقل مقدار فروش را در تمام تراکنش‌ها پیدا می‌کند.

۵. MAX ( ): حداکثر

تابع `MAX` بزرگ‌ترین مقدار در یک ستون را برمی‌گرداند.

```
Max Sales Amount = MAX(FactSales [SalesAmount])
```

توضیح: این Measure حداکثر مقدار فروش را در تمام تراکنش‌ها پیدا می‌کند.

نحوه ایجاد **Measure** در **Power BI Desktop**:

1. در پنل `Fields`، بر روی جدولی که می‌خواهید Measure را در آن ایجاد کنید (معمولاً جدول حقیقت یا یک جدول Measures جداگانه) کلیک راست کنید.
2. گزینه `New measure` را انتخاب کنید.
3. در نوار فرمول (Formula Bar) که در بالای صفحه ظاهر می‌شود، فرمول DAX خود را بنویسید.
4. پس از اتمام، کلید `Enter` را فشار دهید یا بر روی آیکون تیک در نوار فرمول کلیک کنید.

Measureجدید در پنل `Fields` با یک آیکون ماشین حساب در کنار آن ظاهر می‌شود. اکنون می‌توانید آن را به بصری‌سازی‌های خود اضافه کنید.

۳-۵:توابع DISTINCTCOUNT وCOUNTROWS

علاوه بر `COUNT`، دو تابع شمارش دیگر نیز در DAX بسیار پرکاربرد هستند:

۱. DISTINCTCOUNT ( ):شمارش مقادیر متمایز

تابع `DISTINCTCOUNT` تعداد مقادیر متمایز (Unique) در یک ستون را می‌شمارد. این تابع برای شمارش موجودیت‌های منحصربه‌فرد مانند مشتریان، محصولات یا سفارشات بسیار مفید است.

```
Number of Unique Customers = DISTINCTCOUNT(FactSales [CustomerID])
```

توضیح **:**این Measure تعداد مشتریان منحصربه‌فردی را که در جدول `FactSales` تراکنش داشته‌اند، محاسبه می‌کند.

۲. COUNTROWS ( ):شمارش ردیف‌ها

تابع `COUNTROWS` تعداد ردیف‌ها در یک جدول را می‌شمارد. این تابع اغلب برای شمارش تعداد تراکنش‌ها یا رویدادها در یک جدول حقیقت استفاده می‌شود.

```
Total Rows in Sales Table = COUNTROWS(FactSales)
```

توضیح **:**این Measure تعداد کل ردیف‌ها در جدول `FactSales` را برمی‌گرداند. این می‌تواند برای بررسی حجم داده‌ها یا شمارش کلی تراکنش‌ها مفید باشد.

تفاوت بین **COUNT** و**:COUNTROWS**

- `COUNT(Column)`:تعداد ردیف‌های غیرخالی در یک ستون خاص را می‌شمارد.
- `COUNTROWS(Table)`:تعداد کل ردیف‌ها در یک جدول را می‌شمارد، صرف نظر از اینکه ستون‌های آن خالی باشند یا نه.

در بیشتر موارد، برای شمارش تراکنش‌ها `COUNTROWS(FactTable)`، ترجیح داده می‌شود، زیرا کارایی بهتری دارد و واضح‌تر است که هدف شمارش ردیف‌های جدول است.

۳-۶:توابع CALCULATE و ALL: قلبDAX

توابع `CALCULATE` و `ALL` از قدرتمندترین و پرکاربردترین توابع در DAX هستند و درک آن‌ها برای تسلط بر DAX ضروری است. این توابع به شما امکان می‌دهند تا زمینه فیلتر (Filter Context) را تغییر دهید و محاسبات پیچیده‌ای را انجام دهید.

۳-۶-۱:زمینه فیلتر(Filter Context)

قبل از پرداختن به `CALCULATE`، باید مفهوم زمینه فیلتر **(Filter Context)** را درک کنید. در Power BI، هر بصری‌سازی (مانند یک نمودار ستونی یا یک جدول) دارای یک زمینه فیلتر است. این زمینه فیلتر توسط موارد زیر تعیین می‌شود:

- فیلترهای اعمال شده در بصری‌سازی **:**مانند فیلترکردن بر روی یک سال خاص در یک نمودار.
- فیلترهای اعمال شده در صفحه **:**مانند فیلترکردن یک صفحه گزارش بر اساس یک منطقه خاص.
- فیلترهای اعمال شده در گزارش **:**فیلترهایی که بر روی کل گزارش اعمال می‌شوند.
- روابط بین جداول **:**فیلترها از جداول ابعاد به جداول حقیقت از طریق روابط منتقل می‌شوند.

هنگامی که یک Measure محاسبه می‌شود، این کار در زمینه فیلتر فعلی انجام می‌شود. به‌عنوان‌مثال، اگر `[Total Sales]`را در یک جدول نمایش دهید که بر اساس `Year` سال) گروه‌بندی شده است `[Total Sales]`، برای هر سال، مجموع فروش آن سال را برمی‌گرداند، زیرا زمینه فیلتر برای هر ردیف جدول، به آن سال خاص محدود شده است.

۳-۶-۲:تابعCALCULATE

تابع `CALCULATE` یکی از مهم‌ترین توابع در DAX است. این تابع به شما امکان می‌دهد تا زمینه فیلتر را برای یک عبارت (Expression)تغییر دهید. سینتکس آن به‌صورت زیر است:

```
CALCULATE(<expression>, <filter1>, <filter2>, ...)
```

- `<expression>`:عبارتی است که می‌خواهید محاسبه کنید (معمولاً یک Measure یا یک تابع تجمیعی.(
- `<filter1>, <filter2>, ...`:یک یا چند فیلتر جدید که زمینه فیلتر را تغییر می‌دهند. این فیلترها می‌توانند جداول، ستون‌ها یا عبارات بولی (Boolean Expressions) باشند.

CALCULATEچگونه کار می‌کند؟

`CALCULATE`ابتدا زمینه فیلتر موجود را ارزیابی می‌کند. سپس، فیلترهای جدیدی که به‌عنوان آرگومان به آن داده شده‌اند را اعمال می‌کند. این فیلترهای جدید می‌توانند فیلترهای موجود را تغییر دهند **(Modify)** یا نادیده بگیرند .**(Override)** سپس، عبارت `<expression>` در زمینه فیلتر جدید ارزیابی می‌شود.

مثال: فروش سال **۲۰۱۹**

فرض کنید می‌خواهید مجموع فروش را فقط برای سال ۲۰۱۹ محاسبه کنید، صرف نظر از فیلترهای اعمال شده در گزارش.

```
Sales 2019 = CALCULATE([Total Sales] , DimDate [Year] = 2019)
```

توضیح **:**این `[Total Sales]` ،Measure را محاسبه می‌کند، اما زمینه فیلتر را به‌گونه‌ای تغییر می‌دهد که فقط ردیف‌هایی که `Year` آن‌ها برابر با ۲۰۱۹ است، در نظر گرفته شوند. حتی اگر گزارش شما بر اساس سال‌های دیگر فیلتر شده باشد، این Measure همیشه فروش ۲۰۱۹ را نشان می‌دهد.

مثال: فروش محصولات با تخفیف بالا

فرض کنید می‌خواهید فروش محصولاتی را محاسبه کنید که تخفیف آن‌ها بیشتر از ۱۰٪ بوده است.

```
High Discount Sales = CALCULATE([Total Sales] , FactSales [DiscountPercentage] > 0.1)
```

توضیح **:**این `[Total Sales]` ،Measure را محاسبه می‌کند، اما فقط برای ردیف‌هایی که `DiscountPercentage`آن‌ها بیشتر از ۰.۱ است.

۳-۶-۳تابعALL:

تابع ALL یکی دیگر از توابع بسیار مهم در DAX است که اغلب در ترکیب با CALCULATE استفاده می‌شود. ALL تمام فیلترها را از یک جدول یا ستون مشخص حذف می‌کند.

سینتکس آن به‌صورت زیر است:

```
ALL(<table> or <column>)
```

- `<table>`: تمام فیلترها را از تمام ستون‌های جدول مشخص شده حذف می‌کند.
- `<column>`: تمام فیلترها را فقط از ستون مشخص شده حذف می‌کند.

مثال: درصد فروش از کل

فرض کنید می‌خواهید درصد فروش هر محصول را نسبت به کل فروش محاسبه کنید.

```
Sales Percentage of Total =

DIVIDE(

[Total Sales] ,

CALCULATE([Total Sales] , ALL(DimProduct))

)
```

توضیح:

- [Total Sales]: فروش محصول فعلی را در زمینه فیلتر فعلی برمی‌گرداند (مثلاً فروش محصول A).
- CALCULATE([Total Sales] , ALL(DimProduct)): این بخش [Total Sales] را محاسبه می‌کند، اما با استفاده از ALL(DimProduct)، تمام فیلترهای اعمال شده بر روی جدول DimProduct را نادیده می‌گیرد. این بدان معناست که این بخش همیشه کل فروش را برای تمام محصولات (بدون توجه به محصولی که در ردیف فعلی جدول یا نمودار نمایش داده می‌شود) برمی‌گرداند.
- DIVIDE: این تابع برای جلوگیری از خطای تقسیم بر صفر استفاده می‌شود و در صورت تقسیم بر صفر، مقدار جایگزین (مثلا BLANK) را برمی‌گرداند.

این Measure به شما امکان می‌دهد تا سهم هر محصول را از کل فروش مشاهده کنید. ALL در سناریوهای مقایسه‌ای (مانند مقایسه با کل، مقایسه با میانگین) بسیار پرکاربرد است.

مثال: فروش کل بدون فیلتر سال

فرض کنید می‌خواهید مجموع فروش را بدون درنظرگرفتن فیلتر سال محاسبه کنید.

```
Total Sales All Years = CALCULATE([Total Sales] , ALL(DimDate [Year]))
```

توضیح: این [Total Sales] ،Measure را محاسبه می‌کند، اما فیلتر اعمال شده بر روی ستون Year از جدول DimDate را نادیده می‌گیرد. این بدان معناست که حتی اگر گزارش شما بر اساس یک سال خاص فیلتر شده باشد، این Measure همیشه مجموع فروش را برای تمام سال‌ها برمی‌گرداند.

۳-۷-توابع Iterator (تکرارکننده SUMX, AVERAGEX, COUNTX):

توابع تجمیعی که تاکنون بررسی کردیم (SUM, AVERAGE, COUNT و غیره) توابع ستونی (Column Functions) هستند، به این معنی که بر روی یک ستون واحد عمل می‌کنند. اما در DAX، توابعی به نام توابع **Iterator** (تکرارکننده) نیز وجود دارند که به شما امکان می‌دهند تا یک عبارت را برای هر ردیف از یک جدول ارزیابی کنید و سپس نتیجه را تجمیع کنید. این توابع با پسوند X مشخص می‌شوند (مانند: SUMX, AVERAGEX, COUNTX).

سینتکس عمومی توابع Iterator به‌صورت زیر است:

```
FUNCTIONX(<table>, <expression>)
```

- <table>: جدولی که می‌خواهید عبارت را برای هر ردیف آن ارزیابی کنید.
- <expression>: عبارتی که برای هر ردیف از جدول ارزیابی می‌شود.

۳-۷-۱: SUMX

تابع SUMX یک عبارت را برای هر ردیف از یک جدول ارزیابی می‌کند و سپس مجموع نتایج را برمی‌گرداند. این تابع برای محاسبه مجموع‌هایی که نیاز به ضرب یا محاسبات ردیف به ردیف دارند، بسیار مفید است.

مثال: مجموع فروش (مقدار * قیمت)

فرض کنید در جدول FactSales (ستون Quantity تعداد) و UnitPrice قیمت واحد) را دارید و می‌خواهید مجموع فروش را با ضرب این دو ستون در هر ردیف محاسبه کنید.

```
Total Sales Amount = SUMX(FactSales, FactSales [Quantity] * FactSales [UnitPrice])
```

توضیح:

- FactSales: جدول FactSales را به‌عنوان جدول برای تکرار مشخص می‌کند.
- FactSales [Quantity] * FactSales [UnitPrice]: این عبارت برای هر ردیف از جدول FactSales ارزیابی می‌شود. یعنی برای هر ردیف، در Quantity در UnitPrice ضرب می‌شود.
- SUMX: مجموع نتایج حاصل از ارزیابی عبارت برای تمام ردیف‌ها را برمی‌گرداند.

چرا **SUMX** به‌جای **Calculated Column**؟

می‌توانستید یک ستون محاسباتی به نام LineTotal در جدول FactSales ایجاد کنید (LineTotal = FactSales [Quantity] * FactSales [UnitPrice]) و سپس SUM(FactSales [LineTotal]) را محاسبه کنید. اما SUMX مزایایی دارد:

- فضای حافظه SUMX: یک Measure است و مقادیر را در مدل ذخیره نمی‌کند، بنابراین فضای حافظه را اشغال نمی‌کند. ستون محاسباتی فضای حافظه را اشغال می‌کند.
- انعطاف‌پذیری SUMX: به زمینه فیلتر پاسخ می‌دهد. اگر گزارش شما بر اساس محصول یا مشتری فیلتر شود SUMX ، فقط برای ردیف‌های فیلتر شده محاسبه می‌شود.

۳-۷-۲: AVERAGEX

تابع AVERAGEX یک عبارت را برای هر ردیف از یک جدول ارزیابی می‌کند و سپس میانگین نتایج را برمی‌گرداند.

مثال: میانگین سود هر تراکنش

فرض کنید می‌خواهید میانگین سود هر تراکنش را محاسبه کنید، جایی که سود برای هر ردیف به‌صورت SalesAmount - CostAmount محاسبه می‌شود.

```
Average Profit Per Transaction = AVERAGEX(FactSales, FactSales [SalesAmount] - FactSales [CostAmount])
```

توضیح: این Measure، عبارت FactSales [SalesAmount] - FactSales [CostAmount] را برای هر ردیف از جدول FactSales ارزیابی می‌کند و سپس میانگین سود حاصل از هر تراکنش را برمی‌گرداند.

۳-۷-۳: COUNTX

تابع COUNTX تعداد ردیف‌های یک جدول را که در آن‌ها یک عبارت غیرخالی (Non-Blank) ارزیابی می‌شود، می‌شمارد.

مثال: تعداد تراکنش‌های با سود مثبت

فرض کنید می‌خواهید تعداد تراکنش‌هایی را بشمارید که سود آن‌ها مثبت بوده است.

```
Positive Profit Transactions = COUNTX(FactSales, IF(FactSales [SalesAmount] - FactSales [CostAmount] > 0, 1))
```

توضیح:

- `FactSales`: جدول `FactSales` را برای تکرار مشخص می‌کند.
- `IF(FactSales [SalesAmount] - FactSales [CostAmount] > 0, 1)`: این عبارت برای هر ردیف ارزیابی می‌شود. اگر سود مثبت باشد `1`، را برمی‌گرداند، در غیر این صورت) `BLANK` به دلیل عدم وجود بخش `ELSE` در `IF`).
- `COUNTX`: فقط ردیف‌هایی را می‌شمارد که عبارت برای آن‌ها یک مقدار غیرخالی (یعنی `1` برگردانده است.

توابع Iterator بسیار قدرتمند هستند و به شما امکان می‌دهند تا محاسبات پیچیده‌ای را در سطح ردیف انجام دهید و سپس نتایج را تجمیع کنید. این توابع در بسیاری از سناریوهای پیشرفته DAX مورد استفاده قرار می‌گیرند.

۳-۸: توابع منطقی و شرطی IF, AND, OR, NOT :

توابع منطقی و شرطی به شما امکان می‌دهند تا منطق تصمیم‌گیری را در فرمول‌های DAX خود پیاده‌سازی کنید. این توابع برای ایجاد محاسبات پویا و پاسخگو به شرایط مختلف بسیار مفید هستند.

۳-۸-۱: IF

تابع `IF` یک شرط را بررسی می‌کند و بر اساس اینکه شرط درست (TRUE) یا نادرست (FALSE) باشد، دو نتیجه متفاوت را برمی‌گرداند. سینتکس آن به‌صورت زیر است:

IF(<logical_test>, <value_if_true>, <value_if_false>)

- `<logical_test>`: عبارتی که یک مقدار بولی (TRUE/FALSE) را برمی‌گرداند.
- `<value_if_true>`: مقداری که در صورت درست بودن شرط برگردانده می‌شود.
- `<value_if_false>`: مقداری که در صورت نادرست بودن شرط برگردانده می‌شود (اختیاری، اگر حذف شود ، BLANK برگردانده می‌شود.)

مثال: دسته‌بندی مشتریان بر اساس فروش

فرض کنید می‌خواهید مشتریان را بر اساس مجموع فروش آن‌ها به دو دسته `High Value` و `Low Value` تقسیم کنید. (این یک ستون محاسباتی خواهد بود)

Customer Value Category =

IF(

[Total Sales] > 1000000, -- فرض کنید Total Sales یک Measure است

"High Value",

"Low Value"

)

توضیح: این ستون محاسباتی (که باید در جدول مشتریان ایجاد شود) بررسی می‌کند که آیا [Total Sales] (که در زمینه فیلتر مشتری فعلی محاسبه می‌شود) بیشتر از ۱,۰۰۰,۰۰۰ است یا خیر. اگر باشد "High Value"، و در غیر این صورت "Low Value" را برمی‌گرداند.

نکته: در مثال بالا [Total Sales]، یک Measure است. اگر می‌خواهید این کار را به‌عنوان یک ستون محاسباتی در جدول DimCustomer انجام دهید، باید از SUMX یا CALCULATE برای جمع‌آوری فروش برای هر مشتری استفاده کنید، زیرا ستون محاسباتی در زمینه ردیف ارزیابی می‌شود و به طور مستقیم به [Total Sales] دسترسی ندارد.

Customer Value Category (Calculated Column) =

VAR CurrentCustomerSales = CALCULATE([Total Sales], RELATEDTABLE(FactSales))

RETURN

IF(

CurrentCustomerSales > 1000000,

"High Value",

"Low Value"

)

توضیح RELATEDTABLE(FactSales): تمام ردیف‌های مرتبط از FactSales را برای مشتری فعلی برمی‌گرداند و CALCULATE مجموع فروش را برای آن مشتری محاسبه می‌کند.

۳-۸-۲: AND (&&)

تابع) AND یا عملگر (&& دو یا چند شرط بولی را ترکیب می‌کند و تنها زمانی TRUE را برمی‌گرداند که تمام شرایط درست باشند.

```
Sales High Value and High Quantity =
CALCULATE(
[Total Sales] ,
FactSales [SalesAmount] > 100000 && FactSales [Quantity] > 50
)
```

توضیح :این Measure مجموع فروش را برای تراکنش‌هایی محاسبه می‌کند که هم مقدار فروش آن‌ها بیشتر از ۱۰۰۰۰۰ باشد و هم تعداد اقلام فروخته شده بیشتر از ۵۰ باشد.

۳-۸-۳: OR (||)

تابع) OR یا عملگر (|| دو یا چند شرط بولی را ترکیب می‌کند و زمانی TRUE را برمی‌گرداند که حداقل یکی از شرایط درست باشد.

```
Sales Online or Store =
CALCULATE(
[Total Sales] ,
FactSales [Channel] = "Online" || FactSales [Channel] = "Store"
)
```

توضیح :این Measure مجموع فروش را برای تراکنش‌هایی محاسبه می‌کند که کانال فروش آن‌ها "Online" یا "Store"باشد.

۳-۸-۴: NOT

تابع) NOT یا عملگر (!نتیجه یک عبارت بولی را معکوس می‌کند TRUE) را به FALSE و FALSE را به TRUE تبدیل می‌کند.(

```
Sales Not Online =
CALCULATE(
[Total Sales] ,
NOT(FactSales [Channel] = "Online")
)
```

توضیح **:**این Measure مجموع فروش را برای تمام تراکنش‌هایی محاسبه می‌کند که کانال فروش آن‌ها "Online" نباشد.

۳-۹:توابع متنی CONCATENATE, LEFT, RIGHT, MID, LEN, FIND, REPLACE :

توابع متنی در DAX به شما امکان می‌دهند تا با رشته‌های متنی کار کنید، آن‌ها را ترکیب کنید، بخشی از آن‌ها را استخراج کنید یا تغییر دهید. این توابع برای پاک‌سازی داده‌ها، فرمت‌بندی و ایجاد ستون‌های توصیفی جدید مفید هستند.

۳-۹-۱: CONCATENATE (و عملگر(&

تابع CONCATENATE دو رشته متنی را به هم متصل می‌کند. عملگر & نیز همین کار را انجام می‌دهد و معمولاً ترجیح داده می‌شود زیرا کوتاه‌تر و خواناتر است.

```
Full Product Name = DimProduct [ProductName] & " (" & DimProduct [Brand] & ") "
```

توضیح **:**این ستون محاسباتی (که در جدول DimProduct ایجاد می‌شود) نام محصول و برند آن را در یک رشته ترکیب می‌کند (مثلاً

"لپ‌تاپ.("(HP)

۳-۹-۲: LEFT, RIGHT, MID

این توابع برای استخراج بخشی از یک رشته متنی استفاده می‌شوند:

- **LEFT(,):**تعداد مشخصی از کاراکترها را از سمت چپ یک رشته برمی‌گرداند.

```
Product Code Prefix = LEFT(DimProduct [ProductCode] , 3)
```

توضیح **:**سه کاراکتر اول کد محصول را استخراج می‌کند.

- **RIGHT(,):**تعداد مشخصی از کاراکترها را از سمت راست یک رشته برمی‌گرداند.

```
Last Four Digits = RIGHT(FactSales [InvoiceNumber] , 4)
```

توضیح **:**چهار کاراکتر آخر شماره فاکتور را استخراج می‌کند.

- **MID(, ,):**تعداد مشخصی از کاراکترها را از یک موقعیت شروع مشخص در یک رشته برمی‌گرداند.

```
Middle Part of Code = MID(DimProduct [ProductCode] , 4, 2)
```

توضیح **:**دو کاراکتر را از موقعیت چهارم کد محصول استخراج می‌کند.

۳-۹-۳: LEN

تابع `LEN` طول (تعداد کاراکترها) یک رشته متنی را برمی‌گرداند.

```
Product Name Length = LEN(DimProduct [ProductName])
```

توضیح **:**طول نام محصول را محاسبه می‌کند.

۳-۹-۴: FIND

تابع `FIND` موقعیت شروع یک رشته متنی را در داخل رشته متنی دیگر پیدا می‌کند. این تابع به حروف کوچک و بزرگ حساس است.

```
Position of Dash = FIND("-", DimProduct [ProductCode])
```

توضیح **:**موقعیت اولین خط تیره را در کد محصول پیدا می‌کند.

۳-۹-۵: REPLACE

تابع `REPLACE` بخشی از یک رشته متنی را با یک رشته متنی دیگر جایگزین می‌کند.

```
Cleaned Product Code = REPLACE(DimProduct [ProductCode] , FIND("-", DimProduct [ProductCode]), 1, "_")
```

توضیح: اولین خط تیره را در کد محصول با یک آندرلاین _ جایگزین می‌کند. **FIND** موقعیت شروع را می‌دهد، 1 تعداد کاراکترهایی است که باید جایگزین شوند، و "_" رشته جدید است.

۳-۱۰: توابع تاریخ و زمان TODAY, NOW, DATE, YEAR, MONTH, DAY :

توابع تاریخ و زمان در DAX برای کار با مقادیر تاریخ و زمان استفاده می‌شوند. این توابع برای تحلیل‌های زمانی و ایجاد ستون‌های محاسباتی مرتبط با تاریخ بسیار مهم هستند.

۳-۱۰-۱: TODAY و NOW

- **TODAY():** تاریخ فعلی را برمی‌گرداند (بدون زمان.)

```
Today's Date = TODAY()
```

- **NOW():** تاریخ و زمان فعلی را برمی‌گرداند.

```
Current DateTime = NOW()
```

۳-۱۰-۲: DATE, YEAR, MONTH, DAY

- **DATE(, ,):** یک مقدار تاریخ را از سال، ماه و روز مشخص شده ایجاد می‌کند.

```
Custom Date = DATE(2023, 7, 16)
```

- **YEAR():** سال را از یک مقدار تاریخ برمی‌گرداند.

```
Order Year = YEAR(FactSales [OrderDate])
```

- **MONTH():** ماه را از یک مقدار تاریخ برمی‌گرداند (به‌صورت عدد ۱ تا ۱۲.)

```
Order Month Number = MONTH(FactSales [OrderDate])
```

- **DAY():** روز را از یک مقدار تاریخ برمی‌گرداند (به‌صورت عدد ۱ تا ۳۱.)

```
Order Day = DAY(FactSales [OrderDate])
```

مثال: محاسبه سن مشتری

فرض کنید ستون) `BirthDate` تاریخ تولد) در جدول `DimCustomer` دارید.

```
Customer Age =

VAR TodayDate = TODAY()

VAR BirthYear = YEAR(DimCustomer [BirthDate])

VAR BirthMonth = MONTH(DimCustomer [BirthDate])

VAR BirthDay = DAY(DimCustomer [BirthDate])

RETURN

IF(

MONTH(TodayDate) < BirthMonth || (MONTH(TodayDate) = BirthMonth && DAY(TodayDate) < BirthDay) ,

YEAR(TodayDate) - BirthYear - 1,

YEAR(TodayDate) - BirthYear

)
```

توضیح **:**این ستون محاسباتی سن مشتری را بر اساس تاریخ تولد و تاریخ امروز محاسبه می‌کند. این فرمول بررسی می‌کند که آیا روز تولد مشتری در سال جاری گذشته است یا خیر تا سن دقیق را محاسبه کند.

۳-۱۱:توابع اطلاعاتی IF, ISNONBLANK, ISBLANK :

توابع اطلاعاتی برای بررسی وضعیت مقادیر (مانند خالی بودن یا نبودن) و انجام عملیات بر اساس آن استفاده می‌شوند.

۳-۱۱-۱: ISBLANK وISNONBLANK

- **ISBLANK():**اگر مقدار خالی (BLANK) باشد `TRUE` ،و در غیر این صورت `FALSE` را برمی‌گرداند.

```
Has Discount = ISNONBLANK(FactSales [DiscountAmount])
```

توضیح :این ستون محاسباتی `TRUE` را برمی‌گرداند اگر `DiscountAmount` خالی نباشد (یعنی تخفیف اعمال شده باشد.)

- **ISNONBLANK():**اگر مقدار خالی نباشد `TRUE` ،و در غیر این صورت `FALSE` را برمی‌گرداند.

۲-۱۱-۳: BLANK()

تابع `BLANK()` یک مقدار خالی را برمی‌گرداند. این تابع اغلب در ترکیب با `IF` برای مدیریت سناریوهایی که نتیجه‌ای نباید نمایش داده شود، استفاده می‌شود.

```
Profit Margin =

IF(

ISBLANK([Total Sales]),

BLANK() ,

DIVIDE([Total Profit] , [Total Sales])

)
```

توضیح :این Measure حاشیه سود را محاسبه می‌کند، اما اگر `[Total Sales]` خالی باشد، به‌جای تقسیم بر صفر یا نمایش خطا، یک مقدار خالی برمی‌گرداند.

۳-۱۲:متغیرها (Variables) درDAX

استفاده از متغیرها (Variables) در فرمول‌های DAX یک بهترین روش بسیار مهم است. متغیرها به شما امکان می‌دهند تا:

- خوانایی فرمول را بهبود بخشید :با نام‌گذاری بخش‌های مختلف فرمول، درک آن آسان‌تر می‌شود.
- عملکرد را بهبود بخشید :یک متغیر فقط یک بار محاسبه می‌شود، حتی اگر چندین بار در فرمول استفاده شود. این امر از محاسبات تکراری جلوگیری می‌کند.
- دیباگینگ را آسان‌تر کنید :می‌توانید مقادیر متغیرها را در طول فرآیند دیباگینگ بررسی کنید.

برای تعریف یک متغیر از کلمه کلیدی `VAR` و برای برگرداندن نتیجه از کلمه کلیدی `RETURN` استفاده می‌شود.

سینتکس:

```
VAR <variable_name> = <expression>
VAR <variable_name2> = <expression2>
RETURN
<final_expression_using_variables>
```

مثال: محاسبه فروش سال گذشته با متغیرها

```
Sales Last Year (with Variables) =
VAR CurrentYearSales = [Total Sales]
VAR LastYearSales = CALCULATE(
[Total Sales] ,
SAMEPERIODLASTYEAR(DimDate [Date])
)
RETURN
IF(
ISBLANK(CurrentYearSales),
BLANK() ,
LastYearSales
)
```

توضیح:

- `CurrentYearSales`و `LastYearSales` دو متغیر هستند که نتایج محاسبات را در خود نگه می‌دارند.
- `RETURN`نتیجه نهایی را که از این متغیرها استفاده می‌کند، برمی‌گرداند.

۳-۱۳: خلاصه فصل ۳

در این فصل، وارد دنیای هیجان‌انگیز DAX شدیم و با مفاهیم پایه آن آشنا شدیم. آموختیم که ستون‌های محاسباتی ، Measuresو جداول محاسباتی چه کاربردی دارند و چه زمانی باید از هر یک استفاده کرد. با سینتکس پایه ،DAX عملگرها و نحوه کامنت‌گذاری آشنا شدیم. سپس، به سراغ اولین توابع پرکاربرد DAX مانند `SUM`, `AVERAGE`, `COUNT`, `MIN`, `MAX`, `DISTINCTCOUNT`, `COUNTROWS`رفتیم.

مهم‌تر از همه، با توابع قدرتمند `CALCULATE` و `ALL` آشنا شدیم که قلب DAX محسوب می‌شوند و به ما امکان می‌دهند تا زمینه فیلتر را تغییر دهیم و محاسبات پیچیده‌ای را انجام دهیم. همچنین، توابع Iterator مانند `SUMX` را برای محاسبات ردیف به ردیف بررسی کردیم. در نهایت، با توابع منطقی، متنی و تاریخ و زمان آشنا شدیم و اهمیت استفاده از متغیرها را در فرمول‌های DAXدرک کردیم.

در فصل‌های بعدی، به طور عمیق‌تر به توابع DAX خواهیم پرداخت و سناریوهای پیچیده‌تری را با استفاده از این توابع پیاده‌سازی خواهیم کرد. اکنون که با اصول اولیه DAX آشنا شده‌اید، آماده‌اید تا به سراغ مفاهیم پیشرفته‌تر بروید.

فصل ۴: توابع تکرارکننده (Iterators) و توابع جدول(Table Functions)

در فصل قبل، با توابع پایه DAX و همچنین توابع `CALCULATE` و `ALL` آشنا شدیم که به ما امکان می‌دهند تا زمینه فیلتر را کنترل کنیم. همچنین به طور مختصر به توابع تکرارکننده (Iterators) مانند `SUMX` اشاره کردیم. در این فصل، به طور عمیق‌تر به توابع تکرارکننده و همچنین توابع جدول (Table Functions) خواهیم پرداخت. این توابع به شما امکان می‌دهند تا عملیات پیچیده‌تری را بر روی جداول و ردیف‌ها انجام دهید و نتایج قدرتمندی را استخراج کنید.

۴-۱: درک عمیق‌تر توابع Iterator (X-Functions)

همانطور که در فصل قبل اشاره شد، توابع Iterator) که اغلب به آن‌ها X-Functions نیز گفته می‌شود) توابعی هستند که یک عبارت را برای هر ردیف از یک جدول ارزیابی می‌کنند و سپس نتیجه را تجمیع می‌کنند. این توابع برای سناریوهایی که نیاز به محاسبات ردیف به ردیف قبل از تجمیع دارید، ضروری هستند.

تفاوت کلیدی با توابع تجمیعی ساده:

- توابع تجمیعی ساده (مانند **SUM, AVERAGE):** بر روی یک ستون از یک جدول عمل می‌کنند و مجموع، میانگین و غیره را برمی‌گردانند. آن‌ها نمی‌توانند محاسبات ردیف به ردیف را انجام دهند. مثال : `SUM(FactSales [SalesAmount])` فقط ستون `SalesAmount` را جمع می‌کند.

- توابع **Iterator** مانند **SUMX, AVERAGEX):** یک جدول و یک عبارت را به‌عنوان آرگومان می‌گیرند. آن‌ها برای هر ردیف از جدول مشخص شده، عبارت را ارزیابی می‌کنند و سپس نتایج را تجمیع می‌کنند. مثال : `SUMX(FactSales, FactSales [Quantity] * FactSales [UnitPrice])` برای هر ردیف `Quantity`، را در `UnitPrice` ضرب می‌کند و سپس مجموع این حاصل‌ضرب‌ها را برمی‌گرداند.

چرا توابع **Iterator** قدرتمند هستند؟

قدرت اصلی توابع Iterator در این است که آن‌ها می‌توانند زمینه ردیف **(Row Context)** را ایجاد کنند. زمینه ردیف به این معنی است که فرمول در حال ارزیابی، به مقادیر ستون‌های ردیف فعلی که در حال پردازش است، دسترسی دارد. این قابلیت به شما امکان می‌دهد تا محاسبات پیچیده‌ای را در سطح جزئیات ردیف انجام دهید که با توابع تجمیعی ساده امکان‌پذیر نیست.

مثال‌های رایج از توابع **Iterator:**

- **SUMX:** برای محاسبه مجموع حاصل‌ضرب‌ها یا مجموع‌هایی که نیاز به منطق ردیف به ردیف دارند.
- **AVERAGEX:** برای محاسبه میانگین یک عبارت که برای هر ردیف ارزیابی می‌شود.
- **COUNTX:** برای شمارش ردیف‌هایی که یک شرط خاص را برآورده می‌کنند.
- **MAXX / MINX:** برای پیدا کردن حداکثر/حداقل مقدار یک عبارت که برای هر ردیف ارزیابی می‌شود.

SUMX :۱-۱-۴در عمل: محاسبه سود ناخالص

فرض کنید در جدول `FactSales`، ستون) `SalesAmount` مبلغ فروش) و) `CostAmount` مبلغ هزینه) را دارید و می‌خواهید مجموع سود ناخالص را محاسبه کنید.

```
Total Gross Profit = SUMX(FactSales, FactSales [SalesAmount] - FactSales [CostAmount])
```

توضیح `SUMX` **:**برای هر ردیف از جدول `SalesAmount`، `FactSales` را از `CostAmount` کم می‌کند و سپس مجموع تمام این تفاوت‌ها را برمی‌گرداند. این Measure به‌درستی مجموع سود ناخالص را محاسبه می‌کند، حتی اگر فیلترهایی بر روی محصول، مشتری یا تاریخ اعمال شده باشد.

۴-۱-۲: AVERAGEX در عمل: میانگین تعداد اقلام در هر سفارش

فرض کنید می‌خواهید میانگین تعداد اقلام فروخته شده در هر سفارش را محاسبه کنید. اگر جدول FactSales شما هر ردیف را به‌عنوان یک آیتم خطی (Line Item) از یک سفارش در نظر بگیرد، می‌توانید از AVERAGEX استفاده کنید.

```
Average Items Per Order = AVERAGEX(FactSales, FactSales [Quantity])
```

نکته **:**اگر هر ردیف FactSales یک سفارش کامل باشد AVERAGE(FactSales [Quantity]) ،نیز همین کار را انجام می‌دهد. اما اگر هر ردیف یک آیتم از سفارش باشد و یک سفارش شامل چندین آیتم باشد، باید ابتدا مجموع Quantityرا برای هر سفارش پیدا کنید و سپس میانگین آن را بگیرید. این سناریو پیچیده‌تر است و نیاز به توابع جدول دارد که در ادامه به آن می‌پردازیم.

۴-۱-۳: MAXX و MINX: پیدا کردن حداکثر/حداقل مقدار یک عبارت

توابع MAXX و MINX حداکثر و حداقل مقدار یک عبارت را که برای هر ردیف از یک جدول ارزیابی می‌شود، برمی‌گردانند.

مثال: بزرگ‌ترین تخفیف اعمال شده در یک تراکنش

```
Max Discount Amount = MAXX(FactSales, FactSales [SalesAmount] * FactSales [DiscountPercentage])
```

توضیح **:**این Measure برای هر ردیف از جدول ،FactSales مبلغ تخفیف را محاسبه می‌کند (SalesAmount * DiscountPercentage) و سپس بزرگ‌ترین مقدار تخفیف محاسبه شده را برمی‌گرداند.

۴-۲: توابع جدول (Table Functions)

توابع جدول در DAX، توابعی هستند که به‌جای یک مقدار اسکالر (Scalar Value) یا یک ستون، یک جدول را برمی‌گردانند. این جداول می‌توانند به‌عنوان ورودی برای سایر توابع DAX (به خصوص توابع Iterator) استفاده شوند یا برای ایجاد جداول محاسباتی جدید به کار روند. درک توابع جدول برای نوشتن فرمول‌های DAX پیچیده و قدرتمند بسیار مهم است.

مثال‌های رایج از توابع جدول**:**

- **FILTER:** یک جدول فیلتر شده را برمی‌گرداند.
- **ALL / ALLEXCEPT:** تمام فیلترها را از یک جدول/ستون حذف می‌کند و یک جدول کامل را برمی‌گرداند.
- **VALUES / DISTINCT:** یک جدول تک ستونی از مقادیر متمایز را برمی‌گرداند.
- **SUMMARIZE / SUMMARIZECOLUMNS:** یک جدول خلاصه شده را برمی‌گرداند.

- **ADDCOLUMNS:** ستون‌های جدیدی را به یک جدول موجود اضافه می‌کند.
- **CALENDAR / CALENDARAUTO:** یک جدول تاریخ را برمی‌گرداند.

FILTER: ۱-۲-۴فیلترکردن جداول

تابع FILTER یک جدول را به‌عنوان ورودی می‌گیرد و یک جدول جدید را برمی‌گرداند که فقط شامل ردیف‌هایی است که یک شرط مشخص را برآورده می‌کنند. FILTER اغلب در ترکیب با CALCULATE یا توابع Iterator استفاده می‌شود.

سینتکس:

```
FILTER(<table>, <filter_expression>)
```

- `<table>`: جدولی که می‌خواهید فیلتر کنید.
- `<filter_expression>`: یک عبارت بولی که برای هر ردیف از جدول ارزیابی می‌شود. ردیف‌هایی که این عبارت برای آن‌ها `TRUE` باشد، در جدول نتیجه گنجانده می‌شوند.

مثال: مجموع فروش محصولات گران‌قیمت

فرض کنید می‌خواهید مجموع فروش محصولاتی را محاسبه کنید که قیمت واحد آن‌ها بیشتر از ۱۰۰۰ تومان است.

```
Total Sales High Value Products =

CALCULATE(

[Total Sales] ,

FILTER(FactSales, FactSales [UnitPrice] > 1000)

)
```

توضیح:

- `FILTER(FactSales, FactSales [UnitPrice] > 1000)`: این بخش یک جدول موقت از `FactSales` ایجاد می‌کند که فقط شامل ردیف‌هایی است که `UnitPrice` آن‌ها بیشتر از ۱۰۰۰ است.

- CALCULATE([Total Sales] , ...): سپس [Total Sales] در زمینه فیلتر جدید (یعنی فقط برای محصولات گران‌قیمت) محاسبه می‌شود.

نکته FILTER: یک تابع Iterator است، به این معنی که برای هر ردیف از جدول ورودی filter_expression ، را ارزیابی می‌کند. این امر می‌تواند در جداول بسیار بزرگ، تأثیر عملکردی داشته باشد. در بسیاری از موارد، می‌توان فیلترها را مستقیماً به CALCULATE داد که کارایی بهتری دارد، مانند CALCULATE([Total Sales] , FactSales [UnitPrice] > 1000). اما FILTER زمانی ضروری است که نیاز به فیلترکردن بر اساس یک Measure یا یک عبارت پیچیده دارید که مستقیماً در CALCULATE قابل استفاده نیست.

۲-۲-۴: ALLSELECTED: حذف فیلترها به جز فیلترهای بصری‌سازی

تابع ALLSELECTED فیلترها را از ستون‌ها و جداول در زمینه پرس‌وجو فعلی حذف می‌کند، اما فیلترهایی که به طور مستقیم در بصری‌سازی اعمال شده‌اند را حفظ می‌کند. این تابع برای محاسبه درصد از کل در یک زیرمجموعه انتخاب شده از داده‌ها بسیار مفید است.

مثال: درصد فروش هر محصول در دسته انتخاب شده

فرض کنید می‌خواهید سهم فروش هر محصول را نسبت به کل فروش در دسته محصولی که کاربر در اسلایسر انتخاب کرده است، محاسبه کنید.

```
Sales % of Selected Category =

DIVIDE(

[Total Sales] ,

CALCULATE([Total Sales] , ALLSELECTED(DimProduct [Category]))

)
```

توضیح:

- [Total Sales]: فروش محصول فعلی را برمی‌گرداند.
- CALCULATE([Total Sales] , ALLSELECTED(DimProduct [Category])): این بخش مجموع فروش را محاسبه می‌کند، اما فیلترهای اعمال شده بر روی ستون

Category از جدول DimProduct را نادیده می‌گیرد، درحالی‌که سایر فیلترها (مانند فیلترهای اعمال شده بر روی سایر ستون‌ها یا فیلترهای بصری‌سازی) حفظ می‌شوند. این به شما امکان می‌دهد تا سهم هر محصول را در دسته انتخاب شده مشاهده کنید.

۳-۲-۴: VALUES و DISTINCT: استخراج مقادیر متمایز

- **VALUES():**یک جدول تک ستونی از مقادیر متمایز از ستون یا جدول مشخص شده را برمی‌گرداند. اگر هیچ فیلتری اعمال نشده باشد، تمام مقادیر متمایز را برمی‌گرداند. اگر فیلتری اعمال شده باشد، مقادیر متمایز را در زمینه فیلتر فعلی برمی‌گرداند.
- **DISTINCT():**مشابه `VALUES` است، اما فقط یک ستون را به‌عنوان آرگومان می‌گیرد و همیشه مقادیر متمایز را از آن ستون برمی‌گرداند، صرف نظر از زمینه فیلتر.

این توابع اغلب در ترکیب با CALCULATE یا برای ایجاد جداول محاسباتی استفاده می‌شوند.

مثال: شمارش تعداد محصولات فروخته شده در یک تراکنش

```
Products Sold in Transaction =

COUNTROWS(DISTINCT(FactSales [ProductID]))
```

توضیح: این **Measure** تعداد محصولات متمایزی را که در هر تراکنش فروخته شده‌اند، محاسبه می‌کند. **DISTINCT(FactSales [ProductID])** یک جدول موقت از **ProductID**های متمایز در زمینه فیلتر فعلی (یعنی برای هر تراکنش) ایجاد می‌کند و **COUNTROWS** تعداد ردیف‌های آن جدول را می‌شمارد.

۴-۲-۴: SUMMARIZE و SUMMARIZECOLUMNS: خلاصه کردن داده‌ها

این توابع برای ایجاد جداول خلاصه شده (Summary Tables) استفاده می‌شوند. آن‌ها به شما امکان می‌دهند تا داده‌ها را بر اساس یک یا چند ستون گروه‌بندی کنید و سپس محاسبات تجمیعی را بر روی گروه‌ها انجام دهید.

- **SUMMARIZE(, ..., , , ...):**یک جدول خلاصه شده را برمی‌گرداند. این تابع قدیمی‌تر است و در برخی سناریوها ممکن است عملکرد بهینه‌ای نداشته باشد.

- **SUMMARIZECOLUMNS(<groupBy_columnName1>, ..., , , ...):** نسخه جدیدتر و بهینه‌تر `SUMMARIZE` است که برای ایجاد جداول خلاصه شده استفاده می‌شود. این تابع به طور کلی توصیه می‌شود.

مثال: مجموع فروش بر اساس دسته محصول و سال

```
Sales By Category And Year =
SUMMARIZECOLUMNS(
DimProduct [Category] ,
DimDate [Year] ,
"Total Sales", [Total Sales]
)
```

توضیح: این فرمول یک جدول محاسباتی جدید ایجاد می‌کند که شامل ستون‌های `Category` و `Year` است و برای هر ترکیب منحصر به فرد از این دو `Total Sales`، را محاسبه می‌کند. این جدول می‌تواند برای تحلیل‌های بیشتر یا بصری‌سازی‌های خاص استفاده شود.

۴-۲-۵: ADDCOLUMNS: افزودن ستون به یک جدول

تابع `ADDCOLUMNS` ستون‌های محاسباتی جدیدی را به یک جدول موجود (یا یک جدول که توسط یک تابع جدول دیگر برگردانده شده است) اضافه می‌کند. این تابع اغلب در ترکیب با توابع Iterator استفاده می‌شود.

سینتکس:

```
ADDCOLUMNS(<table>, <name1>, <expression1>, <name2>, <expression2>, ...)
```

- `<table>`: جدولی که می‌خواهید ستون‌ها را به آن اضافه کنید.
- `<name>`: نام ستون جدید.
- `<expression>`: عبارتی که مقدار ستون جدید را برای هر ردیف از جدول ورودی محاسبه می‌کند.

مثال: محاسبه سود برای هر محصول در یک جدول موقت

فرض کنید می‌خواهید یک جدول موقت از محصولات ایجاد کنید که شامل نام محصول و سود ناخالص آن باشد.

```
Product Profit Table =
ADDCOLUMNS(
DimProduct,
"Gross Profit", CALCULATE([Total Gross Profit] , FactSales)
)
```

توضیح: این فرمول یک جدول محاسباتی جدید ایجاد می‌کند `ADDCOLUMNS`. به جدول `DimProduct` یک ستون جدید به نام `Gross Profit` اضافه می‌کند. برای هر ردیف از `DimProduct`، `CALCULATE([Total Gross Profit] , FactSales)` مجموع سود ناخالص را برای آن محصول خاص محاسبه می‌کند (زیرا `CALCULATE` زمینه فیلتر را به محصول فعلی محدود می‌کند.)

۴-۳: سناریوهای پیشرفته با توابع Iterator و جدول

ترکیب توابع Iterator و توابع جدول به شما امکان می‌دهد تا سناریوهای تحلیلی بسیار پیچیده‌ای را پیاده‌سازی کنید. در اینجا به چند مثال پیشرفته‌تر می‌پردازیم.

۴-۳-۱: محاسبه میانگین فروش روزانه برای هر ماه

این سناریو نیاز به محاسبه مجموع فروش برای هر روز و سپس گرفتن میانگین آن مجموع‌ها برای هر ماه دارد.

```
Average Daily Sales Per Month =
AVERAGEX(
SUMMARIZE(
FactSales,
DimDate [Date] ,
"Daily Sales", [Total Sales]
```

) ,

[Daily Sales]

)

توضیح:

1. `SUMMARIZE(FactSales, DimDate [Date] , "Daily Sales", [Total Sales])`: این بخش یک جدول موقت ایجاد می‌کند که برای هر `Date` روز (آن روز را)، `Total Sales` محاسبه می‌کند و آن را به‌عنوان `Daily Sales` نام‌گذاری می‌کند.
2. `AVERAGEX(<table>, [Daily Sales])`: سپس `AVERAGEX` بر روی این جدول موقت تکرار می‌شود و میانگین ستون `Daily Sales` را برمی‌گرداند. این Measure در زمینه فیلتر ماه (یا هر فیلتر دیگری) محاسبه می‌شود، بنابراین میانگین فروش روزانه برای آن ماه خاص را نشان می‌دهد.

۴-۳-۲: شمارش تعداد مشتریان فعال در یک دوره

مشتری فعال به مشتری‌ای گفته می‌شود که در یک دوره مشخص (مثلاً ماه جاری) حداقل یک تراکنش داشته باشد.

```
Active Customers =

CALCULATE(

DISTINCTCOUNT(FactSales [CustomerID]),

FILTER(

ALL(DimDate [Date]),

DimDate [Date] >= MIN(DimDate [Date]) && DimDate [Date] <= MAX(DimDate [Date])

)

)
```

توضیح:

- `DISTINCTCOUNT(FactSales [CustomerID])`: تعداد مشتریان متمایز را می‌شمارد.
- `FILTER(ALL(DimDate [Date]), ...)`: این بخش تمام فیلترهای اعمال شده بر روی ستون `Date` را حذف می‌کند و سپس آن را با یک فیلتر جدید جایگزین می‌کند که فقط تاریخ‌های بین حداقل و حداکثر تاریخ در زمینه فیلتر فعلی را شامل می‌شود. این تضمین می‌کند که شمارش مشتریان فعال در محدوده زمانی انتخاب شده (مثلاً ماه جاری) انجام شود.

نکته: این فرمول برای شمارش مشتریان فعال در زمینه فیلتر فعلی (مثلاً برای هر ماه در یک جدول) کار می‌کند. برای سناریوهای پیچیده‌تر هوش زمانی، مانند مشتریان فعال در ماه گذشته، از توابع هوش زمانی استفاده خواهیم کرد.

۳-۳-۴ محاسبه مجموع فروش برای هر محصول و نمایش آن در جدول محصولات:

فرض کنید می‌خواهید یک ستون محاسباتی در جدول `DimProduct` ایجاد کنید که مجموع فروش هر محصول را نشان دهد.

```
Total Sales (Calculated Column) =

CALCULATE(

[Total Sales] ,

RELATEDTABLE(FactSales)

)
```

توضیح:

- `RELATEDTABLE(FactSales)`: این تابع یک جدول موقت از تمام ردیف‌های مرتبط از `FactSales` را برای ردیف فعلی `DimProduct` برمی‌گرداند.
- `CALCULATE([Total Sales] , ...)`: سپس `[Total Sales]` در زمینه فیلتر این جدول موقت محاسبه می‌شود، که در واقع مجموع فروش برای محصول فعلی را برمی‌گرداند.

نکته: اگرچه این کار با ستون محاسباتی امکان‌پذیر است، اما به طور کلی توصیه می‌شود که مجموع فروش را به‌عنوان یک Measure (`[Total Sales]`) نگه دارید و آن را مستقیماً در بصری‌سازی‌ها استفاده کنید. ستون‌های محاسباتی فضای حافظه را اشغال می‌کنند و برای محاسبات تجمیعی که به زمینه فیلتر پاسخ می‌دهند، کمتر کارآمد هستند.

۴-۴:خلاصه فصل ۴

در این فصل، به طور عمیق‌تر به توابع تکرارکننده (Iterators) مانند `SUMX`, `AVERAGEX`, `MAXX`, `MINX` پرداختیم و درک کردیم که چگونه آن‌ها با ایجاد زمینه ردیف، امکان محاسبات پیچیده ردیف به ردیف را فراهم می‌کنند. همچنین، با توابع جدول (Table Functions) مانند `FILTER`, `ALLSELECTED`, `VALUES`, `DISTINCT`, `SUMMARIZECOLUMNS`و `ADDCOLUMNS` آشنا شدیم که به ما امکان می‌دهند تا جداول موقت ایجاد کرده و داده‌ها را به روش‌های جدیدی دستکاری کنیم.

ترکیب این توابع، قدرت فوق‌العاده‌ای به فرمول‌نویسی DAX شما می‌بخشد و شما را قادر می‌سازد تا به سؤالات تحلیلی پیچیده‌ای پاسخ دهید. در فصل بعدی، به طور مفصل به توابع فیلتر و توابع رابطه خواهیم پرداخت که کنترل شما را بر زمینه فیلتر و نحوه جریان فیلترها در مدل داده افزایش می‌دهند.

فصل ۵: توابع فیلتر (Filter Functions) و توابع رابطه (Relationship Functions)

در فصول گذشته، با مفاهیم پایه DAX، توابع تجمیعی و توابع تکرارکننده آشنا شدیم. همچنین، اهمیت زمینه فیلتر (Filter Context)و نقش حیاتی تابع `CALCULATE` در تغییر آن را درک کردیم. در این فصل، به طور عمیق‌تر به توابع فیلتر و توابع رابطه خواهیم پرداخت. این توابع به شما امکان می‌دهند تا کنترل دقیقی بر نحوه فیلتر شدن داده‌ها در مدل خود داشته باشید و از روابط بین جداول به بهترین شکل ممکن برای تحلیل‌های پیچیده استفاده کنید.

۵-۱:درک عمیق‌تر زمینه فیلتر (Filter Context) و زمینه ردیف(Row Context)

قبل از پرداختن به توابع فیلتر، لازم است که تفاوت و تعامل بین زمینه فیلتر **(Filter Context)** و زمینه ردیف **(Row Context)**را به طور کامل درک کنیم. این دو مفهوم، هسته اصلی نحوه عملکرد DAX هستند.

۵-۱-۱:زمینه فیلتر(Filter Context)

زمینه فیلتر، مجموعه‌ای از فیلترهایی است که در حال حاضر بر روی مدل داده اعمال شده‌اند. این فیلترها می‌توانند از منابع مختلفی نشأت بگیرند:

- بصری‌سازی‌ها **:**فیلترهایی که توسط محورها، افسانه‌ها (Legends)، اسلایسرها (Slicers) یا سایر عناصر بصری در گزارش اعمال می‌شوند.

- فیلترهای صفحه و گزارش **:**فیلترهایی که به طور کلی بر روی یک صفحه یا کل گزارش اعمال می‌شوند.
- روابط بین جداول **:**فیلترها از جداول ابعاد به جداول حقیقت از طریق روابط فعال منتقل می‌شوند.

هنگامی که یک Measure محاسبه می‌شود، این کار در زمینه فیلتر فعلی انجام می‌شود. به‌عنوان‌مثال، اگر `[Total Sales]`را در یک نمودار ستونی که بر اساس `Year` سال) گروه‌بندی شده است قرار دهید، برای هر ستون `[Total Sales]` ، در زمینه فیلتر آن سال خاص محاسبه می‌شود.

۵-۱-۲:زمینه ردیف(Row Context)

زمینه ردیف، یک مفهوم موقتی است که زمانی ایجاد می‌شود که DAX در حال ارزیابی یک عبارت ردیف به ردیف است. این اتفاق در دو سناریوی اصلی رخ می‌دهد:

1. ستون‌های محاسباتی **:(Calculated Columns)** هنگامی که یک ستون محاسباتی ایجاد می‌کنید، فرمول آن برای هر ردیف از جدول پایه ارزیابی می‌شود. در هر ردیف، فرمول به مقادیر ستون‌های آن ردیف دسترسی دارد.
2. توابع **:Iterator (X-Functions)** توابعی مانند `SUMX`, `AVERAGEX`, `FILTER`, `ADDCOLUMNS` و غیره، یک عبارت را برای هر ردیف از یک جدول ارزیابی می‌کنند. در طول این تکرار، زمینه ردیف برای هر ردیف از جدول ورودی ایجاد می‌شود.

تفاوت و تعامل**:**

- زمینه فیلتر، همیشه وجود دارد **.**حتی اگر هیچ فیلتری به طور صریح اعمال نشده باشد، یک زمینه فیلتر

خالی وجود دارد.

- زمینه ردیف، موقتی است. فقط زمانی وجود دارد که یک فرمول در حال ارزیابی ردیف به ردیف است.
- انتقال زمینه ردیف به زمینه فیلتر **:**این یکی از مهم‌ترین مفاهیم در DAX است. هنگامی که یک تابع) Iterator که زمینه ردیف ایجاد می‌کند) در داخل یک `CALCULATE` استفاده می‌شود، زمینه ردیف به زمینه فیلتر تبدیل می‌شود. این به `CALCULATE` اجازه می‌دهد تا فیلترها را بر اساس مقادیر ردیف فعلی اعمال کند. این مفهوم به‌عنوان**

**Context Transitionشناخته می‌شود و یکی از پیچیده‌ترین و در عین حال قدرتمندترین جنبه‌های DAX است.

۵-۲-توابع تغییردهنده فیلتر(Filter Modifiers):

توابع تغییردهنده فیلتر، توابعی هستند که در داخل `CALCULATE` استفاده می‌شوند تا نحوه اعمال فیلترها را تغییر دهند. این توابع به شما امکان می‌دهند تا کنترل دقیقی بر زمینه فیلتر داشته باشید.

۵-۲-۱-حذف فیلترها: ALL:

همانطور که در فصل ۳ اشاره شد، تابع `ALL` تمام فیلترها را از یک جدول یا ستون مشخص حذف می‌کند. این تابع برای محاسبه نسبت‌ها یا مقایسه با کل بسیار مفید است.

سینتکس:

```
ALL(<table> or <column>)
```

- `ALL(<table>)`: تمام فیلترها را از تمام ستون‌های جدول مشخص شده حذف می‌کند.
- `ALL(<column>)`: تمام فیلترها را فقط از ستون مشخص شده حذف می‌کند.

مثال: فروش کل شرکت (بدون فیلتر)

```
Total Sales All Filters Removed = CALCULATE([Total Sales] , ALL(FactSales))
```

توضیح: این Measure مجموع فروش را بدون درنظرگرفتن هیچ فیلتری که بر روی جدول `FactSales` اعمال شده باشد، محاسبه می‌کند. این می‌تواند برای محاسبه یک مقدار ثابت (مانند کل فروش شرکت) استفاده شود که با انتخاب‌های کاربر در گزارش تغییر نمی‌کند.

۵-۲-۲-حذف فیلترها به جز موارد مشخص: ALLEXCEPT:

تابع `ALLEXCEPT` تمام فیلترها را از یک جدول حذف می‌کند، به جز فیلترهایی که بر روی ستون‌های مشخص شده اعمال شده‌اند. این تابع برای سناریوهایی مفید است که می‌خواهید فیلترها را از بیشتر ستون‌ها حذف کنید، اما چند ستون خاص را فیلتر شده نگه دارید.

سینتکس:

```
ALLEXCEPT(<table>, <column1>, <column2>, ...)
```

- `<table>`: جدولی که می‌خواهید فیلترها را از آن حذف کنید.

- <column1>, <column2>, ...: ستون‌هایی که می‌خواهید فیلترهای آن‌ها حفظ شود.

مثال: فروش کل برای هر محصول (صرف نظر از مشتری و تاریخ)

فرض کنید می‌خواهید مجموع فروش را برای هر محصول محاسبه کنید، اما فیلترهای اعمال شده بر روی مشتری و تاریخ را نادیده بگیرید.

```
Product Sales Ignoring Customer And Date =
CALCULATE(
[Total Sales] ,
ALLEXCEPT(FactSales, FactSales [ProductID])
)
```

توضیح: این Measure مجموع فروش را محاسبه می‌کند، اما تمام فیلترها را از جدول FactSales حذف می‌کند، به جز فیلترهایی که بر روی ستون ProductID اعمال شده‌اند. این بدان معناست که اگر گزارش شما بر اساس ProductID گروه‌بندی شده باشد، این Measure مجموع فروش را برای هر محصول (صرف نظر از مشتری یا تاریخ) برمی‌گرداند.

۵-۲-۳: ALLNOBLANKROW: حذف فیلترها و ردیف خالی

تابع ALLNOBLANKROW مشابه ALL است، اما علاوه بر حذف تمام فیلترها از یک جدول، ردیف خالی (Blank Row) را نیز حذف می‌کند. ردیف خالی معمولاً در جداول ابعاد ظاهر می‌شود تا تراکنش‌هایی را که کلید خارجی آن‌ها در جدول حقیقت با هیچ کلید اصلی در جدول بعد مطابقت ندارد، مدیریت کند. در بیشتر موارد ALL ،و ALLNOBLANKROW نتایج یکسانی دارند، مگر اینکه ردیف‌های خالی در مدل شما وجود داشته باشد.

سینتکس:

```
ALLNOBLANKROW(<table>)
```

۵-۲-۴: KEEPFILTERS: حفظ فیلترهای موجود

تابع KEEPFILTERS یک فیلتر جدید را به زمینه فیلتر موجود اضافه می‌کند، به‌جای اینکه آن را جایگزین کند. این تابع زمانی مفید است که می‌خواهید یک فیلتر را اعمال کنید، اما نمی‌خواهید فیلترهای موجود را نادیده بگیرید.

سینتکس:

KEEPFILTERS(<expression>)

مثال: فروش محصولات خاص در زمینه فیلتر فعلی

فرض کنید می‌خواهید فروش محصولات دسته

"لوازم خانگی" را در زمینه فیلتر فعلی (مثلاً برای یک شهر خاص) محاسبه کنید.

Sales Home Appliances In Current Context =

CALCULATE(

[Total Sales] ,

KEEPFILTERS(DimProduct [Category] = "لوازم خانگی"

)

توضیح: این Measure مجموع فروش را محاسبه می‌کند KEEPFILTERS. تضمین می‌کند که فیلتر DimProduct[Category] = "لوازم خانگی" به فیلترهای موجود در زمینه (مثلاً فیلتر شهر) اضافه شود، نه اینکه آن‌ها را جایگزین کند. بنابراین، اگر شهر

"تهران" فیلتر شده باشد، این Measure فروش لوازم خانگی در تهران را نشان می‌دهد.

۵-۳-توابع جدول فیلتر(Table Filter Functions):

توابع جدول فیلتر، توابعی هستند که یک جدول را به‌عنوان ورودی می‌گیرند و یک جدول فیلتر شده را برمی‌گردانند. این توابع اغلب در ترکیب با CALCULATE یا توابع Iterator استفاده می‌شوند.

۵-۳-۱-فیلترکردن جداول: FILTER:

همانطور که در فصل ۴ اشاره شد، تابع FILTER یک جدول را به‌عنوان ورودی می‌گیرد و یک جدول جدید را برمی‌گرداند که فقط شامل ردیف‌هایی است که یک شرط مشخص را برآورده می‌کنند. FILTER یک تابع Iterator است و برای هر ردیف از جدول ورودی، عبارت فیلتر را ارزیابی می‌کند.

سینتکس:

FILTER(<table>, <filter_expression>)

مثال: مجموع فروش محصولات با حاشیه سود بالا

فرض کنید می‌خواهید مجموع فروش محصولاتی را محاسبه کنید که حاشیه سود آن‌ها بیشتر از ۲۰٪ است. (فرض کنید `[Profit Margin]` یک Measure است که قبلاً تعریف شده است.)

```
Total Sales High Margin Products =
CALCULATE(
[Total Sales] ,
FILTER(FactSales, [Profit Margin] > 0.20)
)
```

توضیح: در اینجا `FILTER`، بر روی جدول `FactSales` تکرار می‌شود و برای هر ردیف `[Profit Margin]`، را محاسبه می‌کند. سپس `CALCULATE`، `[Total Sales]` را فقط برای ردیف‌هایی که شرط `[Profit Margin] > 0.20` را برآورده می‌کنند، محاسبه می‌کند.

۵-۳-۲: ALLNOBLANKROW: حذف فیلترها و ردیف خالی

تابع `ALLNOBLANKROW` مشابه `ALL` است، اما علاوه بر حذف تمام فیلترها از یک جدول، ردیف خالی (Blank Row) را نیز حذف می‌کند. ردیف خالی معمولاً در جداول ابعاد ظاهر می‌شود تا تراکنش‌هایی را که کلید خارجی آن‌ها در جدول حقیقت با هیچ کلید اصلی در جدول بعد مطابقت ندارد، مدیریت کند. در بیشتر موارد `ALL`، و `ALLNOBLANKROW` نتایج یکسانی دارند، مگر اینکه ردیف‌های خالی در مدل شما وجود داشته باشد.

سینتکس:

ALLNOBLANKROW(<table>)

۵-۳-۳: CALCULATETABLE: تغییر زمینه فیلتر برای یک جدول

تابع `CALCULATETABLE` مشابه `CALCULATE` است، اما به‌جای یک مقدار اسکالر، یک جدول را برمی‌گرداند. این تابع به شما امکان می‌دهد تا زمینه فیلتر را برای یک جدول تغییر دهید و سپس از آن جدول فیلتر شده در سایر محاسبات استفاده کنید.

سینتکس:

```
CALCULATETABLE(<table>, <filter1>, <filter2>, ...)
```

- `<table>`: جدولی که می‌خواهید زمینه فیلتر آن را تغییر دهید.
- `<filter1>, <filter2>, ...`: یک یا چند فیلتر جدید که زمینه فیلتر را تغییر می‌دهند.

مثال: جدول فروش محصولات پرفروش

فرض کنید می‌خواهید یک جدول محاسباتی ایجاد کنید که فقط شامل فروش محصولات پرفروش (مثلاً فروش بالای ۱۰۰۰۰۰ تومان) باشد.

```
High Value Sales Table =
CALCULATETABLE(
    FactSales,
    FactSales [SalesAmount] > 100000
)
```

توضیح: این فرمول یک جدول جدید به نام `High Value Sales Table` ایجاد می‌کند که شامل تمام ردیف‌های جدول `FactSales` است که `SalesAmount` آن‌ها بیشتر از ۱۰۰۰۰۰ تومان است. این جدول می‌تواند سپس برای تحلیل‌های بیشتر یا بصری‌سازی‌های خاص استفاده شود.

۵-۴: توابع رابطه (Relationship Functions)

توابع رابطه در DAX به شما امکان می‌دهند تا با روابط بین جداول در مدل داده خود تعامل داشته باشید. این توابع برای سناریوهایی که نیاز به عبور از روابط، فعال کردن روابط غیرفعال یا بررسی وضعیت روابط دارید، ضروری هستند.

RELATED :۵-۴-۱وRELATEDTABLE

این دو تابع برای دسترسی به داده‌ها از جداول مرتبط استفاده می‌شوند:

- **RELATED():**یک مقدار از یک ستون در یک جدول مرتبط را برمی‌گرداند. این تابع فقط در زمینه ردیف (مانند ستون‌های محاسباتی یا توابع Iterator) قابل استفاده است و برای کار کردن نیاز به یک رابطه فعال یک به چند دارد.

مثال: اضافه‌کردن نام محصول به جدول فروش

فرض کنید می‌خواهید یک ستون محاسباتی در جدول `FactSales` ایجاد کنید که نام محصول را از جدول `DimProduct`نمایش دهد.

```
Product Name = RELATED(DimProduct [ProductName])
```

توضیح :این فرمول برای هر ردیف از `FactSales`، به جدول) `DimProduct` از طریق رابطه (`ProductID`نگاه می‌کند و `ProductName` مربوطه را برمی‌گرداند.

- **RELATEDTABLE(**

مثال: شمارش تعداد سفارشات برای هر مشتری

فرض کنید می‌خواهید یک ستون محاسباتی در جدول `DimCustomer` ایجاد کنید که تعداد سفارشات هر مشتری را نشان دهد.

```
Number of Orders = COUNTROWS(RELATEDTABLE(FactSales))
```

توضیح :این فرمول برای هر ردیف از `DimCustomer`، تمام ردیف‌های مرتبط از `FactSales` را (از طریق رابطه (`CustomerID` برمی‌گرداند و سپس `COUNTROWS` تعداد این ردیف‌ها را می‌شمارد.

USERELATIONSHIP: ۵-۴-۲فعال کردن روابط غیرفعال

تابع `USERELATIONSHIP` به شما امکان می‌دهد تا یک رابطه غیرفعال (Inactive Relationship) را به طور موقت در داخل یک `CALCULATE` فعال کنید. این تابع زمانی مفید است که چندین رابطه بین دو جدول دارید و می‌خواهید در سناریوهای مختلف از روابط متفاوتی استفاده کنید.

سینتکس:

USERELATIONSHIP(<column1>, <column2>)

- <column1>و :<column2> ستون‌هایی که رابطه بین آن‌ها را می‌خواهید فعال کنید.

مثال: فروش بر اساس تاریخ ارسال**(Ship Date)**

فرض کنید در مدل خود، یک رابطه فعال بین DimDate [Date] و FactSales [OrderDate] دارید و یک رابطه غیرفعال بین DimDate [Date] و .FactSales [ShipDate] می‌خواهید مجموع فروش را بر اساس تاریخ ارسال محاسبه کنید.

Total Sales by Ship Date =

CALCULATE(

[Total Sales] ,

USERELATIONSHIP(DimDate [Date] , FactSales [ShipDate])

)

توضیح :این [Total Sales]، Measure را محاسبه می‌کند، اما به‌جای استفاده از رابطه فعال ،(OrderDate) رابطه غیرفعال بین DimDate [Date] و FactSales [ShipDate] را فعال می‌کند و فیلترها را از طریق آن اعمال می‌کند.

۳-۴-۵تغییر جهت یا غیرفعال کردن روابط :CROSSFILTER:

تابع CROSSFILTER به شما امکان می‌دهد تا جهت فیلتر یک رابطه را تغییر دهید یا آن را به طور کامل غیرفعال کنید. این تابع نیز در داخل CALCULATE استفاده می‌شود.

سینتکس:

CROSSFILTER(<column1>, <column2>, <direction>)

- <column1>و :<column2> ستون‌هایی که رابطه بین آن‌ها را می‌خواهید تغییر دهید.

- <direction>: می‌تواند None (غیرفعال کردن رابطه)، OneWay (تک‌جهته) یا Both (دوجهته) باشد.

مثال: شمارش محصولات فروخته شده توسط مشتریان خاص

فرض کنید می‌خواهید تعداد محصولات فروخته شده را برای مشتریانی که در شهر خاصی (مثلاً تهران) زندگی می‌کنند، محاسبه کنید. اگر رابطه بین DimCustomer و FactSales تک‌جهته باشد (از مشتری به فروش)، نمی‌توانید به‌راحتی از FactSales به DimCustomer فیلتر کنید. در اینجا CROSSFILTER می‌تواند کمک کند.

```
Products Sold to Tehran Customers =
CALCULATE(
DISTINCTCOUNT(FactSales [ProductID]),
CROSSFILTER(FactSales [CustomerID] , DimCustomer [CustomerID] , BOTH),
DimCustomer [City] = "تهران"
)
```

توضیح:

- CROSSFILTER(FactSales [CustomerID] , DimCustomer [CustomerID] , BOTH): این بخش به طور موقت رابطه بین FactSales و DimCustomer را دوجهته می‌کند.
- DimCustomer [City] = "تهران": سپس فیلتر شهر تهران اعمال می‌شود. با دوجهته شدن رابطه، این فیلتر می‌تواند از DimCustomer به FactSales منتقل شود و DISTINCTCOUNT(FactSales [ProductID]) تعداد محصولات فروخته شده به مشتریان تهران را برمی‌گرداند.

نکته: استفاده از CROSSFILTER با جهت BOTH باید با احتیاط انجام شود، زیرا می‌تواند منجر به ابهام در مدل و نتایج غیرمنتظره شود. در صورت امکان، سعی کنید مدل خود را به‌گونه‌ای طراحی کنید که نیاز به روابط دوجهته را به حداقل برساند.

۵-۵:توابع اطلاعاتی زمینه فیلتر(Filter Context Information Functions)

این توابع به شما امکان می‌دهند تا اطلاعاتی در مورد زمینه فیلتر فعلی به دست آورید. این توابع برای ایجاد منطق‌های شرطی بر اساس وضعیت فیلترها بسیار مفید هستند.

۵-۵-۱: HASONEVALUE وHASONEFILTER

- **HASONEVALUE():**اگر ستون مشخص شده در زمینه فیلتر فعلی فقط یک مقدار متمایز داشته باشد ، `TRUE`را برمی‌گرداند. این تابع برای کنترل نمایش گزارش‌ها بر اساس انتخاب کاربر مفید است.

مثال: نمایش جزئیات فقط در صورت انتخاب یک محصول

```
Product Details =
IF(
HASONEVALUE(DimProduct [ProductName]),
    " جزئیات محصول,([DimProduct [ProductName] & " : VALUES(
    " لطفاً یک محصول را انتخاب کنید تا جزئیات را مشاهده کنید".
)
```

توضیح :این Measure یک پیام را برمی‌گرداند. اگر کاربر فقط یک محصول را انتخاب کرده باشد، نام آن محصول را نمایش می‌دهد. در غیر این صورت، از کاربر می‌خواهد که یک محصول را انتخاب کند.

- **HASONEFILTER():**اگر ستون مشخص شده در زمینه فیلتر فعلی فیلتر شده باشد و فقط یک مقدار متمایز را شامل شود `TRUE` ،را برمی‌گرداند. این تابع کمی متفاوت از `HASONEVALUE` است، زیرا `HASONEVALUE`به تعداد مقادیر متمایز در زمینه فیلتر نگاه می‌کند، درحالی‌که `HASONEFILTER` به وجود یک فیلتر صریح بر روی ستون نگاه می‌کند.

۵-۵-۲: ISFILTERED وISCROSSFILTERED

- **ISFILTERED():**اگر ستون یا جدول مشخص شده در زمینه فیلتر فعلی فیلتر شده باشد `TRUE` ،را برمی‌گرداند.

مثال: نمایش پیام بر اساس فیلتر شدن منطقه

```
Region Filter Status =
IF(
ISFILTERED(DimCustomer [Region]),
    "  منطقه فیلتر شده است,".
    "  هیچ منطقه‌ای فیلتر نشده است".
)
```

- **ISCROSSFILTERED():**اگر ستون یا جدول مشخص شده توسط یک فیلتر متقاطع (Cross-filter) از یک جدول دیگر فیلتر شده باشد `TRUE` ،را برمی‌گرداند.

۵-۶خلاصه فصل ۵:

در این فصل، به طور عمیق‌تر به توابع فیلتر و توابع رابطه در DAX پرداختیم. درک عمیق زمینه فیلتر و زمینه ردیف، پایه و اساس تسلط بر این توابع است. با توابع تغییردهنده فیلتر مانند ALLNOBLANKROW, ALLEXCEPT, ALL و KEEPFILTERSآشنا شدیم که به ما امکان می‌دهند تا کنترل دقیقی بر نحوه اعمال فیلترها داشته باشیم. همچنین، توابع جدول فیلتر مانند FILTER و CALCULATETABLE را بررسی کردیم که برای ایجاد جداول فیلتر شده موقت استفاده می‌شوند.

در نهایت، با توابع رابطه مانند USERELATIONSHIP, RELATEDTABLE, RELATED و CROSSFILTER آشنا شدیم که به ما امکان می‌دهند تا با روابط بین جداول تعامل داشته باشیم و سناریوهای پیچیده مدل‌سازی را مدیریت کنیم. همچنین، توابع اطلاعاتی زمینه فیلتر مانند HASONEVALUE و ISFILTERED را بررسی کردیم که برای ایجاد منطق‌های شرطی بر اساس وضعیت فیلترها مفید هستند.

تسلط بر این توابع، شما را قادر می‌سازد تا فرمول‌های DAX بسیار قدرتمند و انعطاف‌پذیری بنویسید که به طور دقیق به سؤالات کسب‌وکار شما پاسخ می‌دهند. در فصل بعدی، به یکی از مهم‌ترین و پرکاربردترین جنبه‌های تحلیل داده، یعنی هوش زمانی (Time Intelligence)خواهیم پرداخت و نحوه پیاده‌سازی آن را با تقویم شمسی بررسی خواهیم کرد.

فصل ۶: هوش زمانی (Time Intelligence) با تقویم شمسی

هوش زمانی (Time Intelligence) یکی از قدرتمندترین قابلیت‌های DAX است که به شما امکان می‌دهد تا تحلیل‌های مبتنی بر زمان را به‌راحتی انجام دهید. این تحلیل‌ها شامل مقایسه عملکرد در دوره‌های مختلف (مانند سال‌به‌سال، ماه‌به‌ماه)، محاسبه مجموع‌های تجمعی (مانند MTD، QTD، YTD) و بررسی روندها در طول زمان است. با این حال، برای کسب‌وکارهای ایرانی، چالش اصلی در این بخش، عدم تطابق تقویم میلادی (که Power BI به طور پیش‌فرض از آن پشتیبانی می‌کند) با تقویم رسمی کشور، یعنی تقویم شمسی (جلالی)است. در این فصل، به طور جامع به نحوه پیاده‌سازی هوش زمانی در Power BI با استفاده از تقویم شمسی خواهیم پرداخت.

۶-۱: اهمیت هوش زمانی در تحلیل کسب‌وکار

تحلیل‌های زمانی برای هر کسب‌وکاری حیاتی هستند، زیرا به مدیران و تحلیلگران کمک می‌کنند تا:

- روندها را شناسایی کنند :آیا فروش در حال رشد است یا کاهش؟ آیا الگوهای فصلی وجود دارد؟
- عملکرد را مقایسه کنند :عملکرد فعلی را با دوره‌های گذشته (مثلاً سال گذشته، ماه گذشته) مقایسه کنند تا رشد یا افت را ارزیابی کنند.
- پیش‌بینی‌ها را ارزیابی کنند :عملکرد واقعی را با پیش‌بینی‌ها مقایسه کنند.
- تصمیمات استراتژیک بگیرند :بر اساس بینش‌های زمانی، استراتژی‌های بازاریابی، تولید یا فروش را تنظیم کنند.

مثال‌هایی از سؤالات کسب‌وکار که با هوش زمانی پاسخ داده می‌شوند:

- فروش ما در این ماه چقدر بوده است؟
- فروش این ماه نسبت به ماه گذشته چقدر رشد داشته است؟
- فروش ما از ابتدای سال تا کنون چقدر است؟
- عملکرد فروش ما در این فصل نسبت به فصل مشابه سال گذشته چگونه بوده است؟
- کدام محصولات در فصل تابستان بیشترین فروش را دارند؟

۶-۲: پیش‌نیاز: جدول تاریخ شمسی(Persian Date Table)

همانطور که در فصل ۲ اشاره شد، یک جدول تاریخ مناسب، ستون فقرات تمام تحلیل‌های هوش زمانی است. برای کار با تقویم شمسی، ما نیاز به یک جدول تاریخ شمسی داریم که شامل تمام تاریخ‌های شمسی و ویژگی‌های مرتبط با آن‌ها (مانند سال شمسی، ماه شمسی، روز شمسی، فصل شمسی و غیره) باشد. Power BI به طور پیش‌فرض از تقویم میلادی استفاده می‌کند، بنابراین باید این جدول را به‌صورت سفارشی ایجاد کنیم.

چندین روش برای ایجاد جدول تاریخ شمسی وجود دارد:

1. استفاده از **Power Query (M Language):** این روش انعطاف‌پذیری بالایی دارد و به شما امکان می‌دهد تا یک جدول تاریخ شمسی کامل را با استفاده از توابع M ایجاد کنید. این روش برای اکثر سناریوها توصیه می‌شود.
2. استفاده از **DAX:** می‌توانید با استفاده از توابع DAX و تبدیل تاریخ میلادی به شمسی، یک جدول تاریخ شمسی ایجاد کنید. این روش ممکن است کمی پیچیده‌تر باشد، اما برای کسانی که با DAX راحت‌تر هستند، قابل استفاده است.
3. استفاده از اسکریپت‌های آماده **:**بسیاری از توسعه‌دهندگان، اسکریپت‌های آماده‌ای برای تولید جدول تاریخ شمسی در Power Queryیا DAX ایجاد کرده‌اند که می‌توانید از آن‌ها استفاده کنید.

در این بخش، ما بر روی روش Power Query تمرکز خواهیم کرد، زیرا برای ایجاد یک جدول تاریخ کامل و قابل نگهداری، کارآمدتر است.

ساخت جدول تاریخ شمسی با**Power Query:**

برای ساخت جدول تاریخ شمسی، مراحل زیر را دنبال کنید:

1. باز کردن **Power Query Editor:** در Power BI Desktop، به تب `Home` بروید و بر روی `Transform data`کلیک کنید. این کار Power Query Editor را باز می‌کند.
2. ایجاد یک **Query** جدید **:**در Power Query Editor، به تب `Home` بروید و بر روی `New Source` -> `Blank Query`کلیک کنید.
3. نوشتن کد **M:** در نوار فرمول (Formula Bar) یا در `Advanced Editor`، کد M زیر را وارد کنید. این کد یک تابع سفارشی برای تبدیل تاریخ میلادی به شمسی و سپس ایجاد یک جدول تاریخ شمسی کامل ایجاد می‌کند. (این کد یک مثال ساده است و ممکن است نیاز به سفارشی‌سازی داشته باشد.(

```
let

// Define a function to convert Gregorian to Persian date

GregorianToPersian = (gregorianDate as date) as record =>

let

year = Date.Year(gregorianDate),
```

```
    month = Date.Month(gregorianDate),
    day = Date.Day(gregorianDate),
    // This is a simplified conversion. For full accuracy, consider a more robust library or function.
    // For demonstration, let's assume a basic conversion logic or external function
    // In a real scenario, you'd use a more complex algorithm or a custom function from a shared library.
    // For simplicity, we'll just return the Gregorian date components for now and add Persian logic later.
    // This part needs a proper Persian calendar conversion algorithm.
    // For a real-world solution, you'd integrate a robust Persian calendar conversion.
    // Example: using a custom function from a shared M module for jalaali conversion.
    // Let's assume we have a function 'Jalaali.FromGregorian' available.
    // For now, we'll simulate it.
    persianDate = Date.ToText(gregorianDate, "yyyy/MM/dd") , // Placeholder
    persianYear = Text.Start(persianDate, 4), // Placeholder
    persianMonth = Text.Middle(persianDate, 5, 2), // Placeholder
    persianDay = Text.End(persianDate, 2) // Placeholder
in
```

```
[PersianDate = persianDate, PersianYear = persianYear, PersianMonth =
persianMonth, PersianDay = persianDay] ,

// Define the start and end dates for your calendar

StartDate = #date(2010, 1, 1),

EndDate = #date(2030, 12, 31),

// Generate a list of dates

DateList = List.Dates(StartDate, Duration.Days(EndDate - StartDate) + 1,
#duration(1, 0, 0, 0)) ,

// Convert the list to a table

DateTable = Table.FromList(DateList, Splitter.SplitByNothing() , null, null,
ExtraValues.Error),

#"Renamed Columns" = Table.RenameColumns(DateTable,{{"Column1",
"Date"}}) ,

// Add date attributes

#"Added Year" = Table.AddColumn(#"Renamed Columns", "Year", each
Date.Year([Date]), type number),

#"Added Month" = Table.AddColumn(#"Added Year", "Month", each
Date.Month([Date]), type number),

#"Added Day" = Table.AddColumn(#"Added Month", "Day", each
Date.Day([Date]), type number),

#"Added Quarter" = Table.AddColumn(#"Added Day", "Quarter", each
Date.QuarterOfYear([Date]), type number),

#"Added Day Name" = Table.AddColumn(#"Added Quarter", "Day Name", each
Date.DayOfWeekName([Date]), type text),
```

```
#"Added Month Name" = Table.AddColumn(#"Added Day Name", "Month Name", each Date.MonthName([Date]), type text),

#"Added Week of Year" = Table.AddColumn(#"Added Month Name", "Week of Year", each Date.WeekOfYear([Date]), type number),

#"Added Day of Year" = Table.AddColumn(#"Added Week of Year", "Day of Year", each Date.DayOfYear([Date]), type number),

// Add Persian Date attributes (This part needs a proper Persian calendar conversion logic)

#"Added Persian Date" = Table.AddColumn(#"Added Day of Year", "Persian Date", each GregorianToPersian([Date]) [PersianDate] , type text),

#"Added Persian Year" = Table.AddColumn(#"Added Persian Date", "Persian Year", each GregorianToPersian([Date]) [PersianYear] , type text),

#"Added Persian Month" = Table.AddColumn(#"Added Persian Year", "Persian Month", each GregorianToPersian([Date]) [PersianMonth] , type text),

#"Added Persian Day" = Table.AddColumn(#"Added Persian Month", "Persian Day", each GregorianToPersian([Date]) [PersianDay] , type text),

// Order Month Name

#"Added Month Sort" = Table.AddColumn(#"Added Persian Day", "Month Sort", each Date.Month([Date]), type number),

#"Added Quarter Sort" = Table.AddColumn(#"Added Month Sort", "Quarter Sort", each Date.QuarterOfYear([Date]), type number),

// Set data types

#"Changed Type" = Table.TransformColumnTypes(#"Added Quarter Sort",{

{"Date", type date},

{"Year", Int64.Type},
```

{"Month", Int64.Type},

{"Day", Int64.Type},

{"Quarter", Int64.Type},

{"Week of Year", Int64.Type},

{"Day of Year", Int64.Type},

{"Persian Date", type text},

{"Persian Year", type text},

{"Persian Month", type text},

{"Persian Day", type text},

{"Month Sort", Int64.Type},

{"Quarter Sort", Int64.Type}

})

in

#"Changed Type"

نکته بسیار مهم **:**کد M بالا برای تبدیل تاریخ میلادی به شمسی، یک تابع `GregorianToPersian` را به‌صورت) Placeholder جایگزین موقت) قرار داده است. برای یک راه‌حل واقعی و دقیق، شما نیاز به یک تابع تبدیل تاریخ شمسی دقیق دارید. بهترین راهکار، استفاده از یک تابع سفارشی M است که توسط جامعه Power BI برای این منظور توسعه یافته است. می‌توانید با جستجوی `Power Query Persian Calendar` یا `M function Jalaali`در اینترنت، به این توابع دسترسی پیدا کنید. این توابع معمولاً شامل منطق پیچیده‌ای برای محاسبه سال، ماه و روز شمسی بر اساس تاریخ میلادی هستند.

مثال (فرضی) از یک تابع دقیق‌تر تبدیل تاریخ شمسی در**M:**

// This is a conceptual example. Actual implementation requires a robust algorithm.

```
// Function: Jalaali.FromGregorian(year as number, month as number, day as number) as record

// Returns: [Year = ..., Month = ..., Day = ..., MonthName = ..., DayName = ...]

// You would typically load this from a shared M module or a pre-built template.
```

پس از وارد کردن کد، نام Query را به `DimDate` تغییر دهید و `Close & Apply` را بزنید تا جدول تاریخ شمسی به مدل داده شما اضافه شود.

4. علامت‌گذاری جدول تاریخ **(Mark as Date Table):** پس از بارگذاری جدول `DimDate` در Power BI Desktop، باید آن را به‌عنوان یک جدول تاریخ علامت‌گذاری کنید. این کار به Power BI کمک می‌کند تا توابع هوش زمانی DAX را به‌درستی اعمال کند.
   - در نمای مدل (Model View)، جدول `DimDate` را انتخاب کنید.
   - در پنل Properties (سمت راست)، بخش `Table tools`، بر روی `Mark as date table` کلیک کنید.
   - در پنجره باز شده، ستون `Date` (که تاریخ میلادی است) را به‌عنوان ستون تاریخ انتخاب کنید و `OK` را بزنید.
5. ایجاد رابطه **:** یک رابطه یک به چند از `DimDate [Date]` به ستون تاریخ تراکنش در جدول حقیقت خود (مثلاً `FactSales [OrderDate]`) ایجاد کنید. این رابطه باید فعال (Active) باشد.

۶-۳: توابع هوش زمانی (Time Intelligence Functions)

اکنون که جدول تاریخ شمسی ما آماده است و به جدول حقیقت متصل شده است، می‌توانیم از توابع هوش زمانی DAX استفاده کنیم. این توابع به طور خودکار فیلترها را بر روی جدول تاریخ اعمال می‌کنند تا محاسبات را برای دوره‌های زمانی مختلف انجام دهند.

۶-۳-۱: مجموع‌های تجمعی (Year-to-Date, Quarter-to-Date, Month-to-Date)

این توابع مجموع یک Measure را از ابتدای دوره (سال، فصل، ماه) تا تاریخ فعلی در زمینه فیلتر محاسبه می‌کنند.

- **TOTALYTD(, , [] , []):** مجموع عبارت را از ابتدای سال تا تاریخ فعلی محاسبه می‌کند.

- Measure :<expression>یا عبارتی که می‌خواهید جمع کنید (مثلا.([Total Sales]ً
- :<dates>ستون تاریخ از جدول تاریخ شما (مثلا.([DimDate [Date]ً
-) :<year_end_date>اختیاری) برای تقویم‌های غیرمیلادی مانند شمسی، می‌توانید تاریخ پایان سال را مشخص کنید (مثلا "03-20"ً برای ۲۹ اسفند.(

```
Sales YTD = TOTALYTD([Total Sales] , DimDate [Date] , "03-20")
```

توضیح :این Measure مجموع فروش را از ابتدای سال شمسی (اول فروردین) تا تاریخ فعلی در زمینه فیلتر محاسبه می‌کند.

- **:(TOTALQTD(, , [])**مجموع عبارت را از ابتدای فصل تا تاریخ فعلی محاسبه می‌کند.

```
Sales QTD = TOTALQTD([Total Sales] , DimDate [Date])
```

توضیح :این Measure مجموع فروش را از ابتدای فصل شمسی تا تاریخ فعلی محاسبه می‌کند.

- **:(TOTALMTD(, , [])**مجموع عبارت را از ابتدای ماه تا تاریخ فعلی محاسبه می‌کند.

```
Sales MTD = TOTALMTD([Total Sales] , DimDate [Date])
```

توضیح :این Measure مجموع فروش را از ابتدای ماه شمسی تا تاریخ فعلی محاسبه می‌کند.

۶-۳-۲:مقایسه با دوره‌های گذشته(Previous Period Comparisons)

این توابع به شما امکان می‌دهند تا عملکرد فعلی را با دوره‌های مشابه در گذشته مقایسه کنید.

- **:()SAMEPERIODLASTYEAR**یک جدول از تاریخ‌ها را برمی‌گرداند که یک سال قبل از تاریخ‌های موجود در زمینه فیلتر فعلی هستند.

```
Sales Last Year = CALCULATE([Total Sales] , SAMEPERIODLASTYEAR(DimDate [Date]))
```

توضیح :این Measure مجموع فروش را برای همان دوره زمانی در سال گذشته شمسی محاسبه می‌کند.

- **:(DATEADD(, ,)**یک جدول از تاریخ‌ها را برمی‌گرداند که به تعداد مشخصی از فواصل زمانی (روز، ماه، فصل، سال) به جلو یا عقب منتقل شده‌اند.

- <interval>: می‌تواند DAY, MONTH, QUARTER, YEAR باشد.

```
Sales Previous Month = CALCULATE([Total Sales] , DATEADD(DimDate [Date] , -1, MONTH))
```

توضیح: این Measure مجموع فروش را برای ماه گذشته شمسی محاسبه می‌کند.

```
Sales Previous Quarter = CALCULATE([Total Sales] , DATEADD(DimDate [Date] , -1, QUARTER))
```

توضیح: این Measure مجموع فروش را برای فصل گذشته شمسی محاسبه می‌کند.

۶-۳-۳: محاسبه رشد (Growth Calculations)

با استفاده از Measures بالا، می‌توانید Measures برای محاسبه رشد ایجاد کنید.

- رشد سال‌به‌سال **(Year-over-Year Growth):**

```
Sales YoY Growth =
DIVIDE(
[Total Sales] - [Sales Last Year] ,
[Sales Last Year]
)
```

توضیح: این Measure درصد رشد فروش را نسبت به سال گذشته شمسی محاسبه می‌کند.

- رشد ماه‌به‌ماه **(Month-over-Month Growth):**

```
Sales MoM Growth =
DIVIDE(
[Total Sales] - [Sales Previous Month] ,
```

[Sales Previous Month]

)

توضیح :این Measure درصد رشد فروش را نسبت به ماه گذشته شمسی محاسبه می‌کند.

۴-۳-۶توابع برای شروع و پایان دوره(Start/End of Period):

- **STARTOFMONTH():**اولین تاریخ ماه را در زمینه فیلتر فعلی برمی‌گرداند.
- **ENDOFMONTH():**آخرین تاریخ ماه را در زمینه فیلتر فعلی برمی‌گرداند.
- **STARTOFQUARTER():**اولین تاریخ فصل را در زمینه فیلتر فعلی برمی‌گرداند.
- **ENDOFQUARTER():**آخرین تاریخ فصل را در زمینه فیلتر فعلی برمی‌گرداند.
- **STARTOFYEAR():**اولین تاریخ سال را در زمینه فیلتر فعلی برمی‌گرداند.
- **ENDOFYEAR():**آخرین تاریخ سال را در زمینه فیلتر فعلی برمی‌گرداند.

این توابع اغلب در ترکیب با `CALCULATE` برای ایجاد Measures سفارشی استفاده می‌شوند.

مثال: فروش در آخرین روز ماه

```
Sales Last Day of Month =

CALCULATE(

[Total Sales] ,

LASTDATE(DimDate [Date])

)
```

توضیح :این Measure مجموع فروش را برای آخرین تاریخ موجود در زمینه فیلتر فعلی (که معمولاً آخرین روز ماه است) محاسبه می‌کند.

حتماً! اینجا یک پست آماده برای اضافه شدن به فصل هوش زمانی کتابت هست—با عنوان، ساختار تمیز، و کدهایی که به‌راحتی قابل استفاده در Power BI هستند. این بخش به‌عنوان «پیوست کاربردی برای تقویم شمسی در تحلیل‌های زمانی» قابل درج در انتهای فصل هوش زمانی خواهد بود:

📌 پیوست: ساخت جدول تاریخ شمسی کامل برای هوش زمانی در Power BI

در این بخش، یک جدول تاریخ شمسی کامل در Power Query معرفی می‌شود که برای سناریوهای هوش زمانی در کسب‌وکارهای ایرانی کاربردی است. این جدول شامل تبدیل تاریخ میلادی به شمسی و ستون‌های کمکی برای تحلیل زمانی و شناسایی تعطیلات رسمی است.

🧠 ویژگی‌های جدول تاریخ شمسی

ستون	توضیح
Gregorian Date	تاریخ میلادی
PersianYear, PersianMonth, PersianDay	اجزای تاریخ شمسی
Persian Month Name	نام فارسی ماه
Persian Weekday Name	نام فارسی روز هفته
Persian Quarter	فصل شمسی (بهار، تابستان، ...)
YearMonth	نمایش ماه به‌صورت 1402-01 برای گروهبندی ساده
IsWeekend	آیا روز جمعه یا شنبه است
IsHoliday	پرچم تعطیلی رسمی (بر اساس لیست ثابت عمومی)

🔧 تعریف تابع تبدیل تاریخ میلادی به شمسی در Power Query

ابتدا تابع GregorianToPersian را در Power Query تعریف کنید. از تب **Home > Advanced Editor** استفاده کرده و کد زیر را وارد نمایید:

```
let
GregorianToPersian = (inputDate as date) as record =>
let
year = Date.Year(inputDate),
month = Date.Month(inputDate),
day = Date.Day(inputDate),
// g_day_no = ... الگوریتم تبدیل جلالی
// کد کامل الگوریتم جلالی در نسخه توسعه‌یافته قرار داده می‌شود
result = [PersianYear=jy, PersianMonth=jm, PersianDay=jd]
in
result
in
GregorianToPersian
```

📌 *نسخه کامل الگوریتم جلالی را می‌توان از منابعی مانند GitHub جلالی یا SQLBI تهیه کرد. یا از کتابخانه‌های آماده در M استفاده نمود.*

ساخت جدول تاریخ شمسی کامل در Power BI

بعد از تعریف تابع، جدول تاریخ شمسی را با کد زیر ایجاد کنید:

```
let
StartDate = #date(2010, 3, 21),
EndDate = #date(2030, 3, 19),
ListDates = List.Dates(StartDate, Duration.Days(EndDate - StartDate) + 1, #duration(1,0,0,0)) ,
BaseTable = Table.FromList(ListDates, Splitter.SplitByNothing() , {"Gregorian Date"}) ,

AddConversion = Table.AddColumn(BaseTable, "Persian", each GregorianToPersian([Gregorian Date])) ,
PersianExpanded = Table.ExpandRecordColumn(AddConversion, "Persian", {"PersianYear", "PersianMonth", "PersianDay"}) ,

AddWeekday = Table.AddColumn(PersianExpanded, "Persian Weekday Name", each Date.DayOfWeekName([Gregorian Date]), type text),
AddMonthName = Table.AddColumn(AddWeekday, "Persian Month Name", each Text.Replace(Text.PadStart(Text.From([PersianMonth]), 2, "0") ,
{
{"01", "فروردین"}, {"02", "اردیبهشت"}, {"03", "خرداد"},
{"04", "تیر"}, {"05", "مرداد"}, {"06", "شهریور"},
{"07", "مهر"}, {"08", "آبان"}, {"09", "آذر"},
{"10", "دی"}, {"11", "بهمن"}, {"12", "اسفند"}
}) , type text),

AddQuarter = Table.AddColumn(AddMonthName, "Persian Quarter", each
" بهار" if [PersianMonth] <= 3 then "
" تابستان" else if [PersianMonth] <= 6 then "
" پاییز" else if [PersianMonth] <= 9 then "
" زمستان" else ", type text),

AddIsWeekend = Table.AddColumn(AddQuarter, "IsWeekend", each Date.DayOfWeek([Gregorian Date]) in {5,6}, type logical),
AddYearMonth = Table.AddColumn(AddIsWeekend, "YearMonth", each Text.From([PersianYear]) & "-" & Text.PadStart(Text.From([PersianMonth]),2,"0") , type text),
```

```
AddHolidayFlag = Table.AddColumn(AddYearMonth, "IsHoliday", each
let
m = [PersianMonth] ,
d = [PersianDay] ,
staticHolidays = {
{1,1}, {1,2}, {1,3}, {1,4}, {1,12}, {1,13},
{11,22}, {3,14}, {3,15}
},
isHoliday = List.Contains(staticHolidays, {m,d})
in
isHoliday, type logical)
in
AddHolidayFlag
```

نکات پایانی و توصیه برای استفاده

- جدول را با نام مثلاً DimDatePersianوارد مدل کنید.
- از ستون‌های YearMonth, Persian Quarter, Persian Weekday Nameبرای بصری‌سازی زمانی استفاده کنید.
- ستون IsHolidayبرای فیلتر کردن یا رنگ‌آمیزی گزارش‌ها در مناسبت‌ها بسیار کاربردی است.
- اگر بخواهید پرچم تعطیلات مذهبی قمری را هم اضافه کنید، پیشنهاد می‌شود از فایل JSONیا API تقویم رسمی استفاده شود.

با استفاده از این جدول، سناریوهای هوش زمانی مانند رشد ماهانه، تحلیل فروش سال‌به‌سال و مقایسه بودجه و عملکرد به‌طور کامل پشتیبانی می‌شود—آن هم با تقویم رسمی ایران.

۶-۴:سناریوهای پیشرفته هوش زمانی با تقویم شمسی

پیاده‌سازی هوش زمانی با تقویم شمسی می‌تواند چالش‌برانگیز باشد، اما با استفاده از جدول تاریخ شمسی و توابع DAX، می‌توان سناریوهای پیچیده‌ای را مدیریت کرد.

۶-۴-۱:محاسبه میانگین متحرک(Moving Average)

میانگین متحرک برای هموار کردن نوسانات داده‌ها و شناسایی روندها استفاده می‌شود. مثلاً میانگین متحرک ۳۰ روزه فروش.

```
Sales 30-Day Moving Average =
```

```
AVERAGEX(
DATESINPERIOD(
DimDate [Date] ,
LASTDATE(DimDate [Date]),
-30,
DAY
) ,
[Total Sales]
)
```

توضیح:

- `LASTDATE(DimDate [Date])`: آخرین تاریخ در زمینه فیلتر فعلی را برمی‌گرداند.
- `DATESINPERIOD(...)`: یک جدول از تاریخ‌ها را برمی‌گرداند که شامل ۳۰ روز قبل از `LASTDATE` تا `LASTDATE` است.
- `AVERAGEX(...)`: میانگین `[Total Sales]` را برای هر روز در این دوره ۳۰ روزه محاسبه می‌کند.

۶-۴-۲: مقایسه با دوره مشابه سال گذشته (Same Period Last Year) با سال مالی شمسی

اگر سال مالی شما با سال تقویمی شمسی (اول فروردین تا ۲۹ اسفند) متفاوت باشد، نیاز به رویکرد متفاوتی دارید. در این حالت، باید از توابع `DATEADD` یا `PARALLELPERIOD` با دقت بیشتری استفاده کنید و یا یک جدول تاریخ سفارشی برای سال مالی خود ایجاد کنید.

فرض کنید سال مالی شما از مهر ماه شروع می‌شود.

```
Sales Same Fiscal Period Last Year =
CALCULATE(
[Total Sales] ,
```

SAMEPERIODLASTYEAR(DimDate [Date]),

-- این بخش نیاز به منطق سفارشی برای سال مالی دارد

-- مثلاً اگر سال مالی از مهر شروع می‌شود، باید فیلترها را بر اساس آن تنظیم کنید.

-- این سناریو پیچیده‌تر است و معمولاً با یک جدول تاریخ مالی سفارشی بهتر مدیریت می‌شود.

)

راه‌حل بهتر برای سال مالی سفارشی:

بهترین راهکار برای سال مالی سفارشی، ایجاد ستون‌های مربوط به سال مالی (مانند `Fiscal Year`, `Fiscal Quarter`, `Fiscal Month`) در جدول تاریخ شمسی خود در Power Query است. سپس می‌توانید از این ستون‌ها برای فیلترکردن و گروه‌بندی استفاده کنید و Measures هوش زمانی را بر اساس آن‌ها بنویسید.

۳-۴-۶-محاسبه موجودی پایان دوره(End-of-Period Inventory):

محاسبه موجودی در پایان هر دوره (مثلاً پایان ماه) یک سناریوی رایج است. این کار نیاز به درک جریان موجودی (ورودی و خروجی) و استفاده از توابع هوش زمانی دارد.

فرض کنید یک جدول `FactInventory` دارید که شامل `Date`, `ProductID`, `QuantityIn`, `QuantityOut`است.

Inventory End of Month =

CALCULATE(

SUM(FactInventory [QuantityIn]) - SUM(FactInventory [QuantityOut]),

LASTDATE(DimDate [Date])

)

توضیح: این Measure مجموع ورودی‌ها و خروجی‌ها را برای آخرین تاریخ در زمینه فیلتر فعلی (که معمولاً آخرین روز ماه است) محاسبه می‌کند تا موجودی پایان ماه را برآورد کند. برای موجودی دقیق، نیاز به یک مدل داده‌ای پیچیده‌تر باSnapshot های موجودی یا محاسبات تجمعی دارید.

۶-۵:بهترین روش‌ها برای هوش زمانی

- همیشه از یک جدول تاریخ **(Date Table)** استفاده کنید: هرگز به قابلیت Power BI `Auto Date/Time` تکیه نکنید، به خصوص برای تقویم شمسی. یک جدول تاریخ کامل و علامت‌گذاری شده، ضروری است.
- جدول تاریخ را به‌درستی علامت‌گذاری کنید: مطمئن شوید که جدول تاریخ شما به‌عنوان یک جدول تاریخ علامت‌گذاری شده است و ستون تاریخ اصلی آن مشخص شده است.
- روابط صحیح را ایجاد کنید: جدول تاریخ باید به ستون‌های تاریخ در جداول حقیقت شما با رابطه یک به چند متصل باشد.
- از توابع هوش زمانی **DAX** استفاده کنید: این توابع بهینه‌سازی شده‌اند و استفاده از آن‌ها نسبت به نوشتن منطق فیلتر پیچیده، کارآمدتر و خواناتر است.
- سال مالی را در نظر بگیرید: اگر سال مالی شما با سال تقویمی متفاوت است، ستون‌های مربوط به سال مالی را در جدول تاریخ خود ایجاد کنید و Measures هوش زمانی را بر اساس آن‌ها تنظیم کنید.
- عملکرد را پایش کنید: در مدل‌های بزرگ، محاسبات هوش زمانی می‌توانند بر عملکرد تأثیر بگذارند. از ابزارهایی مانند DAX Studioبرای پایش و بهینه‌سازی آن‌ها استفاده کنید.

۶-۶:خلاصه فصل ۶

در این فصل، به طور جامع به مفهوم هوش زمانی (Time Intelligence) در DAX پرداختیم و اهمیت آن را برای تحلیل کسب‌وکار درک کردیم. چالش اصلی کار با تقویم شمسی را مطرح کردیم و نحوه ساخت یک جدول تاریخ شمسی کامل و قابل نگهداری را با استفاده از Power Query توضیح دادیم. سپس، با توابع کلیدی هوش زمانی DAX مانند `TOTALYTD`, `SAMEPERIODLASTYEAR`, `DATEADD`, `TOTALMTD`, `TOTALQTD`و توابع شروع/پایان دوره آشنا شدیم و نحوه استفاده از آن‌ها را برای محاسبه مجموع‌های تجمعی، مقایسه با دوره‌های گذشته و محاسبه رشد بررسی کردیم. در نهایت، به سناریوهای پیشرفته‌تر و بهترین روش‌ها برای پیاده‌سازی هوش زمانی پرداختیم.

با تسلط بر مفاهیم و توابع این فصل، شما قادر خواهید بود تا تحلیل‌های زمانی قدرتمندی را در Power BI انجام دهید و بینش‌های ارزشمندی را از داده‌های خود استخراج کنید، حتی با پیچیدگی‌های تقویم شمسی. در فصل بعدی، به سناریوهای تحلیلی پیشرفته‌تر خواهیم پرداخت که فراتر از تحلیل‌های زمانی هستند.

فصل ۷: سناریوهای پیشرفته تحلیلی

در فصول گذشته، با اصول مدل‌سازی داده، مفاهیم پایه DAX و هوش زمانی آشنا شدیم. اکنون زمان آن رسیده که این دانش را برای حل سناریوهای تحلیلی پیچیده‌تر و استخراج بینش‌های عمیق‌تر از داده‌ها به کار بگیریم. این فصل به بررسی چندین سناریوی پیشرفته می‌پردازد که اغلب در دنیای واقعی کسب‌وکار با آن‌ها مواجه می‌شوید و نحوه پیاده‌سازی آن‌ها را با DAX نشان می‌دهد.

۷-۱: تحلیل سبد خرید (Market Basket Analysis)

تحلیل سبد خرید، تکنیکی است که برای شناسایی اقلامی که اغلب با هم خریداری می‌شوند، استفاده می‌شود. این تحلیل به خرده‌فروشان کمک می‌کند تا استراتژی‌های چیدمان فروشگاه، پیشنهادهای محصول و کمپین‌های بازاریابی را بهینه کنند. هدف اصلی، پاسخ به سوالاتی مانند

"اگر مشتری X را بخرد، احتمال دارد Y را هم بخرد؟" است.

برای پیاده‌سازی تحلیل سبد خرید در DAX، معمولاً از توابع تکرارکننده و توابع جدول استفاده می‌شود. یک رویکرد رایج، شناسایی اقلامی است که همراه با یک محصول خاص خریداری می‌شوند.

مثال: محصولات خریداری شده همراه با "لپ‌تاپ"

فرض کنید می‌خواهید بدانید مشتریانی که "لپ‌تاپ" خریده‌اند، چه محصولات دیگری را نیز خریداری کرده‌اند. برای این کار، نیاز به یک جدول `FactSales` دارید که شامل `OrderID` (شناسه سفارش) و `ProductID` (شناسه محصول) باشد.

```
Products Bought With Laptop =
CALCULATE(
    DISTINCTCOUNT(FactSales [ProductID]),
    FILTER(
        ALL(FactSales [OrderID]),
        FactSales [OrderID] IN
        CALCULATETABLE(
            VALUES(FactSales [OrderID]),
```

```
" = [ProductName] FactSales                لپ‌تاپ"

)

) ,

" <> [ProductName] FactSales   لپ‌تاپ"

)
```

توضیح:

1. `CALCULATETABLE(VALUES(FactSales [OrderID]), FactSales [ProductName] = "لپ‌تاپ"):` این بخش ابتدا تمام `OrderID` هایی را پیدا می‌کند که شامل محصول "لپ‌تاپ" هستند. این یک جدول موقت از `OrderID` ها را برمی‌گرداند.
2. `FILTER(ALL(FactSales[OrderID]), FactSales[OrderID] IN ...):` سپس، این بخش جدول `FactSales` را فیلتر می‌کند تا فقط شامل ردیف‌هایی باشد که `OrderID` آن‌ها در لیست `OrderID` های حاوی "لپ‌تاپ" قرار دارد `ALL(FactSales[OrderID])` تضمین می‌کند که فیلترهای موجود بر روی `OrderID` نادیده گرفته شوند تا تمام سفارشات مرتبط با لپ‌تاپ در نظر گرفته شوند .
3. `FactSales [ProductName] <> "لپ‌تاپ":` در نهایت، این فیلتر اضافه می‌شود تا خود "لپ‌تاپ" از شمارش نهایی حذف شود، زیرا ما به دنبال محصولات دیگری هستیم که همراه با آن خریداری شده‌اند.
4. `DISTINCTCOUNT(FactSales [ProductID]):` تعداد محصولات متمایز (به جز لپ‌تاپ) را در این سفارشات مشترک می‌شمارد.

این Measure را می‌توانید در یک جدول یا ماتریس در کنار `ProductName` از جدول (`DimProduct`) قرار دهید تا ببینید هر محصول با چه محصولات دیگری خریداری شده است.

۷-۲تحلیل(ABC Analysis):

تحلیل ABC یک روش طبقه‌بندی موجودی است که اقلام را بر اساس اهمیت آن‌ها (معمولاً بر اساس ارزش فروش یا سود) به سه دسته A، B و C تقسیم می‌کند:

- دسته **A:** اقلام بسیار مهم که درصد کمی از اقلام را تشکیل می‌دهند اما درصد زیادی از ارزش را دارند (مثلاً ۲۰٪ اقلام، ۸۰٪ ارزش).

- دسته **B:** اقلام با اهمیت متوسط.
- دسته **C:** اقلام با اهمیت کم که درصد زیادی از اقلام را تشکیل می‌دهند اما درصد کمی از ارزش را دارند.

این تحلیل به شرکت‌ها کمک می‌کند تا منابع خود را به طور مؤثرتری مدیریت کنند.

پیاده‌سازی تحلیل **ABC** بر اساس فروش**:**

برای پیاده‌سازی تحلیل ABC، نیاز به محاسبه مجموع فروش برای هر محصول و سپس رتبه‌بندی و طبقه‌بندی آن‌ها داریم.

1. محاسبه مجموع فروش برای هر محصول**:(Measure)**

```
Product Sales = CALCULATE([Total Sales] , DimProduct)
```

توضیح **:**این Measure مجموع فروش را برای هر محصول در زمینه فیلتر فعلی (مثلاً در یک جدول که `ProductName` در آن قرار دارد) محاسبه می‌کند.

2. محاسبه درصد تجمعی فروش **:(Cumulative Percentage of Sales)** این یک Measure پیچیده‌تر است که نیاز به رتبه‌بندی و جمع تجمعی دارد.

```
Cumulative Sales Percentage =
VAR CurrentProductSales = [Product Sales]
VAR AllProducts = ALL(DimProduct [ProductName])
VAR TotalSalesAllProducts = CALCULATE([Total Sales] , AllProducts)
RETURN
IF(
HASONEVALUE(DimProduct [ProductName]),
DIVIDE(
SUMX(
FILTER(
```

AllProducts,

[Product Sales] >= CurrentProductSales

) ,

[Product Sales]

) ,

TotalSalesAllProducts

) ,

BLANK()

)

توضیح:

- `CurrentProductSales`: فروش محصول فعلی را ذخیره می‌کند.
- `AllProducts`: یک جدول از تمام نام‌های محصول را برمی‌گرداند.
- `TotalSalesAllProducts`: مجموع فروش کل را برای تمام محصولات محاسبه می‌کند.
- `FILTER(AllProducts, [Product Sales] >= CurrentProductSales)`: این بخش تمام محصولاتی را فیلتر می‌کند که فروش آن‌ها بیشتر یا مساوی فروش محصول فعلی است.
- `SUMX(...)`: مجموع فروش این محصولات فیلتر شده را محاسبه می‌کند (یعنی مجموع تجمعی فروش از بالا به پایین).
- `DIVIDE(...)`: این مجموع تجمعی را بر کل فروش تقسیم می‌کند تا درصد تجمعی را به دست آورد.

3. طبقه‌بندی **ABC (Calculated Column** در **DimProduct):** اکنون می‌توانید یک ستون محاسباتی در جدول `DimProduct` ایجاد کنید تا محصولات را به دسته‌های A، B یا C طبقه‌بندی کنید. (این کار را می‌توان با Measure هم انجام داد، اما ستون محاسباتی برای طبقه‌بندی ثابت محصول مناسب‌تر است.)

ABC Category =

```
VAR CurrentProductSales = CALCULATE([Total Sales] , FactSales)
VAR AllProducts = ALL(DimProduct [ProductName])
VAR TotalSalesAllProducts = CALCULATE([Total Sales] , AllProducts)
VAR CumulativeSalesRank =
DIVIDE(
SUMX(
FILTER(
AllProducts,
CALCULATE([Total Sales] , DimProduct) >= CurrentProductSales
) ,
CALCULATE([Total Sales] , DimProduct)
) ,
TotalSalesAllProducts
)
RETURN
SWITCH(
TRUE() ,
CumulativeSalesRank <= 0.8, "A",
CumulativeSalesRank <= 0.95, "B",
"C"
```

)

توضیح **:**این ستون محاسباتی، درصد تجمعی فروش را برای هر محصول محاسبه می‌کند و سپس بر اساس آستانه‌های مشخص (مثلاً ۸۰٪ برای A، ۹۵٪ برای B)، محصول را طبقه‌بندی می‌کند. توجه داشته باشید که `CALCULATE([Total Sales] , DimProduct)` در اینجا برای محاسبه فروش هر محصول در زمینه ردیف استفاده شده است.

۷-۳: تحلیل RFM (Recency, Frequency, Monetary)

تحلیل RFM یک تکنیک بازاریابی است که برای تقسیم‌بندی مشتریان بر اساس سه معیار استفاده می‌شود:

- **Recency (** تازگی **:(**آخرین باری که مشتری خرید کرده است (هرچه تازه‌تر، بهتر.(
- **Frequency (** تکرار **:(**تعداد دفعاتی که مشتری خرید کرده است (هرچه بیشتر، بهتر.(
- **Monetary (** ارزش پولی **:(**مجموع پولی که مشتری خرج کرده است (هرچه بیشتر، بهتر.(

این تحلیل به شرکت‌ها کمک می‌کند تا مشتریان با ارزش بالا را شناسایی کرده و استراتژی‌های بازاریابی هدفمندتری را اجرا کنند.

پیاده‌سازی تحلیل **RFM** در **:DAX**

برای پیاده‌سازی RFM، نیاز به محاسبه سه Measure اصلی برای هر مشتری داریم:

1. **Recency (** تازگی **:(**تعداد روز از آخرین خرید مشتری تا تاریخ امروز (یا آخرین تاریخ در داده‌ها.(

```
Customer Recency =

VAR LastPurchaseDate = MAXX(FILTER(FactSales, FactSales [CustomerID] =
VALUES(DimCustomer [CustomerID])) , FactSales [OrderDate])

VAR TodayDate = TODAY()

RETURN

IF(

NOT ISBLANK(LastPurchaseDate),
```

```
TodayDate - LastPurchaseDate,
BLANK()
)
```

توضیح **:**این Measure آخرین تاریخ خرید هر مشتری را پیدا می‌کند و سپس تعداد روزهای گذشته از آن تاریخ تا امروز را محاسبه می‌کند.

2. **Frequency (** **):**تکرار تعداد سفارشات متمایز هر مشتری.

```
Customer Frequency =
CALCULATE(
DISTINCTCOUNT(FactSales [OrderID]),
FILTER(FactSales, FactSales [CustomerID] = VALUES(DimCustomer [CustomerID]))
)
```

توضیح **:**این Measure تعداد`OrderID` های متمایز را برای هر مشتری می‌شمارد.

3. **Monetary (** **):**ارزش پولی مجموع فروش هر مشتری.

```
Customer Monetary =
CALCULATE(
[Total Sales] ,
FILTER(FactSales, FactSales [CustomerID] = VALUES(DimCustomer [CustomerID]))
)
```

توضیح **:**این Measure مجموع فروش را برای هر مشتری محاسبه می‌کند.

نمره‌دهی **RFM** و بخش‌بندی مشتریان**:**

پس از محاسبه این سه ،Measure می‌توانید به هر مشتری بر اساس هر معیار یک نمره (مثلاً از ۱ تا ۵) اختصاص دهید و سپس مشتریان را به بخش‌های مختلف (مثلاً مشتریان وفادار، مشتریان در معرض خطر، مشتریان جدید) تقسیم‌بندی کنید. این نمره‌دهی معمولاً با استفاده از ستون‌های محاسباتی یا Measures پیچیده‌تر انجام می‌شود که داده‌ها را به کوانتیل‌ها (Quantiles) تقسیم می‌کنند.

مثال: نمره‌دهی **Recency (** به‌عنوان ستون محاسباتی در**:DimCustomer)**

```
Recency Score =
VAR CurrentRecency = [Customer Recency]
VAR AllRecencies = ALL(DimCustomer [Customer Recency])
RETURN
CALCULATE(
COUNTROWS(FILTER(AllRecencies, DimCustomer [Customer Recency] <= CurrentRecency)) ,
ALL(DimCustomer)
)
```

توضیح **:**این یک رویکرد ساده برای رتبه‌بندی است. برای نمره‌دهی دقیق‌تر (مثلاً به کوانتیل‌ها)، نیاز به توابع پیشرفته‌تر مانند RANKXو تقسیم‌بندی بر اساس توزیع داده‌ها دارید. پس از نمره‌دهی برای هر سه معیار، می‌توانید آن‌ها را ترکیب کرده و بخش‌بندی نهایی مشتریان را انجام دهید.

۷-۴:تخصیص مقادیر(Allocation)

تخصیص مقادیر به معنای توزیع یک مقدار کلی (مثلاً هزینه‌های سربار، بودجه بازاریابی) به زیرمجموعه‌های مختلف (مثلاً محصولات، مناطق، مشتریان) بر اساس یک معیار مشخص است. این سناریو در حسابداری و تحلیل مالی بسیار رایج است.

مثال: تخصیص هزینه‌های بازاریابی بر اساس سهم فروش محصول

فرض کنید مجموع هزینه‌های بازاریابی برای یک دوره مشخص را دارید و می‌خواهید این هزینه را به محصولات مختلف بر اساس سهم هر محصول از کل فروش تخصیص دهید.

1. **Measure**برای کل هزینه‌های بازاریابی (فرض کنید از یک جدول دیگر می‌آید**:(**

```
Total Marketing Cost = SUM(FactMarketing [CostAmount])
```

2. **Measure**برای سهم فروش هر محصول**:**

```
Product Sales Share =
DIVIDE(
[Total Sales] ,
CALCULATE([Total Sales] , ALL(DimProduct))
)
```

3. **Measure**برای هزینه بازاریابی تخصیص یافته به هر محصول**:**

```
Allocated Marketing Cost = [Total Marketing Cost] * [Product Sales Share]
```

توضیح **:**این Measure برای هر محصول، سهم آن محصول از کل فروش را محاسبه می‌کند و سپس این سهم را در کل هزینه‌های بازاریابی ضرب می‌کند تا هزینه بازاریابی تخصیص یافته به آن محصول را به دست آورد.

۷-۵:تحلیل Cohort (Cohort Analysis)

تحلیل) Cohort گروه هم‌سن) به بررسی رفتار گروه‌هایی از کاربران یا مشتریان در طول زمان می‌پردازد که یک ویژگی مشترک (مثلاً تاریخ ثبت‌نام، تاریخ اولین خرید) دارند. این تحلیل برای درک ماندگاری مشتری ،(Customer Retention) ارزش طول عمر مشتری (Customer Lifetime Value) و تأثیر تغییرات محصول یا بازاریابی بسیار مفید است.

پیاده‌سازی تحلیل **Cohort** بر اساس ماه اولین خرید**:**

فرض کنید می‌خواهید ماندگاری مشتریان را بر اساس ماه اولین خرید آن‌ها بررسی کنید.

1. پیدا کردن ماه اولین خرید برای هر مشتری **Calculated Column)** در**:(DimCustomer**

```
First Purchase Month =
CALCULATE(
MIN(FactSales [OrderDate]),
ALLEXCEPT(FactSales, FactSales [CustomerID])
)
```

توضیح **:**این ستون محاسباتی، اولین تاریخ خرید هر مشتری را پیدا می‌کند. سپس می‌توانید از این تاریخ برای استخراج ماه و سال اولین خرید استفاده کنید.

2. ایجاد یک جدول**:(Calculated Table) Cohort**

```
Cohort Table =
SUMMARIZECOLUMNS(
DimCustomer [First Purchase Month] ,
DimDate [Date] ,
"Customers", DISTINCTCOUNT(FactSales [CustomerID])
)
```

توضیح **:**این یک رویکرد ساده است. برای تحلیل Cohort واقعی، نیاز به یک ماتریس دارید که در محور X، ماه‌های بعدی از اولین خرید را نشان دهد و در محور Y، ماه Cohort را. این کار با Measures پیچیده‌تر و استفاده از توابع هوش زمانی انجام می‌شود.

Measureبرای تعداد مشتریان فعال در هر ماه**:Cohort**

```
Active Customers in Cohort =
VAR CurrentCohortMonth = VALUES(DimCustomer [First Purchase Month])
```

```
VAR CurrentMonth = VALUES(DimDate [Date])
RETURN
CALCULATE(
DISTINCTCOUNT(FactSales [CustomerID]),
FILTER(
ALL(FactSales),
FactSales [OrderDate] >= CurrentCohortMonth &&
FactSales [OrderDate] < EDATE(CurrentCohortMonth, 1) &&
FactSales [CustomerID] IN
CALCULATETABLE(
VALUES(FactSales [CustomerID]),
FILTER(
ALL(FactSales),
FactSales [OrderDate] >= CurrentCohortMonth &&
FactSales [OrderDate] < EDATE(CurrentCohortMonth, 1)
)
)
)
)
```

توضیح :این Measure بسیار پیچیده است و هدف آن شمارش مشتریانی است که در یک ماه Cohort خاص، در ماه فعلی گزارش (CurrentMonth) فعال بوده‌اند. این کار با فیلترکردن مشتریانی که اولین خرید آن‌ها در

CurrentCohortMonthبوده و سپس بررسی اینکه آیا آن‌ها در CurrentMonth خریدی داشته‌اند، انجام می‌شود. این نوع تحلیل معمولاً با استفاده از یک ماتریس بصری‌سازی می‌شود که در آن ردیف‌هاCohort ها و ستون‌ها ماه‌های بعدی از اولین خرید هستند.

۶-۷تحلیل(What-If Analysis) What-If:

تحلیل What-If به شما امکان می‌دهد تا تأثیر تغییر یک یا چند پارامتر را بر روی نتایج کسب‌وکار شبیه‌سازی کنید. این تحلیل برای برنامه‌ریزی، بودجه‌بندی و تصمیم‌گیری استراتژیک بسیار مفید است.

پیاده‌سازی تحلیل **What-If** با پارامترها**:**

Power BIدارای قابلیت What-if parameter است که به شما امکان می‌دهد تا به‌راحتی یک پارامتر را ایجاد کرده و از آن در Measures خود استفاده کنید.

مثال: شبیه‌سازی تأثیر تغییر نرخ تخفیف بر فروش

1. ایجاد پارامتر **What-If:**

- در تب Modeling، بر روی New parameter -> Numeric range کلیک کنید.
- نام پارامتر را Discount Percentage بگذارید.
- Data typeرا Decimal number انتخاب کنید.
- Minimumرا 0، Maximum را 0.2 برای ۲۰٪) و Increment را 0.01 برای ۱٪) تنظیم کنید.
- Add slicer to this pageرا انتخاب کنید و Create را بزنید.

این کار یک جدول جدید به نام Discount Percentage و یک Measure به نام Discount Percentage Valueایجاد می‌کند:

```
Discount Percentage Value = GENERATESERIES(0, 0.2, 0.01)
```

2. استفاده از پارامتر در **Measure** فروش **:**فرض کنید می‌خواهید فروش را با اعمال این نرخ تخفیف شبیه‌سازی کنید.

```
Simulated Sales =
```

```
[Total Sales] * (1 - 'Discount Percentage' [Discount Percentage Value])
```

توضیح: این Measure مجموع فروش را با اعمال نرخ تخفیف انتخاب شده توسط کاربر از اسلایسر `Discount Percentage` محاسبه می‌کند. می‌توانید این Measure را در یک نمودار یا جدول قرار دهید تا تأثیر تغییر نرخ تخفیف را مشاهده کنید.

۷-۷-خلاصه فصل ۷:

در این فصل، به بررسی چندین سناریوی تحلیلی پیشرفته پرداختیم و نحوه پیاده‌سازی آن‌ها را با DAX نشان دادیم. از تحلیل سبد خرید (Market Basket Analysis) برای شناسایی اقلامی که با هم خریداری می‌شوند، تا تحلیل ABC برای طبقه‌بندی موجودی بر اساس اهمیت، و تحلیل RFM برای بخش‌بندی مشتریان بر اساس رفتار خرید آن‌ها. همچنین، به تخصیص مقادیر (Allocation)و تحلیل Cohort برای درک ماندگاری مشتری پرداختیم. در نهایت، نحوه انجام تحلیل What-If با استفاده از پارامترها را بررسی کردیم.

این سناریوها نشان می‌دهند که چگونه با ترکیب توابع مختلف DAX و درک عمیق مدل داده، می‌توانید به سؤالات پیچیده کسب‌وکار پاسخ دهید و بینش‌های ارزشمندی را از داده‌های خود استخراج کنید. در فصل بعدی، به یکی از جنبه‌های حیاتی در کار با DAX، یعنی بهینه‌سازی عملکرد (Performance Tuning) خواهیم پرداخت.

فصل ۸: بهینه‌سازی عملکرد(DAX Performance Tuning)

هنگامی که مدل‌های داده و فرمول‌های DAX شما پیچیده‌تر و بزرگ‌تر می‌شوند، عملکرد می‌تواند به یک چالش تبدیل شود. فرمول‌های DAX ناکارآمد می‌توانند منجر به کندی گزارش‌ها، زمان بارگذاری طولانی و تجربه کاربری نامطلوب شوند. در این فصل، به بررسی تکنیک‌ها و ابزارهایی می‌پردازیم که به شما کمک می‌کنند تا عملکرد مدل‌های DAX خود را بهینه‌سازی کنید و گزارش‌های سریع‌تر و پاسخگوتر ایجاد کنید.

۸-۱-درک موتور VertiPaq و فرمولاسیون DAX:

برای بهینه‌سازی DAX، درک نحوه عملکرد Power BI در پشت صحنه ضروری است .Power BI از یک موتور تحلیلی در حافظه به نام) **VertiPaq** که بخشی از SQL Server Analysis Services Tabular است) استفاده می‌کند . VertiPaqیک موتور ستونی (Columnar Database) است که داده‌ها را به‌صورت ستونی ذخیره می‌کند و از فشرده‌سازی پیشرفته برای کاهش حجم داده‌ها در حافظه استفاده می‌کند. این طراحی ستونی، برای عملیات تجمیعی و فیلترینگ که در تحلیل‌های BI رایج هستند، بسیار بهینه است.

نحوه عملکرد **DAX:**

هنگامی که یک فرمول DAX را می‌نویسید، این فرمول توسط موتور فرمولاسیون DAX (Formula Engine) پردازش می‌شود. موتور فرمولاسیون، درخواست را به موتور ذخیره‌سازی VertiPaq (Storage Engine) ارسال می‌کند. موتور ذخیره‌سازی، داده‌های مورد نیاز را از حافظه (یا دیسک در صورت نیاز) بازیابی کرده و به موتور فرمولاسیون برمی‌گرداند. سپس موتور فرمولاسیون، محاسبات نهایی را انجام داده و نتیجه را برمی‌گرداند.

نکات کلیدی برای عملکرد**:**

- کاهش حجم داده‌ها **:**هرچه داده‌های کمتری در مدل شما وجود داشته باشد VertiPaq ،می‌تواند آن‌ها را سریع‌تر فشرده و پردازش کند.
- بهینه‌سازی مدل داده **:**یک مدل داده ستاره‌ای (Star Schema) بهینه، برای عملکرد VertiPaq ایده‌آل است.
- نوشتن **DAX** کارآمد **:**فرمول‌هایی که نیاز به پردازش کمتری توسط موتور فرمولاسیون دارند و بیشتر کار را به موتور ذخیره‌سازی واگذار می‌کنند، عملکرد بهتری خواهند داشت.

۲-۸بهترین روش‌ها برای مدل‌سازی داده و عملکرد:

عملکرد DAX به شدت به طراحی مدل داده شما بستگی دارد. یک مدل داده ضعیف می‌تواند حتی بهترین فرمول‌های DAX را کند کند.

- استفاده از **Star Schema:** این مهم‌ترین بهترین روش است. جداول حقیقت (Fact Tables) باید شامل مقادیر عددی و کلیدهای خارجی باشند، درحالی‌که جداول ابعاد (Dimension Tables) باید شامل ویژگی‌های توصیفی باشند. این ساختار، فیلترینگ و تجمیع را بهینه می‌کند.
- حذف ستون‌های غیرضروری **:**هر ستون در مدل شما، حافظه را اشغال می‌کند و زمان پردازش را افزایش می‌دهد. فقط ستون‌هایی را نگه دارید که برای تحلیل یا مدل‌سازی شما ضروری هستند.
- انتخاب نوع داده مناسب **:**از دقیق‌ترین نوع داده ممکن برای هر ستون استفاده کنید (مثلا `Whole Number` به‌جای `Decimal Number` اگر فقط اعداد صحیح نیاز دارید). این کار به فشرده‌سازی بهتر کمک می‌کند.
- مدیریت روابط **:**
  - روابط یک به چند **(One-to-Many)** را ترجیح دهید **:**این روابط برای عملکرد بهینه هستند.
  - از روابط چند به چند **(Many-to-Many)** با احتیاط استفاده کنید **:**این روابط می‌توانند پیچیده باشند و بر عملکرد تأثیر بگذارند. در صورت امکان، آن‌ها را با یک جدول پل (Bridge Table) به روابط یک به چند تبدیل کنید.

- روابط دوجهته **(Bi-directional)** را به حداقل برسانید **:**روابط دوجهته می‌توانند ابهام ایجاد کرده و بر عملکرد تأثیر بگذارند. فقط در صورت لزوم از آن‌ها استفاده کنید و سعی کنید آن‌ها را به روابط تک‌جهته تبدیل کنید.
- عدم استفاده از ستون‌های محاسباتی برای تجمیع **:**ستون‌های محاسباتی در زمان رفرش داده‌ها محاسبه می‌شوند و فضای حافظه را اشغال می‌کنند. برای محاسبات تجمیعی که به زمینه فیلتر پاسخ می‌دهند (مانند مجموع فروش)، همیشه از Measuresاستفاده کنید.
- بهینه‌سازی **Power Query:**
  - حذف مراحل غیرضروری **:**هر مرحله در Power Query زمان پردازش را افزایش می‌دهد.
  - فیلترکردن و حذف ستون‌ها در مراحل اولیه **:**داده‌ها را در اسرع وقت فیلتر و ستون‌های غیرضروری را حذف کنید تا حجم داده‌های بارگذاری شده به مدل کاهش یابد.
  - استفاده از Query Folding **:Query Folding** به Power Query اجازه می‌دهد تا عملیات تبدیل را به سیستم منبع (مثلاً پایگاه داده SQL) منتقل کند، که می‌تواند عملکرد را به شدت بهبود بخشد.

۳-۸:بهترین روش‌ها برای نوشتن DAX کارآمد

نحوه نوشتن فرمول‌های DAX تأثیر زیادی بر عملکرد دارد. در اینجا چند بهترین روش آورده شده است:

- استفاده از **Measures** به‌جای ستون‌های محاسباتی برای تجمیع **:**همانطور که قبلاً ذکر شد Measures ،در زمان نیاز محاسبه می‌شوند و به زمینه فیلتر پاسخ می‌دهند، درحالی‌که ستون‌های محاسباتی در زمان رفرش داده‌ها محاسبه و ذخیره می‌شوند.
- به حداقل رساندن استفاده از توابع **Iterator (X-Functions)** بر روی جداول بزرگ **:**توابع Iterator مانند `SUMX`, `AVERAGEX`, `FILTER` و `ADDCOLUMNS` برای هر ردیف از جدول ورودی تکرار می‌شوند. اگر جدول ورودی بسیار بزرگ باشد، این می‌تواند منجر به عملکرد کند شود. در صورت امکان، از توابع تجمیعی ساده‌تر یا فیلترهای مستقیم در `CALCULATE` استفاده کنید.
  - مثال **:**به‌جای `SUMX(FactSales, FactSales [SalesAmount])`، از `SUM(FactSales [SalesAmount])`استفاده کنید.
  - مثال **:**به‌جای `CALCULATE([Total Sales] , FILTER(FactSales, FactSales [SalesAmount] > 100))`، از `CALCULATE([Total Sales] , FactSales [SalesAmount] > 100)`استفاده کنید.
- استفاده از متغیرها **:(Variables)** متغیرها (با `VAR` و `RETURN`) نه‌تنها خوانایی فرمول را بهبود می‌بخشند، بلکه عملکرد را نیز افزایش می‌دهند، زیرا یک عبارت فقط یک بار محاسبه می‌شود و نتیجه آن در متغیر ذخیره می‌شود، حتی اگر چندین بار در فرمول استفاده شود.

- اجتناب از **IF** های تو در تو **(Nested IFs)** پیچیده `IF` **:**های تو در تو می‌توانند خوانایی را کاهش داده و بر عملکرد تأثیر بگذارند. در صورت امکان، از `SWITCH(TRUE() , ...)` استفاده کنید که خواناتر و گاهی اوقات کارآمدتر است.
- بهینه‌سازی `CALCULATE` **CALCULATE:** قدرتمندترین تابع در DAX است، اما استفاده نادرست از آن می‌تواند منجر به مشکلات عملکردی شود. سعی کنید فیلترها را تا حد امکان ساده نگه دارید.
- استفاده از **ALL** به‌جای **ALLSELECTED** در صورت امکان `ALL` **:**ساده‌تر است و فیلترها را به طور کامل حذف می‌کند، درحالی‌که `ALLSELECTED` نیاز به پردازش بیشتری برای حفظ فیلترهای بصری‌سازی دارد.
- پرهیز از روابط دوجهته در **Measures:** اگرچه روابط دوجهته در مدل داده ممکن است ضروری باشند، اما سعی کنید در Measures خود از آن‌ها اجتناب کنید و به‌جای آن از `USERELATIONSHIP` یا `CROSSFILTER` به‌صورت کنترل شده استفاده کنید.
- بررسی فرمول‌های پیچیده **:**فرمول‌هایی که شامل چندین `CALCULATE` تو در تو، توابع Iterator بر روی جداول بزرگ یا توابع رابطه پیچیده هستند، باید به دقت بررسی و بهینه‌سازی شوند.

۸-۴:ابزارهای بهینه‌سازی عملکردDAX

چندین ابزار قدرتمند وجود دارد که به شما کمک می‌کنند تا فرمول‌های DAX خود را تحلیل، پایش و بهینه‌سازی کنید:

۸-۴-۱: DAX Studio

DAX Studioیک ابزار خارجی رایگان و ضروری برای هر توسعه‌دهنده Power BI است. این ابزار به شما امکان می‌دهد تا:

- کوئری‌های **DAX** را اجرا کنید **:**می‌توانید کوئری‌های DAX را مستقیماً بر روی مدل Power BI خود اجرا کنید و نتایج را مشاهده کنید.
- عملکرد کوئری‌ها را پایش کنید **:**با استفاده از قابلیت `Server Timings`، می‌توانید زمان اجرای هر بخش از کوئری DAX خود را مشاهده کنید و گلوگاه‌های عملکردی را شناسایی کنید.
- متریک‌های **VertiPaq** را بررسی کنید DAX Studio **:**به شما امکان می‌دهد تا اطلاعات دقیقی در مورد نحوه فشرده‌سازی و ذخیره‌سازی داده‌ها توسط) VertiPaq مانند حجم ستون‌ها، کاردینالیتی، فشرده‌سازی) را مشاهده کنید. این اطلاعات برای شناسایی ستون‌هایی که فضای زیادی را اشغال می‌کنند یا به خوبی فشرده نمی‌شوند، بسیار مفید است.
- فرمول‌ها را فرمت کنید DAX Studio **:**دارای یک فرمت‌کننده داخلی است که فرمول‌های DAX شما را خواناتر می‌کند.
- مدل را مرور کنید **:**می‌توانید ساختار مدل داده خود را مشاهده کنید.

نحوه استفاده از **DAX Studio:**

1. نصب DAX Studio: **DAX Studio** را از وب‌سایت رسمی آن (daxstudio.org) دانلود و نصب کنید.
2. اتصال به **Power BI Desktop**: پس از باز کردن **Power BI Desktop** و مدل خود، **DAX Studio** را باز کنید. **DAX Studio** به طور خودکار مدل **Power BI Desktop** باز شده را شناسایی می‌کند و می‌توانید به آن متصل شوید.
3. اجرای کوئری و پایش عملکرد: یک کوئری DAX را در پنجره `Query` بنویسید. سپس `Server Timings` را فعال کنید و کوئری را اجرا کنید. نتایج و زمان‌بندی سرور را در تب‌های مربوطه مشاهده خواهید کرد.

۸-۴-۲: Performance Analyzer در Power BI Desktop

Performance Analyzer یک ابزار داخلی در Power BI Desktop است که به شما امکان می‌دهد تا عملکرد عناصر بصری گزارش خود را تحلیل کنید. این ابزار نشان می‌دهد که هر بصری‌سازی چقدر زمان برای رفرش، کوئری DAX و رندر کردن نیاز دارد.

نحوه استفاده از **Performance Analyzer**:

1. فعال کردن **Performance Analyzer**: در تب `View` در Power BI Desktop، بر روی `Performance Analyzer` کلیک کنید.
2. شروع ضبط: بر روی `Start recording` کلیک کنید.
3. تعامل با گزارش: با گزارش خود تعامل کنید (مثلاً اسلایسرها را تغییر دهید، صفحات را عوض کنید). Performance Analyzer زمان اجرای هر بصری‌سازی را ضبط می‌کند.
4. بررسی نتایج: پس از اتمام تعامل، بر روی `Stop` کلیک کنید. می‌توانید جزئیات عملکرد هر بصری‌سازی را مشاهده کنید، از جمله کوئری DAX تولید شده توسط آن. می‌توانید کوئری DAX را کپی کرده و در DAX Studio برای تحلیل عمیق‌تر اجرا کنید.

۸-۴-۳: Tabular Editor

Tabular Editor یک ابزار خارجی قدرتمند است که به شما امکان می‌دهد تا مدل‌های Tabular (مانند مدل‌های Power BI) را به‌صورت برنامه‌نویسی ویرایش کنید. این ابزار برای کارهایی مانند:

- ایجاد **Measures** و ستون‌های محاسباتی به‌صورت انبوه: می‌توانید چندین Measure را به‌سرعت ایجاد یا ویرایش کنید.
- اعمال بهترین روش‌ها: Tabular Editor دارای قابلیت `Best Practice Analyzer` است که مدل شما را برای شناسایی مشکلات عملکردی یا عدم رعایت بهترین روش‌ها اسکن می‌کند.
- مدیریت پوشه‌ها و سلسله مراتب: سازماندهی Measures و ستون‌ها در پوشه‌ها را آسان‌تر می‌کند.

نحوه استفاده از **Tabular Editor:**

1. نصب Tabular Editor **Tabular Editor:** را از وب‌سایت رسمی آن (tabulareditor.com) دانلود و نصب کنید.
2. اتصال به **Power BI Desktop:** پس از باز کردن Power BI Desktop و مدل خود Tabular Editor ، را باز کنید. می‌توانید به مدل Power BI Desktop باز شده متصل شوید.
3. استفاده از **Best Practice Analyzer:** در Tabular Editor، به `Tools` -> `Best Practice Analyzer`بروید و مدل خود را اسکن کنید. این ابزار پیشنهاداتی برای بهبود عملکرد و رعایت بهترین روش‌ها ارائه می‌دهد.

۸-۵:سناریوهای رایج و راه‌حل‌های بهینه‌سازی

- مشکل **:**گزارش کند می‌شود وقتی اسلایسرها یا فیلترهای زیادی اعمال می‌شوند.
- راه‌حل **:**مدل داده را بهینه کنید ،(Star Schema) روابط را بررسی کنید، از Measures به‌جای ستون‌های محاسباتی برای تجمیع استفاده کنید، و از `ALL` به‌جای `ALLSELECTED` در صورت امکان استفاده کنید.
- مشکل **:**رفرش داده‌ها زمان زیادی می‌برد.
- راه‌حل Power Query **:**را بهینه کنید (حذف مراحل غیرضروری، فیلترکردن زودهنگام Query ، ،(Foldingستون‌های غیرضروری را از مدل حذف کنید، و نوع داده‌ها را بهینه کنید.
- مشکل **:**یک Measure خاص بسیار کند است.
- راه‌حل **:**کوئری Measure را در DAX Studio اجرا کنید و `Server Timings` را بررسی کنید تا گلوگاه را شناسایی کنید. ممکن است نیاز به بازنویسی فرمول با استفاده از متغیرها، اجتناب از توابع Iterator بر روی جداول بزرگ، یا ساده‌سازی منطق `CALCULATE` باشد.
- مشکل **:**مدل داده بسیار بزرگ است و حافظه زیادی را اشغال می‌کند.
- راه‌حل **:**از DAX Studio برای بررسی متریک‌های VertiPaq استفاده کنید تا ستون‌هایی که بیشترین حافظه را اشغال می‌کنند (معمولاً ستون‌های با کاردینالیتی بالا یا رشته‌های متنی طولانی) را شناسایی کنید. می‌توانید این ستون‌ها را حذف کنید، فشرده‌سازی آن‌ها را بهبود بخشید (مثلاً با ایجاد جداول ابعاد برای آن‌ها) یا نوع داده آن‌ها را تغییر دهید.

۸-۶: خلاصه فصل ۸

در این فصل، به اهمیت بهینه‌سازی عملکرد (Performance Tuning) در مدل‌های DAX پرداختیم. درک موتور VertiPaqو نحوه عملکرد DAX، پایه و اساس بهینه‌سازی است. بهترین روش‌ها برای مدل‌سازی داده (مانند استفاده از Star Schema، حذف ستون‌های غیرضروری و مدیریت روابط) و بهترین روش‌ها برای نوشتن DAX کارآمد (مانند استفاده از Measures، متغیرها و بهینه‌سازی (CALCULATE را بررسی کردیم.

همچنین، با ابزارهای ضروری مانند Performance Analyzer، DAX Studio و Tabular Editor آشنا شدیم که به شما امکان می‌دهند تا عملکرد مدل‌های خود را تحلیل، پایش و بهینه‌سازی کنید. با به کارگیری این تکنیک‌ها و ابزارها، می‌توانید گزارش‌های Power BI سریع‌تر، پاسخگوتر و کارآمدتری ایجاد کنید که تجربه کاربری بهتری را فراهم می‌کنند.

در فصل بعدی، به بررسی سناریوهای پیشرفته مدل‌سازی داده خواهیم پرداخت که فراتر از Star Schema سنتی هستند و به شما امکان می‌دهند تا با ساختارهای داده‌ای پیچیده‌تر کار کنید.

فصل ۹: سناریوهای پیشرفته مدل‌سازی داده

در فصول گذشته، بر روی مدل‌سازی داده با استفاده از) Star Schema مدل ستاره‌ای) تمرکز کردیم که برای اکثر سناریوهای تحلیلی ایده‌آل است. با این حال، در دنیای واقعی، ممکن است با ساختارهای داده‌ای پیچیده‌تری مواجه شوید که نیاز به رویکردهای مدل‌سازی پیشرفته‌تری دارند. این فصل به بررسی این سناریوها و نحوه مدل‌سازی آن‌ها در Power BI می‌پردازد.

۹-۱: مدل‌سازی) Many-to-Many چند به چند(

رابطه) Many-to-Many چند به چند) زمانی رخ می‌دهد که یک رکورد در جدول A می‌تواند به چندین رکورد در جدول B مرتبط باشد و یک رکورد در جدول B نیز می‌تواند به چندین رکورد در جدول A مرتبط باشد Power BI .به طور مستقیم از روابط Many-to-Many پشتیبانی می‌کند، اما بهترین روش این است که این روابط را با استفاده از یک جدول پل **(Bridge Table)** به روابط) One-to-Many یک به چند) تبدیل کنید. این کار خوانایی مدل را افزایش داده و از مشکلات عملکردی یا ابهام در فیلترینگ جلوگیری می‌کند.

مثال: محصولات و فروشندگان (یک محصول می‌تواند توسط چندین فروشنده فروخته شود و یک فروشنده می‌تواند چندین محصول را بفروشد**(**

فرض کنید دو جدول DimProduct و DimSeller دارید و می‌خواهید بدانید کدام فروشندگان کدام محصولات را فروخته‌اند. اگر یک جدول FactSales دارید که شامل ProductID و SellerID است، این یک رابطه Many-to-Many بین DimProduct و DimSeller از طریق FactSales ایجاد می‌کند.

مدل‌سازی با جدول پل:

بهترین روش برای مدل‌سازی این سناریو، ایجاد یک جدول پل (Bridge Table) است که شامل ترکیب‌های منحصر به فرد ProductID و SellerID باشد. در واقع، جدول FactSales خود می‌تواند به‌عنوان یک جدول پل عمل کند.

- جداول:
  - DimProduct (ProductID, ProductName, ...)
  - DimSeller (SellerID, SellerName, ...)
  - FactSales (OrderID, ProductID, SellerID, SalesAmount, ...)
- روابط:
  - DimProduct [ProductID] (One) به FactSales [ProductID] (Many)
  - DimSeller [SellerID] (One) به FactSales [SellerID] (Many)

در این حالت FactSales، به‌عنوان جدول پل بین DimProduct و DimSeller عمل می‌کند. فیلترها از DimProduct به FactSales و سپس به DimSeller و بالعکس) جریان می‌یابند. این ساختار، یک Star Schema کلاسیک است که روابط Many-to-Many را به طور مؤثر مدیریت می‌کند.

سناریوی پیچیده‌تر: مشتریان و کمپین‌ها (یک مشتری می‌تواند در چندین کمپین شرکت کند و یک کمپین می‌تواند چندین مشتری داشته باشد)

فرض کنید دو جدول DimCustomer و DimCampaign دارید و یک جدول FactCampaignParticipation که نشان می‌دهد کدام مشتری در کدام کمپین شرکت کرده است.

- جداول:
  - DimCustomer (CustomerID, CustomerName, ...)
  - DimCampaign (CampaignID, CampaignName, ...)

- `FactCampaignParticipation` (ParticipationID, CustomerID, CampaignID, ParticipationDate, ...)

- روابط:

- `DimCustomer` به `[CustomerID]` (One) `FactCampaignParticipation [CustomerID]` (Many)
- `DimCampaign` به `[CampaignID]` (One) `FactCampaignParticipation [CampaignID]` (Many)

در این حالت `FactCampaignParticipation` ،به‌عنوان جدول پل عمل می‌کند. اگر بخواهید تعداد مشتریان را بر اساس کمپین فیلتر کنید، فیلتر از `DimCampaign` به `FactCampaignParticipation` و سپس به `DimCustomer` جریان می‌یابد. این یک Star Schema معتبر است.

۹-۲: مدل‌سازی) Role-Playing Dimensions ابعاد نقش‌آفرین(

ابعاد نقش‌آفرین زمانی رخ می‌دهند که یک جدول ابعاد (Dimension Table) واحد، چندین بار در یک جدول حقیقت (Fact Table) استفاده می‌شود، اما هر بار با یک نقش متفاوت. مثال رایج، جدول تاریخ است که می‌تواند برای `OrderDate`، `DeliveryDate`، `ShipDate` و غیره استفاده شود.

مثال: تاریخ سفارش و تاریخ ارسال

فرض کنید یک جدول `FactSales` دارید که شامل `OrderDate` و `ShipDate` است و یک جدول `DimDate`.

- جداول:

- `DimDate` (Date, Year, Month, ...)
- `FactSales` (OrderID, OrderDate, ShipDate, SalesAmount, ...)

- روابط:

- `DimDate [Date]` (One) به `FactSales [OrderDate]` (Many) - رابطه فعال
- `DimDate [Date]` (One) به `FactSales [ShipDate]` (Many) - رابطه غیرفعال

نحوه استفاده در **DAX:**

برای استفاده از رابطه غیرفعال، باید از تابع USERELATIONSHIP در داخل CALCULATE استفاده کنید:

```
Total Sales by Ship Date =
CALCULATE(
[Total Sales],
USERELATIONSHIP(DimDate [Date] , FactSales [ShipDate])
)
```

توضیح **:**این Measure مجموع فروش را بر اساس ShipDate محاسبه می‌کند، زیرا USERELATIONSHIP رابطه غیرفعال را به طور موقت فعال می‌کند. این رویکرد به شما امکان می‌دهد تا از یک جدول ابعاد واحد برای چندین نقش استفاده کنید و مدل خود را ساده نگه دارید.

۹-۳:مدل‌سازی جداول Snapshot (Snapshot Tables)

جداول Snapshot برای ذخیره وضعیت یک سیستم در یک نقطه زمانی خاص استفاده می‌شوند. این جداول برای تحلیل روندها و تغییرات در طول زمان، به خصوص برای موجودی، وضعیت پروژه‌ها یا وضعیت مشتریان، مفید هستند.

مثال: موجودی پایان ماه

فرض کنید می‌خواهید موجودی انبار را در پایان هر ماه تحلیل کنید. به‌جای ذخیره هر تراکنش ورودی و خروجی، می‌توانید یک جدول Snapshot ایجاد کنید که موجودی را در پایان هر ماه ثبت می‌کند.

- جدول **Snapshot:**
  - FactInventorySnapshot (SnapshotDate, ProductID, QuantityOnHand, ...)
- روابط**:**
  - DimDate [Date] (One) به FactInventorySnapshot [SnapshotDate] (Many)

- `DimProduct` به `FactInventorySnapshot` `[ProductID]` (One) `[ProductID]` (Many)

نحوه تحلیل در **DAX:**

برای تحلیل موجودی از این جدول، می‌توانید Measures ساده‌ای مانند `SUM(FactInventorySnapshot[QuantityOnHand])` ایجاد کنید. اگر می‌خواهید موجودی را در یک تاریخ خاص مشاهده کنید، کافی است تاریخ را فیلتر کنید. برای مقایسه موجودی بین دو دوره، می‌توانید از توابع هوش زمانی استفاده کنید.

نکته **:** جداول Snapshot می‌توانند بسیار بزرگ شوند، بنابراین مهم است که فقط داده‌های ضروری را ذخیره کنید و فرکانس Snapshotها را با دقت انتخاب کنید.

۹-۴: مدل‌سازی جداول Bridge برای سلسله مراتب ناهمگون (Ragged Hierarchies)

سلسله مراتب ناهمگون (Ragged Hierarchies) زمانی رخ می‌دهند که سطوح مختلف در یک سلسله مراتب، تعداد فرزندان متفاوتی دارند یا برخی از اعضا در سطوح بالاتری از سلسله مراتب قرار می‌گیرند (مثلاً یک مدیر که خودش هم زیرمجموعه دارد و هم گزارش مستقیم). مدل‌سازی این نوع سلسله مراتب می‌تواند چالش‌برانگیز باشد.

مثال: سلسله مراتب سازمانی (مدیران و کارمندان**(**

فرض کنید یک جدول `DimEmployee` دارید که شامل `EmployeeID` و `ManagerID` است. این یک سلسله مراتب والد-فرزند (Parent-Child Hierarchy) است.

مدل‌سازی با جدول **Bridge (Path Functions):**

DAXدارای توابعی است که به طور خاص برای کار با سلسله مراتب والد-فرزند طراحی شده‌اند:

- **PATH(<ID_column>, <Parent_ID_column>):** یک رشته متنی را برمی‌گرداند که شامل تمام IDهای والد در سلسله مراتب است، از بالاترین سطح تا ID فعلی.
- **PATHITEM(,):** یک ID را از یک مسیر در یک موقعیت مشخص برمی‌گرداند.
- **PATHLENGTH():** طول یک مسیر را برمی‌گرداند.
- **PATHCONTAINS(,):** بررسی می‌کند که آیا یک ID خاص در یک مسیر وجود دارد یا خیر.

مثال: ایجاد سلسله مراتب سازمانی و محاسبه تعداد زیرمجموعه‌ها

1. ایجاد ستون **Path (Calculated Column** در **:(DimEmployee**

```
Employee Path = PATH(DimEmployee [EmployeeID] , DimEmployee [ManagerID])
```

2. محاسبه تعداد زیرمجموعه‌ها **:(Measure)**

```
Number of Direct Reports =
CALCULATE(
COUNTROWS(DimEmployee),
FILTER(
ALL(DimEmployee),
DimEmployee [ManagerID] = VALUES(DimEmployee [EmployeeID])
)
)
```

توضیح **:**این Measure تعداد کارمندانی را می‌شمارد که مدیر مستقیم آن‌ها، کارمند فعلی در زمینه فیلتر است.

3. محاسبه تعداد کل زیرمجموعه‌ها (شامل زیرمجموعه‌های غیرمستقیم **:(Measure) (**

```
Number of All Subordinates =
CALCULATE(
COUNTROWS(DimEmployee),
FILTER(
ALL(DimEmployee),
```

PATHCONTAINS(DimEmployee [Employee Path] , VALUES(DimEmployee [EmployeeID]))

)

)

توضیح :این Measure تمام کارمندانی را می‌شمارد که در مسیر سلسله مراتبی آن‌ها، کارمند فعلی (مدیر) وجود دارد. این کارمندانی را شامل می‌شود که به طور مستقیم یا غیرمستقیم زیرمجموعه هستند.

۹-۵:مدل‌سازی جداول) Factless Fact جداول حقیقت بدون مقدار(

جداول) Factless Fact جداول حقیقت بدون مقدار) جداولی هستند که فقط شامل کلیدهای خارجی (Foreign Keys) هستند و هیچ Measure عددی ندارند. این جداول برای ردیابی رویدادها یا روابط بین ابعاد که شامل یک مقدار عددی نیستند، استفاده می‌شوند.

مثال: حضور دانش‌آموزان در کلاس‌ها

فرض کنید می‌خواهید ردیابی کنید که کدام دانش‌آموز در کدام کلاس حضور داشته است، اما هیچ مقدار عددی برای جمع‌آوری وجود ندارد (مثلاً فروش یا سود.(

- جداول:
  - `DimStudent` (StudentID, StudentName, ...)
  - `DimClass` (ClassID, ClassName, ...)
  - `FactAttendance` (AttendanceID, StudentID, ClassID, Date, ...)
- روابط:
  - `DimStudent [StudentID]` (One) به `FactAttendance [StudentID]` (Many)
  - `DimClass [ClassID]` (One) به `FactAttendance [ClassID]` (Many)
  - `DimDate[Date]` (One) به `FactAttendance[Date]` (Many)

نحوه تحلیل در **DAX:**

می‌توانید از `COUNTROWS(FactAttendance)` برای شمارش تعداد حضورها استفاده کنید. همچنین می‌توانید از `DISTINCTCOUNT(FactAttendance [StudentID])`برای شمارش تعداد دانش‌آموزان حاضر در یک کلاس خاص یا `DISTINCTCOUNT(FactAttendance [ClassID])` برای شمارش تعداد کلاس‌هایی که یک دانش‌آموز در آن‌ها حضور داشته است، استفاده کنید.

۹-۶: خلاصه فصل ۹

در این فصل، به بررسی سناریوهای پیشرفته مدل‌سازی داده پرداختیم که فراتر از Star Schema سنتی هستند. نحوه مدل‌سازی روابط Many-to-Many با استفاده از جداول پل (Bridge Tables) را بررسی کردیم که به بهبود خوانایی و عملکرد مدل کمک می‌کند. همچنین، با ابعاد نقش‌آفرین (Role-Playing Dimensions) آشنا شدیم و نحوه استفاده از `USERELATIONSHIP`برای فعال کردن روابط غیرفعال را دیدیم.

مدل‌سازی جداول Snapshot برای تحلیل روندها و جداول Bridge برای سلسله مراتب ناهمگون (با استفاده از توابع Path) نیز مورد بحث قرار گرفت. در نهایت، با مفهوم جداول Factless Fact آشنا شدیم که برای ردیابی رویدادها یا روابط بدون مقادیر عددی استفاده می‌شوند.

تسلط بر این تکنیک‌های مدل‌سازی پیشرفته، شما را قادر می‌سازد تا با ساختارهای داده‌ای پیچیده‌تر کار کنید و مدل‌های Power BIانعطاف‌پذیرتر و قدرتمندتری را ایجاد کنید که به نیازهای تحلیلی متنوع پاسخ می‌دهند. در فصل بعدی، به مبحث بسیار مهم امنیت در Power BI و DAX خواهیم پرداخت.

فصل ۱۰: امنیت در Power BI و DAX (Row-Level Security)

امنیت داده‌ها یکی از مهم‌ترین جنبه‌ها در هر راهکار هوش تجاری است. در Power BI، امنیت در سطح ردیف (Row-Level Security - RLS)به شما امکان می‌دهد تا دسترسی کاربران به داده‌ها را بر اساس نقش‌ها و قوانین مشخص محدود کنید. این بدان معناست که کاربران مختلف، هنگام مشاهده یک گزارش واحد، فقط می‌توانند زیرمجموعه‌ای از داده‌ها را ببینند که مجاز به دسترسی به آن‌ها هستند. در این فصل، به طور جامع به نحوه پیاده‌سازی RLS در Power BI با استفاده از DAX خواهیم پرداخت.

۱۰-۱مفهوم امنیت در سطح ردیف(RLS):

RLSیک لایه امنیتی است که در سطح مدل داده اعمال می‌شود. هنگامی که RLS را پیاده‌سازی می‌کنید، فیلترهایی را بر روی جداول مدل خود تعریف می‌کنید. این فیلترها به طور خودکار بر روی کوئری‌های DAX اعمال می‌شوند که توسط کاربران اجرا می‌شوند. نتیجه این است که کاربران فقط ردیف‌هایی از داده‌ها را می‌بینند که با فیلترهای نقش آن‌ها مطابقت دارد.

چرا **RLS** مهم است؟

- حفظ حریم خصوصی و محرمانگی داده‌ها **:**اطمینان حاصل می‌کند که اطلاعات حساس فقط توسط افراد مجاز قابل مشاهده است.
- رعایت مقررات **:**به شرکت‌ها کمک می‌کند تا با مقررات مربوط به حفاظت از داده‌ها (مانند GDPR) مطابقت داشته باشند.
- ساده‌سازی مدیریت گزارش **:**به‌جای ایجاد گزارش‌های جداگانه برای هر گروه از کاربران، می‌توانید یک گزارش واحد ایجاد کرده و RLS را برای محدود کردن دسترسی به داده‌ها اعمال کنید.
- مقیاس‌پذیری RLS **:**در سطح مدل داده اعمال می‌شود، بنابراین حتی اگر داده‌ها به‌صورت مستقیم از طریق DAX یا سایر ابزارها کوئری شوند، قوانین امنیتی اعمال می‌شوند.

سناریوهای رایج**RLS:**

- مدیر فروش منطقه **:**فقط می‌تواند داده‌های فروش منطقه خود را ببیند.
- مدیر محصول **:**فقط می‌تواند داده‌های مربوط به محصولات خود را ببیند.
- کارمند **:**فقط می‌تواند اطلاعات مربوط به خود را ببیند.

۱۰-۲پیاده‌سازی RLS درPower BI Desktop:

پیاده‌سازی RLS در Power BI Desktop شامل دو مرحله اصلی است: تعریف نقش‌ها (Roles) و تعریف قوانین (Rules) با استفاده از.DAX

1. تعریف نقش‌ها**:(Roles)**

- در ،Power BI Desktop به تب `Modeling` بروید.
- در گروه ،`Security` بر روی `Manage roles` کلیک کنید.
- در پنجره ،`Manage roles` بر روی `Create` کلیک کنید تا یک نقش جدید ایجاد کنید. نامی برای نقش خود انتخاب کنید (مثلا.(`Product Manager`، `Sales Manager`ً

2. تعریف قوانین **(Rules)** با **DAX:**

- پس از ایجاد نقش، جداول مدل خود را در سمت چپ پنجره مشاهده خواهید کرد.
- جدولی را که می‌خواهید فیلتر کنید، انتخاب کنید (مثلا `DimRegion` برای مدیران فروش منطقه.)
- در کادر `Table filter DAX expression`، یک عبارت DAX بنویسید که ردیف‌هایی را که کاربر مجاز به دیدن آن‌ها است، فیلتر کند. این عبارت باید یک مقدار `TRUE` یا `FALSE` برگرداند.

مثال ۱: محدود کردن دسترسی بر اساس منطقه**(Region)**

فرض کنید در جدول `DimRegion` یک ستون به نام `RegionName` دارید و می‌خواهید مدیران فروش فقط داده‌های منطقه خود را ببینند. همچنین، فرض کنید نام کاربری (UPN) مدیر فروش در یک جدول نگاشت (Mapping Table)ذخیره شده است.

- نقش `Regional Sales Manager` :
- جدول `DimRegion` :
- قانون **DAX:**

```
[RegionName] = LOOKUPVALUE(
'UserRegionMapping' [RegionName] ,
'UserRegionMapping' [UserPrincipalName] ,
USERPRINCIPALNAME()
)
```

توضیح:

- `USERPRINCIPALNAME()`: نام کاربری فعلی (UPN) کاربر وارد شده را برمی‌گرداند.
- `LOOKUPVALUE(...)`: این تابع در جدول `UserRegionMapping` که باید از قبل در مدل شما وجود داشته باشد و شامل ستون‌های `UserPrincipalName` و `RegionName`باشد) به دنبال `RegionName` مرتبط با `USERPRINCIPALNAME()`فعلی می‌گردد.

- سپس، این RegionName با [RegionName] در جدول DimRegion مقایسه می‌شود. فقط ردیف‌هایی که مطابقت دارند، برای کاربر قابل مشاهده خواهند بود.

مثال ۲: محدود کردن دسترسی بر اساس دسته محصول**(Product Category)**

فرض کنید در جدول DimProduct یک ستون به نام Category دارید و می‌خواهید مدیران محصول فقط داده‌های دسته محصول خود را ببینند.

- نقش Product Category Manager :
- جدول DimProduct :
- قانون**DAX:**

"} IN {[Category]لوازم خانگی", "الکترونیک"

توضیح **:**در این مثال، کاربرانی که به نقش Product Category Manager اختصاص داده شده‌اند، فقط می‌توانند داده‌های مربوط به دسته‌های "لوازم خانگی" و "الکترونیک" را ببینند. این رویکرد برای تعداد محدودی از دسته‌ها مناسب است. برای سناریوهای پیچیده‌تر، می‌توانید از یک جدول نگاشت کاربر-دسته مشابه مثال قبل استفاده کنید.

مثال ۳: نمایش تمام داده‌ها برای مدیران**(Admin Role)**

برای نقش‌های مدیریتی که باید به تمام داده‌ها دسترسی داشته باشند، می‌توانید یک نقش Admin ایجاد کنید و هیچ فیلتری را اعمال نکنید، یا یک فیلتر TRUE() را اعمال کنید:

- نقش Admin :
- جدول) FactSales **:**یا هر جدول دیگر(
- قانون**DAX:**

TRUE()

توضیح **:**این قانون همیشه TRUE را برمی‌گرداند، بنابراین هیچ فیلتری اعمال نمی‌شود و کاربر به تمام ردیف‌ها دسترسی دارد.

3. اعمال فیلترها به جداول مرتبط **:**هنگامی که یک فیلتر RLS را بر روی یک جدول ابعاد اعمال می‌کنید، این فیلتر به طور خودکار از طریق روابط فعال به جداول حقیقت مرتبط و سایر جداول ابعاد جریان می‌یابد. نیازی به تعریف قوانین RLS بر روی هر جدول حقیقت نیست.

۱۰-۳: تست RLS در Power BI Desktop

پس از تعریف نقش‌ها و قوانین، می‌توانید آن‌ها را در Power BI Desktop تست کنید تا مطمئن شوید که به‌درستی کار می‌کنند:

1. در تب ،`Modeling` بر روی `View as` کلیک کنید.
2. در پنجره ،`View as roles` می‌توانید نقش‌هایی را که ایجاد کرده‌اید، انتخاب کنید. همچنین می‌توانید `Other user` را انتخاب کرده و یک نام کاربری (UPN) خاص را وارد کنید تا ببینید گزارش برای آن کاربر چگونه به نظر می‌رسد.
3. پس از انتخاب نقش یا کاربر، گزارش شما بر اساس قوانین RLS اعمال شده، فیلتر می‌شود. می‌توانید با تغییر نقش‌ها، نحوه تغییر داده‌ها را مشاهده کنید.
4. برای بازگشت به نمای عادی، بر روی `Stop Viewing` در بالای گزارش کلیک کنید.

۱۰-۴: انتشار و مدیریت RLS در Power BI Service

پس از اینکه RLS را در Power BI Desktop پیاده‌سازی و تست کردید، باید گزارش را در Power BI Service منتشر کنید و کاربران را به نقش‌ها اختصاص دهید.

1. انتشار گزارش **:**گزارش Power BI خود را در Power BI Service منتشر کنید.
2. مدیریت امنیت در **Power BI Service:**

- در ،Power BI Service به فضای کاری (Workspace) که گزارش را در آن منتشر کرده‌اید، بروید.
- در کنار مجموعه داده (Dataset) مربوطه، بر روی سه نقطه (`...`) کلیک کنید و `Security` را انتخاب کنید.
- در صفحه ،`Row-level security` نقش‌هایی را که در Power BI Desktop ایجاد کرده‌اید، مشاهده خواهید کرد.
- بر روی `Add members` در کنار هر نقش کلیک کنید و کاربران یا گروه‌های امنیتی (مانند گروه‌های Azure Active Directory) را که می‌خواهید به آن نقش اختصاص دهید، اضافه کنید.
- کاربران یا گروه‌های اضافه شده، هنگام دسترسی به گزارش، قوانین RLS مربوط به آن نقش را تجربه خواهند کرد.

نکات مهم برای مدیریت **RLS** در **Power BI Service:**

- همگام‌سازی نقش‌ها **:**اگر نقش‌ها یا قوانین RLS را در Power BI Desktop تغییر دهید، باید گزارش را دوباره منتشر کنید تا تغییرات در Power BI Service اعمال شوند.
- تست در **Power BI Service:** پس از اختصاص کاربران به نقش‌ها، می‌توانید با استفاده از قابلیت `Test as role`در Power BI Service، نحوه مشاهده گزارش توسط کاربران مختلف را تست کنید.
- عملکرد RLS **:**می‌تواند بر عملکرد گزارش‌ها تأثیر بگذارد، به خصوص اگر قوانین DAX پیچیده باشند یا بر روی جداول بسیار بزرگ اعمال شوند. از بهترین روش‌های بهینه‌سازی DAX که در فصل ۸ بحث شد، استفاده کنید.
- امنیت مبتنی بر نام کاربری **(Dynamic RLS):** استفاده از `USERPRINCIPALNAME()` یا `USERNAME()`در قوانین DAX، به شما امکان می‌دهد تا RLS پویا (Dynamic RLS) را پیاده‌سازی کنید، جایی که فیلترها بر اساس نام کاربری وارد شده به Power BI Service اعمال می‌شوند. این رویکرد برای مدیریت تعداد زیادی از کاربران با دسترسی‌های متفاوت بسیار کارآمد است.

۱۰-۵: ملاحظات امنیتی پیشرفته

- امنیت در سطح شیء **(Object-Level Security - OLS):** علاوه بر RLS ،Power BI از OLS نیز پشتیبانی می‌کند که به شما امکان می‌دهد تا دسترسی کاربران به جداول یا ستون‌های خاص را به طور کامل محدود کنید (یعنی کاربران حتی نمی‌توانند وجود آن‌ها را ببینند OLS .(معمولاً با استفاده از Tabular Editor پیاده‌سازی می‌شود.
- امنیت در سطح ستون OLS **(Column-Level Security):** شامل امنیت در سطح ستون نیز می‌شود.
- امنیت در **Power Query:** می‌توانید فیلترهای امنیتی را در Power Query نیز اعمال کنید، اما این کار معمولاً برای RLS توصیه نمی‌شود، زیرا RLS در سطح مدل داده اعمال می‌شود و انعطاف‌پذیری بیشتری دارد.
- امنیت داده‌های منبع RLS **:**در Power BI، دسترسی به داده‌ها را در خود گزارش Power BI محدود می‌کند. اگر کاربران به طور مستقیم به منبع داده (مثلاً پایگاه داده SQL) دسترسی دارند، باید امنیت را در سطح منبع داده نیز پیاده‌سازی کنید.
- تست جامع **:**همیشه RLS را به طور جامع تست کنید تا مطمئن شوید که هیچ راهی برای دور زدن قوانین امنیتی وجود ندارد.

۱۰-۶: خلاصه فصل ۱۰

در این فصل، به طور جامع به مفهوم و پیاده‌سازی امنیت در سطح ردیف (Row-Level Security - RLS) در Power BIپرداختیم. درک کردیم که RLS چگونه به شما امکان می‌دهد تا دسترسی کاربران به داده‌ها را بر اساس نقش‌ها و قوانین DAXمحدود کنید و چرا این قابلیت برای حفظ حریم خصوصی، رعایت مقررات و ساده‌سازی مدیریت گزارش‌ها حیاتی است.

نحوه تعریف نقش‌ها و قوانین DAX در Power BI Desktop، تست RLS و سپس انتشار و مدیریت آن در Power BI Serviceرا بررسی کردیم. همچنین، به ملاحظات امنیتی پیشرفته‌تر مانند امنیت در سطح شیء و اهمیت تست جامع اشاره کردیم.

با تسلط بر RLS، شما قادر خواهید بود تا راهکارهای هوش تجاری امن و مقیاس‌پذیری را در Power BI ایجاد کنید که اطمینان حاصل می‌کند داده‌های حساس فقط توسط افراد مجاز قابل مشاهده هستند. در فصل بعدی، به بررسی سناریوهای پیشرفته هوش زمانی و مالی خواهیم پرداخت.

فصل ۱۱: سناریوهای پیشرفته هوش زمانی و مالی

در فصل ۶، به اصول هوش زمانی و نحوه پیاده‌سازی آن با تقویم شمسی پرداختیم. در این فصل، به سناریوهای پیشرفته‌تر هوش زمانی و همچنین کاربردهای DAX در تحلیل‌های مالی خواهیم پرداخت. این سناریوها به شما امکان می‌دهند تا بینش‌های عمیق‌تری را از داده‌های زمانی و مالی خود استخراج کنید.

۱۱-۱: مقایسه دوره‌های دلخواه(Custom Period Comparisons)

گاهی اوقات، نیاز به مقایسه عملکرد بین دوره‌های زمانی دارید که با توابع استاندارد هوش زمانی (مانند سال گذشته یا ماه گذشته) پوشش داده نمی‌شوند. این سناریو نیاز به کنترل دقیق‌تری بر زمینه فیلتر دارد.

مثال: مقایسه فروش در دو دوره انتخاب شده توسط کاربر

فرض کنید می‌خواهید به کاربر اجازه دهید دو دوره زمانی دلخواه را انتخاب کند و فروش را بین این دو دوره مقایسه کند. این کار با استفاده از پارامترهای What-If یا جداول تاریخ سفارشی انجام می‌شود.

رویکرد با پارامترهای **What-If)** برای انتخاب تاریخ شروع و پایان**:(**

1. ایجاد دو پارامتر **What-If** برای تاریخ شروع و پایان دوره**:۱**

- `Start Date 1` (نوع `Date`, حداقل: اولین تاریخ داده‌ها، حداکثر: آخرین تاریخ داده‌ها، گام: ۱ روز)
- `End Date 1` (نوع `Date`, حداقل: اولین تاریخ داده‌ها، حداکثر: آخرین تاریخ داده‌ها، گام: ۱ روز)

2. ایجاد دو پارامتر **What-If** برای تاریخ شروع و پایان دوره **۲:**

- `Start Date 2`
- `End Date 2`

3. **Measure** برای فروش دوره **۱:**

```
Sales Period 1 =
CALCULATE(
[Total Sales] ,
DATESBETWEEN(
DimDate [Date] ,
'Start Date 1' [Start Date 1 Value] ,
'End Date 1' [End Date 1 Value]
)
)
```

4. **Measure** برای فروش دوره **۲:**

```
Sales Period 2 =
CALCULATE(
[Total Sales] ,
DATESBETWEEN(
```

DimDate [Date] ,

'Start Date 2' [Start Date 2 Value] ,

'End Date 2' [End Date 2 Value]

)

)

5. **Measure** برای تفاوت یا رشد**:**

Sales Period Difference = [Sales Period 1] - [Sales Period 2]

توضیح: این رویکرد به کاربر امکان می‌دهد تا با استفاده از اسلایسرها، تاریخ‌های شروع و پایان دو دوره را انتخاب کند و **Measures** مربوطه فروش را برای آن دوره‌ها محاسبه می‌کنند. **DATESBETWEEN** یک جدول از تاریخ‌ها را بین دو تاریخ مشخص برمی‌گرداند.

۱۱-۲: تحلیل) Rolling Period دوره‌های متحرک(

تحلیل) Rolling Period مانند میانگین متحرک ۳۰ روزه یا مجموع فروش ۱۲ ماه گذشته) برای هموار کردن نوسانات و شناسایی روندهای بلندمدت مفید است. در فصل ۶ به میانگین متحرک اشاره شد، در اینجا به مجموع متحرک می‌پردازیم.

مثال: مجموع فروش ۱۲ ماه گذشته**(Rolling 12-Month Sales)**

Rolling 12-Month Sales =

CALCULATE(

[Total Sales] ,

DATESINPERIOD(

DimDate [Date] ,

LASTDATE(DimDate [Date]),

-12,

MONTH

)

)

توضیح: این **Measure** مجموع فروش را برای ۱۲ ماه گذشته (شامل ماه فعلی) محاسبه می‌کند. **DATESINPERIOD** یک جدول از تاریخ‌ها را در یک دوره مشخص (در اینجا ۱۲ ماه قبل از آخرین تاریخ در زمینه فیلتر) برمی‌گرداند.

۱۱-۳: تحلیل) Budget vs. Actual بودجه در مقابل عملکرد واقعی(

مقایسه عملکرد واقعی با بودجه، یک تحلیل مالی حیاتی است. برای این کار، نیاز به یک جدول بودجه در مدل داده خود دارید.

مدل‌سازی جدول بودجه:

جدول بودجه (مثلا `FactBudget`) باید شامل ستون‌هایی مانند `Date`, `ProductID`, `RegionID` و `BudgetAmount`باشد. این جدول باید به جداول ابعاد مربوطه (تاریخ، محصول، منطقه) مرتبط باشد.

- جداول:
  - `FactBudget` (Date, ProductID, RegionID, BudgetAmount)
  - `DimDate`
  - `DimProduct`
  - `DimRegion`
- روابط: روابط یک به چند از جداول ابعاد به `FactBudget`.

Measures برای تحلیل **Budget vs. Actual**:

1. **Measure** برای مجموع بودجه:

```
Total Budget = SUM(FactBudget [BudgetAmount])
```

2. **Measure** برای تفاوت **(Variance)**:

```
Sales Variance = [Total Sales] - [Total Budget]
```

3. **Measure**برای درصد تحقق بودجه**:**

```
Sales Budget Attainment % = DIVIDE([Total Sales] , [Total Budget])
```

توضیح **:**این Measures به شما امکان می‌دهند تا عملکرد واقعی فروش را با بودجه مقایسه کنید و تفاوت و درصد تحقق را مشاهده کنید. می‌توانید این Measures را با ابعاد مختلف (محصول، منطقه، زمان) تحلیل کنید.

۴-۱۱:تحلیل سودآوری(Profitability Analysis)

تحلیل سودآوری به شما کمک می‌کند تا بفهمید کدام محصولات، مشتریان، مناطق یا کانال‌ها بیشترین سود را برای کسب‌وکار شما ایجاد می‌کنند.

Measuresکلیدی برای تحلیل سودآوری**:**

1. سود ناخالص**:(Gross Profit)**

```
Total Gross Profit = SUMX(FactSales, FactSales [SalesAmount] - FactSales [CostAmount])
```

)این Measure قبلاً در فصل ۴ بحث شد.(

2. حاشیه سود ناخالص**:(Gross Profit Margin %)**

```
Gross Profit Margin % = DIVIDE([Total Gross Profit] , [Total Sales])
```

3. سود خالص **:(Net Profit)** برای محاسبه سود خالص، نیاز به جمع‌آوری تمام هزینه‌ها (عملیاتی، بازاریابی، اداری و غیره) دارید. فرض کنید یک جدول `FactExpenses` دارید.

```
Total Expenses = SUM(FactExpenses [ExpenseAmount])
```

```
Net Profit = [Total Gross Profit] - [Total Expenses]
```

4. حاشیه سود خالص**:(Net Profit Margin %)**

```
Net Profit Margin % = DIVIDE([Net Profit] , [Total Sales])
```

توضیح :با استفاده از این Measures، می‌توانید سودآوری را بر اساس ابعاد مختلف تحلیل کنید. مثلاً، می‌توانید یک ماتریس ایجاد کنید که سود ناخالص و حاشیه سود را برای هر محصول یا هر منطقه نمایش دهد.

۱۱-۵: تحلیل جریان نقدی (Cash Flow Analysis)

تحلیل جریان نقدی برای مدیریت نقدینگی و سلامت مالی یک کسب‌وکار حیاتی است. DAX می‌تواند برای محاسبه جریان‌های نقدی ورودی و خروجی و موجودی نقدی استفاده شود.

مدل‌سازی داده برای جریان نقدی:

نیاز به جداولی دارید که تراکنش‌های نقدی ورودی (مانند دریافت از مشتریان) و خروجی (مانند پرداخت به تامین‌کنندگان، هزینه‌ها) را ثبت کنند. این جداول باید شامل `Date`, `Amount` و `TransactionType` ورودی/خروجی) باشند.

- جدول(... ,Date, Amount, TransactionType) `FactCashFlow` :
- رابطه (One) `DimDate [Date]` به(Many) `FactCashFlow [Date]`

Measures برای تحلیل جریان نقدی:

1. جریان نقدی ورودی **(Cash Inflow):**

```
Cash Inflow = CALCULATE(SUM(FactCashFlow [Amount]), FactCashFlow [TransactionType] = "ورودی")
```

2. جریان نقدی خروجی **(Cash Outflow):**

```
Cash Outflow = CALCULATE(SUM(FactCashFlow [Amount]), FactCashFlow [TransactionType] = "خروجی")
```

3. جریان نقدی خالص **(Net Cash Flow):**

```
Net Cash Flow = [Cash Inflow] - [Cash Outflow]
```

4. موجودی نقدی پایان دوره **(End-of-Period Cash Balance):** این Measure پیچیده‌تر است و نیاز به جمع تجمعی جریان نقدی خالص از ابتدای زمان تا تاریخ فعلی دارد.

```
Cash Balance EOP =
CALCULATE(
[Net Cash Flow] ,
ALL(DimDate [Date]),
DimDate [Date] <= MAX(DimDate [Date])
)
```

توضیح: این **Measure** مجموع تجمعی **Net Cash Flow** را از ابتدای زمان تا آخرین تاریخ در زمینه فیلتر فعلی محاسبه می‌کند. **ALL(DimDate [Date])** فیلترهای تاریخ را حذف می‌کند و **DimDate [Date] <= MAX(DimDate [Date])** تضمین می‌کند که فقط تاریخ‌های تا تاریخ فعلی در نظر گرفته شوند.

۱۱-۶: خلاصه فصل ۱۱

در این فصل، به سناریوهای پیشرفته هوش زمانی و کاربردهای DAX در تحلیل‌های مالی پرداختیم. نحوه مقایسه دوره‌های دلخواه با استفاده از پارامترهای What-If و محاسبه دوره‌های متحرک (Rolling Periods) را بررسی کردیم. همچنین، به تحلیل‌های مالی حیاتی مانند مقایسه بودجه در مقابل عملکرد واقعی (Budget vs. Actual)، تحلیل سودآوری (Profitability Analysis) و تحلیل جریان نقدی (Cash Flow Analysis) پرداختیم.

با تسلط بر این تکنیک‌ها، شما قادر خواهید بود تا بینش‌های مالی عمیق‌تری را از داده‌های خود استخراج کنید و به تصمیم‌گیری‌های استراتژیک در کسب‌وکار کمک کنید. این فصل نشان می‌دهد که DAX چقدر می‌تواند در تحلیل‌های پیچیده مالی قدرتمند باشد.

فصل ۱۲: بهترین روش‌ها و نکات پیشرفته در DAX

در طول این کتاب، با مفاهیم پایه تا پیشرفته DAX آشنا شدیم و نحوه پیاده‌سازی سناریوهای مختلف تحلیلی را بررسی کردیم. در این فصل پایانی، به جمع‌بندی بهترین روش‌ها، نکات پیشرفته و توصیه‌هایی می‌پردازیم که به شما کمک می‌کنند تا به یک متخصص DAX تبدیل شوید و مدل‌های داده و گزارش‌های Power BI خود را به بالاترین سطح کارایی و خوانایی برسانید.

۱۲-۱: بهترین روش‌ها برای نوشتن DAX

- خوانایی را در اولویت قرار دهید: فرمول‌های **DAX** شما باید برای خودتان و دیگران قابل‌درک باشند. از نام‌گذاری معنی‌دار برای **Measures**، ستون‌ها و متغیرها استفاده کنید. از فرمت‌بندی مناسب (مانند تورفتگی و خطوط خالی) استفاده کنید. **DAX Studio** دارای یک فرمت‌کننده داخلی است که به این کار کمک می‌کند.
- از متغیرها **(Variables)** استفاده کنید **:**همانطور که در فصل ۳ و ۸ بحث شد، متغیرها (با `VAR` و `RETURN`) خوانایی را بهبود می‌بخشند و عملکرد را افزایش می‌دهند. آن‌ها از محاسبات تکراری جلوگیری می‌کنند و دیباگینگ را آسان‌تر می‌کنند.
- کامنت‌گذاری کنید **:**فرمول‌های پیچیده را با کامنت‌ها (با استفاده از `//` برای یک خط یا `/* ... */` برای چند خط) توضیح دهید. این کار به شما کمک می‌کند تا در آینده فرمول‌های خود را به خاطر بیاورید و به دیگران کمک می‌کند تا آن‌ها را درک کنند.
- از **Measures** به‌جای ستون‌های محاسباتی برای تجمیع استفاده کنید: این یک قانون طلایی است. **Measures** پویا هستند و به زمینه فیلتر پاسخ می‌دهند، درحالی‌که ستون‌های محاسباتی ثابت هستند و فضای حافظه را اشغال می‌کنند.
- از **`CALCULATE`** به طور مؤثر استفاده کنید **:**`CALCULATE` قدرتمندترین تابع در DAX است. درک عمیق آن و نحوه تغییر زمینه فیلتر، کلید تسلط بر DAX است. سعی کنید فیلترها را تا حد امکان ساده نگه دارید و از فیلترهای مستقیم به‌جای `FILTER` بر روی جداول بزرگ استفاده کنید.
- به حداقل رساندن استفاده از توابع **Iterator (X-Functions)** بر روی جداول بزرگ **:**این توابع برای هر ردیف تکرار می‌شوند و می‌توانند بر عملکرد تأثیر بگذارند. در صورت امکان، از توابع تجمیعی ساده‌تر استفاده کنید.
- از **`SWITCH(TRUE() , ...)`** به‌جای **`IF`** های تو در تو استفاده کنید **:**برای منطق‌های شرطی پیچیده ، `SWITCH(TRUE() , ...)` خواناتر و گاهی اوقات کارآمدتر است.
- از **`DIVIDE`** به‌جای عملگر تقسیم **(/)** استفاده کنید **:**تابع `DIVIDE` به طور خودکار تقسیم بر صفر را مدیریت می‌کند و در صورت وقوع `BLANK()` ، را برمی‌گرداند، که از خطاها جلوگیری می‌کند.
- از **`BLANK()`** برای مدیریت نتایج خالی استفاده کنید **:**به‌جای نمایش صفر یا خطا، در سناریوهایی که نتیجه‌ای منطقی وجود ندارد `BLANK()` ، را برگردانید.

۱۲-۲: بهترین روش‌ها برای مدل‌سازی داده

- **Star Schema**را در اولویت قرار دهید **:**این ساختار مدل‌سازی برای Power BI و موتور VertiPaq بهینه است. جداول حقیقت و ابعاد را به‌درستی شناسایی و طراحی کنید.
- حذف ستون‌ها و ردیف‌های غیرضروری **:**داده‌ها را در Power Query تمیز و بهینه کنید. فقط داده‌هایی را بارگذاری کنید که برای تحلیل شما ضروری هستند.

- انتخاب نوع داده مناسب **:**از دقیق‌ترین و کوچک‌ترین نوع داده ممکن برای هر ستون استفاده کنید. این کار به فشرده‌سازی بهتر و کاهش حجم مدل کمک می‌کند.
- مدیریت روابط **:**
  - روابط یک به چند **(One-to-Many)** را ترجیح دهید**.**
  - از روابط دوجهته **(Bi-directional)** با احتیاط استفاده کنید و آن‌ها را به حداقل برسانید.
  - روابط **Many-to-Many** را با جداول پل **(Bridge Tables)** مدیریت کنید**.**
- ایجاد یک جدول تاریخ **(Date Table)** مناسب **:**این یک عنصر حیاتی برای هر مدل داده است، به خصوص برای هوش زمانی. مطمئن شوید که جدول تاریخ شما کامل، بدون شکاف و به‌درستی علامت‌گذاری شده است.
- پنهان کردن ستون‌های کلید خارجی **:**پس از ایجاد روابط، ستون‌های کلید خارجی (مانند `ProductID` در جدول حقیقت) را از نمای گزارش پنهان کنید تا مدل برای کاربران نهایی تمیزتر و قابل فهم‌تر باشد.
- استفاده از پوشه‌ها **(Display Folders)** برای سازماندهی Measures **:Measures** و ستون‌های محاسباتی را در پوشه‌های منطقی گروه‌بندی کنید تا کاربران به‌راحتی بتوانند آن‌ها را پیدا کنند.

۱۲-۳: نکات پیشرفته و تکنیک‌ها

- **Context Transition (**انتقال زمینه **:(**این مفهوم را به طور کامل درک کنید. این اتفاق زمانی رخ می‌دهد که یک تابع Iterator در داخل `CALCULATE` استفاده می‌شود و زمینه ردیف به زمینه فیلتر تبدیل می‌شود. این یکی از قدرتمندترین و در عین حال پیچیده‌ترین جنبه‌های DAX است.
- استفاده از **`ALLSELECTED`** برای محاسبات درصد از کل در زیرمجموعه‌ها **:**این تابع برای سناریوهایی که می‌خواهید درصد از کل را در یک زیرمجموعه فیلتر شده توسط کاربر محاسبه کنید، بسیار مفید است.
- استفاده از **`USERELATIONSHIP`** برای روابط نقش‌آفرین **:**این تابع به شما امکان می‌دهد تا روابط غیرفعال را به طور موقت فعال کنید و از یک جدول ابعاد برای چندین نقش استفاده کنید.
- استفاده از **`CROSSFILTER`** با احتیاط **:**این تابع می‌تواند برای تغییر جهت فیلتر یا غیرفعال کردن روابط استفاده شود، اما باید با دقت و درک کامل از تأثیرات آن بر مدل استفاده شود.
- پیاده‌سازی **Row-Level Security (RLS):** برای محدود کردن دسترسی کاربران به داده‌ها بر اساس نقش‌ها و قوانین DAX، RLS را پیاده‌سازی کنید.
- استفاده از ابزارهای خارجی Performance Analyzer، DAX Studio **:**و Tabular Editor ابزارهای ضروری برای هر توسعه‌دهنده Power BI هستند. از آن‌ها برای تحلیل، پایش و بهینه‌سازی مدل و فرمول‌های خود استفاده کنید.
- یادگیری مداوم **:**دنیای DAX و Power BI به‌سرعت در حال تغییر است. منابع آنلاین، وبلاگ‌ها، انجمن‌ها و دوره‌های آموزشی را دنبال کنید تا دانش خود را به روز نگه دارید.

۱۲-۴: منابع یادگیری و جامعه DAX

- وب‌سایت **SQLBI (sqlbi.com):** این وب‌سایت توسط آلبرتو فراری و مارکو روسو اداره می‌شود و بهترین منبع برای یادگیری DAX و مدل‌سازی داده است. آن‌ها مقالات، ویدئوها، کتاب‌ها و دوره‌های آموزشی بسیار با کیفیتی ارائه می‌دهند.
- کتاب **"The Definitive Guide to DAX":** این کتاب توسط آلبرتو فراری و مارکو روسو نوشته شده و به‌عنوان مرجع اصلی DAX شناخته می‌شود. اگرچه ممکن است کمی فنی باشد، اما برای درک عمیق DAX ضروری است.
- انجمن **Power BI (community.powerbi.com):** یک منبع عالی برای پرسیدن سؤالات، یافتن پاسخ‌ها و یادگیری از دیگران است.
- کانال‌های **YouTube:** بسیاری از متخصصان Power BI و DAX کانال‌های YouTube دارند که آموزش‌ها و نکات مفیدی را ارائه می‌دهند.
- وبلاگ‌ها **:**وبلاگ‌های زیادی در مورد DAX و Power BI وجود دارد که می‌توانید از آن‌ها استفاده کنید.

۱۲-۵: کلام آخر

تبریک می‌گویم! شما اکنون دانش و ابزارهای لازم را برای تسلط بر DAX و ساخت مدل‌های داده قدرتمند در Power BI در اختیار دارید. DAX یک‌زبان بسیار قدرتمند است که به شما امکان می‌دهد تا بینش‌های عمیقی را از داده‌های خود استخراج کنید و به تصمیم‌گیری‌های آگاهانه در کسب‌وکار کمک کنید.

به یاد داشته باشید که یادگیری DAX یک سفر است، نه یک مقصد. با تمرین مداوم، حل مسائل واقعی و استفاده از منابع موجود، می‌توانید مهارت‌های خود را به طور پیوسته بهبود بخشید. از چالش‌ها نترسید و همیشه به دنبال راه‌های جدید برای استفاده از DAX برای حل مسائل کسب‌وکار باشید.

امیدوارم این کتاب به شما در این سفر کمک کرده باشد و شما را برای تبدیل شدن به یک متخصص DAX توانمند یاری رسانده باشد. موفق باشید!

پیوست الف: نصب و راه‌اندازی Power BI Desktop

Power BI Desktop ابزار اصلی برای توسعه گزارش‌ها و مدل‌های داده در Power BI است. نصب آن بسیار ساده است و می‌توانید آن را به‌صورت رایگان از وب‌سایت مایکروسافت دانلود کنید.

الف-۱: دانلود Power BI Desktop

1. مراجعه به وب‌سایت مایکروسافت: به آدرس `https://powerbi.microsoft.com/desktop/` بروید.

2. انتخاب گزینه دانلود: دو گزینه برای دانلود وجود دارد:

- **Download from Download Center (**توصیه می‌شود**):** این گزینه به شما امکان می‌دهد تا فایل نصب (MSI) را دانلود کنید که برای نصب آفلاین یا توزیع در سازمان‌ها مناسب است. همچنین می‌توانید نسخه ۶۴ بیتی یا ۳۲ بیتی را انتخاب کنید.
- **Download from Microsoft Store:** این گزینه Power BI Desktop را از Microsoft Store نصب می‌کند. مزیت این روش، به‌روزرسانی خودکار است، اما ممکن است در برخی محیط‌های سازمانی محدودیت‌هایی داشته باشد.

توصیه: برای اکثر کاربران، دانلود از Download Center و انتخاب نسخه ۶۴ بیتی (اگر سیستم عامل شما ۶۴ بیتی است) بهترین گزینه است.

3. انتخاب زبان: پس از کلیک بر روی `Download from Download Center`، می‌توانید زبان موردنظر خود را انتخاب کنید. زبان فارسی نیز موجود است، اما برای یادگیری DAX و استفاده از منابع انگلیسی، ممکن است نسخه انگلیسی را ترجیح دهید.

4. شروع دانلود: بر روی `Download` کلیک کنید و فایل نصب را در یک مکان مناسب ذخیره کنید.

الف-۲: نصب Power BI Desktop

1. اجرای فایل نصب: پس از اتمام دانلود، فایل (`PBIDesktopSetup.exe` یا `PBIDesktopSetup_x64.exe`) را اجرا کنید.
2. مراحل نصب:
   - پنجره خوش‌آمدگویی را مشاهده خواهید کرد. بر روی `Next` کلیک کنید.
   - شرایط مجوز را مطالعه کرده و `I accept the terms in the License Agreement` را انتخاب کنید، سپس `Next`.
   - مکان نصب را انتخاب کنید. می‌توانید از مکان پیش‌فرض استفاده کنید یا یک مسیر دیگر را انتخاب کنید، سپس `Next`.

- بر روی `Install` کلیک کنید تا فرآیند نصب آغاز شود.
- پس از اتمام نصب، بر روی `Finish` کلیک کنید. می‌توانید گزینه `Launch Microsoft Power BI Desktop now` را فعال نگه دارید تا برنامه بلافاصله اجرا شود.

الف-۳: اولین نگاه به محیط Power BI Desktop

پس از باز کردن Power BI Desktop، با صفحه خوش‌آمدگویی (Welcome Screen) مواجه خواهید شد که گزینه‌هایی برای `Recent sources`، `Get data` و `Open other reports` را ارائه می‌دهد. می‌توانید این صفحه را ببندید تا وارد محیط اصلی Power BI Desktop شوید.

بخش‌های اصلی محیط **Power BI Desktop:**

1. نوار ریبون **(Ribbon):** در بالای صفحه قرار دارد و شامل تب‌های مختلفی مانند `Home`, `Insert`, `Modeling`, `View` و `Help` است. هر تب شامل گروه‌هایی از ابزارها و گزینه‌ها است.
2. نماهای اصلی **(Views):** در سمت چپ صفحه، سه آیکون اصلی وجود دارد:
   - **Report View (** نمای گزارش **):** جایی که گزارش‌ها و بصری‌سازی‌های خود را طراحی می‌کنید.
   - **Data View (** نمای داده **):** جایی که می‌توانید داده‌های خام جداول خود را مشاهده کنید، ستون‌های محاسباتی ایجاد کنید و نوع داده‌ها را تغییر دهید.
   - **Model View (** نمای مدل **):** جایی که روابط بین جداول را مدیریت می‌کنید Measures، ایجاد می‌کنید و RLS را پیاده‌سازی می‌کنید.
3. پنل‌های سمت راست**:**
   - **Filters (** فیلترها **):** برای اعمال فیلترها بر روی بصری‌سازی‌ها، صفحات یا کل گزارش.
   - **Visualizations (** بصری‌سازی‌ها **):** شامل انواع مختلف بصری‌سازی‌ها (نمودارها، جداول، کارت‌ها و غیره) که می‌توانید به گزارش خود اضافه کنید.
   - **Data (** داده‌ها **):** شامل تمام جداول و ستون‌های مدل داده شما. می‌توانید ستون‌ها را به بصری‌سازی‌ها بکشید و رها کنید یا Measures جدید ایجاد کنید.

اولین گام‌ها**:**

- **Get data:** برای اتصال به منابع داده مختلف. (Excel, SQL Server, Web, etc.)
- **Transform data:** برای باز کردن Power Query Editor و تمیز کردن و تبدیل داده‌ها.
- **New Measure / New Column / New Table:** برای ایجاد محاسبات DAX.

با این مقدمه، شما آماده‌اید تا سفر خود را در دنیای Power BI و DAX آغاز کنید!

پیوست ب: منابع و ابزارهای تکمیلیDAX

در طول این کتاب، به برخی از ابزارها و منابع مهم اشاره شد. در این پیوست، لیستی جامع‌تر از این منابع و ابزارها را ارائه می‌دهیم که به شما در ادامه مسیر یادگیری و تسلط بر DAX کمک خواهند کرد.

ب-۱: ابزارهای خارجی(External Tools)

Power BI Desktopاز ابزارهای خارجی پشتیبانی می‌کند که قابلیت‌های آن را گسترش می‌دهند. این ابزارها برای هر متخصص Power BIضروری هستند.

1. **DAX Studio:**
   - وب‌سایت: https://daxstudio.org/
   - کاربرد: اجرای کوئری‌های DAX، تحلیل عملکرد (Server Timings)، بررسی متریک‌های VertiPaq، فرمت‌بندی DAX.
   - چرا مهم است: ابزار شماره یک برای دیباگینگ و بهینه‌سازی فرمول‌های DAX.
2. **Tabular Editor:**
   - وب‌سایت: https://tabulareditor.com/
   - کاربرد: ویرایش مدل‌های Tabular (Power BI) به‌صورت برنامه‌نویسی، ایجاد Measures و ستون‌ها به‌صورت انبوه، اعمال بهترین روش‌ها (Best Practice Analyzer)، مدیریت پوشه‌ها و سلسله مراتب.
   - چرا مهم است: برای توسعه سریع و اعمال بهترین روش‌ها در مدل‌سازی.
3. **ALM Toolkit:**
   - وب‌سایت: https://almtoolkit.com/
   - کاربرد: مقایسه، ادغام و استقرار مدل‌های Power BI. برای مدیریت تغییرات در محیط‌های تیمی و کنترل نسخه بسیار مفید است.
   - چرا مهم است: برای توسعه تیمی و مدیریت چرخه عمر برنامه (Application Lifecycle Management).

ب-۲: منابع آموزشی آنلاین

1. **SQLBI (sqlbi.com):**

- وب‌سایت: https://sqlbi.com/
- توضیحات: منبع اصلی و معتبرترین منبع برای یادگیری DAX و مدل‌سازی داده. شامل مقالات، ویدئوها، کتاب‌ها و دوره‌های آموزشی (رایگان و پولی).
- توصیه: حتماً بخش DAX Guide و DAX Patterns را بررسی کنید.

2. **Microsoft Learn:**

- وب‌سایت: https://learn.microsoft.com/en-us/power-bi/
- توضیحات: مستندات رسمی مایکروسافت برای Power BI، شامل آموزش‌ها، راهنماها و مثال‌ها. برای شروع و درک مفاهیم پایه بسیار مفید است.

3. **Power BI Community:**

- وب‌سایت: https://community.powerbi.com/
- توضیحات: انجمن رسمی Power BI که می‌توانید سؤالات خود را بپرسید، به سؤالات دیگران پاسخ دهید و از تجربیات جامعه استفاده کنید.

4. **YouTube Channels:**

- **Guy in a Cube:** کانالی با ویدئوهای هفتگی در مورد Power BI، شامل نکات، ترفندها و به‌روزرسانی‌ها.
- **SQLBI:** کانال رسمی SQLBI با ویدئوهای آموزشی عمیق در مورد DAX و مدل‌سازی.
- **Curbal:** کانالی با آموزش‌های عملی و کاربردی در مورد Power BI و DAX.

ب-۳: کتاب‌ها

1. **The Definitive Guide to DAX (by Alberto Ferrari & Marco Russo):**

- توضیحات: این کتاب به‌عنوان "انجیل DAX" شناخته می‌شود و برای هر کسی که می‌خواهد به یک متخصص DAX تبدیل شود، ضروری است. مفاهیم را به‌صورت عمیق و با جزئیات زیاد توضیح می‌دهد.

2. **Power BI Desktop Step-by-Step (by Errin O'Connor):**

- توضیحات **:**برای شروع کار با Power BI Desktop و درک مفاهیم پایه، یک منبع عالی است.

ب-۴: وبلاگ‌ها و منابع دیگر

- **Radacad (radacad.com):**وبلاگی با مقالات بسیار مفید در مورد Power BI، DAX، Power Query و SQL Server.
- **Reid Havens (reidhavens.com):**وبلاگی با نکات و ترفندهای کاربردی در مورد Power BI.
- **DAX Patterns (daxpatterns.com):**مجموعه‌ای از الگوهای DAX برای حل مسائل رایج تحلیلی. بسیار مفید برای یادگیری نحوه پیاده‌سازی سناریوهای مختلف
- **Getbi.ir**سایت فروش داشبوردهای آماده برای دیتابیس‌های نرم افزارهای حسابداری.

ب-۵: تمرین و پروژه‌های عملی

بهترین راه برای یادگیری DAX، تمرین و کار بر روی پروژه‌های واقعی است. سعی کنید:

- داده‌های خود را تحلیل کنید **:**از داده‌های شخصی، کاری یا عمومی برای ساخت مدل‌ها و گزارش‌ها استفاده کنید.
- چالش‌های **DAX** را حل کنید **:**در انجمن‌ها یا وب‌سایت‌های چالش DAX شرکت کنید.
- پروژه‌های نمونه را بازسازی کنید **:**گزارش‌ها و مدل‌های نمونه را دانلود کرده و سعی کنید آن‌ها را از ابتدا بازسازی کنید.
- برای کارآموزی می‌توانید از طریق لینکدین نویسنده درج در معرفی گردآورنده کتاب اقدام نمایید.

با استفاده از این منابع و ابزارها، و با تمرین مداوم، می‌توانید مهارت‌های DAX خود را به طور چشمگیری افزایش دهید و به یک متخصص تحلیل داده در Power BI تبدیل شوید.

واژه‌نامه(Glossary)

این واژه‌نامه شامل اصطلاحات کلیدی است که در این کتاب مورد استفاده قرار گرفته‌اند:

- **DAX (Data Analysis Expressions):**زبان فرمول‌نویسی برای Power BI، SQL Server Analysis Services Tabular و Excel Power Pivot. برای ایجاد Measures، ستون‌های محاسباتی و جداول محاسباتی استفاده می‌شود.
- **Power BI Desktop:**ابزار رایگان مایکروسافت برای توسعه گزارش‌ها و مدل‌های داده Power BI.
- **Power Query:**ابزاری برای اتصال، تبدیل و تمیز کردن داده‌ها در Power BI.

- مدل داده **(Data Model):** ساختار داده‌ها در Power BI که شامل جداول، ستون‌ها، روابط و محاسبات است.
- جدول حقیقت **(Fact Table):** جدولی در مدل داده که شامل مقادیر عددی (Measures) و کلیدهای خارجی (Foreign Keys)است. معمولاً شامل تراکنش‌ها یا رویدادها است.
- جدول ابعاد **(Dimension Table):** جدولی در مدل داده که شامل ویژگی‌های توصیفی (Attributes) است که برای فیلترکردن و گروه‌بندی داده‌ها استفاده می‌شود (مثلاً محصولات، مشتریان، تاریخ.(
- **Star Schema (**مدل ستاره‌ای **:(**یک الگوی طراحی مدل داده که در آن یک جدول حقیقت مرکزی توسط جداول ابعاد احاطه شده است.
- **Measure:**یک محاسبه پویا که در زمان نیاز (بر اساس زمینه فیلتر) محاسبه می‌شود و نتیجه عددی را برمی‌گرداند (مثلاً مجموع فروش، میانگین سود.(
- ستون محاسباتی **(Calculated Column):** یک ستون جدید که مقادیر آن با استفاده از یک فرمول DAX محاسبه می‌شوند و در مدل داده ذخیره می‌شوند.
- جدول محاسباتی **(Calculated Table):** یک جدول جدید که با استفاده از یک فرمول DAX ایجاد می‌شود.
- زمینه فیلتر **(Filter Context):** مجموعه‌ای از فیلترهایی که در حال حاضر بر روی مدل داده اعمال شده‌اند و بر نتیجه Measures تأثیر می‌گذارند.
- زمینه ردیف **(Row Context):** یک مفهوم موقتی که زمانی ایجاد می‌شود که DAX در حال ارزیابی یک عبارت ردیف به ردیف است (مانند ستون‌های محاسباتی یا توابع.(Iterator
- **Context Transition (**انتقال زمینه **:(**فرآیندی که در آن زمینه ردیف به زمینه فیلتر تبدیل می‌شود، معمولاً زمانی که یک تابع Iterator در داخل `CALCULATE` استفاده می‌شود.
- **Iterator Function (X-Function):**تابعی که یک عبارت را برای هر ردیف از یک جدول ارزیابی می‌کند و سپس نتیجه را تجمیع می‌کند (مثلا.(`SUMX`, `AVERAGEX`ً
- **Table Function:**تابعی که یک جدول را به‌عنوان خروجی برمی‌گرداند (مثلا `FILTER`, `ALL`, ً `SUMMARIZECOLUMNS`).
- **Time Intelligence (**هوش زمانی **:(**قابلیت‌های DAX برای انجام تحلیل‌های مبتنی بر زمان (مانند YTD، مقایسه با سال گذشته.(
- **Row-Level Security (RLS):**قابلیتی در Power BI که دسترسی کاربران به ردیف‌های داده را بر اساس نقش‌ها و قوانین DAX محدود می‌کند.
- **VertiPaq:**موتور ذخیره‌سازی ستونی در حافظه که توسط Power BI برای ذخیره و پردازش داده‌ها استفاده می‌شود.
- **DAX Studio:**یک ابزار خارجی برای تحلیل و بهینه‌سازی فرمول‌های.DAX
- **Tabular Editor:**یک ابزار خارجی برای ویرایش مدل‌های Power BI به‌صورت برنامه‌نویسی و اعمال بهترین روش‌ها.

- **Many-to-Many Relationship (**رابطه چند به چند **:(**رابطه‌ای که در آن یک رکورد در یک جدول می‌تواند به چندین رکورد در جدول دیگر مرتبط باشد و بالعکس.
- **Bridge Table (**جدول پل **:(**یک جدول میانی که برای تبدیل روابط Many-to-Many به روابط One-to-Many استفاده می‌شود.
- **Role-Playing Dimension (**بعد نقش‌آفرین **:(**یک جدول ابعاد واحد که چندین بار در یک جدول حقیقت با نقش‌های متفاوت استفاده می‌شود.
- **Snapshot Table (**جدول **Snapshot):** جدولی که وضعیت یک سیستم را در یک نقطه زمانی خاص ذخیره می‌کند.
- **Factless Fact Table (**جدول حقیقت بدون مقدار **:(**جدولی که فقط شامل کلیدهای خارجی است و هیچ Measure عددی ندارد و برای ردیابی رویدادها استفاده می‌شود.
- **USERELATIONSHIP:**تابع DAX برای فعال کردن موقت یک رابطه غیرفعال.
- **CROSSFILTER:**تابع DAX برای تغییر جهت فیلتر یک رابطه یا غیرفعال کردن آن.
- **USERPRINCIPALNAME():**تابع DAX که نام کاربری (UPN) کاربر وارد شده را برمی‌گرداند.
- **CALCULATETABLE:**تابع DAX که یک جدول را برمی‌گرداند و زمینه فیلتر را تغییر می‌دهد.
- **DATESBETWEEN:**تابع هوش زمانی DAX که یک جدول از تاریخ‌ها را بین دو تاریخ مشخص برمی‌گرداند.
- **DATESINPERIOD:**تابع هوش زمانی DAX که یک جدول از تاریخ‌ها را در یک دوره مشخص (مثلاً ۱۲ ماه گذشته) برمی‌گرداند.
