Overview

The dataset represents the core operational and sales data of an e-commerce platform similar to Olist. It integrates multiple aspects of the business — customers, sellers, products, orders, payments, marketing, and employee interactions — into a unified structure suitable for analytical and relational database modeling.

The data contains several interrelated tables, including:

Customers and Geolocation, which describe user information and their regional mapping.

Orders, Order Items, and Order Payments, which record transaction-level details from purchase to delivery.

Products and Sellers, representing the supply side of the platform.

Closed Deals and Marketing Qualified Leads (MQLs), capturing sales and marketing performance data.

Employees, describing company staff involved in sales or lead handling.

Together, these entities provide a complete view of the customer journey — from marketing lead acquisition to order fulfillment — enabling advanced analytics on sales performance, delivery efficiency, product demand, and business operations.

ERD Process

To design the Entity Relationship Diagram (ERD), we followed a structured process:

Data Exploration:

We examined all tables and their columns to understand their attributes, data types, and constraints such as primary and foreign keys.

Entity Identification:

Each table in the dataset was treated as an entity:

Customers, Geolocation, Orders, Order Items, Order Payments, Products, Sellers, Closed Deals, Marketing Qualified Leads, Employees, and Reviews.

Primary & Foreign Keys Mapping:

Primary keys were identified (e.g., order_id, customer_id, seller_id), and foreign key relationships were traced to connect entities logically (e.g., customer_id links Customers and Orders).

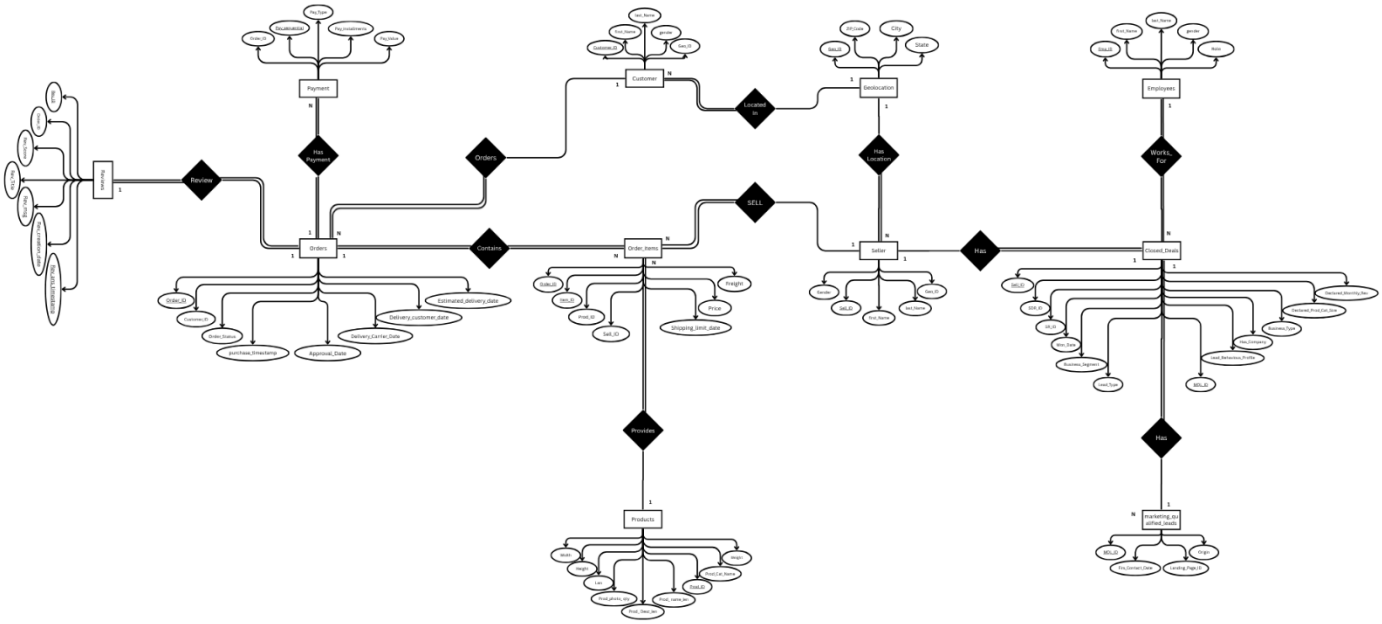
Cardinality and Participation:

We defined how entities relate to each other based on the relationship table:

- One-to-Many: e.g., one customer can have many orders.
- One-to-One: e.g., each order has one review.
- Optional vs. Mandatory participation was also specified as “May” or “Must”.

ERD Design:

Using the identified relationships, we created a visual ERD illustrating how data flows between tables — showing relationships between marketing, sales, and delivery stages in a cohesive system.



Physical Mapping

1-Location (geo_id PK, zip_code, latitude, longitude, city, state)

2-Customers (customer_id PK, customer_unique_id, customer_name, gender, geo_id FK)

3-Sellers (seller_id PK, seller_first_name ,seller_last_name , gender, geo_id FK)

4-Products (product_id PK, product_category_name, product_name_length, product_description_length, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm)

5-Orders (order_id PK, customer_id FK, order_status, purchase_timestamp, order_approved_at, delivered_carrier_date, delivered_customer_date, estimated_delivery_date,review_id, review_score, review_comment_title, review_comment_message, creation_date, answer_timestamp)

6-Order_Items ((order_id FK, order_item_id) PK, product_id fk, seller_id fk, shipping_limit_date, price, freight_value)

7- Payments (payment_id,(order_id FK, payment_sequential)PK, payment_type, payment_installments, payment_value)

8-Employees (employee_id PK, employee_name, role FK)

9- Marketing_Qualified_Leads (mql_id PK, first_contact_date, landing_page_id, origin)

10- Closed_Deals (mql_id PK FK, seller_id FK, sdr_id, sr_id, won_date, business_segment, lead_type, lead_behavior_profile, has_company, average_stock, business_type, product_catalog_size, declared_monthly_revenue)

Data Implementation

After finalizing the Mapping, the next step was to implement the database structure on **Microsoft SQL Server**. The goal was to translate the logical design into a physical database that maintains referential integrity, supports data analysis, and reflects real business operations.

1. **Database Creation**
   We started by creating a new database named **Olist_DB** on SQL Server using the CREATE DATABASE statement. This served as the container for all the related tables and data objects.
2. **Table Creation**
   Each entity from the ERD was then implemented as a table.
   We defined **primary keys** and **foreign keys** to maintain relationships, along with appropriate **data types** and **constraints** such as NOT NULL, CHECK, and DEFAULT where necessary.

```
CREATE TABLE [dbo].[closed_deals](
    [mql_id] [nvarchar](255) NOT NULL,
    [seller_id] [nvarchar](255) NULL,
    [sdr_id] [nvarchar](255) NULL,
    [sr_id] [nvarchar](255) NULL,
    [won_date] [date] NULL,
    [business_segment] [nvarchar](255) NULL,
    [lead_type] [nvarchar](255) NULL,
    [lead_behavior_profile] [nvarchar](255) NULL,
    [has_company] [bit] NULL,
    [business_type] [nvarchar](255) NULL,
    [declared_monthly_revenue] [decimal](18, 2) NULL,
    [product_catalog_size] [int] NULL,
PRIMARY KEY CLUSTERED
(
    [mql_id] ASC
```

3. **Relationship Implementation**
   Based on the ERD, foreign key constraints were added to link related tables such as:
   - customer_id in **Orders** referencing **Customers**
   - order_id in **Order_Items** and **Order_Payments** referencing **Orders**
   - product_id in **Order_Items** referencing **Products**
   - seller_id in **Order_Items** referencing **Sellers**

These relationships ensured **referential integrity** across the database and prevented inconsistent or orphaned records.

Data Preparation and Generation

After completing the data model for the database and creating the tables, the next step is to validate the Olist dataset and ensure that it fits our model. In this part, we will check the columns that serve as primary keys to confirm that they are unique and not null. We will also examine the cardinality and verify that the relationships between the tables in the Olist dataset align with those defined in our data model. Any issues that violate the database constraints will be identified and resolved. Afterward, we will generate data for the additional columns we included in our data model that do not exist in the Olist dataset. Finally, we will enrich some columns in the Olist dataset that contain null values and could provide valuable insights for our analysis.

Data Assessing

In this part, we examined each table in the Olist dataset to identify key information such as the number of rows, the number of columns, the number of duplicates, and the number of null values, along with which columns contain those nulls.

	dataset	n_rows	n_cols	n_duplicates	null_counts	null_cols
0	closed_deals	842	14	0	3300	business_segment, lead_type, lead_behaviour_profile, has_company, has_gtin, average_stock, business_type, declared_product_catalog_size
1	customers	99441	5	0	0	
2	geolocation	1000163	5	261831	0	
3	marketing_qualified_leads	8000	4	0	60	origin
4	orders	99441	8	0	4908	order_approved_at, order_delivered_carrier_date, order_delivered_customer_date
5	order_items	112650	7	0	0	
6	order_payments	103886	6	0	0	
7	order_reviews	99224	7	0	145903	review_comment_title, review_comment_message
8	products	32951	9	0	2448	product_category_name, product_name_lenght, product_description_lenght, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm
9	sellers	3095	4	0	0	
10	product_category_name_translation	71	2	0	0	

We found that the only table containing duplicates is the geolocation table, and there are null values in the closed_deals, orders, order_reviews, and products tables. In the next steps, we will remove the

duplicates and check whether any of the null values exist in candidate primary key columns so we can handle them properly.

After that, we looked at each table to see which columns have missing values. The closed_deals table has several columns with more than 90% null values, such as lead_behaviour_profile, has_company, and declared_product_catalog_size. The order_reviews table has the review_comment_title column, where about 88% of the values are null, and review_comment_message, where 58.7% of the values are null.

Also, in this step, we looked at each column that will serve as a primary key and verified that none of them have missing values.

Primary Key Integrity Check

In this part, we focus on verifying the **uniqueness** and **completeness** of primary keys. We already confirmed that all primary key columns do not contain missing values. To check their uniqueness, I applied the same logic shown in the image below to each primary key column.

```
datasets['orders'].duplicated(subset='order_id').sum()
# Primary key (order_id)
```

```
0
```

If the result is **0**, it means there are no duplicates, and the column can be used as a primary key.

Primary Key Validation and Corrections:

- All candidate primary key columns were unique, except for the geolocation_zip_code_prefix column in the geolocation table, which contained 719,317 duplicate values.
- Since the geolocation, customers, and sellers datasets shared three common columns: geolocation_zip_code_prefix, city, and state, and our data model did not include latitude or longitude, I decided to drop those two columns.
- After that, I checked the customers and sellers datasets to see if they contained any records that did not exist in the geolocation dataset. Any missing records were appended to the geolocation dataset as part of a data enrichment step to ensure there were no missing values when performing joins.
- Then, I removed the duplicate rows from the geolocation dataset so that it contained a unique combination of geolocation_zip_code_prefix, city, and state.
- After that, I created a new primary key column named geo_id.

- Finally, I joined the geolocation dataset twice, once with the customers table and once with the sellers table, using the three shared columns to add the geo_id field. Once this was done, I dropped the geolocation_zip_code_prefix, city, and state columns from both the customers and sellers datasets.

```
# drop duplicate rows
datasets['geolocation'].drop_duplicates(inplace=True)

# test
datasets['geolocation'].duplicated().sum()
```

```
0
```

```
datasets['geolocation'].shape
```

```
(28221, 3)
```

```
# create a new primary key column `geo_id`
datasets['geolocation']['geo_id'] = list(range(0, datasets['geolocation'].shape[0]))
```

Checking Cardinality Constraint

Orders and Reviews

In our data model, the relationship between the reviews and orders tables was designed to be one-to-one with total participation from both sides. Therefore, I joined the reviews and orders tables to align with this design.

Before performing the join, I examined the order_id column in the reviews table to check if it was unique, but I found that there were multiple reviews for the same order. This issue needed to be handled before joining the two tables.

```
datasets['order_reviews'].order_id.value_counts()
```

```
order_id
c88b1d1b157a9999ce368f218a407141    3
8e17072ec97ce29f0e1f111e598b0c85    3
df56136b8031ecd28e200bb18e6ddb2e    3
03c939fd7fd3b38f8485a0f95798f1f6    3
5cb890a68b91b6158d69257e4e2bc359    2
                                   ..
5b4e9a12d219f34f5c2de9f8d620b19d    1
a6da096d974acc000962856d7386448a    1
75e0647c26de647eca3421e9cc66c9da    1
bad0467c52f23cdc71e9fa139d4a8afd    1
90531360ecb1eec2a1fbb265a0db0508    1
Name: count, Length: 98673, dtype: int64
```

To align with the data model, I kept only the review with the most recent review_answer_timestamp, as it represented the customer's final feedback and reflected the most accurate impression.

Customers and Orders

The relationship between the customers table and the orders table in the data model was designed to be one-to-many. However, in the Olist dataset, it was one-to-one because a new record was added for each customer whenever they made an order. There was also another column named customer_unique_id that tracked each unique customer.

To fix this, I joined the two tables on the customer_id column to bring the customer_unique_id into the orders table.

```
datasets['orders'] = datasets['orders'].merge(datasets['customers'][['customer_id', 'customer_unique_id']],
                                              on='customer_id', how='left')
```

After that, I dropped the customer_id column from both tables and removed duplicates from the customers table to ensure that each customer was represented only once.

```
datasets['customers'].drop('customer_id', axis=1, inplace=True)
datasets['orders'].drop('customer_id', axis=1, inplace=True)
```

```
print(('customer_id' in datasets['customers'].columns) | ('customer_id' in datasets['orders'].columns))
```

```
False
```

```
datasets['customers'].duplicated().sum()
```

```
3089
```

```
datasets['customers'].drop_duplicates(subset='customer_unique_id', inplace=True)
```

```
datasets['customers'].duplicated().sum()
```

```
0
```

Next, I checked the relationship between the sellers and closed_deals tables to make sure it was one-to-one, as indicated in the data model, and I confirmed that it was correct.

Data Generation and Enrichment

In this part, we added new columns from other tables to align with our data model. We also used a Python package named Faker to generate new data for the columns that were newly created and did not exist in the Olist dataset. In addition, we filled in missing values in existing columns that were considered useful for the analysis.

Products Table

First, I obtained the English name of each product category by joining the products table with the product_category_name_translation table.

Join `products` and `product_category_name_translation`

```
datasets['products'] = datasets['products'].merge(datasets['product_category_name_translation'],
                                                  on='product_category_name', how='left')
```

```
datasets['products'].drop('product_category_name', axis=1, inplace=True)
```

```
datasets['products'] = datasets['products'].rename(columns={'product_category_name_english': 'product_category_name'})
```

Closed_Deals Table

We addressed the data integrity issue where some sellers existed in the closed_deals table but not in the main sellers table. We imputed these missing seller records using data from valid sellers found in the sellers table.

We also filled in the missing values in the declared_product_catalog_size column for each seller by calculating their actual count of unique products sold.

In addition, we filled in the missing values in the declared_monthly_revenue column for each seller using their total revenue, which we calculated as the sum of price and freight_value for all items they sold and multiplied it by a random number to look like a different value than the one we calculated from the Olist dataset.

Customers Table

We generated data for the customer_name and customer_gender columns to enrich the customers table and make it more suitable for analysis.

```
datasets['customers']['first_name'] = df_customer_generated['first_name']
datasets['customers']['last_name'] = df_customer_generated['last_name']
datasets['customers']['gender'] = df_customer_generated['gender']

datasets['customers'].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96096 entries, 0 to 96095
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   index               96096 non-null  int64 
 1   customer_unique_id  96096 non-null  object
 2   geo_id              96096 non-null  int64 
 3   first_name          96096 non-null  object
 4   last_name           96096 non-null  object
 5   gender              96096 non-null  object
dtypes: int64(2), object(4)
memory usage: 4.4+ MB
```

Sellers Table

We generated data for the seller_name and seller_gender columns to complete the missing information and enhance the dataset for future reporting and analysis.

Employees Table

We created a new employee's table based on the sales representatives and sales development representatives' IDs extracted from the data. We then generated names for those employees and added Facebook page URLs for some of the sales representatives.

```
df_emps = generate_names(41, locale='en_US', seed=123)
df_emps['employee_id'] = emps_ids
df_emps['role'] = emps_roles
print(df_emps.full_name.duplicated().sum())
```

```
0
```

Using Quadratic for Data Enrichment

To create realistic product data for this project, I used **Quadratic AI** to generate additional attributes for an existing dataset containing product IDs. I provided the model with a CSV file listing the product IDs and a set of instructions specifying the desired number of products per category. The AI was prompted to generate corresponding **product names** and **brands** while maintaining the required **category distribution**. This approach ensured that the resulting dataset was both diverse and balanced across categories, making it suitable for analysis and visualization.

Prompt:

I have a file that contains two columns:

- The first column is the **product category name**.
- The second column is the **number of products** required in that category.

I want you to generate **real, existing product names** (not generic ones) for each category.

Please follow these rules carefully:

- Use **authentic, real-world product names** that actually exist and belong to well-known brands.

- The number of products for each category must exactly match the number in the “product count” column.
- Choose products that are **commonly sold in global or local markets**, depending on what fits the category.
- Avoid repeating product names.
- Output the result as a **CSV-style table** with two columns:
  Category Name | **Product Name**

Similarly, for the customer data, I used the same AI tool to generate **customer ages** based on an existing customer CSV file, providing a specific **age range** to ensure the generated values remained realistic and consistent with typical demographic patterns. This process resulted in balanced, coherent datasets suitable for analysis and visualization.

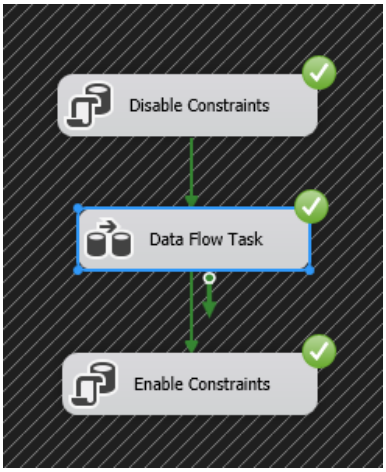
Prompt:

I have a customer dataset, and I want to add a new column called **Age**.
Please generate a realistic age for each customer, randomly distributed between **21 and 65 years old**.

Keep all other data in the same order, and just add the new **Age** column at the end.
Output the result in the same table format (CSV style).

Loading the data into the database

After we completed the database model, we prepared the data and implemented the database. Now, it is time to load the data into the database and to do that I used SQL Server Integration Service (SSIS). The results of the data preparation were 10 Excel files as shown in the image below, one for each table in the database **schema**. I saved the data in Excel format to avoid any problems that may arise because of the data type in SSIS. In SSIS, I created a package to load the data from each file to its intended table in the database.



First, I added an Execute SQL Task Component to disable database constraints and truncate the tables from any data. Then, I added a dataflow which contains **10 separate flows** to load the data from an Excel source to an **OLE DB Destination**. At the end, I added an Execute SQL Task to enable the database constraints.

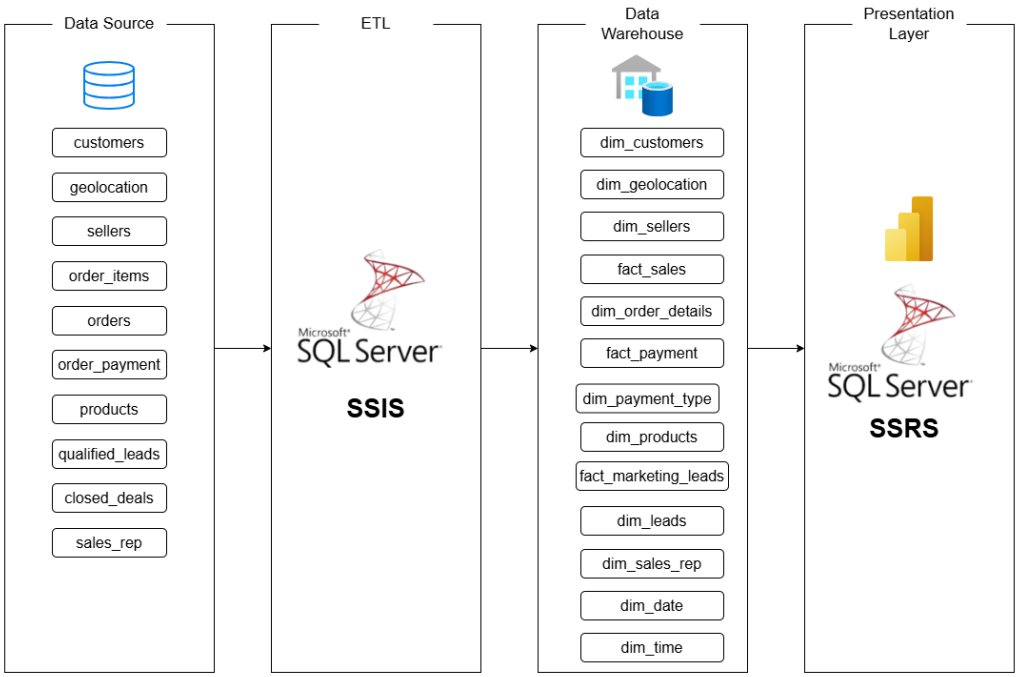
Data Warehouse Implementation

For this project, we first needed to design the high-level architecture. This plan illustrated how all the different parts of the system were connected, starting from the original raw data and ending with the final reports.

Data Architecture

I decided to use a four-layer architecture following Kimball Approach. The diagram I created shows the complete flow of data through the system. Below is a simple breakdown of what each layer does:

1. **Data Source:** This was the starting point. It represented the OLTP database of the Olist system.
2. **ETL (Extract, Transform, Load):** In this layer, I used SQL Server Integration Services (SSIS) to build packages that extracted, transformed, and loaded the data into the data warehouse.
3. **Data Warehouse:** This was the central database where the transformed data was stored after being processed by SSIS. I designed the schema for this data warehouse using a Galaxy Schema that included three fact tables and several shared dimension tables.
4. **Presentation Layer:** In this final layer, I used SQL Server Reporting Services (SSRS) and Power BI to connect to the data warehouse to build reports and dashboards to present the results clearly and effectively.



Data Warehouse Schema

To design the data warehouse, I began by analyzing the Olist database tables to determine which ones serve as **fact tables** (storing measurable business events) and which ones serve as **dimension tables** (storing descriptive context). Based on the business processes, I developed **three data marts**, **Sales**, **Payments**, and **Marketing Leads**, each designed as a star schema or snowflake schema depending on the relationships among dimensions. When combined, they form a **Galaxy Schema** that supports cross-functional analysis across sales, payments, and marketing performance.

Sales Data Mart

Fact Sales

This fact table stores one record per item sold, enabling analysis of key performance metrics such as item price, freight value, review score, and delivery time.
It was created by joining the orders and order_items tables.
Although the table includes two levels of granularity (order and item), this issue was handled in the presentation layer to ensure accurate measure calculations.

Dimensions

Order Details Dimension

Tracks descriptive information about orders, such as order status, review title, and review message.
This dimension is shared between the Sales Data Mart and the Payments Data Mart, ensuring consistency across analyses.

Products Dimension

Contains descriptive details about the products sold by sellers on the Olist e-commerce platform, including product name, brand, and category name.

Customers Dimension

Stores descriptive information about customers who placed orders, such as name, gender, age, and location.

Sellers Dimension

Holds descriptive details about sellers, including name, gender, and location.
This dimension is shared between the Sales Data Mart and the Marketing Data Mart to allow seller-level performance analysis across both domains.

Geolocation Dimension

Contains geographical information about states and cities where customers and sellers are located.

Time Dimension

Tracks the time of day when orders are placed, allowing analysis of peak or rush hours throughout the day.

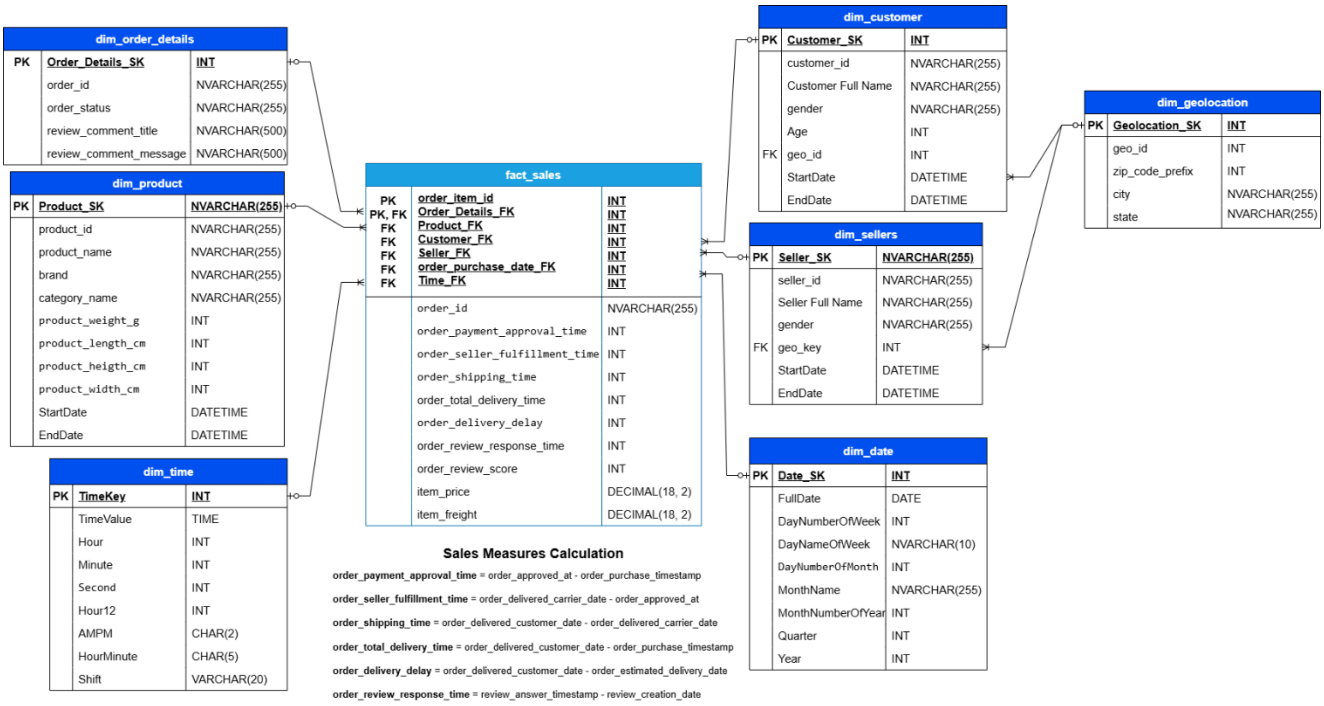
Date Dimension

Tracks the date when an order was placed and the date when a deal was closed with a seller.
It is a shared dimension between the Sales Data Mart and the Marketing Data Mart, enabling time-based analysis across both areas.

Sales Calculated Measures

- **order_payment_approval_time** = order_approved_at - order_purchase_timestamp
  Measures how long it takes to approve a customer's payment.
- **order_seller_fulfillment_time** = order_delivered_carrier_date - order_approved_at
  Tracks the time sellers take to prepare and ship an order after approval.
- **order_shipping_time** = order_delivered_customer_date - order_delivered_carrier_date
  Measures the duration of shipping from carrier pickup to customer delivery.
- **order_total_delivery_time** = order_delivered_customer_date - order_purchase_timestamp
  Calculates the total time from order placement to delivery.
- **order_delivery_delay** = order_delivered_customer_date - order_estimated_delivery_date
  Shows how late or early the order was delivered compared to the estimated date.
- **order_review_response_time** = review_answer_timestamp - review_creation_date
  Measures the time taken to respond to a customer review.

The image below shows the Sales Data Mart



Payment Data Mart

Fact Payment

This fact table captures each payment transaction, allowing analysis of payment value and installment information.

Dimensions

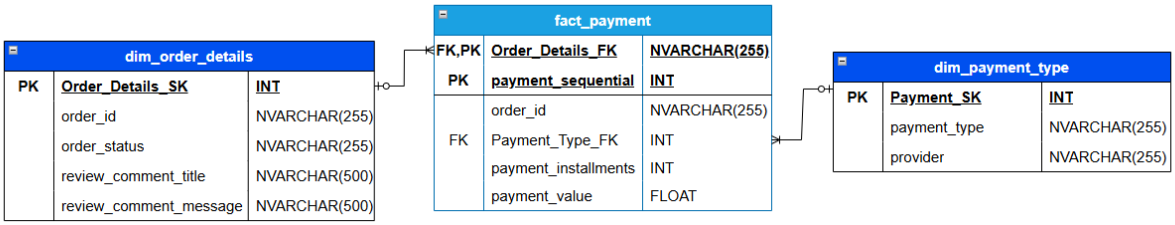
Order Details Dimension

As previously described, this dimension is shared between the Sales Data Mart and the Payment Data Mart to maintain consistency across both analyses.

Payment Type Dimension

Contains descriptive information about the payment methods used by customers, including the payment type and its associated provider.

The image below shows the Payment Data Mart



Marketing Data Mart

Fact Marketing Leads

This fact table tracks the marketing funnel, allowing me to measure conversion rates, average time to close a deal, and evaluate the performance of marketing channels and sales representatives.

Leads Dimension

Contains descriptive information about the leads such as the lead type, lead behavior profile, business type, business segment, and origin.

Sales Rep Dimension

Contains descriptive information about the sales representative responsible and sales development representative such as name, gender, role, Facebook user id.

Sellers, Geolocation and Date Dimensions

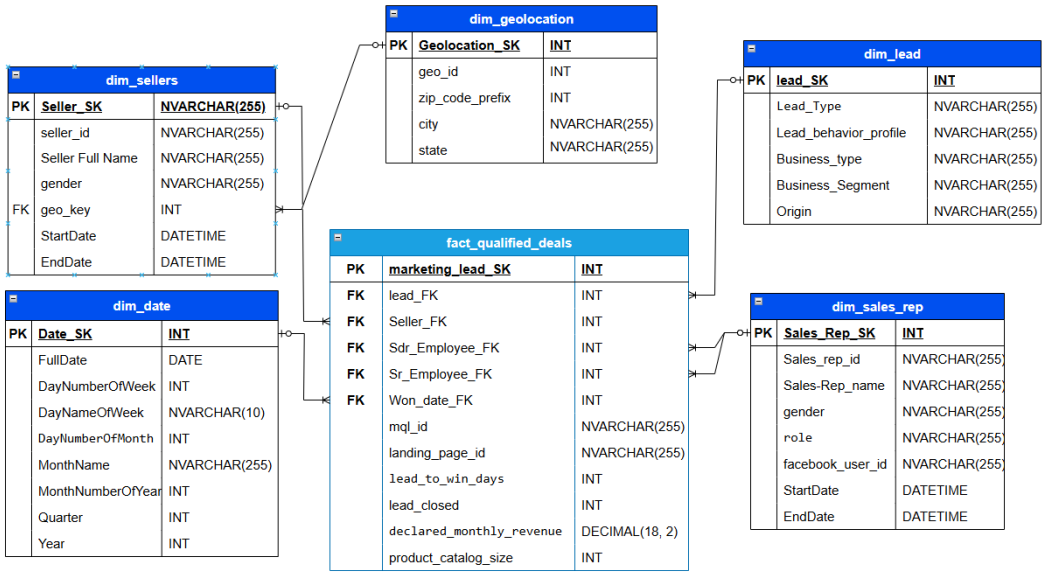
Common dimensions between the Sales Data Mart and the Marketing Data Mart.

Marketing Calculated Measures

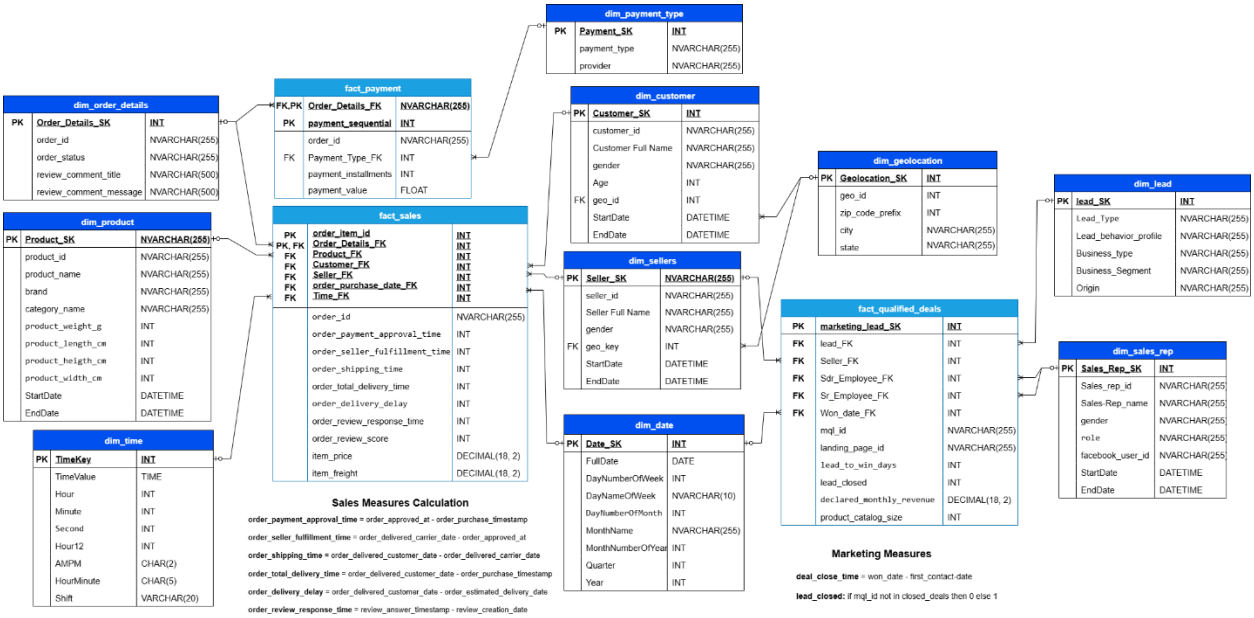
- **deal_close_time** = *won_date − first_contact_date*
  Measures how long it takes to close a deal from the first contact date.

- **lead_closed** = *IF(mql_id NOT IN closed_deals, 0, 1)*
  Indicates whether a lead was successfully closed (1) or not (0).

The image below shows the Marketing Data Mart



After completing the data marts, I connected them using the shared dimensions to form the final schema for the entire data warehouse.



ETL Process (SSIS)

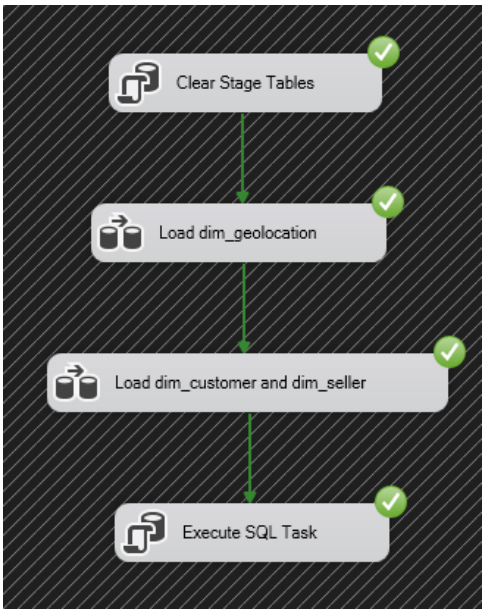
We used SQL Server Integration Services (SSIS) to extract, transform, and load the data into the database. I created a separate package for each dimension or group of related dimensions, as well as for each fact table. Then, I built a master package that runs all of them in the correct order. In this master package, the dimensions are loaded first, followed by the facts, since the fact tables rely on

the dimensions to retrieve their surrogate keys. I'm going to explain the purpose of each package and what it does.

ETL Package: Dim_Geolocation_Sellers_Customers.dtsx

This package is responsible for loading three dimensions: dim_geolocation, dim_customers, and dim_sellers. Since there is a relationship between dim_geolocation and the other two tables, dim_geolocation must be loaded first.

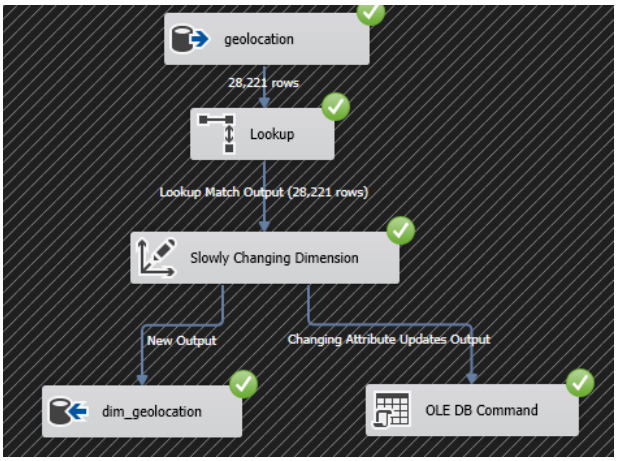
Control Flow



1. **Execute SQL Task:** Runs a SQL script to clear the data in the staging tables as part of the Slowly Changing Dimension (SCD) logic used to load dim_customers.
2. **Data Flow Task:** Extracts, transforms, and loads the data into dim_geolocation.
3. **Data Flow Task:** Extracts, transforms, and loads the data into dim_customers and dim_sellers.
4. **Execute SQL Task:** Runs a SQL script to insert the data from the staging tables into dim_customers.

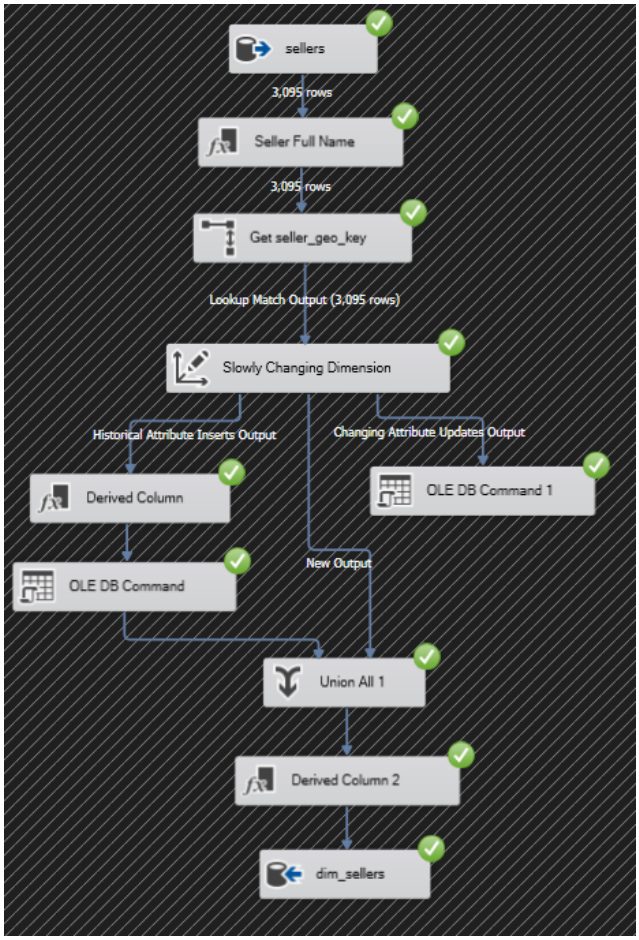
Data Flow (Load dim_location)

In this Data Flow, we extracted the data from the **geolocation** table and then used a Lookup component to get the full state name and then we used a **Slowly Changing Dimension (SCD)** component to handle changing attributes. The **state** and **city** attributes were treated as changing attributes so that if their names change, the SCD component will automatically handle the update logic.



Data Flow (Load dim_customers and dim_sellers)

In this Data Flow, we loaded two dimensions. In the **dim_sellers** table, we treated **geo_id** as a historical attribute and **seller_name** as a changing attribute. We used an **SCD** component to handle the logic for tracking these changes.

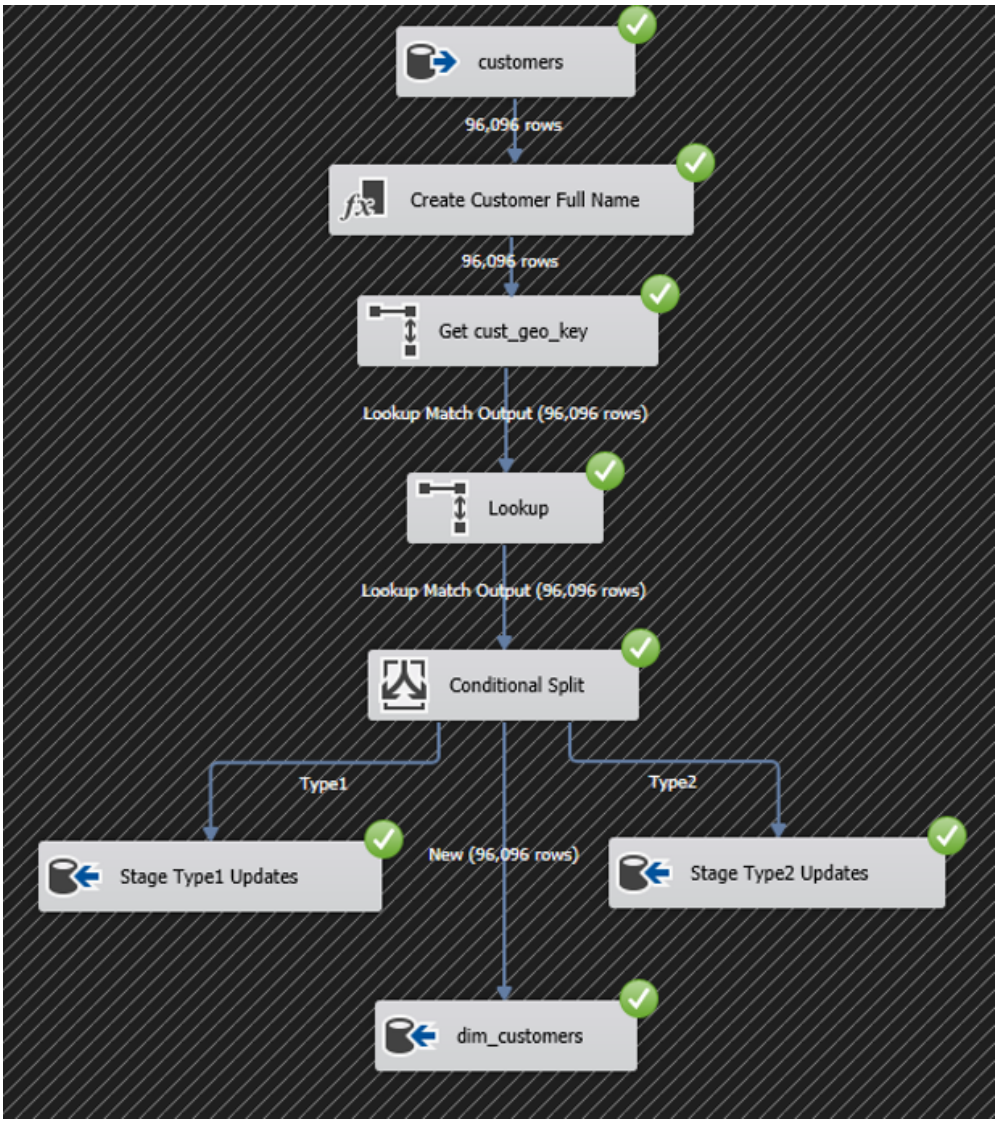


For the **dim_customers** table, I used a different approach to handle the different types of attributes. The **customers** table contains around 100,000 rows, and the standard **SCD** component processes data row by row, which makes it slow for large datasets.

Instead, I used a **Lookup** component to compare the incoming data with the existing records in the dimension. Then, I applied a **Conditional Split** component to classify the records based on the type of change (Type 1 or Type 2).

I created two **staging tables** to temporarily store the data for Type 1 and Type 2 attributes, using **OLE DB Destination** components for faster bulk loading instead of the **OLE DB Command**, which processes each row individually.

Finally, the new data was loaded into the dimension directly using **OLE DB Destination**, with the data access mode set to **Table or View – Fast Load** to enable batch inserts and improve performance. The updated data will be loaded using an Execute SQL task at the end of the control flow.



ETL Package: Dim_Products.dtsx

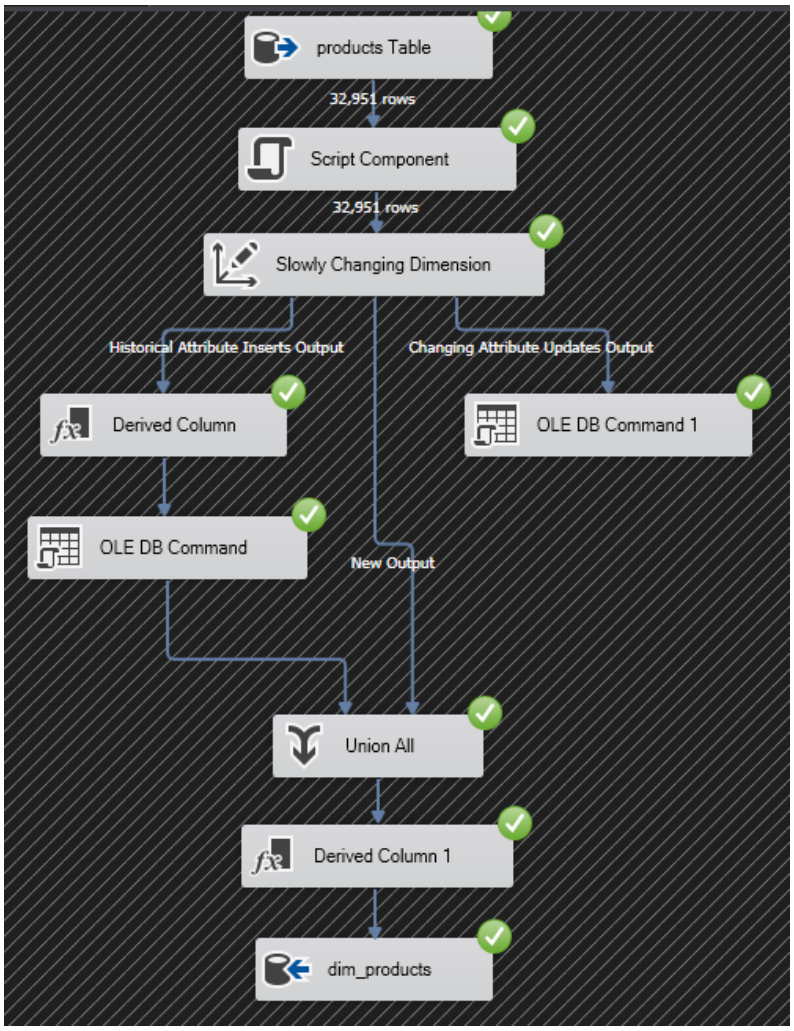
This package is responsible for loading the **dim_products** table.

Control Flow

It contains a single **Data Flow** task that extracts, transforms, and loads the data from the **products** table into **dim_products**.

Data Flow

We used a **Script Component** to convert attribute values from lowercase to title case. In this dimension, the **product_category** and **brand** attributes were treated as **historical attributes**, while the remaining attributes were considered **changing attributes**. An **SCD** component was used to handle this logic.

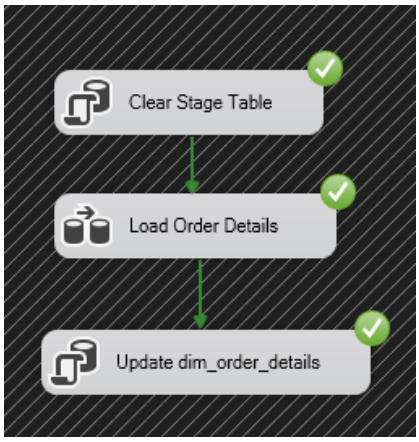


ETL Package: Dim_Order_Details.dtsx

The purpose of this package is to load the descriptive attributes of the orders from the **orders** table.

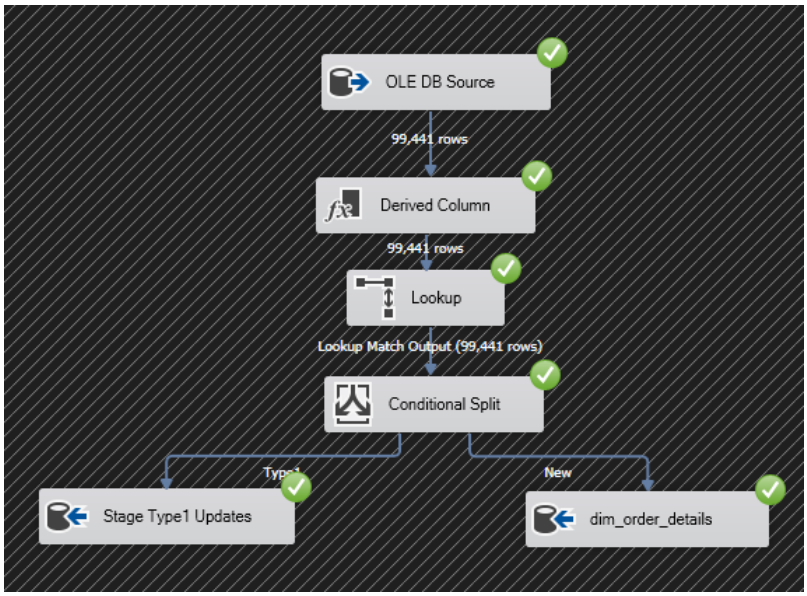
Control Flow

1. **Execute SQL Task:** Clears the staging table used to store Type 1 attribute changes.
2. **Data Flow Task:** Extracts, transforms, and loads the data into the **dim_order_details** table.
3. **Execute SQL Task:** Updates the **dim_order_details** table if there are any changes to existing records.



Data Flow

All attributes in this dimension were treated as **changing attributes**. Since the table contains a large number of rows and the **SCD** component would be slow, we implemented the same logic used earlier for the **dim_customers** package to handle updates efficiently.

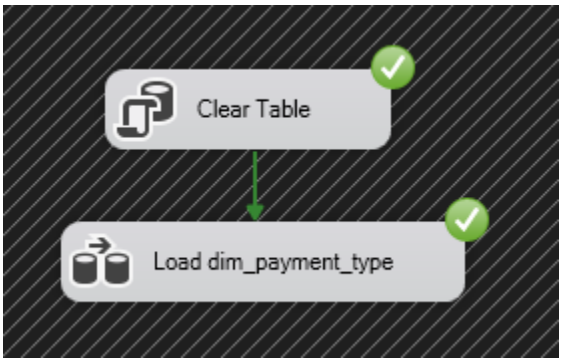


ETL Package: Dim_Payment_Type.dtsx

The purpose of this package is to load descriptive information related to payment transactions, such as **payment type** and **provider**, from the **order_payments** table.

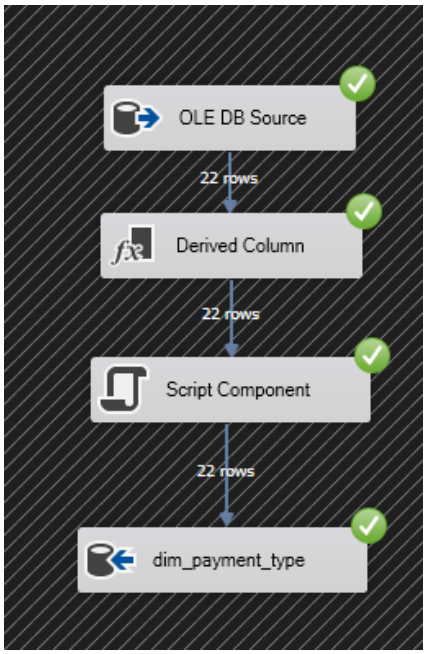
Control Flow

1. **Execute SQL Task:** Clears the **dim_payment_type** table before loading the data, since this dimension is fully reloaded each time.
2. **Data Flow Task:** Extracts, transforms, and loads the data into the **dim_payment_type** table.



Data Flow

We applied transformations to clean and standardize the column values, such as removing underscores and converting text to **Title Case** for consistency.



ETL Package: Dim_Sales_Rep.dtsx

The purpose of this package is to load sales representative data from the sales_rep source table into the dim_sales_rep dimension table.

Control Flow

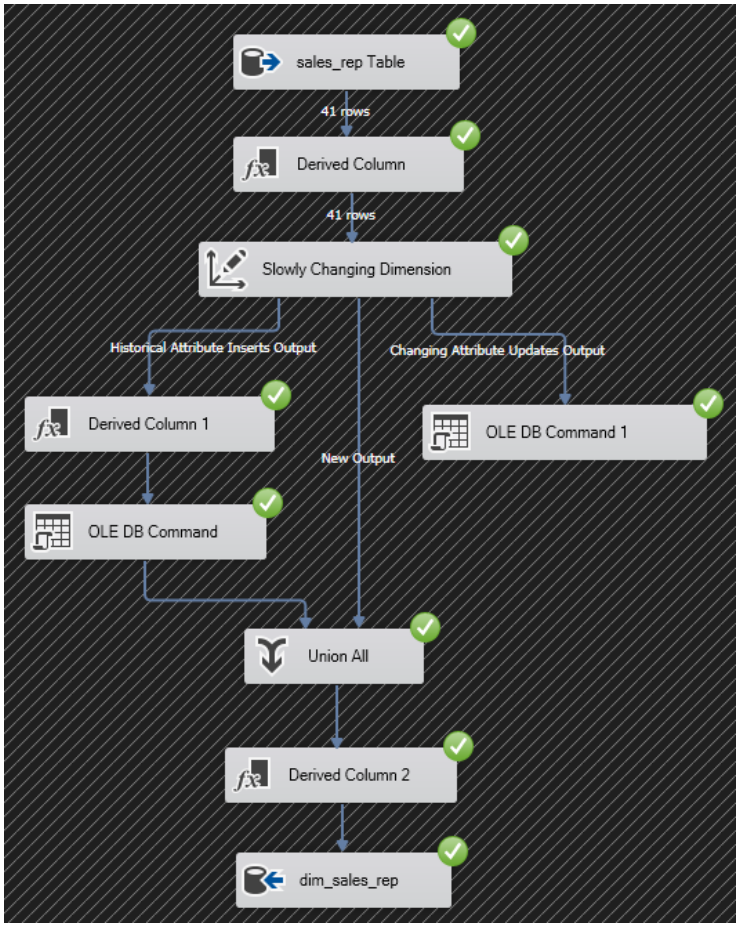
The package contains a single **Data Flow Task** responsible for extracting, transforming, and loading the data into the dim_sales_rep table.

Data Flow

Derived Column Transformation:
A new column was created to combine the first and last names into a single **Full Name** field for better readability and analysis.

Slowly Changing Dimension (SCD) Component:

The **Name** and **Facebook User ID** were treated as **changing attributes**. The **Role** was treated as a **historical attribute** to preserve history whenever it changes. The **SCD component** was used to implement this logic, ensuring proper handling of both Type 1 and Type 2 changes.



ETL Package: Dim_Leads

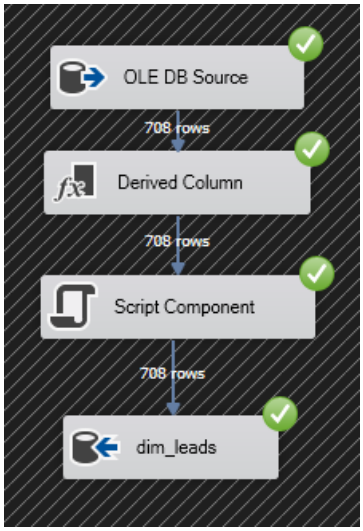
The purpose of this package is to load the descriptive attributes for marketing leads into the dim_leads table.

Control Flow

1. **Execute SQL Task:** Clears the dim_leads table before loading the data.
2. **Data Flow Task:** Extracts, transforms, and loads the data into the dim_leads table.

Data Flow

1. **Derived Column Transformation:** Replaces all NULL values with "N/A".
2. **Script Component:** Converts all text values to **Title Case** instead of lowercase.

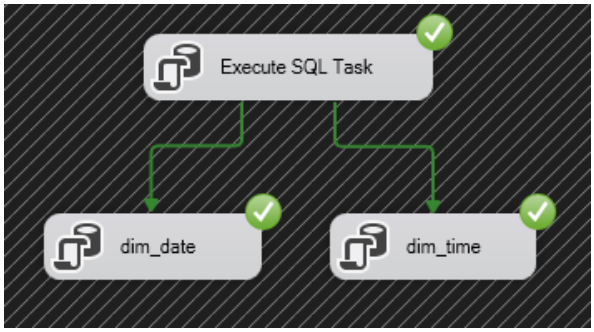


ETL Package: Dim_Date_Time

The purpose of this package is to create and load the date and time dimension tables used across the data warehouse.

Control Flow

- **Two Execute SQL Task:** Runs a SQL script that generates the dim_date and dim_time tables, including all necessary attributes for date and time analysis.



ETL Package: Fact_Sales.dtsx

Purpose:

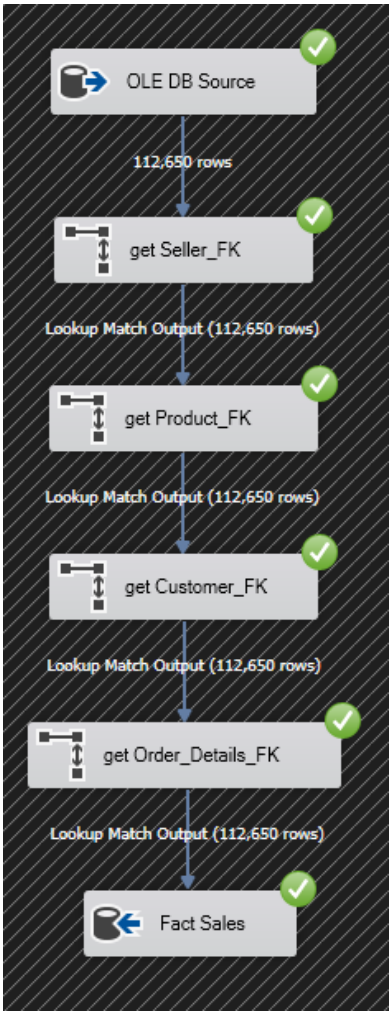
The purpose of this package is to load the **Fact_Sales** table by extracting and calculating all necessary measures using a SQL query as the data source.

Control Flow

1. **Execute SQL Task:** Clears the existing data from the fact_sales table before loading new records.
2. **Data Flow Task:** Extracts, transforms, and loads data into the fact_sales table.

Data Flow

- The data flow begins with an **OLE DB Source** that retrieves and calculates all required sales measures using a SQL script.
- Several **Lookup** components are used to fetch the corresponding **foreign keys** for each dimension table by matching business keys.
- After all transformations and lookups are completed, the data is loaded into the fact_sales table using an **OLE DB Destination** configured for fast batch loading.



ETL Package: Fact_Payment.dtsx

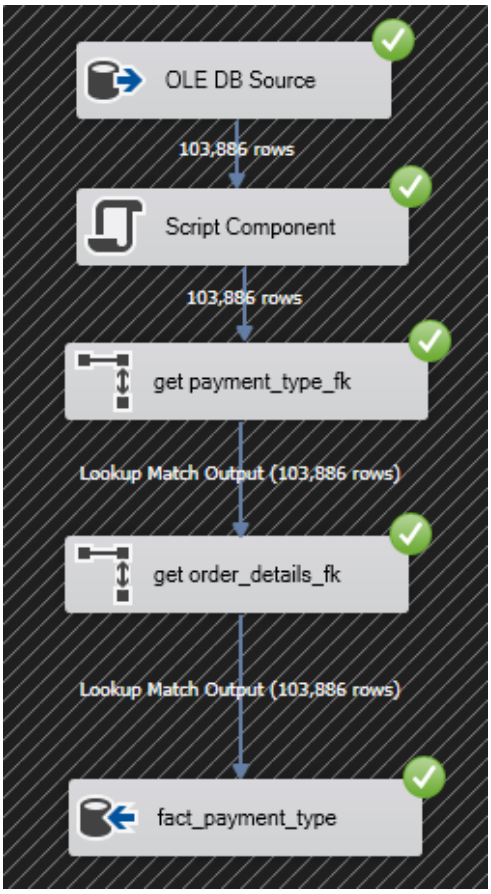
The purpose of this package is to load the **Fact_Payment** table with all payment-related transactional data.

Control Flow

1. **Execute SQL Task:** Clears the existing data from the fact_payment table before loading new records.
2. **Data Flow Task:** Extracts, transforms, and loads the data into the fact_payment table.

Data Flow

- A **Script Component** is used to apply the same transformations that were performed on the related dimension table columns. This ensures that lookups and key matching function correctly in the following steps.
- Multiple **Lookup Components** are used to retrieve the corresponding **foreign keys** from each dimension table by matching business keys.



ETL Package: Fact_Marketing_Leads.dtsx

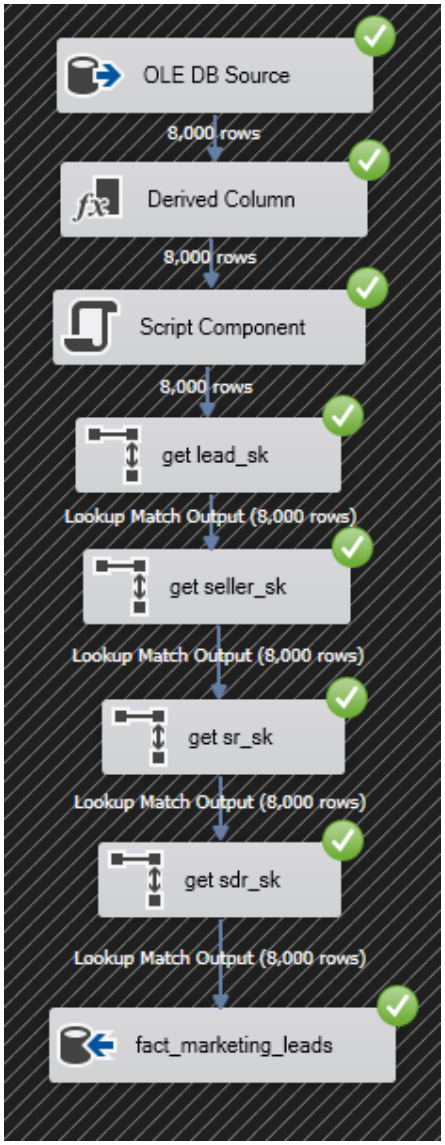
The purpose of this package is to load the **fact_marketing_leads** table.

Control Flow

3. **Execute SQL Task:** Clears the existing data from the **fact_marketing_leads** table before loading new records.
4. **Data Flow Task:** Extracts, transforms, and loads the data into the **fact_marketing_leads** table.

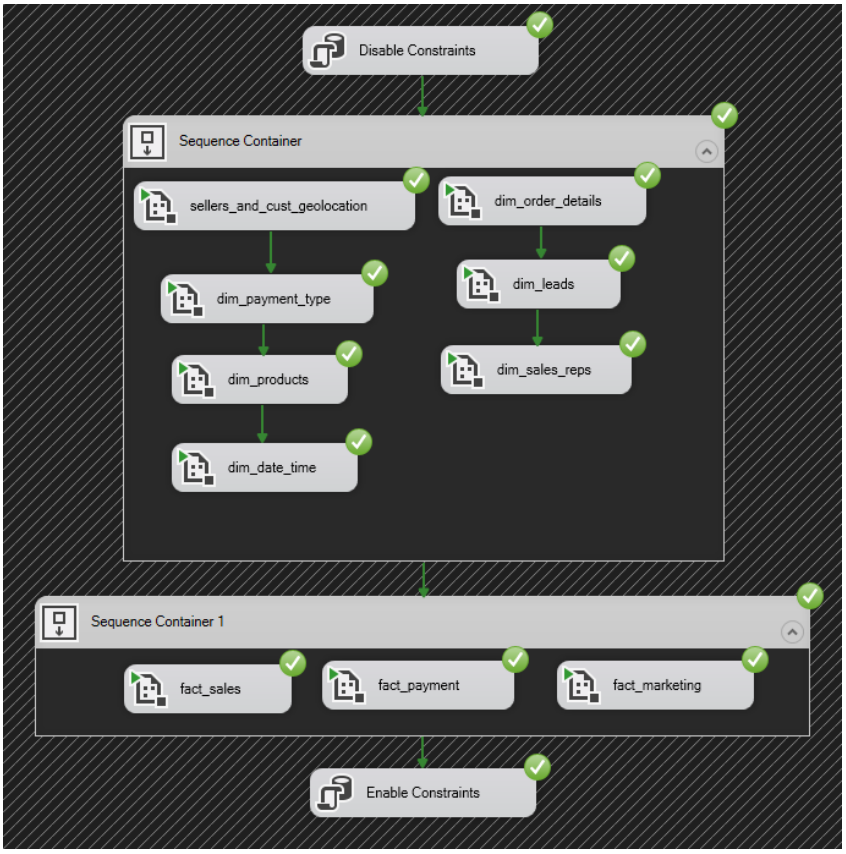
Data Flow

- A **Script Component** is used to apply the same transformations that were performed on the related dimension table columns. This ensures that lookups and key matching function correctly in the following steps.
- Multiple **Lookup Components** are used to retrieve the corresponding **foreign keys** from each dimension table by matching business keys.



ETL Package: Master.dtsx

The master package automates the **complete ETL process** by controlling the execution of all dimension and fact packages in the correct sequence. It ensures **referential integrity** by managing the order of loading and by disabling/enabling constraints as needed.



Control Flow

1. Disable Constraints

This task temporarily disables **foreign key constraints** in the database. It allows dimension and fact tables to be truncated and reloaded without constraint violations.

2. Sequence Container (Dimensions)

The purpose is to ensure that all dimensions are fully loaded before any fact tables, since **fact tables depend on dimension surrogate keys**.

3. Sequence Container 1 (Facts)

After all dimensions have been successfully loaded, this container executes the packages that load **fact tables**.

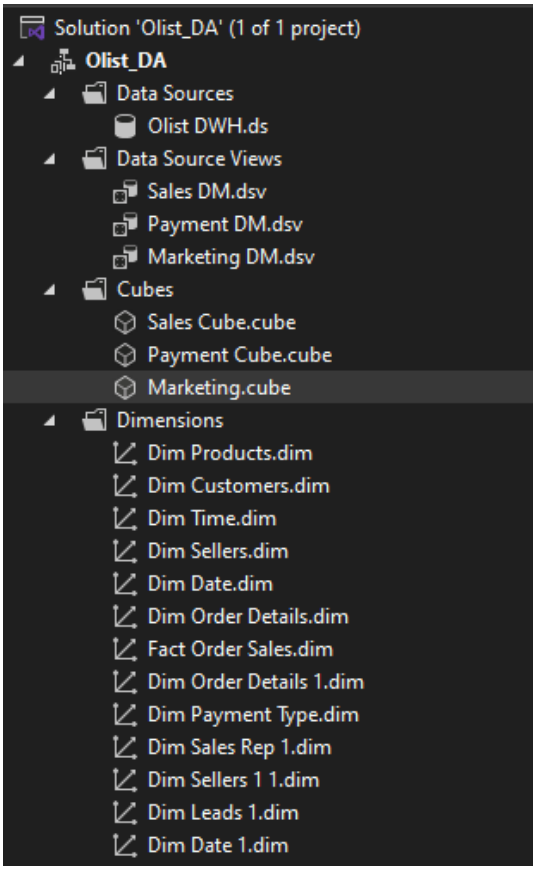
4. Enable Constraints

This task re-enables the **foreign key constraints** in the database after all ETL operations are completed.

Analyzing Data in SSAS

After successfully loading all the data into the data warehouse, the next step was to perform multidimensional analysis to answer key business questions. To achieve this, I used **SQL Server Analysis Services (SSAS)** to connect to the **Olist Data Warehouse** as the data source.

Within SSAS, I created three views (one for Sales, one for Marketing, and one for Payments) and then built three cubes based on these views. Each cube was designed to support analysis within its specific business domain.



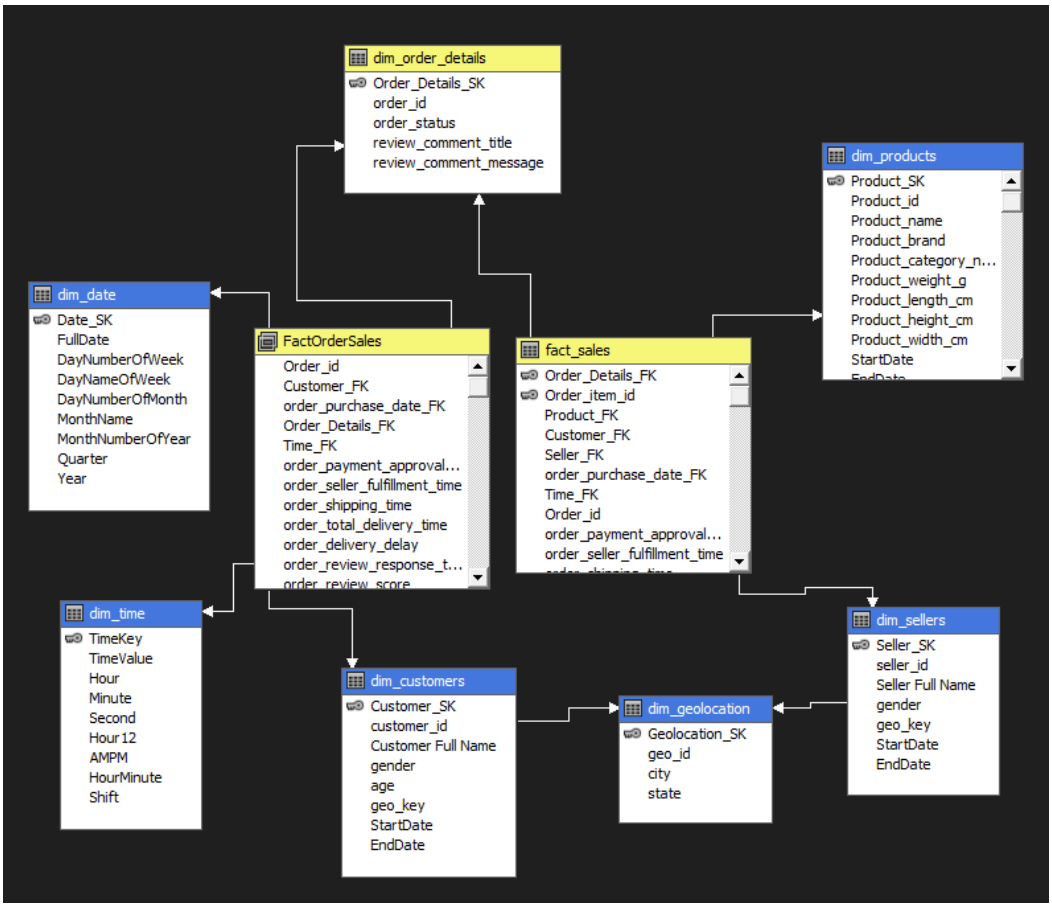
The views serve as an abstraction layer between the data warehouse and the cubes, simplifying data modeling, improving maintainability, and ensuring efficient cube processing.

The cubes were developed to deliver powerful analytical capabilities, enabling deeper insight into sales performance, payment trends, and marketing effectiveness through measures, calculated members, and dimension hierarchies.

Sales View

This view brings together all the tables needed to analyze the entire sales process. Its purpose is to answer questions like:

- What are the total sales by product, category, and brand?
- Who are my top customers by region (state and city)?
- Whatt is the average delivery time and review score for different states?



Since the Fact_Sales table contains two levels of granularity: item level and order level, it was important to ensure that order-level measures were calculated correctly without duplication.

To address this, I created a Named Query in SSAS that aggregates data at the order level for each customer within a specific time. This query summarizes order-related measures. By doing this, I eliminated duplicate entries and ensured that all order-level calculations and aggregations in the cube are accurate and consistent.

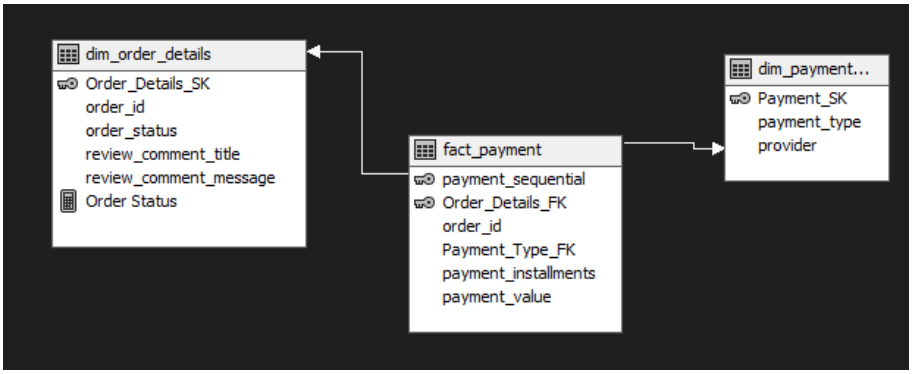
```
SELECT Order_id, Customer_FK, order_purchase_date_FK, Order_Details_FK, Time_FK,
    -- Use MAX() to get the single, non-duplicated value for each order
    MAX(order_payment_approval_time) AS order_payment_approval_time,
    MAX(order_seller_fulfillment_time) AS order_seller_fulfillment_time,
    MAX(order_shipping_time) AS order_shipping_time,
    MAX(order_total_delivery_time) AS order_total_delivery_time,
    MAX(order_delivery_delay) AS order_delivery_delay,
    MAX(order_review_response_time) AS order_review_response_time,
    MAX(order_review_score) AS order_review_score
FROM
    fact_sales
GROUP BY Order_id, Customer_FK, order_purchase_date_FK, Order_Details_FK, Time_FK
```

Figure 1: Fact Order Sales Logical Table Code

Payment View

This view is designed for financial analysis. Its purpose is to answer questions about customer payments, such as:

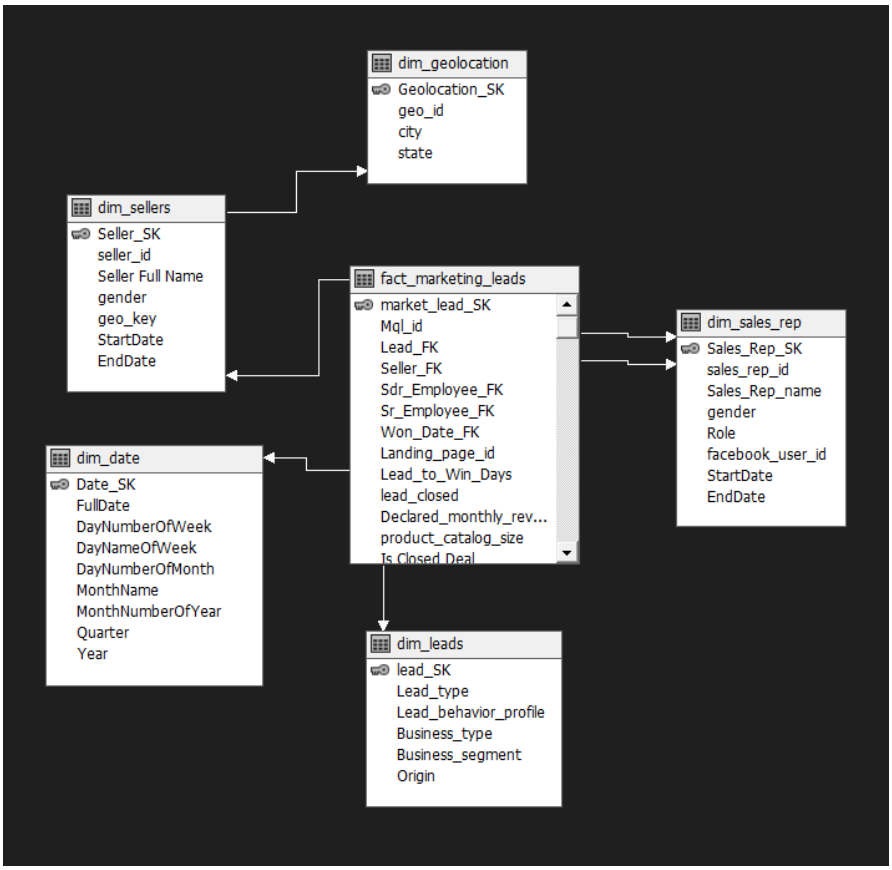
- What is the total payment value by payment type?
- What is the average number of payment installments?



Marketing View

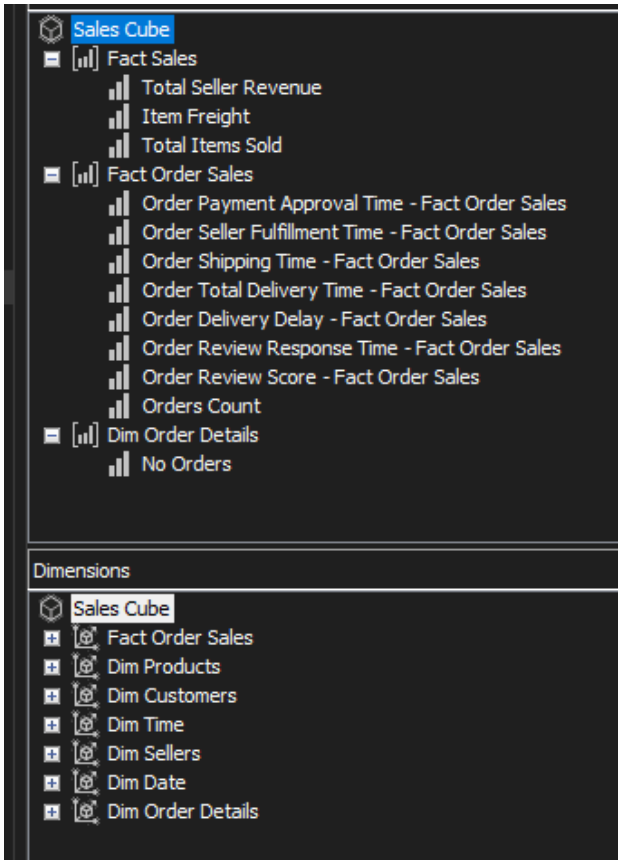
This view is focused on the marketing funnel. Its purpose is to track the performance of leads and the sales representatives who manage them. It's designed to answer specific questions, such as:

- What is the conversion rate per origin?
- What is the average time to close a deal per sales rep?



Sales Cube

This cube was built using Sales View. It combines measures from both the order level and the item level to ensure accurate and flexible analysis. The order-level measures were selected from the logical table created to resolve granularity issues, while the item-level measures, such as Item Freight and Item Price, were sourced directly from the Fact_Sales table. This design allows the cube to support detailed item-level insights while maintaining accurate aggregated metrics at the order level.



In this cube, I created several calculated measures to support three analytical reports:

1. Geolocation Report

- **Description:** A report that aggregates sales data by customer state. It shows the Avg Order Value, Freight %, and Total Revenue for each state.
- **Insights Provided:** This report is for geographical analysis. It helps identify the most valuable states and states with the highest average spending (Avg Order Value). You can also analyze the relationship between Freight % and revenue to see if high shipping costs in certain areas might be impacting sales.

2. Seller Performance Report

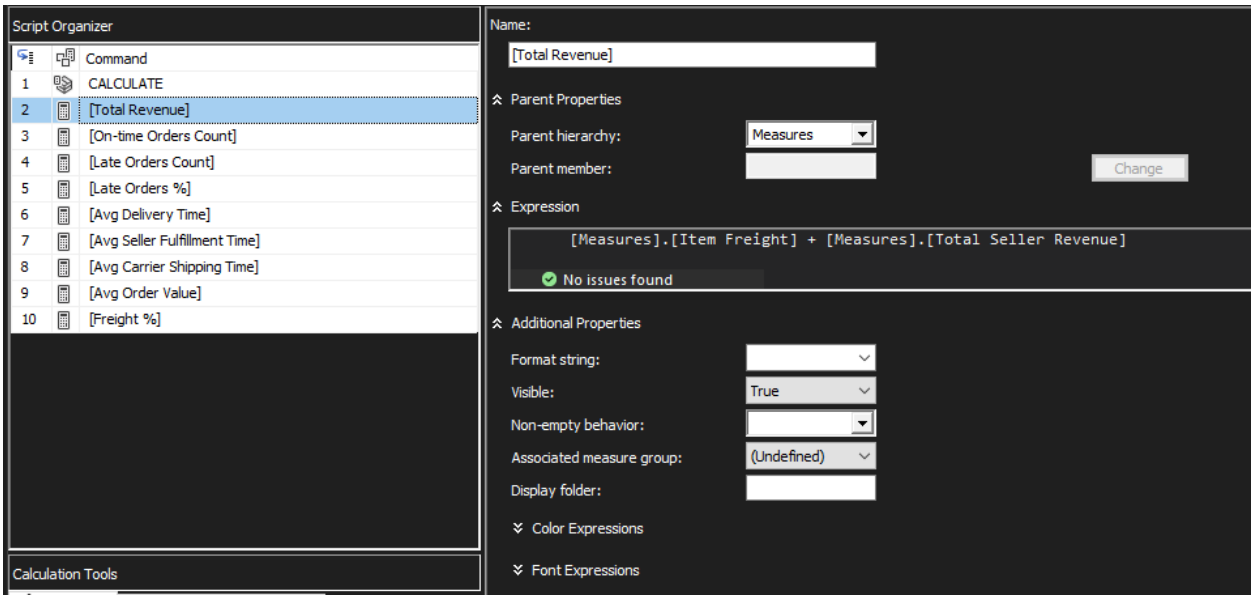
- **Description:** A report that lists individual sellers and their key performance metrics: No Orders, Freight %, and Total Seller Revenue.
- **Insights Provided:** This report is used to identify top and bottom performers. You can easily sort by Total Seller Revenue or No Orders to find your most valuable sellers. You can also use it to find sellers who may be problematic, for example, a seller with a very high Freight % but low revenue.

3. Delivery Performance Report

- **Description:** A report that tracks sales and logistics performance over time. It shows Orders Count, Total Revenue, Late Orders %, and the average time for fulfillment and shipping.
- **Insights Provided:** This report is perfect for identifying seasonal trends in sales and for monitoring logistics efficiency. You can spot problem months with high late orders % and see if delays are caused by sellers or the shipping carriers.

Calculated Members

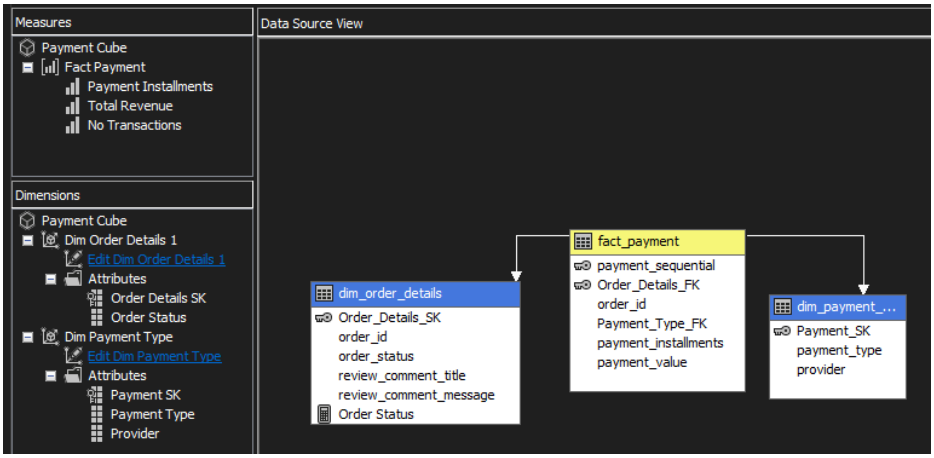
The following calculated measures were created in SSAS using **MDX** to support these reports:



Measure	*Formula*	*Description*
Total Revenue	[Measures].[Item Freight] + [Measures].[Total Seller Revenue]	Combines product and freight revenue for total sales value.
On-time Orders Count	SUM(FILTER([Fact Order Sales].[Order Id].[Order Id].MEMBERS, [Measures].[Order Delivery Delay - Fact Order Sales] <= 0), [Measures].[Orders Count])	Counts all orders delivered on or before the estimated date.
Late Orders Count	SUM(FILTER([Fact Order Sales].[Order Id].[Order Id].MEMBERS, [Measures].[Order Delivery Delay - Fact Order Sales] > 0), [Measures].[Orders Count])	Counts all orders delivered after the estimated delivery date.
Late Orders %	[Measures].[Late Orders Count] / [Measures].[Orders Count]	Calculates the percentage of delayed deliveries.
Avg Delivery Time	[Measures].[Order Total Delivery Time - Fact Order Sales] / [Measures].[Orders Count]	Measures average total delivery duration per order.
Avg Seller Fulfillment Time	[Measures].[Order Seller Fulfillment Time - Fact Order Sales] / [Measures].[Orders Count]	Measures average time sellers take to prepare and dispatch orders.
Avg Carrier Shipping Time	[Measures].[Order Shipping Time - Fact Order Sales] / [Measures].[Orders Count]	Calculates average shipping duration handled by the carrier.
Avg Order Value	[Measures].[Total Revenue] / [Measures].[No Orders]	Measures average revenue per order.
Freight %	[Measures].[Item Freight] / [Measures].[Total Revenue]	Indicates how much freight contributes to total revenue.

Payment Cube

This cube was built from the Payment View and is designed to support financial transaction analysis. It provides insights into payment patterns, provider performance, and order cancellations.



Payment Report

Description:
This report displays the Number of Transactions, Total Revenue, and Number of Canceled Orders, grouped by Payment Provider and filtered by Payment Type.

Insights Provided:
It helps identify the most valuable payment providers for the business and serves as a risk analysis tool by highlighting providers with a high number of canceled orders.

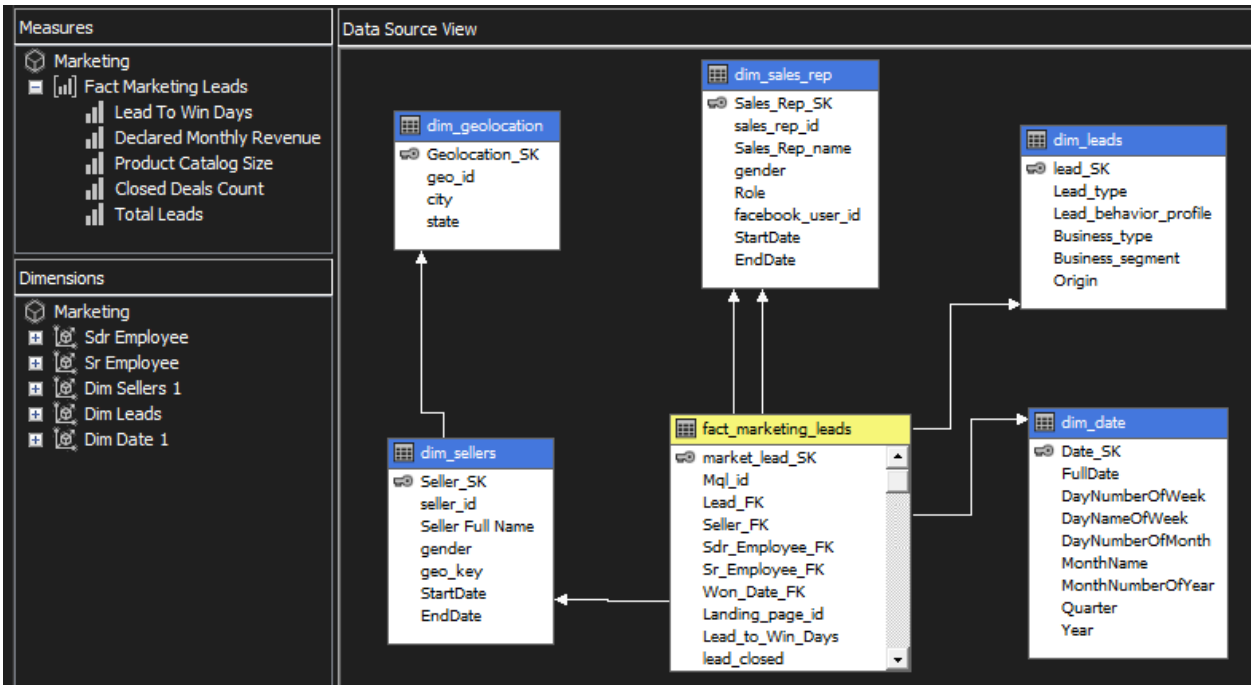
Calculated Member

```
[No Canceled Orders] = IIF(
    IsEmpty(([Measures].[No Transactions], [Dim Order Details 1].[Order Status].&[canceled])),
    0,
    ([Measures].[No Transactions], [Dim Order Details 1].[Order Status].&[canceled])
)
```

This calculated member counts the number of canceled orders by checking transactions with an order status of "canceled" and assigning zero where no such transactions exist.

Marketing Cube

This cube was built from Marketing View to support analysis of the marketing funnel and sales representative performance. It provides insights into lead generation, deal conversion, and team efficiency.



We used this cube to create the following reports:

1. Origin Performance Report

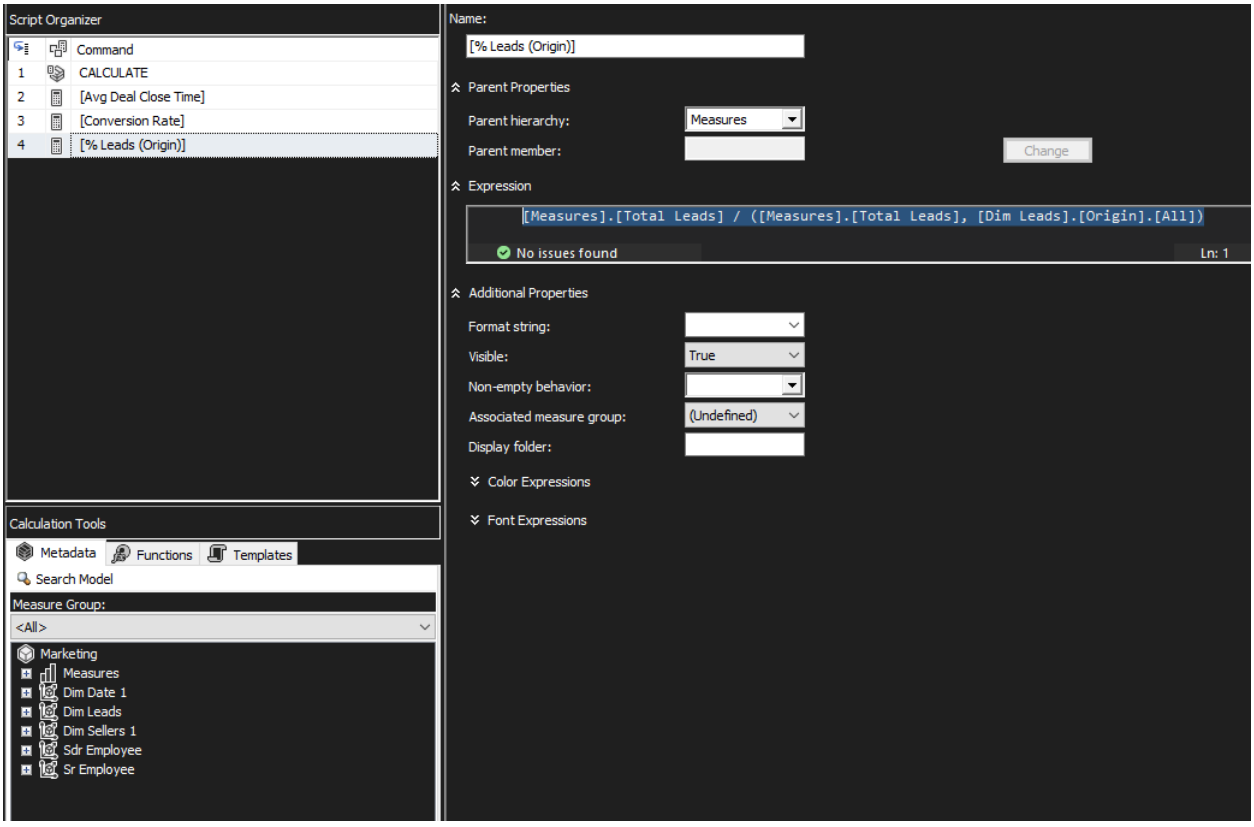
- **Description:** A report that analyzes the effectiveness of different marketing channels by grouping leads by their Origin. It displays the Closed Deals Count, % Leads (Origin), Conversion Rate, and Avg Deal Close Time for each channel.
- **Insights Provided:** This report is essential for marketing strategy. It shows which channels generate the most closed deals and which channels have the highest Conversion Rate It also identifies which channels produce the *fastest* deals.

2. Sales Rep Performance Report

- **Description:** This report breaks down performance by individual sales representative. It shows Sales Rep Name, Role, Gender, Closed Deals Count, and Avg Deal Close Time.
- **Insights Provided:** This report is used to identify top-performing sales reps, both by the volume of deals they close and their efficiency.

Calculated Members

Measure Name	*Description*	*Formula (MDX Expression)*
Avg Deal Close Time	Calculates the average number of days it takes to close a deal from the time a lead is created.	[Measures].[Lead To Win Days] / [Measures].[Closed Deals Count]
Conversion Rate	Calculates the percentage of leads that were successfully converted into closed deals.	[Measures].[Closed Deals Count] / [Measures].[Total Leads]
% Leads (Origin)	Calculates the proportion of leads from each marketing origin compared to total leads.	[Measures].[Total Leads] / ([Measures].[Total Leads], [Dim Leads].[Origin].[All])

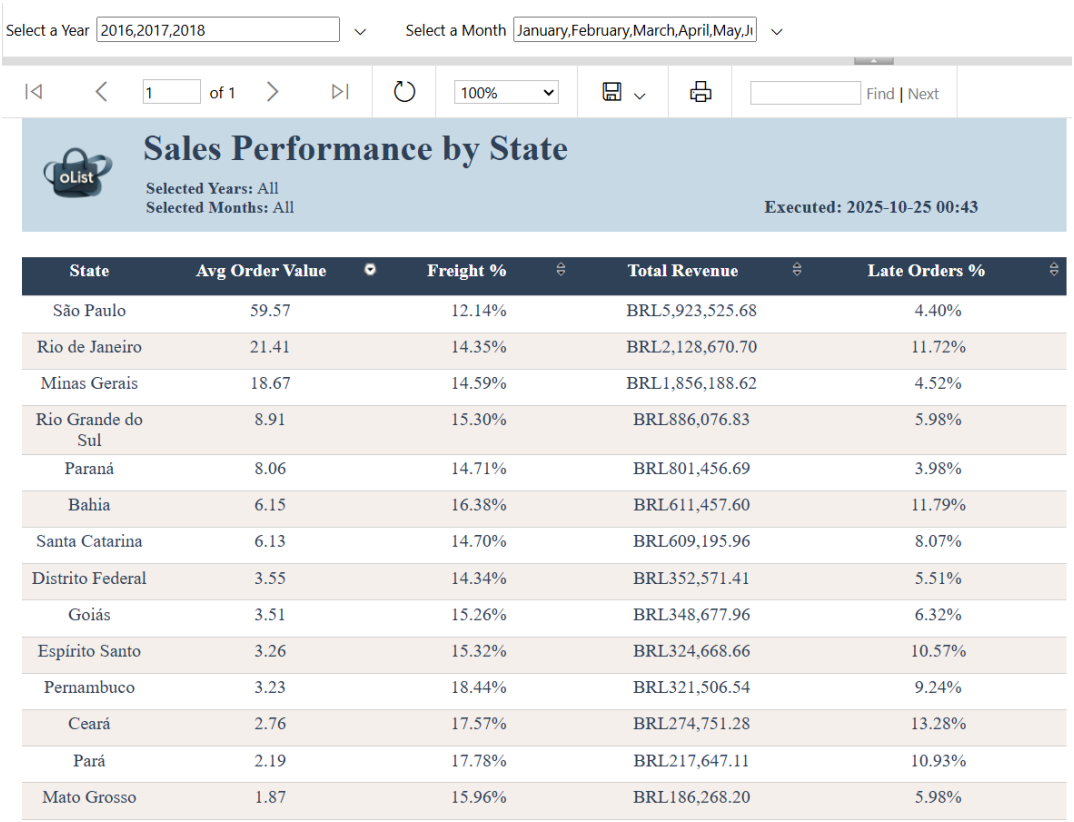


Reporting in SSRS

After we completed the analysis and prepared the necessary measures for six reports. It is time to create the reports in SSRS.

Geolocation Report

A report that aggregates sales data by customer state. It shows the Avg Order Value, Freight %, and Total Revenue for each state. It gives the user the ability to filter by year and month.



State	Avg Order Value	Freight %	Total Revenue	Late Orders %
São Paulo	59.57	12.14%	BRL5,923,525.68	4.40%
Rio de Janeiro	21.41	14.35%	BRL2,128,670.70	11.72%
Minas Gerais	18.67	14.59%	BRL1,856,188.62	4.52%
Rio Grande do Sul	8.91	15.30%	BRL886,076.83	5.98%
Paraná	8.06	14.71%	BRL801,456.69	3.98%
Bahia	6.15	16.38%	BRL611,457.60	11.79%
Santa Catarina	6.13	14.70%	BRL609,195.96	8.07%
Distrito Federal	3.55	14.34%	BRL352,571.41	5.51%
Goiás	3.51	15.26%	BRL348,677.96	6.32%
Espírito Santo	3.26	15.32%	BRL324,668.66	10.57%
Pernambuco	3.23	18.44%	BRL321,506.54	9.24%
Ceará	2.76	17.57%	BRL274,751.28	13.28%
Pará	2.19	17.78%	BRL217,647.11	10.93%
Mato Grosso	1.87	15.96%	BRL186,268.20	5.98%

Insights

State with the Top Average Order Value:
São Paulo has the highest average order value of 59.57, which is nearly three times higher than the next state.

State with the Lowest Freight %:
São Paulo also records the lowest freight percentage at 12.14%.

State with the Top Total Revenue:
São Paulo leads in total revenue, generating BRL 5,923,525.

State with the Top Late Orders %:
Alagoas has the highest percentage of late orders at 20.63%.

Seller Performance Report

A report that lists individual sellers and their key performance metrics: No Orders, Freight %, and Total Seller Revenue.

Select a State Acre,Amazonas,Bahia,Ceará,Distrito

1 of 2 ? 100% Find | Next

Seller Performance

Selected States: All

Executed: 2025-10-25 01:27

Name	Gender	Total Items Sold	No Orders	Freight %	Total Seller Revenue
Sharon Walton	Female	2033	1854	18.48%	BRL123,305
Robert Burton	Male	1987	1806	14.89%	BRL200,473
Joseph Young	Male	1775	1706	19.75%	BRL104,288
Jacob Chang	Male	1931	1404	24.75%	BRL106,939
Douglas Castillo	Male	1551	1314	13.48%	BRL160,237
Kathy Mayo	Female	1499	1287	15.83%	BRL135,172
Desiree Whitney	Female	1171	1160	12.85%	BRL141,746
Wanda Moody	Female	1203	1146	32.06%	BRL37,178
Denise Dudley	Female	1156	1132	8.08%	BRL229,473
Jacob Young	Male	1147	1080	19.12%	BRL94,914
David Morales	Male	1364	982	21.55%	BRL187,924

Insights

Seller with the Highest Number of Items Sold:

Sharon Walton has the highest number of items sold, totaling 2,033 items.

Seller with the Highest Number of Orders:

Sharon Walton also holds the highest number of orders, with 1,854 orders.

Seller with the Highest Freight Percentage:

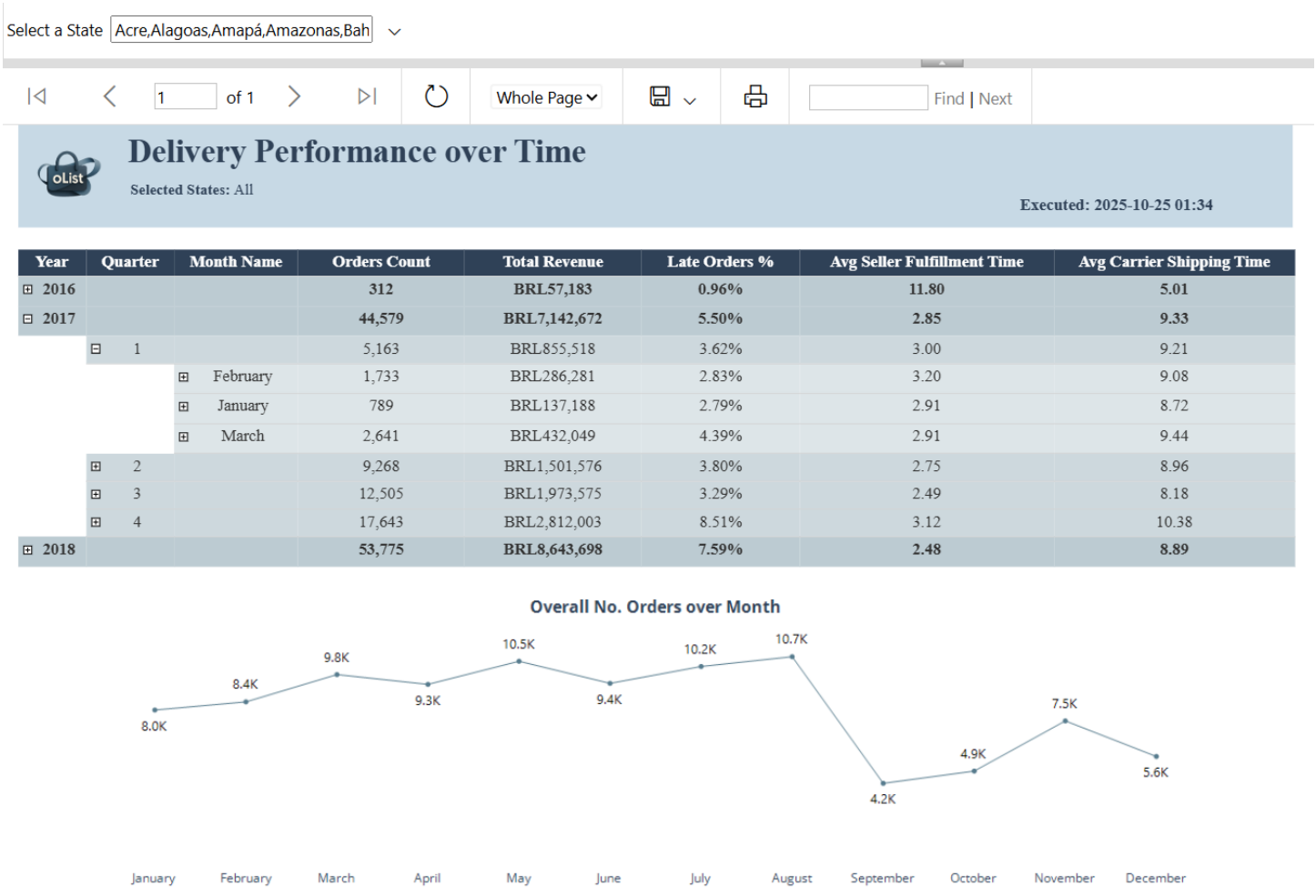
Eric Gonzalez sold 2 items, with freight costs representing 77% of the total cost.

Seller with the Highest Revenue:

Denise Dudley achieved the highest total revenue, amounting to BRL 229,473.

Delivery Performance Report

A report that tracks sales and logistics performance over time. It shows Orders Count, Total Revenue, Late Orders %, and the average time for fulfillment and shipping.



Year	Quarter	Month Name	Orders Count	Total Revenue	Late Orders %	Avg Seller Fulfillment Time	Avg Carrier Shipping Time
⊞ 2016			312	BRL57,183	0.96%	11.80	5.01
⊟ 2017			44,579	BRL7,142,672	5.50%	2.85	9.33
	⊟ 1		5,163	BRL855,518	3.62%	3.00	9.21
		⊞ February	1,733	BRL286,281	2.83%	3.20	9.08
		⊞ January	789	BRL137,188	2.79%	2.91	8.72
		⊞ March	2,641	BRL432,049	4.39%	2.91	9.44
	⊞ 2		9,268	BRL1,501,576	3.80%	2.75	8.96
	⊞ 3		12,505	BRL1,973,575	3.29%	2.49	8.18
	⊞ 4		17,643	BRL2,812,003	8.51%	3.12	10.38
⊞ 2018			53,775	BRL8,643,698	7.59%	2.48	8.89

Insights

From this report, we can gain insights into various business aspects such as delivery performance, revenue trends, and order volume over time.

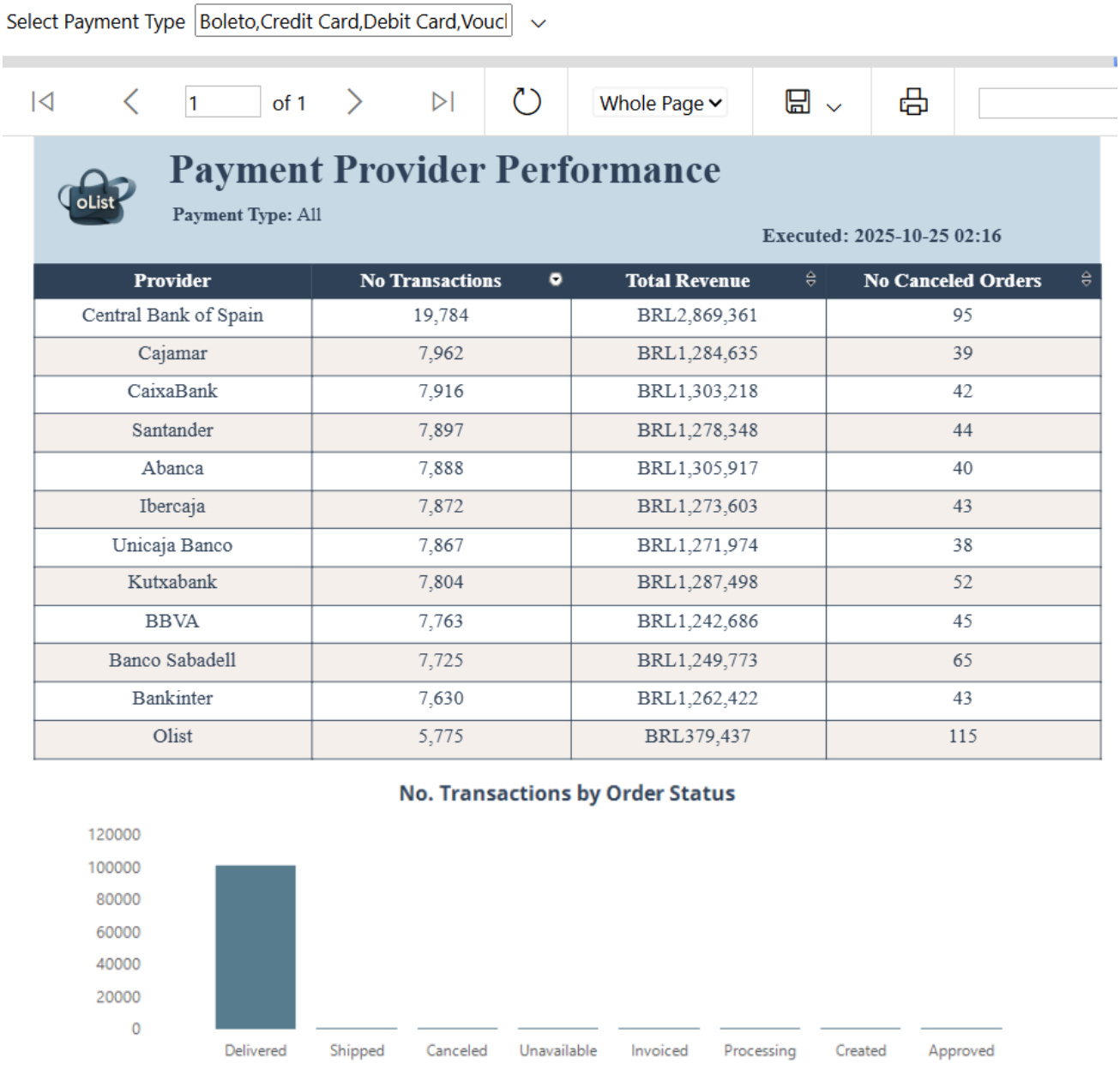
In 2016, there were only 312 orders, indicating that this was the year the company began operations. By 2017, the number of orders increased significantly to 44,579, with 5.5% of them being late. The average seller fulfillment time was 2.85 days, and the average carrier shipping time was 9.33 days.

By drilling down to the quarter and month levels, we can see that in February 2017, there were 1,733 orders placed, with 2.83% of them being late.

When observing the trend across months, August shows the highest number of orders overall, while the period between September and December experiences a noticeable decline in order volume compared to other months.

Payment Report

A report that displays the Number of Transactions, Total Revenue, and Number of Canceled Orders, grouped by Payment Provider and filtered by Payment Type.



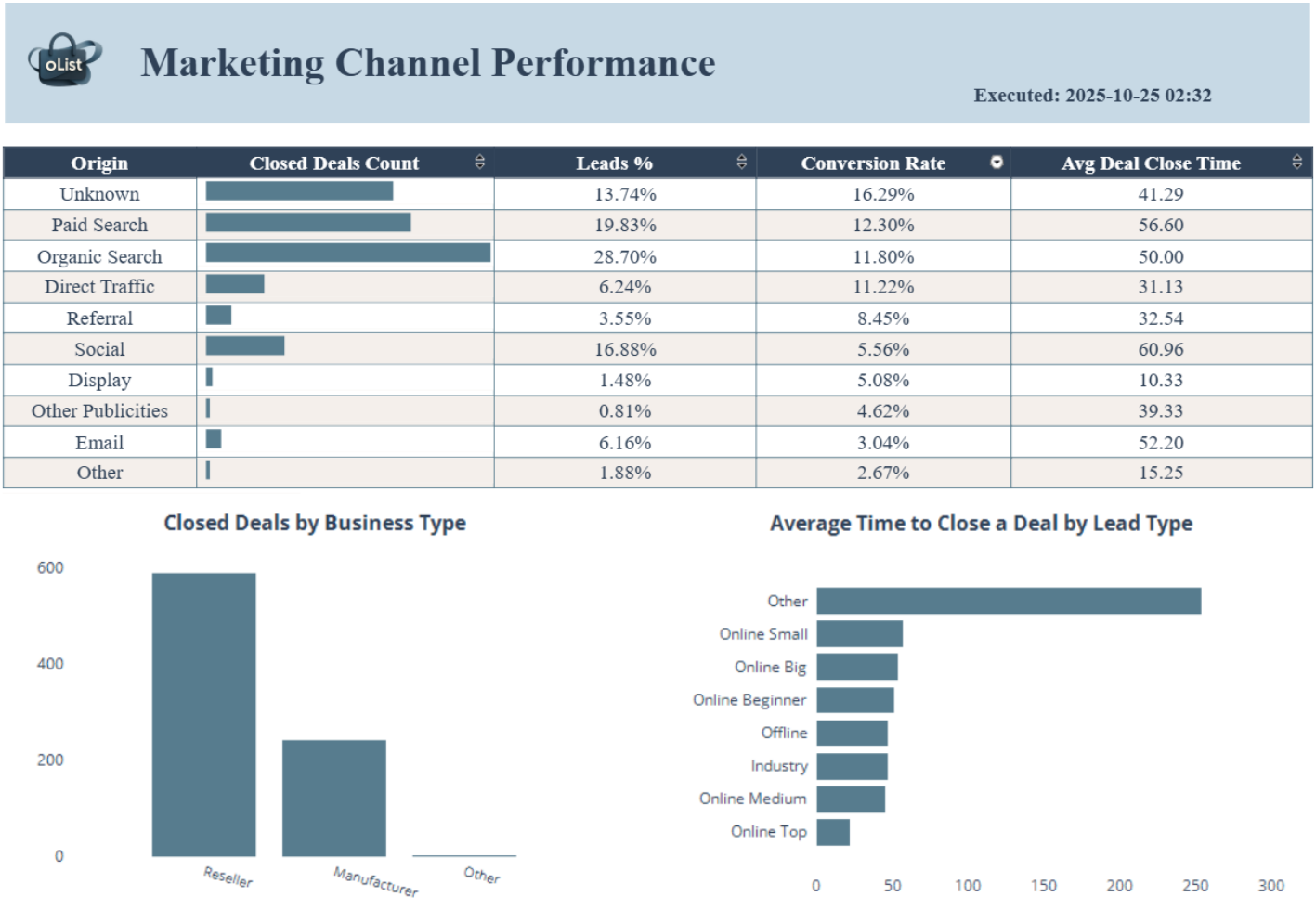
Provider	No Transactions	Total Revenue	No Canceled Orders
Central Bank of Spain	19,784	BRL2,869,361	95
Cajamar	7,962	BRL1,284,635	39
CaixaBank	7,916	BRL1,303,218	42
Santander	7,897	BRL1,278,348	44
Abanca	7,888	BRL1,305,917	40
Ibercaja	7,872	BRL1,273,603	43
Unicaja Banco	7,867	BRL1,271,974	38
Kutxabank	7,804	BRL1,287,498	52
BBVA	7,763	BRL1,242,686	45
Banco Sabadell	7,725	BRL1,249,773	65
Bankinter	7,630	BRL1,262,422	43
Olist	5,775	BRL379,437	115

Insights

The Central Bank of Spain recorded the highest number of transactions, totaling 19,784, with a total payment value of BRL 2,869,361 and 95 canceled orders. The highest number of canceled orders was associated with Olist as a provider, suggesting that these cancellations were connected to vouchers issued by Olist. As shown in the chart, the number of undelivered orders is very low, indicating strong delivery overall.

Marketing Channel Performance Report

A report that analyzes the effectiveness of different marketing channels by grouping leads by their Origin. It displays the Closed Deals Count, % Leads (Origin), Conversion Rate, and Avg Deal Close Time for each channel.

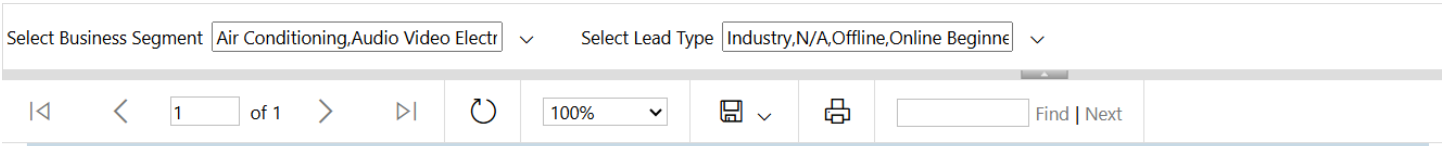


Insights

- The Unknown marketing channel achieved the highest conversion rate at 16.29%, followed by Paid Search with a 12.3% conversion rate.
- Organic Search generated the highest number of closed deals and maintained a conversion rate of 11.80%, representing the largest share of leads at 28.7%.
- Leads originating from Social channels took the longest average time to close, averaging 61 days. Most closed deals were associated with the Reseller business type.
- Among lead types, the “Other” category had the longest average close time, while Online Top leads had the shortest average deal close time.

Sales Rep Performance Report

A report breaks down performance by individual sales representatives. It shows Sales Rep Name, Role, Gender, Closed Deals Count, and Avg Deal Close Time. We can filter by Business Segment and Lead Type.



Sales Rep Performance Overview

Business Segments: All
Lead Types: All

Executed: 2025-10-25 02:40

Sales Rep Name	Gender	Role	Closed Deals Count	Avg Deal Close Time
Randy Hunt	Male	Sales Representative	133	30.49
Sara Mack	Female	Sales Representative	82	46.65
Joseph Pacheco	Male	Sales Representative	74	25.78
Greg Solis	Male	Sales Representative	64	33.70
Emily Sanders	Female	Sales Representative	63	48.59
Melinda Little	Female	Sales Representative	59	21.97
Melvin Wu	Male	Sales Representative	59	48.39
Linda Jones	Female	Both	53	61.74
Emily West	Female	Both	51	41.71
Laura Baker	Female	Sales Representative	36	22.39
Daniel Lynch	Male	Sales Representative	32	25.22
Sondos Mohamed	Female	Both	27	122.33
George Cook	Male	Both	26	96.65
Jessica Miller	Female	Both	24	17.00
Nour Ghanaym	Male	Both	24	55.13
Basma Hesham	Female	Both	10	172.00
Alex Pruitt	Male	Both	9	180.56
Charles Tyler	Male	Both	7	234.00
Reaal Salah	Female	Both	6	214.33
Adblwahed Nasr	Male	Both	1	321.00

Insights

- Randy Hunt closed the highest number of deals, totaling 133, with an average close time of 30 days.
- Joseph Pacheco closed 74 deals with an average close time of 25 days, meaning he closed deals faster than Randy Hunt and Sara Mack.
- Employees who served as both Sales Representatives and Sales Development Representatives tended to take longer to close deals on average compared to those who worked solely as Sales Representatives.

Market Basked Analysis (MBA)

In this section, we describe the process followed to perform Market Basket Analysis (MBA) to uncover product association rules and co-purchase patterns.

Market Basket Analysis is a data mining technique used to identify relationships between items that are frequently purchased together. Applying it to our transaction data helps us understand customer purchasing behavior and generate insights for cross-selling and marketing strategies.

Through this analysis, we aim to answer key business questions such as:

- Which products are frequently bought together?
- What product combinations should sellers recommend for cross-selling?
- Which product bundles can drive promotions or increase sales value?

Implementation Steps

We used Python and applied the Apriori algorithm to discover association rules between products.

We started by connecting to the data warehouse and writing a SQL query to retrieve the required data for analysis, using fact_sales for transaction data and dim_products for product details.

```python
# Define connection string for SQLAlchemy
conn_str = (
    "mssql+pyodbc://localhost\\MSSQLSERVER01/olist_dwh?"
    "driver=ODBC+Driver+17+for+SQL+Server&trusted_connection=yes"
)

# Create SQLAlchemy engine
engine = create_engine(conn_str)

# Query your database
query = """
SELECT Product_SK, product_name, product_category_name, order_id
FROM fact_sales INNER JOIN dim_products ON Product_FK = Product_SK
"""

df = pd.read_sql(query, engine)
```

```python
df.head()
```

	Product_SK	product_name	product_category_name	order_id
0	1	Dior Cologne Gold	Perfumery	f30149f4a8882a08895b6a242aa0d612

Next, we examined the percentage of orders that contained only one item and found that approximately 97% of orders have a single item.

```python
# Count products per order
order_counts = df.groupby('order_id')['Product_SK'].nunique()

print("Percentage of Orders that have only 1 item: ", (order_counts == 1).mean())
```

```
Percentage of Orders that have only 1 item:  0.9672024810978452
```

To focus on transactions that can reveal meaningful product associations, we filtered the dataset to include only orders with multiple items. Orders containing only one product were excluded since they cannot contribute to association rule discovery since meaningful relationships require at least two items in a single transaction.

```
# Keep only orders with more than one product
multi_item_orders = order_counts[order_counts > 1].index
df_filtered = df[df['order_id'].isin(multi_item_orders)]
df_filtered.shape
```

```
(7768, 4)
```

After that, we created the basket dataset required by the Apriori algorithm, where each row represents an order, each column represents a product, and each cell contains 1 or 0 indicating whether that product was purchased in that order.

```
# Count products by order_id, then pivot so product names become columns
basket = (
    df_filtered.groupby(['order_id', 'product_name'])['product_name']
    .count().unstack().fillna(0)
)

# Convert counts to 1 (purchased) or 0 (not purchased)
basket[basket > 0] = 1

# Convert the basket to boolean type for Apriori
basket = basket.astype(bool)

basket
```

product_name	1-800-Flowers Mixed Flowers 796	3M Air Filter 274 V2	3M Air Filter Mini	3M Battery Black V1	3M Battery Generation 9	3M Battery Gray	3M Battery Max	3M Battery Series 4	3M Brake Pad 632	3M Brake Pad Lite	...	Zebra Pen Set Gray	Zebra Pencil Case Max V2	Zebra Planner 2024
order_id														
002f98c0f7efd42638ed6100ca699b42	False	False	False	False	False	False	False	False	False	False	...	False	False	False
00337fe25a3780b3424d9ad7c5a4b35e	False	False	False	False	False	False	False	False	False	False	...	False	False	False

After preparing the basket dataset, we applied the Apriori algorithm to identify frequent itemsets. Because the dataset contained only 3,236 orders, we set the minimum support threshold to 0.001, which corresponds to roughly three orders. This low threshold ensures that even less common but potentially meaningful patterns are captured.

At this support level, the algorithm generated 226 frequent itemsets.

```
frequent_itemsets = apriori(basket, min_support=0.001, use_colnames=True)

frequent_itemsets
```

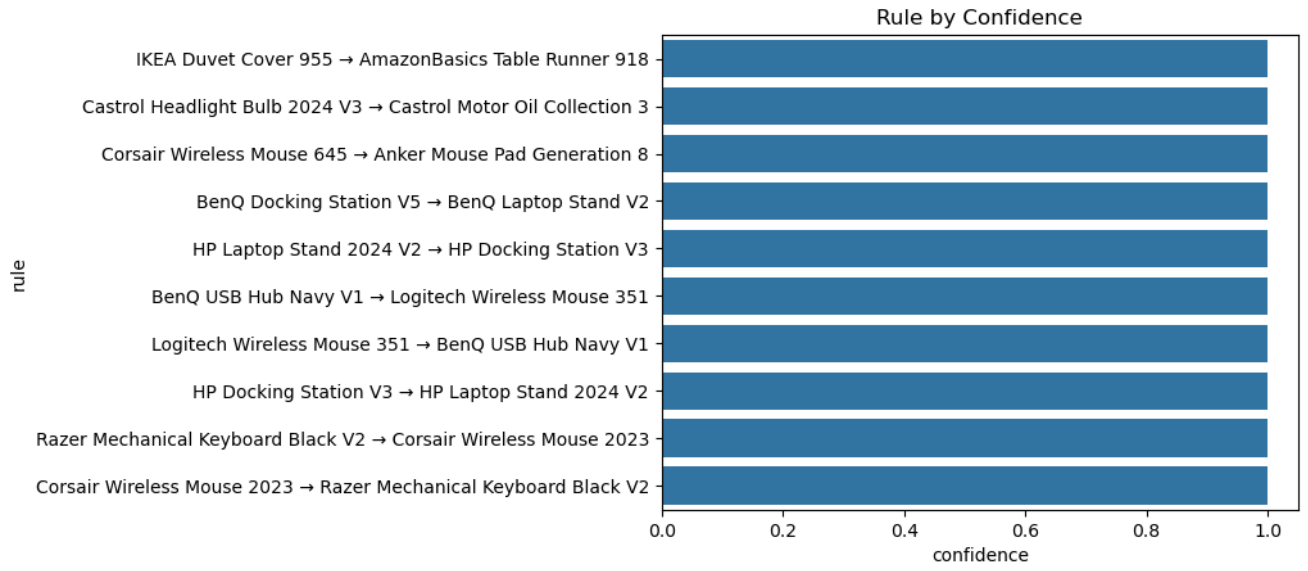
	support	itemsets
0	0.001236	(3M Laminator 470)
1	0.001545	(AmazonBasics Blanket Collection 2 V2)

Next, we used these itemsets to derive association rules, using confidence as the evaluation metric and a minimum confidence threshold of 0.5. This means we only kept rules where the likelihood of one product being purchased given another was at least 50%.

```
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
rules.sort_values(by='lift', ascending=False)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
3	(Corsair Wireless Mouse 645)	(Anker Mouse Pad Generation 8)	0.001236	0.001854	0.001236	1.000000	539.333333
4	(Anker Mouse Pad Generation 8)	(Corsair Wireless Mouse 645)	0.001854	0.001236	0.001236	0.666667	539.333333

The process returned 60 association rules, several of which had a confidence of 1 and a very high lift value, indicating strong and reliable product relationships.



For example, one of the discovered rules states that if a customer buys an “Corsair Wireless Mouse 645”, they are 100% likely to also purchase an “Anker Mouse Pad Generation 8”. This insight highlights a strong co-purchase relationship that can be leveraged for cross-selling strategies and bundle recommendations.

However, because the minimum support corresponds to only about three orders, we must treat these results as exploratory signals rather than fully robust insights.

The rules we found may hold business value (for example as cross-selling or bundling opportunities), and they are interesting given their high confidence and lift; but the extremely low absolute support means we should proceed with caution. We should consider collecting more data (or focusing on a higher support threshold) to increase reliability.

Therefore, our conclusion is:

- We have uncovered valid and promising associations within the multi-item orders subset.
- The results should be considered useful leads, not definitive business rules.
- Next steps should include raising the support threshold (so rules are based on more transactions), validating the associations with business logic, and monitoring over time as more multi-item orders accumulate.

Dashboard Implementation

The Power BI report was developed to provide a comprehensive analytical view of the Olist dataset, covering customer behavior, product and category performance, seller efficiency, and delivery operations. Each page of the report focuses on a specific business area and includes interactive visuals, navigational bookmarks, and dynamic filters to enhance data exploration.

- **Location Page:**
  Features an interactive map supported by a parameter rule that allows switching between main measures — total revenue, total orders, total customers, and total sellers. A KPI slicer enables users to visualize how geographical factors influence key performance metrics.

Measure Name	Measure
total revenue	Total revenue = CALCULATE(SUM(fact_payment[payment_value]), TREATAS(VALUES(fact_sales[Order_Details_FK]), fact_payment[Order_Details_FK]))
total orders	Total orders = DISTINCTCOUNT(fact_sales[Order_id])
total customers	Total customers = DISTINCTCOUNT(dim_customers[customer_id])

total sellers	Total Sellers = CALCULATE(DISTINCTCOUNT(dim_sellers[seller_id]), USERELATIONSHIP(dim_sellers[geo_key],dim_geolocation[Geolocation_SK]))

- **General Overview Page:**
  Provides an overall business summary through cards (total freight, total brands, canceled orders, total orders, and total revenue). It includes a pie chart of revenue by payment type, a revenue trend line chart, and a decomposition tree for category, brand, and product revenue. Navigation bookmarks allow movement between main pages, access to the revenue page, filter pane, and reset views.
- **Revenue Page:**
  Displays financial insights through cards (total sellers, customers, installments, products, and sales reps). Visuals include a revenue distribution by age group, monthly deviation waterfall chart, interest-based revenue trend, and monthly variance bar chart. Interactive bookmarks are used for navigation, filtering, and resetting.
- **Categories Page:**
  Highlights category performance with KPIs for the highest and lowest performing categories, and total category count. It includes charts for top brand revenue, orders by category, average review score, and total revenue by category. Users can drill through to brands and products pages, with dedicated bookmarks for filters and navigation.
- **Brands Page:**
  Focuses on brand-level analysis through KPIs (selected category, top brand, and total products). Visuals show revenue, review scores, orders, installments, and product counts by brand. Drill-through functionality connects to the products page, with bookmarks for filtering and navigation.
- **Products Page:**
  Displays product-specific details through cards (selected brand, top product, and average product dimensions). Charts analyze orders, installments, revenue, and freight by product name. Buttons and bookmarks allow navigation back to the brands page and access to filtering options.
- **Customers Overview Page:**
  Summarizes customer insights through KPIs (total customers, new customers %, repeat customers %). Visuals include customer segmentation, gender distribution, new customer trends, and orders by gender. Navigation buttons lead to customer performance and payment behavior pages.
- **Customer Performance Page:**
  Measures engagement levels with cards for engaging and dormant customers, gender

percentages, average age, and orders per customer. Charts include revenue by gender, revenue by top states, customer segments, repeat trends, and orders by age group.

- **Customer Payment Behavior Page:**
  Focuses on payment dynamics through KPIs (highest and lowest payment provider, total installments, average approval time, and average installment count). Visuals show revenue by payment type, top customers, provider-type decomposition, and revenue trends over time.
- **Seller Overview Page:**
  Presents seller performance through visuals for revenue and orders by seller, plus a matrix showing total orders, average monthly revenue, and revenue variation. Navigation allows access to the seller performance page.
- **Seller Performance Page:**
  Highlights key seller metrics including top and lowest performers, total sellers, on-time shipments, and delivery delays. Charts visualize shipment punctuality, delay trends, and gender distribution among sellers.
- **Order Fulfillment and Delivery Page:**
  Analyzes delivery efficiency with KPIs (total shipments, on-time delivery, delayed delivery, average delivery time, canceled and unavailable orders). Visuals include shipment distribution, delivery status, and delay trends over time, supported by bookmarks for navigation, filtering, and resetting.

"Time Based Analysis" Dashboard

1. **Orders by Weekday vs Weekends**
2. **Insight**: Most orders happen on **weekdays**, with weekends contributing only a small portion → business is weekday-focused.

2. Total Orders by Shift

- **Insight**: **Evening shift handles the majority** of orders, followed by morning, then night → evening is the peak operational period.

3. On-Time Delivery by TimeValue

- **Insight**: On-time performance **drops significantly in the evening**, especially after late afternoon → delivery reliability weakens as the day progresses.

4. Delayed Delivery by Shift

- **Insight**: **Evening shift sees the most delays**, followed by morning and night → evening operations face the greatest pressure.

5. On-Time Delivery by Shift

- **Insight**: Evening shift delivers the **highest volume on time** but also has the most delays → high activity, inconsistent reliability.

6. Orders per Month

- **Insight**: Order volume **grows steadily over time**, showing consistent business expansion.

Business is weekday-driven, peaks in the evening, but delivery reliability suffers during high-volume evening hours. Strong growth is evident, but **service quality does not keep pace with demand**, especially in the evening shift.

Recommendations

- **Strengthen evening shift** with more staff or better processes.
- **Limit same-day orders** late in the day to improve on-time rates.
- **Shift non-urgent orders** to morning or night for smoother execution.
- **Optimize evening workflows** to reduce bottlenecks.
- **Monitor on-time delivery by shift** as a key performance metric.

"Marketing Funnel Analysis" Dashboard

1-**Avg Declared Revenue by Lead Type**

- **Insight**: **Industry** leads have the **highest declared value**, followed by "Other" → high-value segments exist.

2- Avg Catalog Size by Lead Type

- **Insight**: **Other** leads engage with the **largest product catalog**, while smaller segments browse less → interest depth varies.

3- Avg Win Days by Lead Type

- **Insight**: **Other** leads close **fastest**, while "Online Beginner" take longest → speed depends on lead sophistication.

4- Total MQLs by Lead Type

- **Insight**: **Online Medium** generates the **most qualified leads**, while "Top" and "Other" contribute little → MQL volume skewed.
- High lead volume and ambition, but poor conversion and long cycles. Strong declared revenue from certain types (Industry, Other), but lead quality and nurturing are inconsistent. Some segments engage deeply and close fast; others stall or fail.

Recommendations

- **Fix funnel leakage** – improve qualification and follow-up.
- **Shorten sales cycle** for slow lead types with automation or better content.
- **Double down** on high-value, fast-closing segments (Industry, Other).

- **Retire or retrain** worst-performing lead type.
- **Align marketing** to generate more MQLs from proven types (Online Medium).

" Marketing Funnel by Lead Behavior Profile " Dashboard

1. Total Leads / Total Closed Deals / Conversion Rate / Avg Win Days / Avg Declared Revenue / Worst Performing Lead Type

- **Insight**: High lead inflow but modest closure success, with deals taking considerable time and carrying ambitious revenue expectations; one lead type clearly underperforms.

2. Avg Catalog Size by Lead Behavior Profile

- **Insight**: Certain profiles explore the product catalog deeply, while others show limited interest → engagement depth varies by behavior.

3. Avg Declared Revenue by Lead Behavior Profile

- **Insight**: A few profiles signal strong revenue potential, while most project lower value → revenue opportunity is concentrated.

4. Total MQLs by Lead Behavior Profile

- **Insight**: One profile dominates qualified lead generation, others contribute far less → MQL volume is uneven.

5. Avg Win Days by Lead Behavior Profile

- **Insight**: Some profiles close quickly, others drag on significantly → sales cycle speed depends heavily on behavior.

General Understanding

Plenty of leads, but conversion and speed lag. A few high-engagement, high-value behavior profiles stand out, while others stall or fail. The funnel is **top-heavy and inconsistent**, with clear winners and laggards in lead quality and responsiveness.

Recommendations

- **Focus nurturing** on high-engagement, fast-closing profiles.
- **Shorten cycle** for slow-moving profiles with targeted content.
- **Boost MQLs** from proven behavior types.
- **De-prioritize or retrain** the worst-performing profile.

Conclusion

Funnel has strong potential but suffers from uneven lead quality. Prioritize high-performing behavior profiles, streamline slow ones, and cut weak links to improve conversion and speed.

" Marketing Funnel by Origin" Dashboard

- **Total MQLs by Origin**: The bar chart shows Organic Search as the leading source for marketing qualified leads, followed by Paid Search and Social, with diminishing contributions from sources like Unknown, Direct Traffic, Email, Referral, Display, Other Publicities, and N/A.
- **Closed and Lost by Origin**: The line chart illustrates varying success rates across origins, with Organic Search showing higher closed deals relative to losses, while sources like Display and Other Publicities have more losses, indicating inconsistent performance.
- **Average of Product Catalog by Origin**: This bar chart reveals Social as the top performer in product variety or engagement, closely followed by Paid Search, with lower averages for Organic Search, Direct Traffic, and Unknown, suggesting some channels drive broader interest.
- **Top Conversion Rate by Origin**: The horizontal bar chart ranks Unknown as the highest in conversion efficiency, then Paid Search and Organic Search, with Direct Traffic, Referral, Social, and Display trailing, pointing to strengths in certain acquisition methods.
- **Average of Lead to Win Days by Origin**: The bar chart indicates Social takes the longest time from lead to win, followed by Paid Search and Email, while Organic Search and Unknown are quicker, highlighting efficiency differences in nurturing processes.
- **Average of Declared Monthly Revenue by Origin**: This bar chart positions Organic Search as the strongest revenue driver, followed by Paid Search, with lower contributions from Other Publicities, Social, and Unknown, underscoring profitable channels.

Recommendations

- Prioritize investment in organic and paid search channels to capitalize on their strong lead generation and revenue potential.

- Investigate and optimize social and display origins to reduce longer win times and improve conversion rates.
- Explore unknown and referral sources for untapped efficiency, perhaps by better tracking and attribution.
- Focus on streamlining the lead nurturing process across all origins to shorten time to win and boost overall funnel performance.

" Marketing Funnel by Lost Deals " Dashboard

- **Top Lost Leads by Origin**: The pie chart highlights organic search as the dominant source of lost leads, followed by unknown, social, and paid search, indicating these channels contribute most to overall losses.
- **Lost Revenue by Loss Stage**: The bar chart shows loss to competitors as the stage with the highest revenue impact, followed by no response, unqualified leads, and budget issues, suggesting competitive pressures cause the biggest financial setbacks.
- **Overall Lost Leads Summary**: This includes totals for lost leads, their proportion, average catalog size for losses, and average declared revenue from lost deals, with unqualified marked as the primary losing stage, pointing to common qualification challenges.
- **Lost Leads by Loss Stage**: The bar chart ranks unqualified as the top reason for lost leads, then no response, loss to competitor, and budget issue, revealing qualification and engagement as key pain points.
- **Average Catalog Size for Lost Leads by Loss Stage**: The bar chart indicates loss to competitor involves larger catalog sizes, followed by no response and unqualified, with budget issue showing smaller sizes, implying deal complexity influences certain loss types.
- **Total Lost Revenue by Origin**: The horizontal bar chart positions organic search as the source with the most lost revenue, followed by paid search, social, unknown, email, direct traffic, referral, other, and display, underscoring search channels' high-stakes involvement in losses.

Overall Understanding

The dashboard focuses on areas of leakage in the sales process, where search origins like organic and paid drive the bulk of lost opportunities and revenue, primarily due to qualification issues, lack of response, and competition. It reveals inefficiencies in lead handling across stages, with a pattern of higher losses in high-volume channels, suggesting a need for better retention strategies to minimize financial impact.

Recommendations

- Strengthen qualification processes early in the funnel to reduce unqualified losses, especially from search and social sources.
- Improve response times and engagement tactics to address no-response losses, potentially through automation or team training.
- Analyze competitive positioning to mitigate losses to competitors, focusing on unique value propositions in high-revenue channels.

"Sales Reps Performance " Dashboard

- **Closed Deals by Sales Rep Name**: The horizontal bar chart ranks sales representatives by the volume of deals they have closed, with a clear leader at the top followed by a gradual decline among others, indicating varying levels of success in deal completion.
- **Seller Segments**: This tree diagram categorizes sales reps into groups such as micro, small, medium, enterprise, and large sellers, showing how reps are distributed across these segments and highlighting the largest group.
- **Total Revenue by Sales Rep Name**: The horizontal bar chart identifies top revenue contributors among reps, with patterns similar to closed deals but emphasizing financial impact, where some reps stand out for higher earnings generation.
- **Average Lead to Win Days by Role**: The bar chart compares the time taken to convert leads to wins between combined roles and specific sales representatives, revealing longer cycles in broader roles versus shorter ones in focused positions.
- **Number of Lead Behaviors by Sales Rep Name**: The horizontal bar chart measures engagement or activity levels with leads per rep, with some showing higher interaction frequencies, suggesting differences in lead management approaches.

Overall Understanding

The dashboard illustrates performance disparities among sales reps, segmented by seller types, where certain individuals excel in closing deals and generating revenue, while others lag, with variations in lead conversion speed and engagement. It points to a team structure favoring specific segments and highlights efficiency in targeted roles over general ones.

Recommendations

- Recognize and replicate strategies from top-performing reps to uplift underperformers through training or mentoring.
- Adjust resource allocation across seller segments to bolster weaker groups, focusing on micro and small sellers for growth potential.
- Streamline processes in roles with longer lead-to-win times to accelerate conversions and improve overall team velocity.

"Sales Rep Overview" Dashboard

- **Rep Profile**: Displays personal details including name, gender, birthday, and location, giving context to the individual seller's background.
- **Closed Deals**: A single metric showing total deals successfully closed by the rep.
- **Number of Business Types**: Indicates the variety of business categories the rep has engaged with.
- **Number of Lead Type**: Reflects the diversity of lead sources the rep has converted.
- **Total Orders**: Summarizes the overall volume of orders processed by the rep.
- **Total Revenue**: Highlights the cumulative financial contribution from the rep's sales.
- **Closed Deals by Lead Type**: Bar chart ranks performance across different lead categories, with online medium and offline sources leading, followed by beginner, big, small, and industry leads.
- **Total Revenue by Year and Month**: Line chart traces revenue trends over time, showing fluctuations with periods of growth and decline across years.

Overall Understanding

The dashboard provides a snapshot of a single sales rep's performance, revealing consistent activity across multiple lead types and business categories, with stronger results from online medium and offline channels. Revenue shows variability over time, indicating seasonal or market influences, while the rep maintains broad engagement and solid overall output.

Recommendations

- Leverage strengths in online medium and offline leads to scale high-performing channels.
- Explore reasons behind lower conversion in small, big, and industry leads to unlock growth.
- Analyze revenue dips in the timeline to identify external factors or internal gaps for stabilization.

"Advanced Analysis" Dashboard

- **(Market Basket Analysis Rules)**: The horizontal bar chart ranks product association rules by their lift strength, showing strong co-purchase tendencies between items like mouse pads and hubs, docking stations and laptops, wireless mice and stands, keyboards and docks, webcams and runners, watch covers and bulbs, sports watches and oil, laptop stands and monitors, with higher lifts indicating more frequent bundled buys than random chance.

- **Overall Review Metrics**: Summarizes total volume of reviews, split into positive and negative sentiments, highlighting the balance or imbalance in customer feedback.
- **Average Delivery Delay by Review Sentiment**: The bar chart compares delivery time variances across negative, neutral (no review), and positive sentiments, revealing that delays are similarly negative across categories but slightly varied, suggesting delivery issues influence sentiment uniformly.
- **Count of Reviews by Sentiment**: The pie chart distributes reviews into positive, negative, and no-review segments, illustrating the proportion of satisfied versus dissatisfied customers alongside those without feedback.

Overall Understanding

The dashboard combines market basket analysis to uncover product affinities, showing tech accessories like mice, keyboards, and docking stations often bundle together, while sentiment analysis on reviews points to a roughly even split between positive and negative feedback, with delivery delays as a common thread affecting perceptions, indicating opportunities in bundling and logistics improvements.