Spring 2025 - AASD4012

BIG DATA TOOLS & TECHNIQUES II

Group 1 - Final Project

Optimizing Retail Insights with Big Data Pipeline

Akrami, Safoora
Gharacheh, Lily
Dias, Isaac
Ha, Bach
Singh, Shashi Kumar
Tikyani, Gotham
Tran, Bao Phuc
Wong, Lok Yin

Dataset:
https://www.kaggle.com/datasets/devarajv88/target-dataset

Table of Contents

1. Data Selection & Exploration

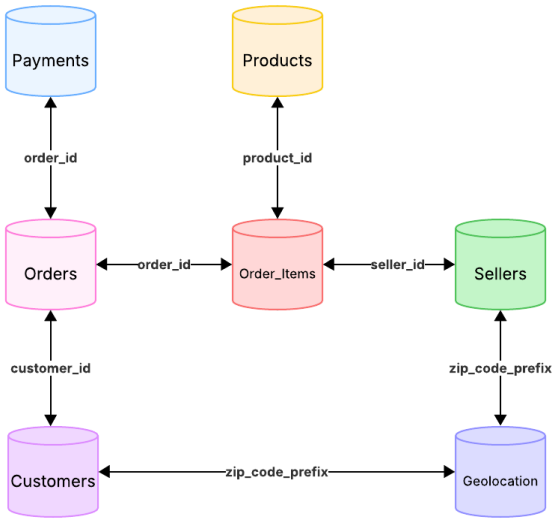
https://www.kaggle.com/datasets/devarajv88/target-dataset

About Target Brazil

Target is a leading retail and e-commerce company offering various products, including electronics, home goods, apparel, and groceries. As one of the major players in Brazil's online marketplace, Target's operations involve high-volume transactions, diverse payment methods, and complex logistics. Understanding its performance in this competitive landscape provides valuable business insights.

Dataset Description

This dataset focuses on Target's operations in Brazil and covers 100,000 orders placed between 2016 and 2018. It includes detailed information on order status, pricing, payment, and shipping performance, as well as customer locations, product attributes, and customer reviews. All the data is stored in the following 7 tables in CSV format:



- **Customers**:
  - `customer_id` (string): ID of the consumer who made the purchase
  - `customer_unique_id` (string): Unique ID of the consumer
  - `customer_zip_code_prefix` (string): Zip code of the consumer's location
  - `customer_city` (string): Name of the city
  - `customer_state` (string): State Code

- **Sellers**:
  - `seller_id` (string): Unique ID of the seller registered
  - `seller_zip_code_prefix` (string): Zip code of the seller's location
  - `seller_city` (string): Name of the city
  - `seller_state` (string): State Code

- **Geolocation**:
  - `geolocation_zip_code_prefix` (string): First 5 digits of Zip Code
  - `geolocation_lat` (double): Latitude
  - `geolocation_lng` (double): Longitude
  - `geolocation_city` (string): City
  - `geolocation_state` (string): State

- **Products**:
  - `product_id` (string): ID of the product
  - `product_category` (string): Product category
  - `product_name_length` (int): Length of the product's name
  - `product_description_length` (int): Length of the product's description
  - `product_photos_qty` (int): Number of photos of the product in the shopping portal
  - `product_weight_g` (int): Weight of the products measured in grams
  - `product_length_cm` (int): Length of the products measured in centimeters
  - `product_height_cm` (int): Height of the products measured in centimeters
  - `product_width_cm` (int): Width of the product measured in centimeters

- **Orders**:
  - `order_id` (string): ID of the order made by the consumers
  - `customer_id` (string): ID of the consumer who made the purchase
  - `order_status` (string): Status of the order made (i.e. delivered, shipped, etc.)
  - `order_purchase_timestamp` (date): Purchase date
  - `order_approved_at` (date): Order approved date
  - `order_delivered_carrier_date` (date): Delivery date at which carrier got the products
  - `order_delivered_customer_date` (date): Delivery date at which customer got the product
  - `order_estimated_delivery_date` (date): Estimated delivery date

- **Order_Items**:
  - `order_id` (string): ID of the order
  - `order_item_id` (int): Unique ID given to each item ordered in the order (surrogate key)
  - `product_id` (string): ID of the product
  - `seller_id` (string): ID of the seller
  - `shipping_limit_date` (date): The deadline when the ordered product must be shipped
  - `price` (double): Actual price of the products ordered
  - `freight_value` (double): Inventory and shipping costs, combined with actual price to get the total price of the order

- **Payments**:
  - `order_id` (string): A Unique ID of the order made by the consumers
  - `payment_sequential` (int): Sequences of payments made in case of EMI
  - `payment_type` (string): Mode of payment used (i.e. Credit Card)
  - `payment_installments` (int): Number of installments in case of EMI purchase
  - `payment_value` (double): Total amount paid for the purchase order

Defining Business Questions

1. What's the customer's purchase pattern?

Purpose: Understand how customers purchase over time (daily, monthly, yearly) and what payment types they prefer. This helps forecast demand, optimize inventory and staffing, and refine marketing strategies.

Supporting Queries:

- Sales volume over time (monthly, yearly)
- Payment method breakdown

2. Which store (seller) locations are the customers most or least satisfied with?

Purpose: Identify which Brazilian states where the stores are located, perform best and worst in terms of customer experience, delays, cancellations, and sales. This supports decisions about expansion, store closures, or logistics improvements.

Supporting Queries:

- Cancel and delay rate by seller location
- Total sales by seller location

3. How to increase revenue?

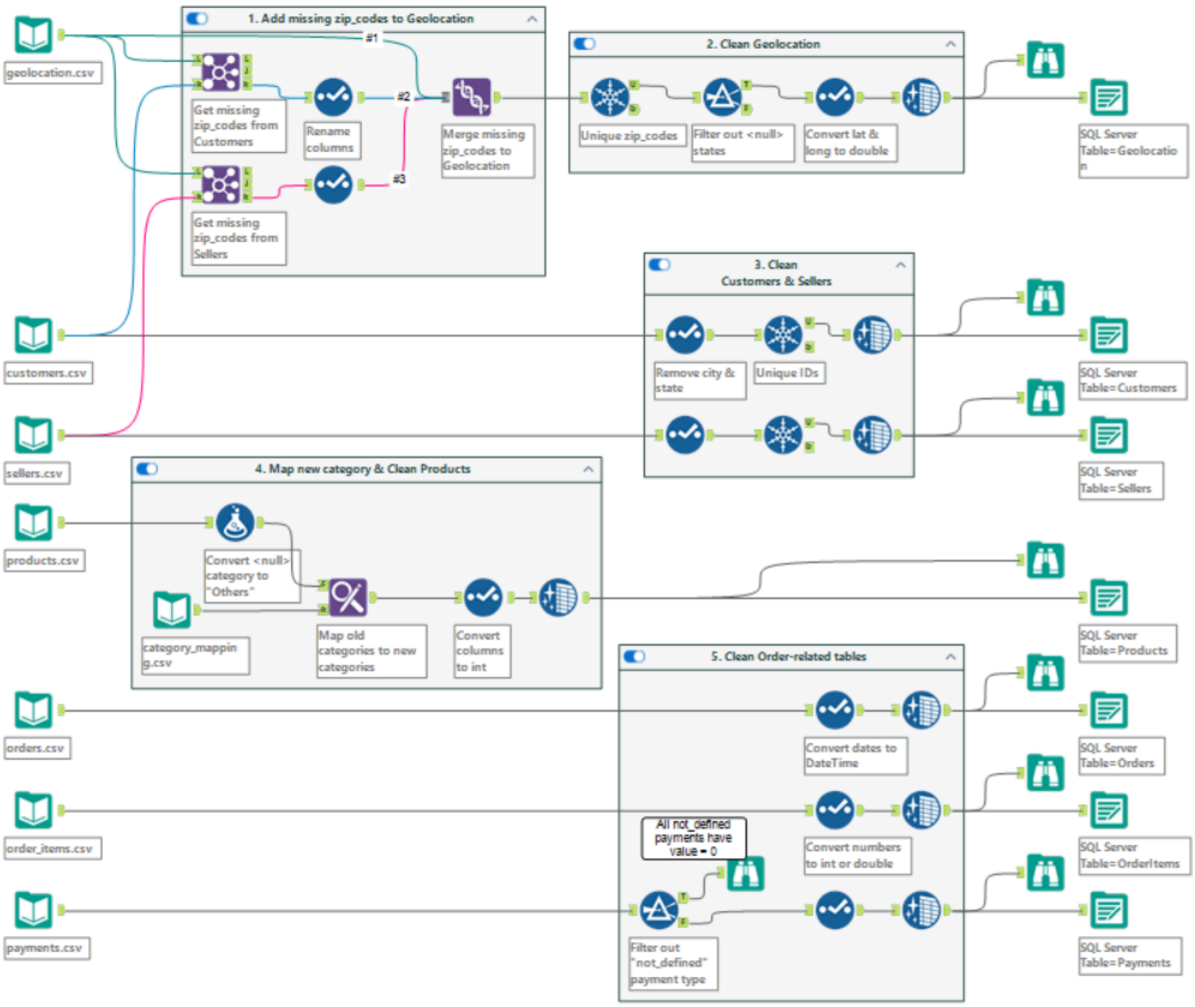
Purpose: Discover which products/categories drive the most revenue and which underperform. This enables better inventory decisions, targeted marketing promotions, and the elimination of unprofitable stock.

Supporting Queries:

- Top and bottom-selling product categories
- Sellers with the best sales record

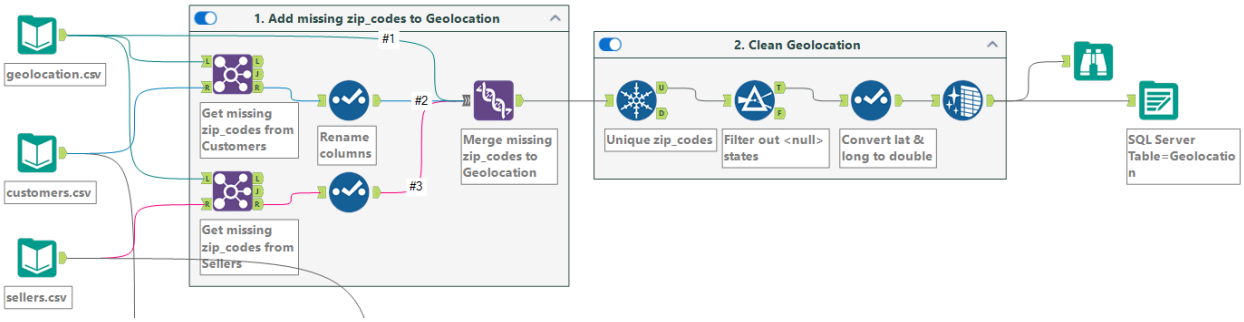
2. Data Preprocessing & ETL with Alteryx

Full Flow Diagram



The ETL pipeline processes seven CSV files from the Target Brazil dataset and prepares them for storage in a SQL Server data warehouse. The workflow includes five main steps stated below.

Step 2.1 & Step 2.2: Add Missing Zip Codes to Geolocation, then clean it



Files Used:

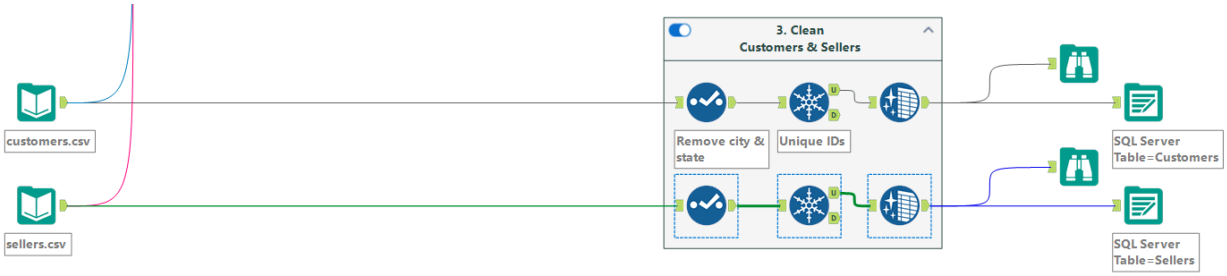
- **`geolocation.csv`**
- **`customers.csv`**
- **`sellers.csv`**

Process Summary:

- Extracted missing **`zip_code_prefix`** entries from both **`customers.csv`** and **`sellers.csv`**.
- Renamed and standardized columns for compatibility.
- Merged these into the **`geolocation.csv`** dataset to fill in missing zip code data using known matches.
- Removed duplicate **`zip_code_prefix`** entries and ensured only unique values are retained.
- Filtered out records with **`NULL`** state values to avoid geographic ambiguity.
- Converted **`latitude`** and **`longitude`** columns from text to numeric format for mapping and spatial analysis.
- Created the Geolocation table in the SQL server, then loaded the cleaned dataset into the created table.

Purpose: Improves the quality and coverage of the geolocation data so downstream joins on location can succeed without **`NULL`** or mismatches.

Step 2.3: Clean Customers & Sellers



Files Used:

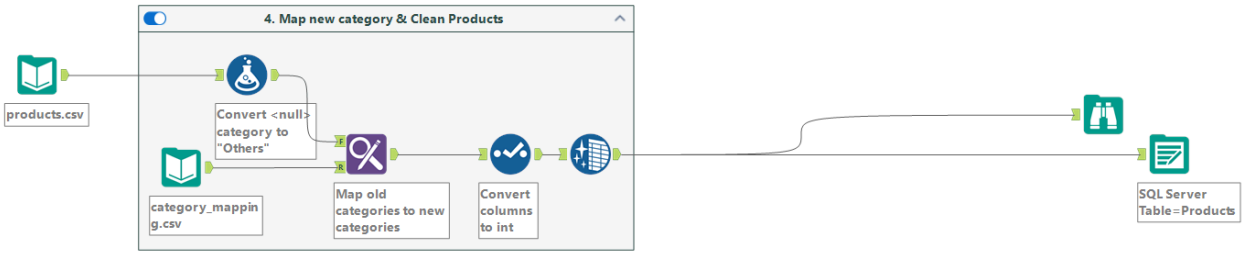
- `customers.csv`
- `sellers.csv`

Process Summary:

- Removed city and state columns (as they're now enriched and standardized via geolocation).
- Ensured only unique IDs remain for each entity.
- Cleaned datasets were written to SQL Server table: Customers & Sellers

Purpose: Standardizes customer and seller information by removing redundant or potentially inaccurate geographic fields, deferring to the cleaned geolocation dataset.

Step 2.4: Map New Category & Clean Products



Files Used:

- **`products.csv`**
- **`category_mapping.csv`** (Manually created to convert Portuguese or duplicated categories into English)

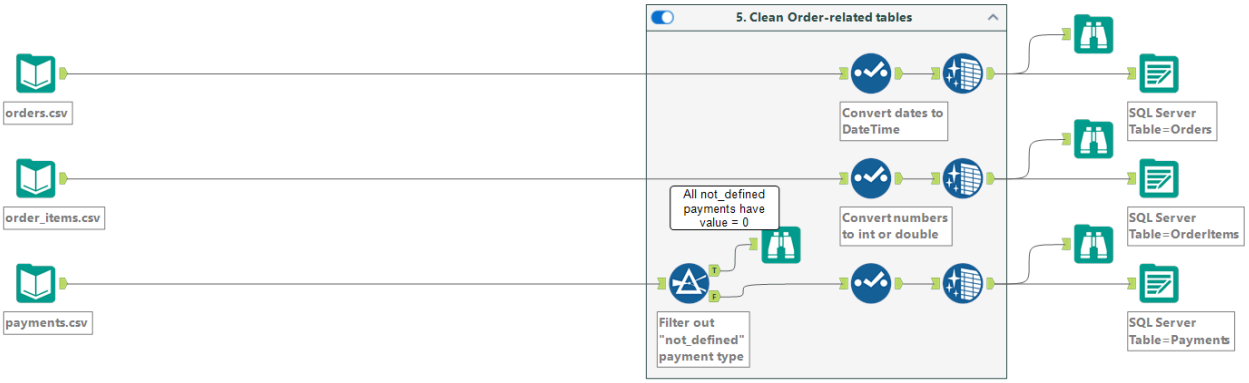
	A	B
1	old_category	new_category
2	Art	Arts
3	PCs	Computers
4	song	Songs & Audio
5	toys	Toys
6	audio	Songs & Audio
7	foods	Food
8	babies	Baby Care
9	drinks	Food
10	flowers	Flowers
11	PC Gamer	Computers
12	pet Shop	Pets
13	Furniture	Furniture
14	perfumery	Personal Care & Beauty

Process Summary:

- Replaced **`NULL`** values in product categories with **`"Others"`** to preserve rows with missing category info.
- Used **`category_mapping.csv`** to remap old category names to new standardized labels.
- Converted key product metadata (like weight and dimensions) to integers for consistency.
- Output was written to the SQL Server Table Products

Purpose: Creates clean, structured product data with standardized categories and numeric fields for analysis, like weight-based delivery cost or category-based sales.

Step 2.5: Clean Order-Related Tables



Files Used:

- **`orders.csv`**
- **`order_items.csv`**
- **`payments.csv`**

Process Summary:

- Converted string dates (e.g. order_purchase_timestamp, order_delivered_customer_date) to proper DateTime format.
- Casted numeric fields (e.g. price, freight, payment values) into integers/doubles for math operations.
- Filtered out records with payment_type = 'not_defined' as these are not useful for analysis.
- Cleaned tables were written to the SQL Server Tables: Orders, OrderItems, Payments

Purpose: Transforms transactional and financial data into an analysis-ready format. Date conversions allow time series analysis, while cleaning payment types ensures valid aggregation of payment data.

3. SSAS Tabular Model

Data Modeling

The implemented schema follows a **star-like schema with elements of a fact constellation**. Centralized fact tables capture transactional data (FactOrderItems, FactOrders, FactPayments), surrounded by descriptive dimension tables. This hybrid star-galaxy structure is common in retail and e-commerce models, balancing performance with data normalization and supports efficient OLAP-style analysis in SSAS Tabular Model.

The primary tables include:

Fact Tables:

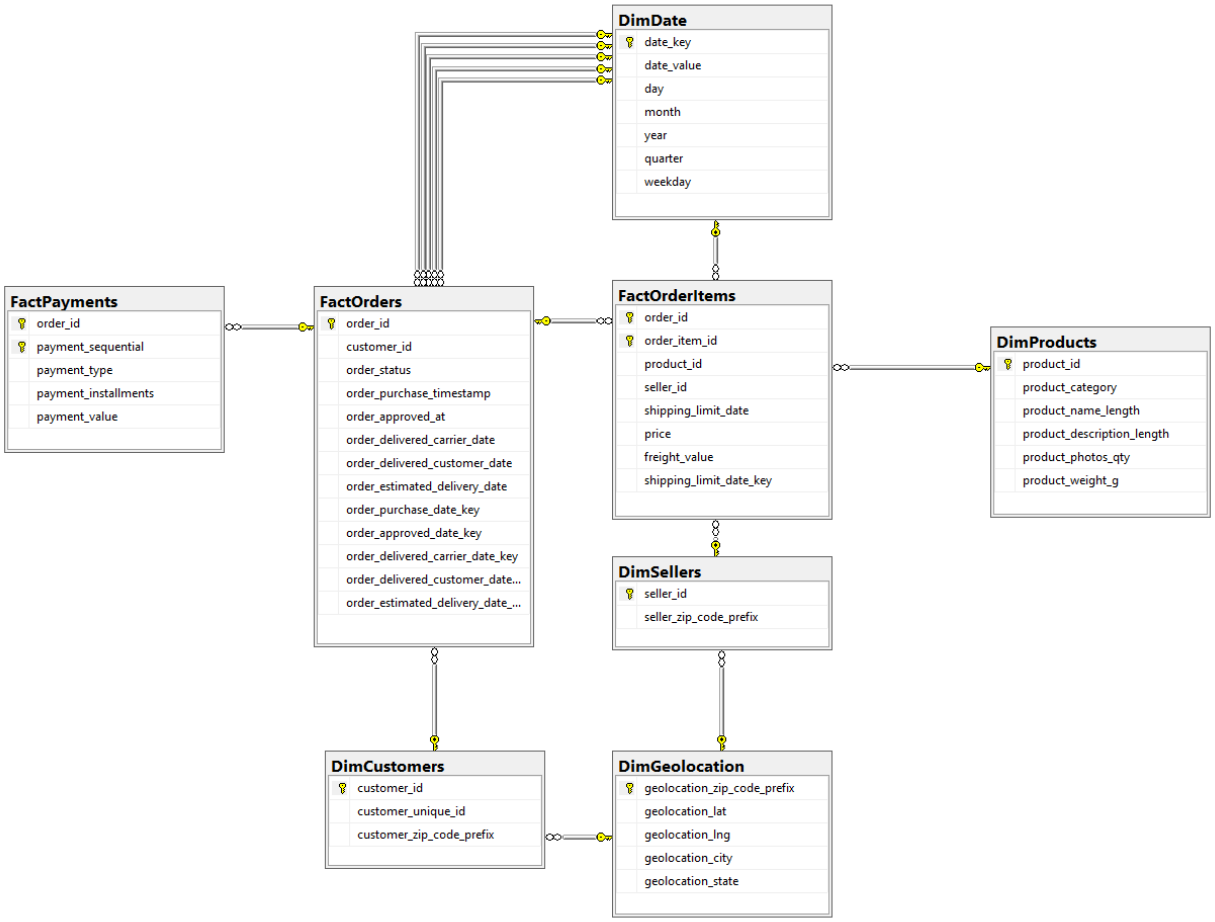
- **`FactOrderItems:`** Core sales and logistics data
- **`FactOrders:`** Order lifecycle metadata
- **`FactPayments:`** Payment method and amount details

Dimension Tables:

- **`DimDate:`** Full calendar table with hierarchical levels (Year, Quarter, Month, Day)
- **`DimProducts:`** Product-related attributes
- **`DimSellers:`** Seller identifiers and zip codes
- **`DimGeolocation:`** Geographic data including city, state, and coordinates
- **`DimCustomers:`** Customer ID and zip prefix for spatial analysis.

ERD Diagram

A relationship diagram (ERD) was created to define the primary and foreign key connections between these tables, supporting robust slicing and dicing operations in the tabular model.



1. Geographic Relationships:

- **`DimCustomers[customer_zip_code_prefix]` → `DimGeolocation[geolocation_zip_code_prefix]`**
- **`DimSellers[seller_zip_code_prefix]` → `DimGeolocation[geolocation_zip_code_prefix]`**

2. Fact-to-Dimension Relationships:

- **`FactOrderItems[product_id]` → `DimProducts[product_id]`**
- **`FactOrderItems[seller_id]` → `DimSellers[seller_id]`**

- **FactOrderItems[order_id] → FactOrders[order_id]**
- **FactOrders[customer_id] → DimCustomers[customer_id]**
- **FactPayments[order_id] → FactOrders[order_id]**

3. Date Relationships:

Each date relationship connects back to a single shared DimDate table, enabling robust time intelligence calculations across the order lifecycle.

- **FactOrderItems[shipping_limit_date_key] → DimDate[date_key]**
- **FactOrders[order_approved_date_key] → DimDate[date_key]**
- **FactOrders[order_delivered_carrier_date_key] → DimDate[date_key]**
- **FactOrders[order_delivered_customer_date_key] → DimDate[date_key]**
- **FactOrders[order_estimated_delivery_date_key] → DimDate[date_key]**
- **FactOrders[order_purchase_date_key] → DimDate[date_key]**

SQL Scripts for schema creation

Create dimension tables

```
CREATE TABLE DimDate (
    date_key INT PRIMARY KEY,           -- YYYYMMDD format
    date_value DATE NOT NULL,
    day TINYINT,
    month TINYINT,
    year SMALLINT,
    quarter TINYINT,
    weekday TINYINT,
);

CREATE TABLE DimCustomers (
    customer_id NVARCHAR(254) PRIMARY KEY,
    customer_unique_id NVARCHAR(254),
    customer_zip_code_prefix INT
);

CREATE TABLE DimSellers (
    seller_id NVARCHAR(254) PRIMARY KEY,
    seller_zip_code_prefix INT
);
```

Link customers and sellers to geolocation

```
ALTER TABLE DimCustomers
ADD FOREIGN KEY (customer_zip_code_prefix) REFERENCES
DimGeolocation(geolocation_zip_code_prefix);

ALTER TABLE DimSellers
ADD FOREIGN KEY (seller_zip_code_prefix) REFERENCES
DimGeolocation(geolocation_zip_code_prefix);
```

Create fact tables

```
CREATE TABLE DimProducts (
    product_id NVARCHAR(254) PRIMARY KEY,
    product_category NVARCHAR(254),
    product_name_length INT,
    product_description_length INT,
    product_photos_qty INT,
    product_weight_g INT
);

CREATE TABLE DimGeolocation (
    geolocation_zip_code_prefix INT PRIMARY KEY,
    geolocation_lat DECIMAL(10, 2),
    geolocation_lng DECIMAL(10, 2),
    geolocation_city NVARCHAR(254),
    geolocation_state NVARCHAR(254)
);

CREATE TABLE FactOrders (
    order_id NVARCHAR(254) PRIMARY KEY,
    customer_id NVARCHAR(254),
    order_status NVARCHAR(254),
    order_purchase_timestamp DATETIME,
    order_approved_at DATETIME,
    order_delivered_carrier_date DATETIME,
    order_delivered_customer_date DATETIME,
    order_estimated_delivery_date DATETIME,
    FOREIGN KEY (customer_id) REFERENCES DimCustomers(customer_id)
);

CREATE TABLE FactOrderItems (
    order_id NVARCHAR(254),
```

```
    order_item_id INT,
    product_id NVARCHAR(254),
    seller_id NVARCHAR(254),
    shipping_limit_date DATETIME,
    price DECIMAL(10, 2),
    freight_value DECIMAL(10, 2),
    PRIMARY KEY (order_id, order_item_id),
    FOREIGN KEY (order_id) REFERENCES FactOrders(order_id),
    FOREIGN KEY (product_id) REFERENCES DimProducts(product_id),
    FOREIGN KEY (seller_id) REFERENCES DimSellers(seller_id)
);

CREATE TABLE FactPayments (
    order_id NVARCHAR(254),
    payment_sequential INT,
    payment_type NVARCHAR(50),
    payment_installments INT,
    payment_value DECIMAL(10, 2),
    PRIMARY KEY (order_id, payment_sequential),
    FOREIGN KEY (order_id) REFERENCES FactOrders(order_id)
);
```

Recursively generate a complete date spine

```
SELECT
    @MinDate = MIN(date_val),
    @MaxDate = MAX(date_val)
FROM (
    SELECT order_purchase_timestamp AS date_val FROM FactOrders
    UNION ALL
    SELECT order_approved_at FROM FactOrders
    UNION ALL
    SELECT order_delivered_carrier_date FROM FactOrders
    UNION ALL
    SELECT order_delivered_customer_date FROM FactOrders
    UNION ALL
    SELECT order_estimated_delivery_date FROM FactOrders
    UNION ALL
    SELECT shipping_limit_date FROM FactOrderItems
) AS AllDates;
WITH DateSpine AS (
    SELECT @MinDate AS date_value
```

```
    UNION ALL
    SELECT DATEADD(DAY, 1, date_value)
    FROM DateSpine
    WHERE date_value < @MaxDate
)
INSERT INTO DimDate (date_key, date_value, day, month, year, quarter,
weekday)
SELECT
    CAST(FORMAT(date_value, 'yyyyMMdd') AS INT),
    date_value,
    DATEPART(DAY, date_value),
    DATEPART(MONTH, date_value),
    DATEPART(YEAR, date_value),
    DATEPART(QUARTER, date_value),
    DATEPART(WEEKDAY, date_value)
FROM DateSpine
OPTION (MAXRECURSION 32767);
```

Add surrogate date keys for fact-dimension joins

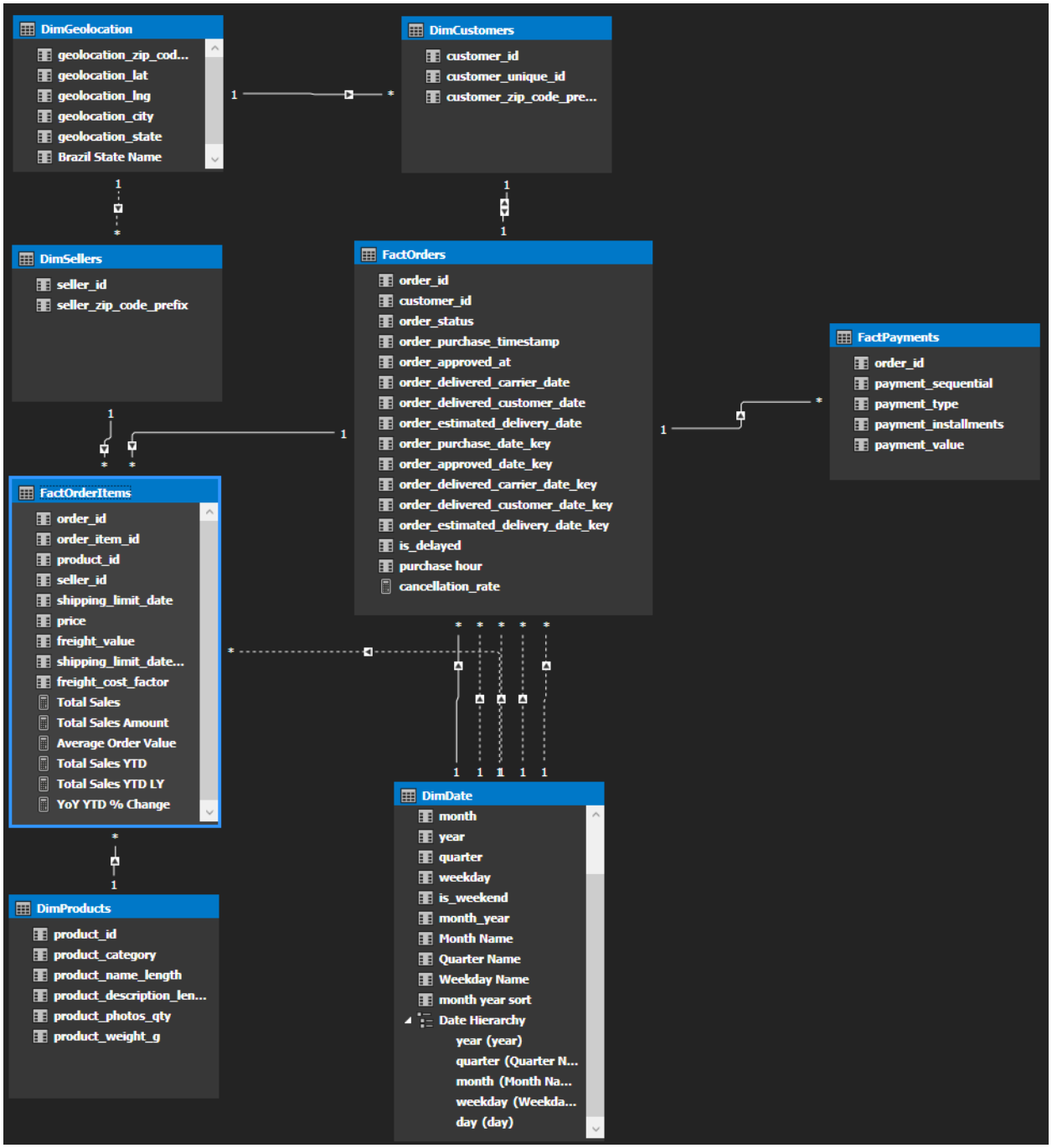
```
ALTER TABLE FactOrders ADD order_purchase_date_key INT;
ALTER TABLE FactOrders ADD order_approved_date_key INT;
ALTER TABLE FactOrders ADD order_delivered_carrier_date_key INT;
ALTER TABLE FactOrders ADD order_delivered_customer_date_key INT;
ALTER TABLE FactOrders ADD order_estimated_delivery_date_key INT;
ALTER TABLE FactOrderItems ADD shipping_limit_date_key INT;
```

Add foreign key constraints for all date keys

```
ALTER TABLE FactOrders
ADD CONSTRAINT FK_FactOrders_order_purchase_date
FOREIGN KEY (order_purchase_date_key) REFERENCES DimDate(date_key);

ALTER TABLE FactOrders
ADD CONSTRAINT FK_FactOrders_order_approved_date
FOREIGN KEY (order_approved_date_key) REFERENCES DimDate(date_key);

ALTER TABLE FactOrders
ADD CONSTRAINT FK_FactOrders_carrier_date
FOREIGN KEY (order_delivered_carrier_date_key) REFERENCES
DimDate(date_key);
```

```
ALTER TABLE FactOrders
ADD CONSTRAINT FK_FactOrders_customer_date
FOREIGN KEY (order_delivered_customer_date_key) REFERENCES
DimDate(date_key);

ALTER TABLE FactOrders
ADD CONSTRAINT FK_FactOrders_estimated_date
FOREIGN KEY (order_estimated_delivery_date_key) REFERENCES
DimDate(date_key);

ALTER TABLE FactOrderItems
ADD CONSTRAINT FK_FactOrderItems_shipping_date
FOREIGN KEY (shipping_limit_date_key) REFERENCES DimDate(date_key);
```

Tabular Model Implementation



Steps:

1. Tables were imported into the model from SQL Server.
2. Relationships were defined as per the schema above.
3. Measures were created using DAX, including:

- **[Total Sales]**
- **[Total Sales Amount]**
- **[Freight Cost Factor]**
- **[Total Sales YTD]**
- **[Total Sales YTD LY]**
- **[YoY YTD % Change]**
- **[Average Order Value]**
- **[cancellation_rate]**

4. Calculated columns were created using DAX, including:
   - **FactOrders[is_delayed]**
   - **DimDate[is_weekend]**
   - **DimDate[month_year]**
5. Date Hierarchy was created in DimDate table (marked as Date Table using DimDate[date_value] as identifier)

DAX Test Scripts for SSAS Cube

Top 5 Months by Sales (Across All Years)

```
EVALUATE
TOPN(
    5,
    ADDCOLUMNS(
        SUMMARIZECOLUMNS(
            DimDate[month_year]
        ),
        "Total Sales", [Total Sales]
    ),
    [Total Sales], DESC
)
```

DimDate[month_...	[Total Sales]
May 2018	7925
Apr 2018	7975
Mar 2018	8217
Nov 2017	8665
Jan 2018	8208

Top 10 Selling Product Categories – Sales and AOV

```
EVALUATE
TOPN(
    10,
    ADDCOLUMNS(
        SUMMARIZECOLUMNS(
            DimProducts[product_category]
        ),
        "Total Sales", [Total Sales],
           "Total Sales Amount", [Total Sales Amount],
        "Average Order Value", [Average Order Value]
    ),
    [Total Sales], DESC
)
```

DimProducts[pro...	[Total Sales]	[Total Sales Amo...	[Average Order ...
Toys	4117	483946.6	124.54
Gardening	4347	485256.46	137.94
Automotive	4235	592720.11	152.1
Computers	8039	1136463.4	165.23
Telephone	4545	323667.53	77.08
Furniture	22071	2180353.66	122.16
Others	6061	890402.62	156.93
Personal Care & ...	13128	1659373.8	138.13
Household	8572	874223.17	118.2
Fashion	18366	2535703.31	151.11

Sales comparison YTD

```
EVALUATE
SELECTCOLUMNS(
    SUMMARIZECOLUMNS(
        DimDate[month_year],
        "SortKey", MIN(DimDate[date_key]),
        "Total Sales", [Total Sales YTD],
        "Sales Last Year", [Total Sales YTD LY],
        "YoY % Change", [YoY YTD % Change]
    ),
    "Month Year", DimDate[month_year],
    "Total Sales", [Total Sales],
    "Sales Last Year", [Sales Last Year],
    "YoY % Change", [YoY % Change],
   "SortKey", [SortKey]
)
```

```
ORDER BY [SortKey]
```

[Month Year]	[Total Sales]	[Sales Last Year]	[YoY % Change]	[SortKey]
Jun 2017	16309			20170601
Jul 2017	20828			20170701
Aug 2017	25738			20170801
Sep 2017	30569	6	5093.83333333...	20170901
Oct 2017	35891	369	96.2655826558...	20171001
Nov 2017	44556	369	119.747967479...	20171101
Dec 2017	50864	370	136.47027027027	20171201
Jan 2018	8208	955	7.59476439790...	20180101
Feb 2018	15880	2906	4.46455609084...	20180201
Mar 2018	24097	5906	3.08008804605...	20180301
Apr 2018	32072	8590	2.73364377182...	20180401
May 2018	39997	12726	2.14293572214...	20180501
Jun 2018	47075	16309	1.88644306824...	20180601
Jul 2018	54167	20828	1.60068177453...	20180701
Aug 2018	61415	25738	1.38616054083...	20180801
Sep 2018	61416	30569	1.00909418037...	20180901
Oct 2018	61416	35891	0.71118107603...	20181001
Nov 2018	61416	44556	0.37840021545...	20181101
Dec 2018	61416	50864	0.20745517458...	20181201

Top 5 selling city

```
EVALUATE
VAR Top5Cities =
    TOPN(
        5,
        ADDCOLUMNS(
            SUMMARIZECOLUMNS(
                DimGeolocation[geolocation_state],
                DimGeolocation[geolocation_city]
            ),
            "Total Sales", CALCULATE(
                [Total Sales],
                USERELATIONSHIP(DimSellers[seller_zip_code_prefix],
DimGeolocation[geolocation_zip_code_prefix])
            ),
            "Total Sales Amount", CALCULATE(
                [Total Sales Amount],
```

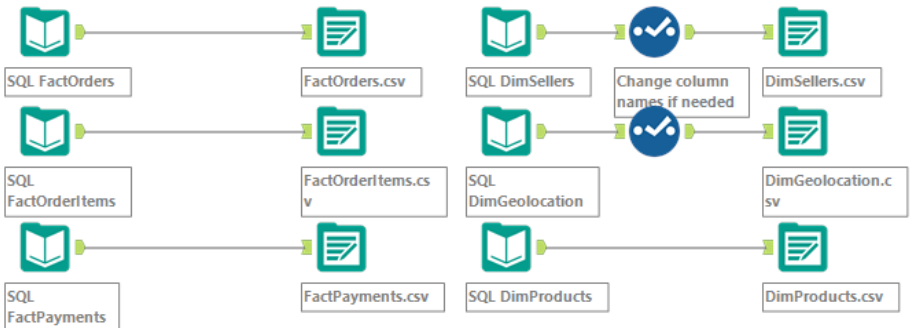
```
                USERELATIONSHIP(DimSellers[seller_zip_code_prefix],
DimGeolocation[geolocation_zip_code_prefix])
            ),
                "Average Freight Cost",
AVERAGE(FactOrderItems[freight_value])
        ),
        [Total Sales], DESC
    )

RETURN
SELECTCOLUMNS(
    Top5Cities,
    "State", DimGeolocation[geolocation_state],
    "City", DimGeolocation[geolocation_city],
    "Total Sales", [Total Sales],
    "Total Sales Amount", [Total Sales Amount],
   "Average Freight Cost", ROUND([Average Freight Cost], 2)
)
ORDER BY [Total Sales] DESC
```

[State]	[City]	[Total Sales]	[Total Sales Amo...	[Average Freight ...
SP	sao paulo	28316	2740651.69	19.99
SP	ibitinga	7750	624592.94	19.99
PR	curitiba	3016	470759.82	19.99
SP	santo andre	2992	230901.84	19.99
SP	sao jose do rio p...	2695	208346.26	19.99

4. Data Analysis & Queries with PySpark

Before Analysis

We need to export our tables from MS SQL Server into CSV files so that we can import them into a PySpark session, using Alteryx again. **Change column names if needed**.



After that, we import the files into the PySpark session, convert each table from a PySpark DataFrame into a temporary SQL view, so that we can directly use SQL queries to get the query results needed.

```
fact_orders = spark.read.csv("./Step 4 Exported CSV/FactOrders.csv",
header=True, inferSchema=True)
fact_orderitems = spark.read.csv("./Step 4 Exported
CSV/FactOrderItems.csv", header=True, inferSchema=True)
fact_payments = spark.read.csv("./Step 4 Exported CSV/FactPayments.csv",
header=True, inferSchema=True)
dim_sellers = spark.read.csv("./Step 4 Exported CSV/DimSellers.csv",
header=True, inferSchema=True)
dim_geolocation = spark.read.csv("./Step 4 Exported
CSV/DimGeolocation.csv", header=True, inferSchema=True)
dim_products = spark.read.csv("./Step 4 Exported CSV/DimProducts.csv",
header=True, inferSchema=True)

fact_orders.createOrReplaceTempView("FactOrders")
fact_orderitems.createOrReplaceTempView("FactOrderItems")
fact_payments.createOrReplaceTempView("FactPayments")
dim_sellers.createOrReplaceTempView("DimSellers")
dim_geolocation.createOrReplaceTempView("DimGeolocation")
dim_products.createOrReplaceTempView("DimProducts")
```

Q1. What's the customer's purchase pattern?

- Query 1.1: Sales volume over time (monthly, yearly)

```
# 1.1.1 Sales volume over time (Yearly)
SELECT
    YEAR (fo.order_purchase_timestamp) AS sales_year,
    SUM (foi.price + foi.freight_value) AS total_sales_volume
FROM FactOrders fo
    JOIN FactOrderItems foi ON fo.order_id = foi.order_id
GROUP BY YEAR (fo.order_purchase_timestamp)
ORDER BY sales_year;
```

```
+----------+------------------+
|sales_year|total_sales_volume|
+----------+------------------+
|2016      |57183.21          |
|2017      |7142672.430000115 |
|2018      |8643697.599999946 |
+----------+------------------+
```

```
# 1.1.2 Sales volume over time (Monthly)
SELECT
    YEAR (fo.order_purchase_timestamp) AS sales_year,
    MONTH (fo.order_purchase_timestamp) AS sales_month,
    SUM (foi.price + foi.freight_value) AS total_sales_volume
FROM FactOrders fo
    JOIN FactOrderItems foi ON fo.order_id = foi.order_id
GROUP BY YEAR (fo.order_purchase_timestamp),
MONTH(fo.order_purchase_timestamp)
ORDER BY sales_year, sales_month;
```

```
+----------+-----------+------------------+
|sales_year|sales_month|total_sales_volume|
+----------+-----------+------------------+
|2016      |9          |354.75            |
|2016      |10         |56808.84          |
|2016      |12         |19.62             |
|2017      |1          |137188.49000000005|
|2017      |2          |286280.61999999976|
|2017      |3          |432048.5899999997 |
|2017      |4          |412422.23999999964|
|2017      |5          |586190.9499999994 |
|2017      |6          |502963.0399999998 |
|2017      |7          |584971.6199999996 |
```

Analysis:

- Sales have been growing consistently each year.
- Sales tend to peak sometime between the middle and the end of the year.

- **Query 1.2: Payment method breakdown**

```
SELECT
    payment_type,
    AVG (payment_value) AS average_payment_value,
    SUM (payment_value) AS total_payment_value,
    COUNT (order_id) AS number_of_transactions,
    AVG (payment_installments) AS average_installments
FROM FactPayments
GROUP BY payment_type
ORDER BY total_payment_value DESC;
```

payment_type	average_payment_value	total_payment_value	number_of_transactions	average_installments
credit_card	163.31902063935948	1.2542084189999961E7	76795	3.507155413763917
UPI	145.03443540234582	2869361.27000001	19784	1.0
voucher	65.70335411255415	379436.8700000002	5775	1.0
debit_card	142.57017004578148	217989.78999999986	1529	1.0

Analysis:

- Credit cards account for the vast majority of both transactions and average payments, with UPI having a close second in average payments.
- Very little of the revenue comes from vouchers and debit cards.
- Customers tend to pay in installments when using credit cards.

Q2. Which store (seller) locations are the customers most or least satisfied with?

- **Query 2.1: Cancel and delay rate by seller location**

```
# 2.1.1 Delay rate by location
SELECT
    query.state,
    query.city,
    query.total_orders,
    query.delayed_orders,
```

```
    query.delay_rate_percentage
FROM (
    SELECT
        dg.state,
        dg.city,
        COUNT(DISTINCT fo.order_id) AS total_orders,
        COUNT(DISTINCT CASE WHEN fo.order_delivered_customer_date >
fo.order_estimated_delivery_date THEN fo.order_id END) AS delayed_orders,
        CAST (
            COUNT(DISTINCT CASE WHEN fo.order_delivered_customer_date >
fo.order_estimated_delivery_date THEN fo.order_id END) * 100.0 /
COUNT(DISTINCT fo.order_id)
            AS DECIMAL(10,2)
        ) AS delay_rate_percentage,
        RANK() OVER (
            PARTITION BY dg.state
            ORDER BY COUNT(DISTINCT CASE WHEN
fo.order_delivered_customer_date > fo.order_estimated_delivery_date THEN
fo.order_id END) * 100.0 / COUNT(DISTINCT fo.order_id) DESC
        ) AS rnk
    FROM FactOrders fo
        JOIN FactOrderItems foi ON fo.order_id = foi.order_id
        JOIN DimSellers ds ON foi.seller_id = ds.seller_id
        JOIN DimGeolocation dg ON ds.zip_code_prefix = dg.zip_code_prefix
    GROUP BY dg.state, dg.city
) AS query
WHERE query.rnk <= 10
ORDER BY query.state, query.rnk;
```

state	city	total_orders	delayed_orders	delay_rate_percentage
AC	rio branco	1	0	0.00
AM	manaus	3	2	66.67
BA	feira de santana	3	1	33.33
BA	ipira	4	1	25.00
BA	arraial d'ajuda	29	4	13.79
BA	salvador	106	10	9.43
BA	porto seguro	14	1	7.14
BA	ilheus	18	1	5.56
BA	lauro de freitas	359	15	4.18
BA	eunapolis	11	0	0.00
BA	guanambi	21	0	0.00

```
# 2.1.2 Cancel rate by location
SELECT
    query.state,
    query.city,
    query.total_orders,
    query.canceled_orders,
    query.cancel_rate_percentage
FROM (
   SELECT
         dg.state,
         dg.city,
         COUNT(DISTINCT fo.order_id) AS total_orders,
         COUNT(DISTINCT CASE WHEN fo.order_status = 'canceled' THEN
fo.order_id END) AS canceled_orders,
         CAST (
               COUNT(DISTINCT CASE WHEN fo.order_status = 'canceled' THEN
fo.order_id END) * 100.0 / COUNT(DISTINCT fo.order_id)
               AS DECIMAL(10,2)
         ) AS cancel_rate_percentage,
         RANK() OVER (
               PARTITION BY dg.state
               -- line below here is the same as cancel_rate_percentage
without 2 decimal places
               ORDER BY COUNT(DISTINCT CASE WHEN fo.order_status =
'canceled' THEN fo.order_id END) * 100.0 / COUNT(DISTINCT fo.order_id)
               DESC
         ) AS rnk
   FROM FactOrders fo
         JOIN FactOrderItems foi ON fo.order_id = foi.order_id
         JOIN DimSellers ds ON foi.seller_id = ds.seller_id
         JOIN DimGeolocation dg ON ds.zip_code_prefix = dg.zip_code_prefix
   GROUP BY dg.state, dg.city
) AS query
WHERE query.rnk <= 10
ORDER BY query.state, query.rnk;
```

state	city	total_orders	canceled_orders	cancel_rate_percentage
AC	rio branco	1	0	0.00
AM	manaus	3	0	0.00
BA	arraial d'ajuda	29	1	3.45
BA	salvador	106	1	0.94
BA	lauro de freitas	359	0	0.00
BA	porto seguro	14	0	0.00
BA	eunapolis	11	0	0.00
BA	guanambi	21	0	0.00
BA	ipira	4	0	0.00
BA	ilheus	18	0	0.00
BA	[illegible]	1	0	0.00

Analysis:

- The states São Paulo (SP) and Minas Gerais (MG) have consistently higher cancellation and delay rates compared to the rest of the country. Focusing on logistics here should yield the highest benefit.
- There are some states, such as Paraná (PR) or Maranhão (MA), that have high delay rates, but low cancellation rates. Focusing on logistics here would be less useful, but still make a positive impact.

- **Query 2.2: Total sales by seller location**

```
SELECT
    query.state,
    query.city,
    query.total_sales_volume
FROM (
    SELECT
        dg.state,
        dg.city,
        CAST (SUM (foi.price + foi.freight_value) AS DECIMAL(20,2) ) AS
total_sales_volume,
        RANK() OVER (
               PARTITION BY dg.state
               ORDER BY SUM (foi.price + foi.freight_value)
               DESC
         ) AS rnk
    FROM FactOrderItems foi
        JOIN DimSellers ds ON foi.seller_id = ds.seller_id
        JOIN DimGeolocation dg ON ds.zip_code_prefix = dg.zip_code_prefix
```

```
    GROUP BY dg.state, dg.city
) AS query
    WHERE query.rnk <=10
    ORDER BY query.state, query.rnk;
```

state	city	total_sales_volume
AC	rio branco	299.84
AM	manaus	1258.80
BA	lauro de freitas	238675.42
BA	salvador	23078.54
BA	guanambi	18733.18
BA	ilheus	9644.62
BA	arraial d'ajuda	8435.21
BA	eunapolis	1880.00
BA	porto seguro	1774.94
BA	ipira	1751.21

Analysis:

- São Paulo (The state and the city) represents the vast majority of total sales volume.
- Most of the volumes focus around highly-dense and coastal regions of Brazil.

Q3. How to increase revenue?

- **Query 3.1: Top and bottom-selling product categories**

```
# 3.1.1 Top selling product categories
SELECT
    dp.product_category,
    CAST (SUM (foi.price + foi.freight_value) AS DECIMAL(20,2) ) AS
total_sales_volume
FROM FactOrderItems foi
   JOIN DimProducts dp ON foi.product_id = dp.product_id
GROUP BY dp.product_category
ORDER BY total_sales_volume DESC
LIMIT 10;
```

product_category	total_sales_volume
Fashion	2877024.84
Furniture	2662598.24
Personal Care & Beauty	1896728.05
Computers	1293751.35
Household	1053599.81
Others	1017178.47
Automotive	685384.32
Gardening	584219.21
Toys	561372.55
Electronics	537411.81

```
# 3.1.2 Bottom selling product categories
SELECT
    dp.product_category,
    CAST (SUM (foi.price + foi.freight_value) AS DECIMAL(20,2) ) AS
total_sales_volume
FROM FactOrderItems foi
   JOIN DimProducts dp ON foi.product_id = dp.product_id
GROUP BY dp.product_category
ORDER BY total_sales_volume ASC
LIMIT 10;
```

product_category	total_sales_volume
Insurance & Services	324.51
Flowers	1598.91
Photography	8189.66
CDs & DVDs	8243.12
Party Articles	17343.27
House Safety	28017.05
Arts	30431.95
Songs & Audio	63123.80
Telephony	64220.81
Agricultural	78374.07

Analysis:

- **Top-Selling** Product Categories: Furniture, Fashion, and Personal Care & Beauty are highly popular → Might consider stocking more inventory for the top 3 categories in sales revenue.

- **Lowest-Selling** Product Categories (excluding Insurance and Services): Flowers, Photography, CDs & DVDs → Consider removing altogether all flower & photography-related products due to low customer interest, while CDs & DVDs are considered obsolete at the current time.

- **Query 3.2: Sellers with the best sales record**

```
SELECT
    foi.seller_id,
   dg.city,
   dg.state,
    CAST (SUM (foi.price + foi.freight_value) AS DECIMAL(20,2) ) AS
total_sales_volume
FROM FactOrderItems foi
   JOIN DimSellers ds ON foi.seller_id = ds.seller_id
   JOIN DimGeolocation dg ON ds.zip_code_prefix = dg.zip_code_prefix
GROUP BY foi.seller_id, dg.state, dg.city
ORDER BY total_sales_volume DESC
LIMIT 10;
```

seller_id	city	state	total_sales_volume
4869f7a5dfa277a7dca6462dcf3b52b2	guariba	SP	249640.70
7c67e1448b00f6e969d365cea6b010ab	itaquaquecetuba	SP	239536.44
53243585a1d6dc2643021fd1853d8905	lauro de freitas	BA	235856.68
4a3ca9315b744ce9f8e9374361493884	ibitinga	SP	235539.96
fa1c13f2614d7b5c4749cbc52fecda94	sumare	SP	204084.73
da8622b14eb17ae2831f4ac5b9dab84a	piracicaba	SP	185192.32
7e93a43ef30c4f03f38b393420bc753a	barueri	SP	182754.05
1025f0e2d44d7041d6cf58b6550e0bfa	sao paulo	SP	172860.69
7a67c85e85bb2ce8582c35f2203ad736	sao paulo	SP	162648.38
955fee9216a65b617aa5c0531780ce60	sao paulo	SP	160602.68

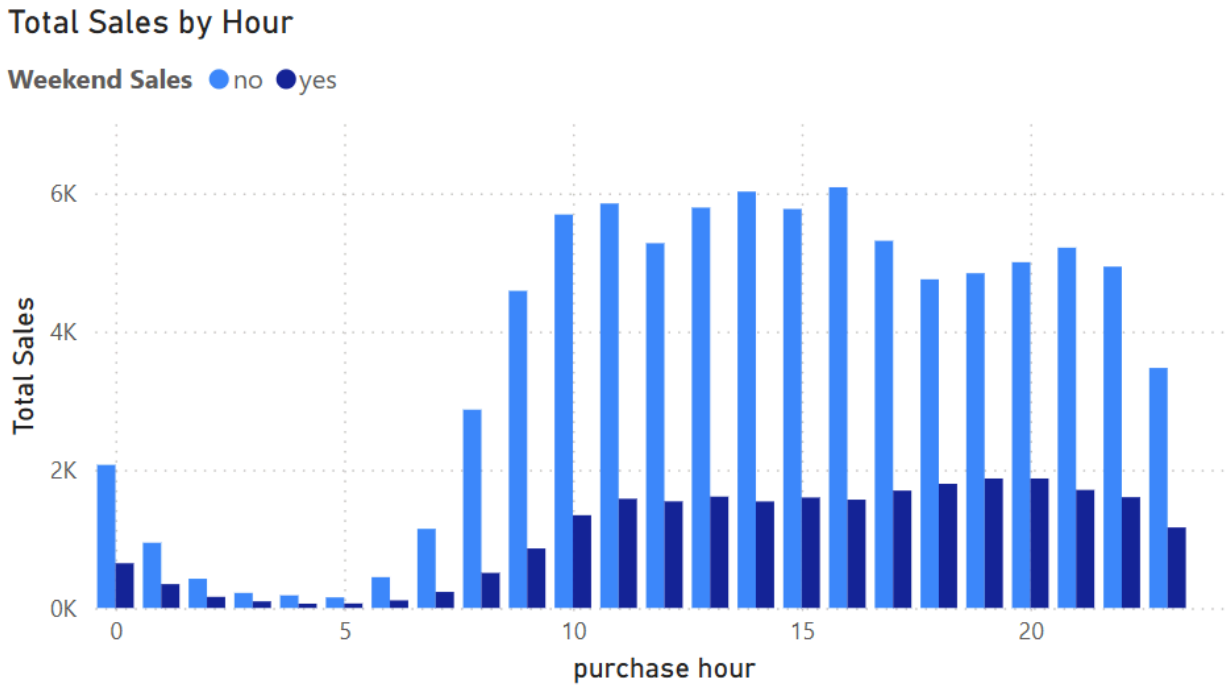
<u>Analysis:</u>

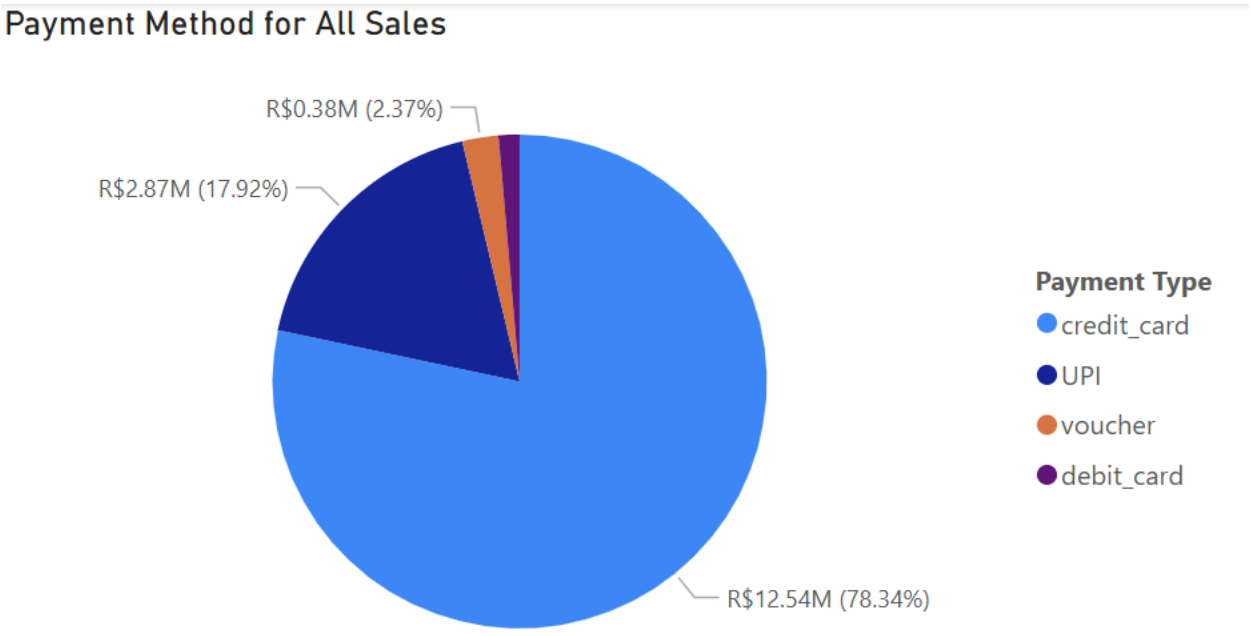
- **The state of São Paulo dominates sales**: Out of the top 10 stores, 9 operate in São Paulo (SP) → This indicates that SP is a central hub for e-commerce transactions.
- While most are from big cities in SP (São Paulo, Piracicaba, Sumaré, Barueri, Itaquaquecetuba), the top 10 also includes **Ibitinga and Guariba are two small**

cities in the regional area → Further research to set up more stores around the regional area of SP might be a good idea.

5. Visualization & Reporting with Power BI

Customer Purchase Pattern





Insight:

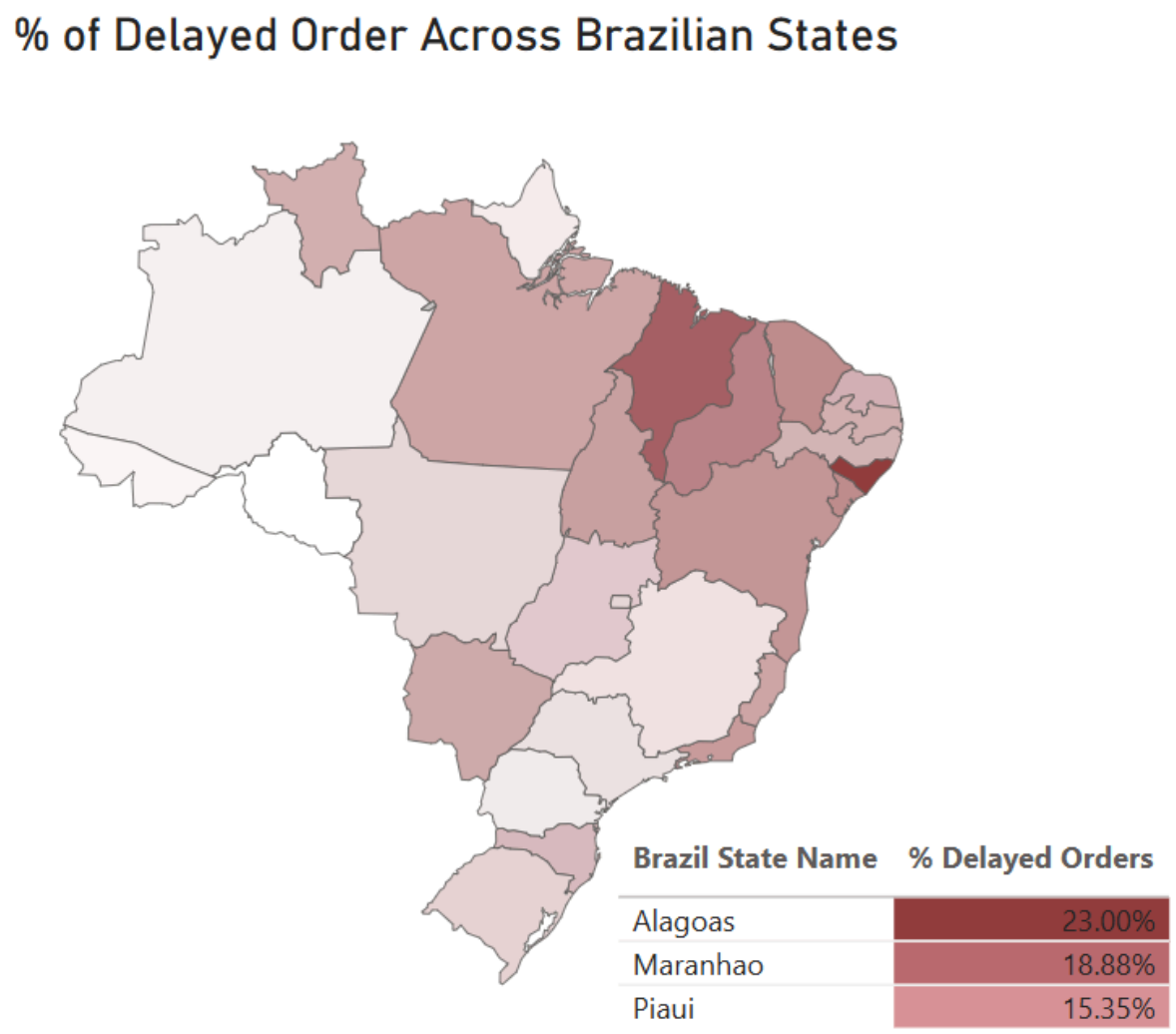
- There's a strong upward trend in total sales from late 2016 through mid-2018.
- Sales peak consistently between November and March, suggesting seasonality in purchasing.
- End-of-year and Q1 demand spikes.
- Credit card dominates as the preferred payment method — 78.34% of all sales.

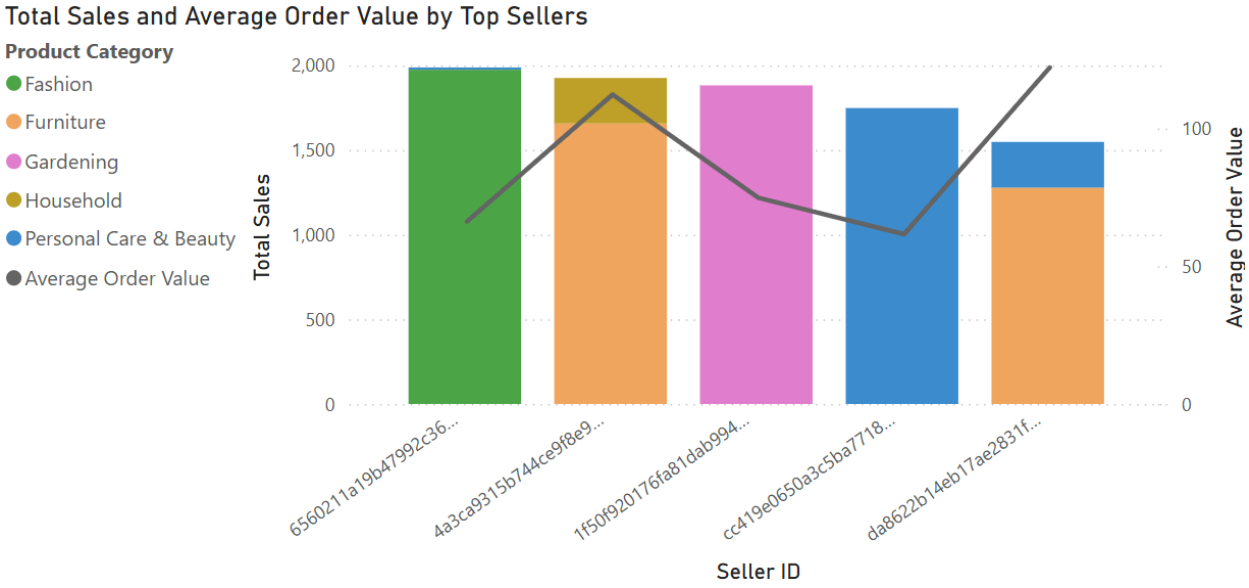
- UPI follows (17.92%), while debit card and voucher usage is negligible.
- Sales activity increases sharply starting at 8 AM, peaking between 10 AM to 4 PM.
- Weekend sales follow a similar hour pattern with weekday sales.

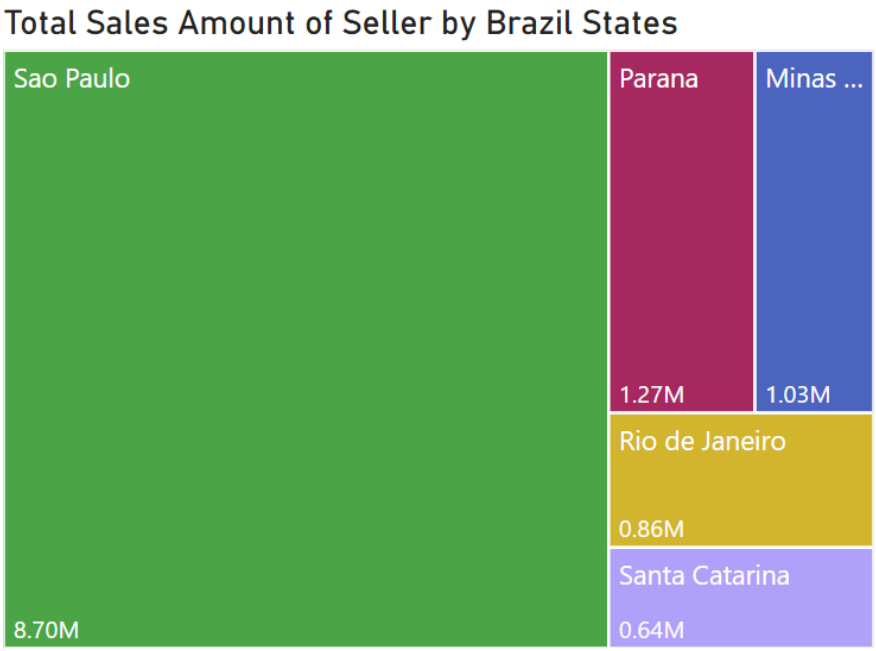
Business Implications:

- Plan inventory, staffing, and marketing efforts around Q4–Q1 (Nov–Mar).
- Investigate and forecast based on seasonal peaks.
- Address low-performance periods (e.g., Q3) with targeted promotions or discounts.
- Ensure checkout processes and promotions are optimized for credit card users.
- Consider partnerships or offers with credit card providers to drive loyalty.
- Evaluate why voucher and debit card usage is low — UX issues? Poor support?
- Customers prefer to shop during daytime hours, especially on weekdays.
- Schedule system maintenance or updates outside these peak hours.
- Run weekday-specific promotions or boost weekend engagement with flash sales.

Seller Performance







Insight:

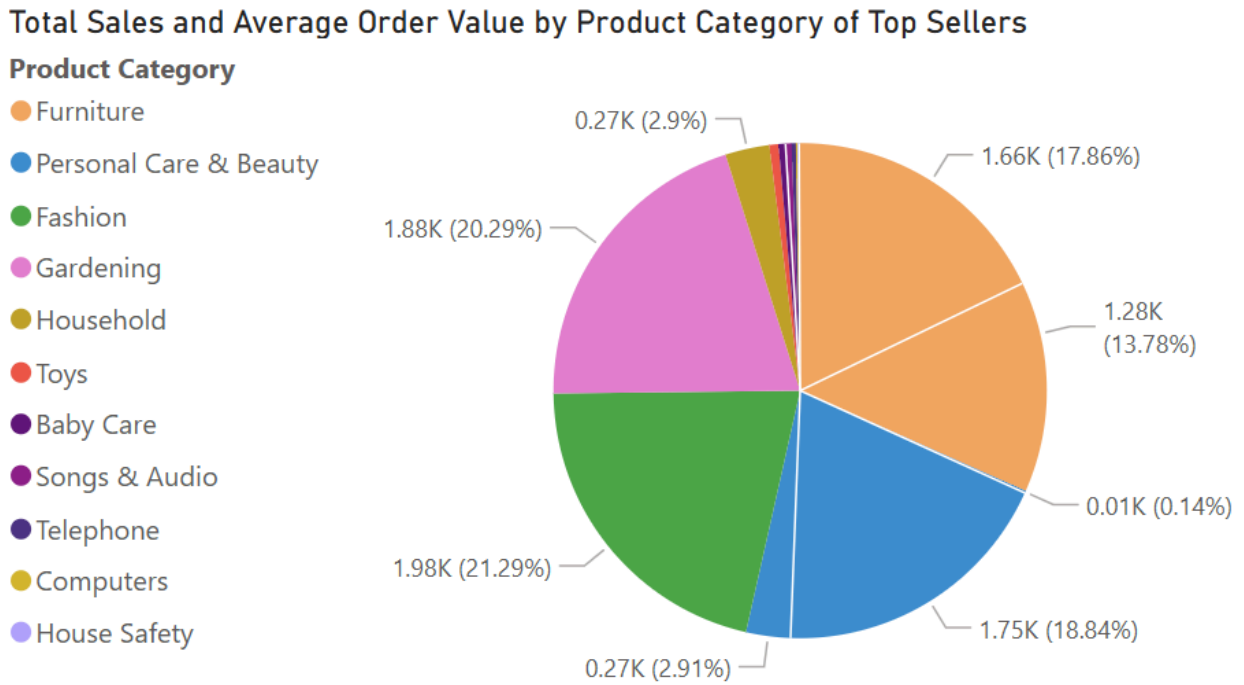
- The northeastern and northern states (e.g., Alagoas: 23%, Maranhão: 18.88%, Piauí: 15.35%) experience the highest order delays.
- Central and southern states show lighter shading, indicating lower delay rates.
- São Paulo dominates with R$8.70M, vastly outpacing all other states, with secondary performing states Paraná (R$1.27M), Minas Gerais (R$1.03M).
- Fashion, Furniture, Gardening and Personal Care & Beauty are among the major categories from the top sellers.
- Furniture has the highest AOV, suggesting that furniture sellers are successfully scaling. Despite lower AOV on Fashion and Gardening sellers, the sales are good in those categories.

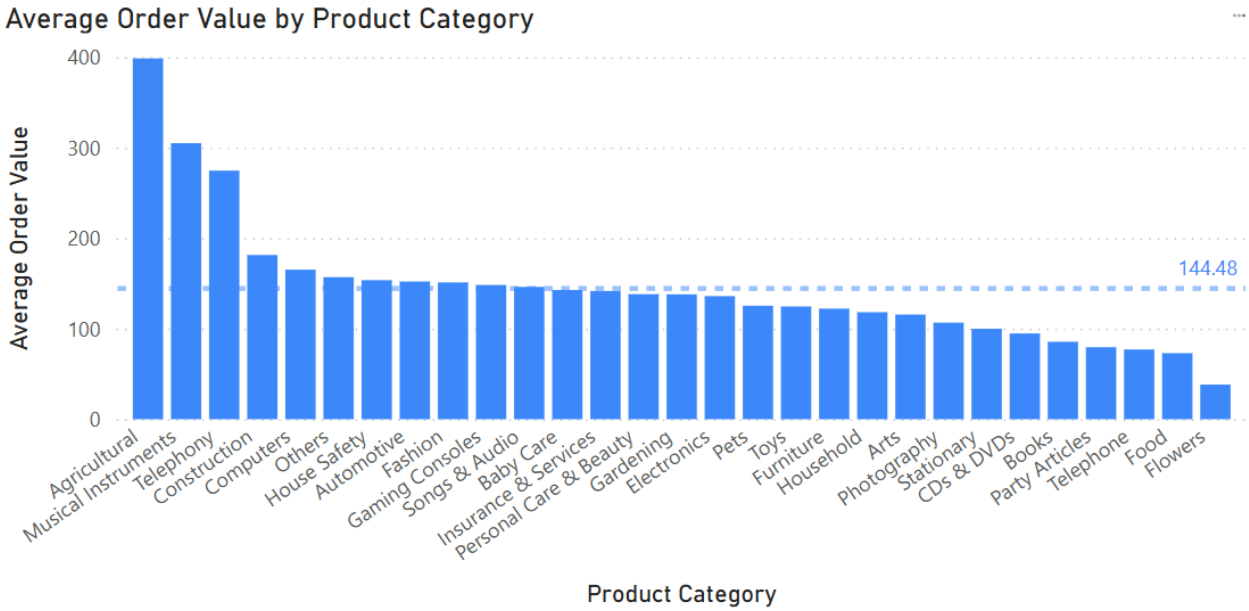
Business Implications:

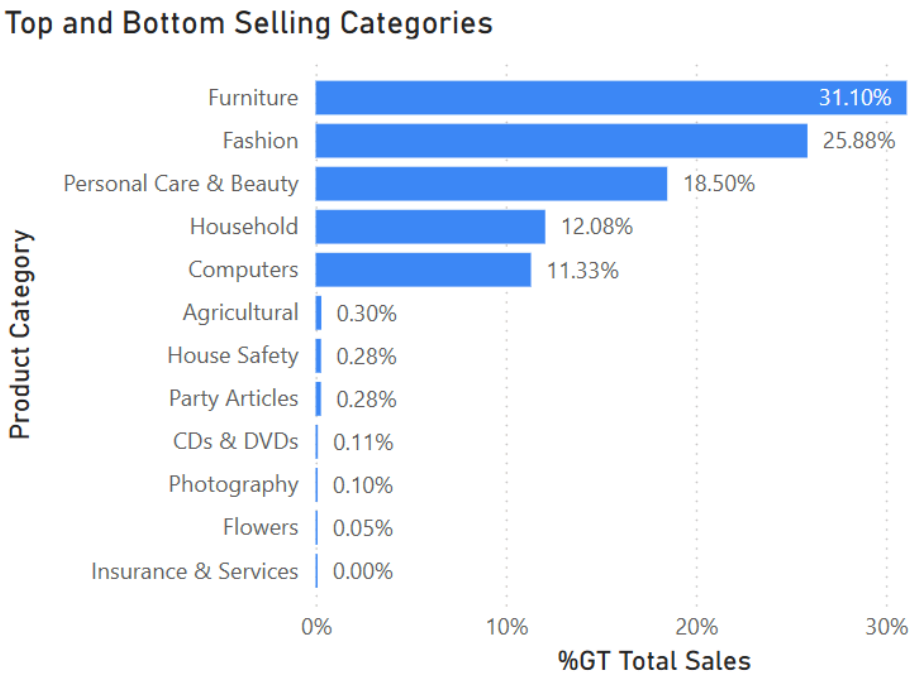
- Logistical challenges in certain regions (e.g., less developed infrastructure or longer delivery routes). Evaluating last-mile carrier performance.
- Investing in local fulfillment centers or partnerships in high-delay regions.
- Providing longer ETA estimates or alerts for customers in these areas.
- São Paulo is the commercial hub — focus for scaling successful strategies.
- States like Paraná and Minas Gerais also represent strong seller bases — potential for growth.
- Underperforming states need to improve seller onboarding, localize marketing and evaluate infrastructure.

- Segment sellers by strength: volume vs. value. Optimize platform layout or promotions to push low-traffic, high-AOV sellers.
- Prioritize reliability improvements in high-delay, low-sales regions (e.g., Alagoas, Maranhão).

Product Analysis







Insight:

- Furniture (31.1%) and Fashion (25.88%) dominate in terms of total sales volume, followed by Personal Care & Beauty (18.5%)
- Agricultural, House Safety, CDs, Photography, and Flowers make up less than 0.5% each, with Insurance & Services contributing 0%.
- The highest AOV comes from Agricultural (~R$400), Musical Instruments, and Telephony.
- Furniture (a top-seller) has a below-average AOV, while Fashion and Personal Care fall near or below the average line (R$144.48).
- Sellers are concentrated across a few strong categories, with Furniture and Fashion making up over 40% combined.

Business Implications:

- Focus marketing, stocking, and promotions on high-volume categories, and evaluate whether low performers should be repositioned, bundled, or phased out.
- High sales in low-AOV categories (e.g., Fashion) indicate success through volume, while underperforming high-AOV categories (e.g., Agricultural, Construction) have potential if properly marketed or repositioned.

- Identify high-margin, underutilized categories (e.g., Musical Instruments or Construction) and test focused seller acquisition or promotion campaigns in those verticals.

6. Summary & Conclusion

Summary

This project analyzed Target Brazil's e-commerce operations using structured data from Kaggle, focusing on customer behavior, seller performance, and product-level sales to derive actionable business insights.

The key steps included:

- **Data Preprocessing with Alteryx:**
  - Merged geolocation data with customer and seller records to fill missing zip codes
  - Removed redundant city/state fields and ensured unique IDs
  - Standardized product category names and cast numerical fields
  - Cleaned payment, order, and item tables by converting date formats and removing undefined types
  - Loaded all cleaned datasets into SQL Server

- **SSAS Tabular Model:**
  - Designed a hybrid star-galaxy schema with 3 fact tables and 5 dimension tables
  - Created relationships for time intelligence, product/seller tracking, and geographic mapping
  - Built DAX measures (e.g., Total Sales, YoY Change) and hierarchies in DimDate
  - Enabled flexible OLAP-style slicing/dicing by category, state, payment, and time

- **Data Analysis & Queries:**
  - Converted the processed tables from MS SQL tables back to CSV files using Alteryx, then imported them to a PySpark session for query analysis.

- Convert the PySpark DataFrame into temporary SQL views to directly run SQL scripts to query the needed results.

- **Power BI Visualizations:**
  - Built dashboards showing sales trends, regional seller performance, product category analysis, and customer behavior
  - Used slicers, KPIs, and trend lines to enhance interactivity and insight delivery
  - Visualized DAX-powered metrics like cancellation rate and sales YoY change

Conclusion

The project successfully uncovered key operational insights across Target Brazil's platform:

- Customer purchases are highly seasonal and daytime-driven
- Sales are concentrated among a few regions and categories
- Seller performance varies widely by geography
- Credit cards dominate transactions, with other methods underutilized
- High-AOV, low-volume categories represent untapped revenue potential

Future improvements could include:

- Integrating customer reviews into the SSAS model for sentiment-based insights
- Expanding analysis to real-time streaming data (e.g., with Azure Data Factory)
- Automating ETL workflows using SSIS pipelines
- Deploying predictive models for demand forecasting and churn prediction
- Incorporating external data (e.g., holidays, promotions) to better explain seasonal trends