TABLE OF CONTENTS

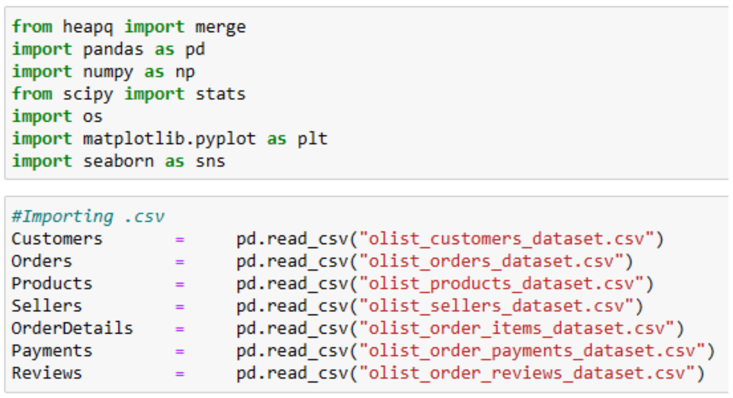
TABLE OF FIGURES

INTRODUCTION

The dataset chosen for data management project is titled Brazillian E-commerce dataset by Olist taken from Kaggle that contains information on 100k orders placed at several Brazilian marketplaces between 2016 and 2018. Its capabilities allow to evaluate an order from numerous perspectives, including order status, pricing, payment, and freight performance, as well as customer location, product qualities, and customer feedback. In total there are 9 .csv file ("customer", "geolocation", "order items", "order payments", "order reviews", "orders", "products", "sellers", "product category"). By conducting data analysis on this data, Olist will be able to support decision-making, assist in the discovery of relevant information, and assist organisations in improving their performance. Therefore, the primary objective of this report is to analyse the data, provide business insights for Olist, and describe how effective actions may be made to further the company's growth.

PRE-PROCESSING

Prior to real use, data must first be pre-processed. The idea of turning raw data into a clean data set is known as data preparation. The dataset is pre-processed to look for missing values, noisy data, and other irregularities before the algorithm is applied to it. The Olist dataset has been pre-processed using Python Jupyter.

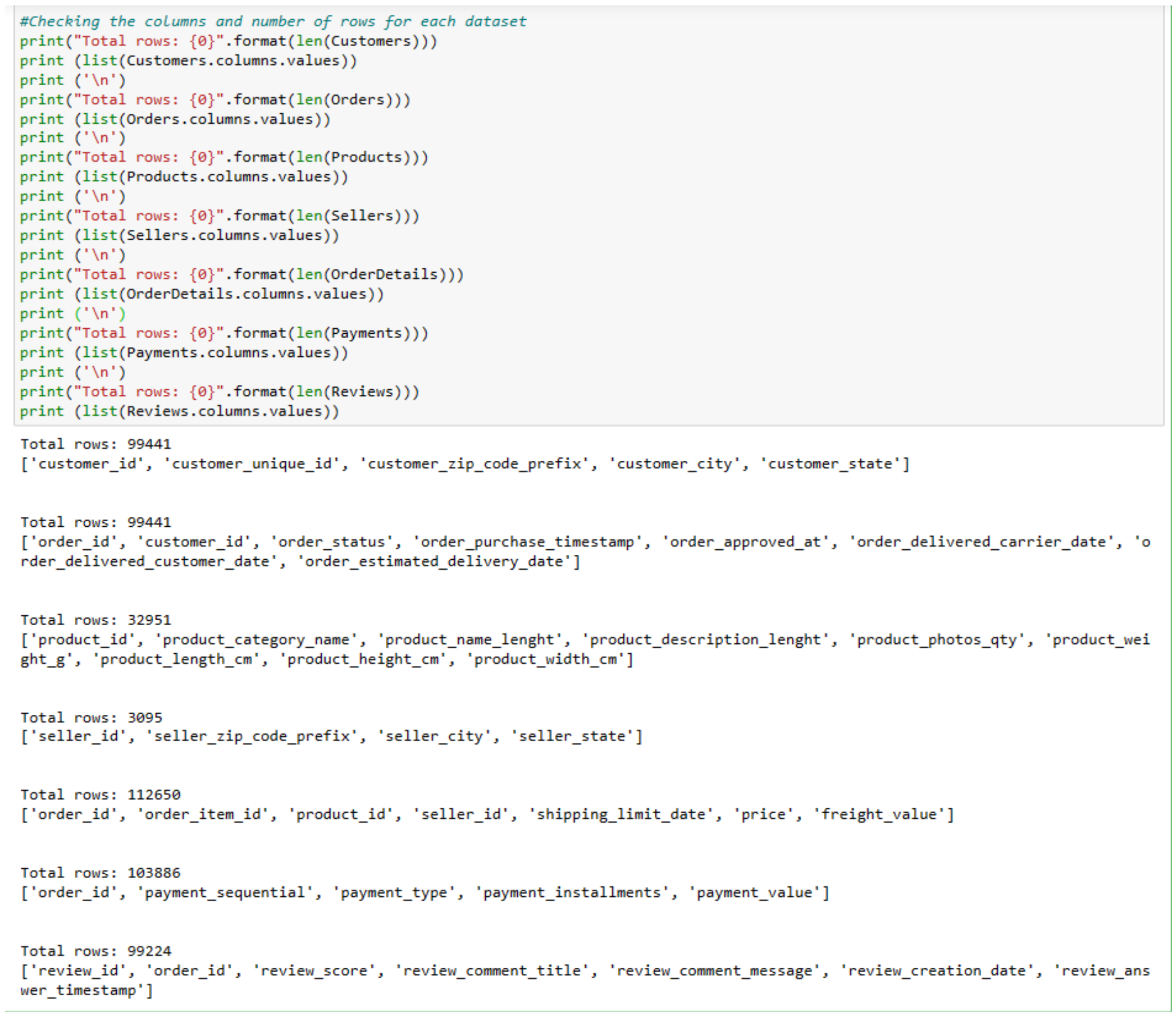


```
from heapq import merge
import pandas as pd
import numpy as np
from scipy import stats
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Importing .csv
Customers       =     pd.read_csv("olist_customers_dataset.csv")
Orders          =     pd.read_csv("olist_orders_dataset.csv")
Products        =     pd.read_csv("olist_products_dataset.csv")
Sellers         =     pd.read_csv("olist_sellers_dataset.csv")
OrderDetails    =     pd.read_csv("olist_order_items_dataset.csv")
Payments        =     pd.read_csv("olist_order_payments_dataset.csv")
Reviews         =     pd.read_csv("olist_order_reviews_dataset.csv")
```

Figure 1: Importing Libraries and Dataset

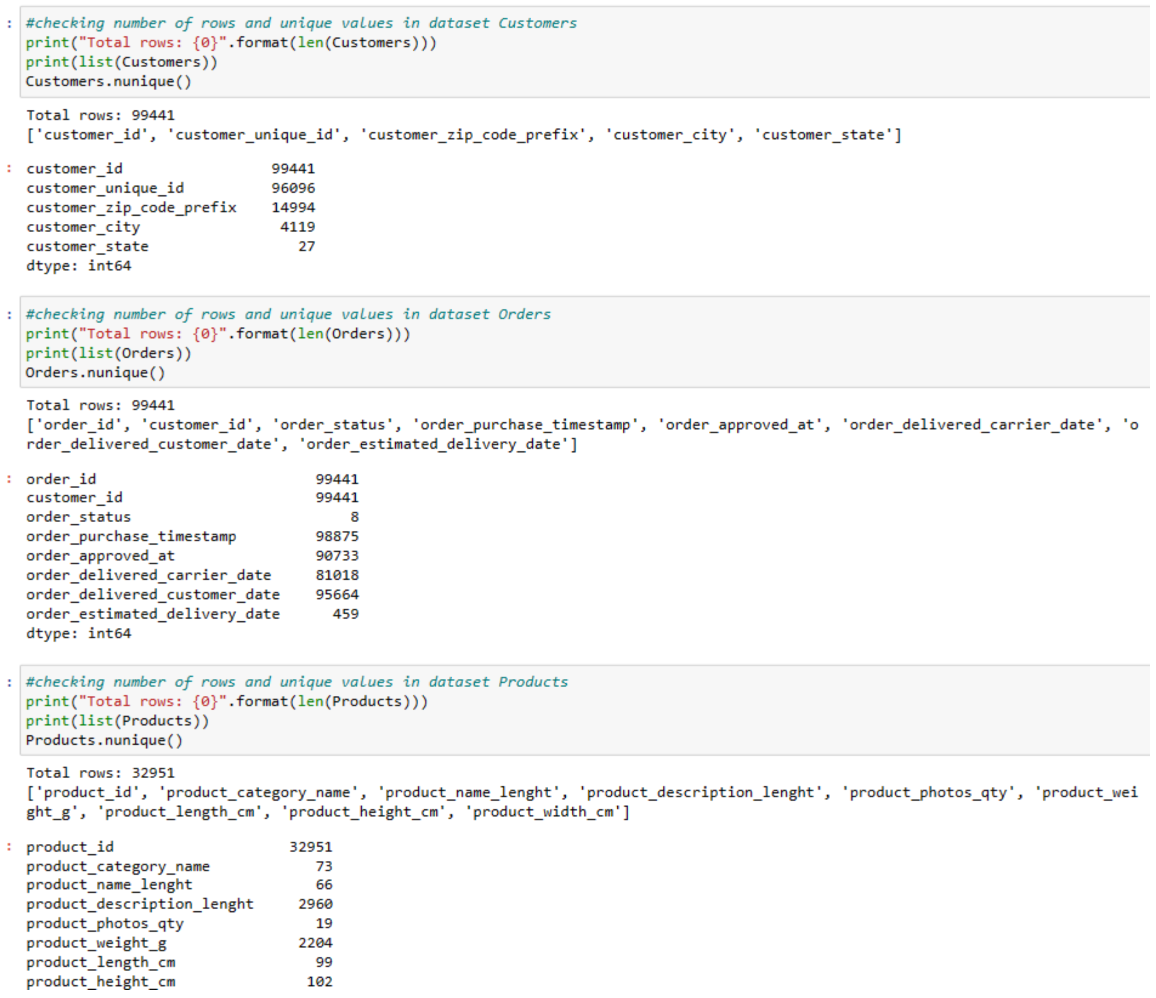
To work with data frames and arrays in the datasets, important libraries such as pandas and numpy are imported. The following datasets are uploaded to Python: "Customer" for the customers dataset, "Orders" for the orders dataset, "Products" for the products dataset, "Sellers" for the sellers dataset, "OrderDetails" for the order items dataset, "Payment" for the payment dataset, and finally "Reviews" for the review dataset as shown in Figure 1.



```python
#Checking the columns and number of rows for each dataset
print("Total rows: {0}".format(len(Customers)))
print (list(Customers.columns.values))
print ('\n')
print("Total rows: {0}".format(len(Orders)))
print (list(Orders.columns.values))
print ('\n')
print("Total rows: {0}".format(len(Products)))
print (list(Products.columns.values))
print ('\n')
print("Total rows: {0}".format(len(Sellers)))
print (list(Sellers.columns.values))
print ('\n')
print("Total rows: {0}".format(len(OrderDetails)))
print (list(OrderDetails.columns.values))
print ('\n')
print("Total rows: {0}".format(len(Payments)))
print (list(Payments.columns.values))
print ('\n')
print("Total rows: {0}".format(len(Reviews)))
print (list(Reviews.columns.values))
```

```
Total rows: 99441
['customer_id', 'customer_unique_id', 'customer_zip_code_prefix', 'customer_city', 'customer_state']


Total rows: 99441
['order_id', 'customer_id', 'order_status', 'order_purchase_timestamp', 'order_approved_at', 'order_delivered_carrier_date', 'o
rder_delivered_customer_date', 'order_estimated_delivery_date']


Total rows: 32951
['product_id', 'product_category_name', 'product_name_lenght', 'product_description_lenght', 'product_photos_qty', 'product_wei
ght_g', 'product_length_cm', 'product_height_cm', 'product_width_cm']


Total rows: 3095
['seller_id', 'seller_zip_code_prefix', 'seller_city', 'seller_state']


Total rows: 112650
['order_id', 'order_item_id', 'product_id', 'seller_id', 'shipping_limit_date', 'price', 'freight_value']


Total rows: 103886
['order_id', 'payment_sequential', 'payment_type', 'payment_installments', 'payment_value']


Total rows: 99224
['review_id', 'order_id', 'review_score', 'review_comment_title', 'review_comment_message', 'review_creation_date', 'review_ans
wer_timestamp']
```

Figure 2: Checking Columns and Rows For Each Dataset

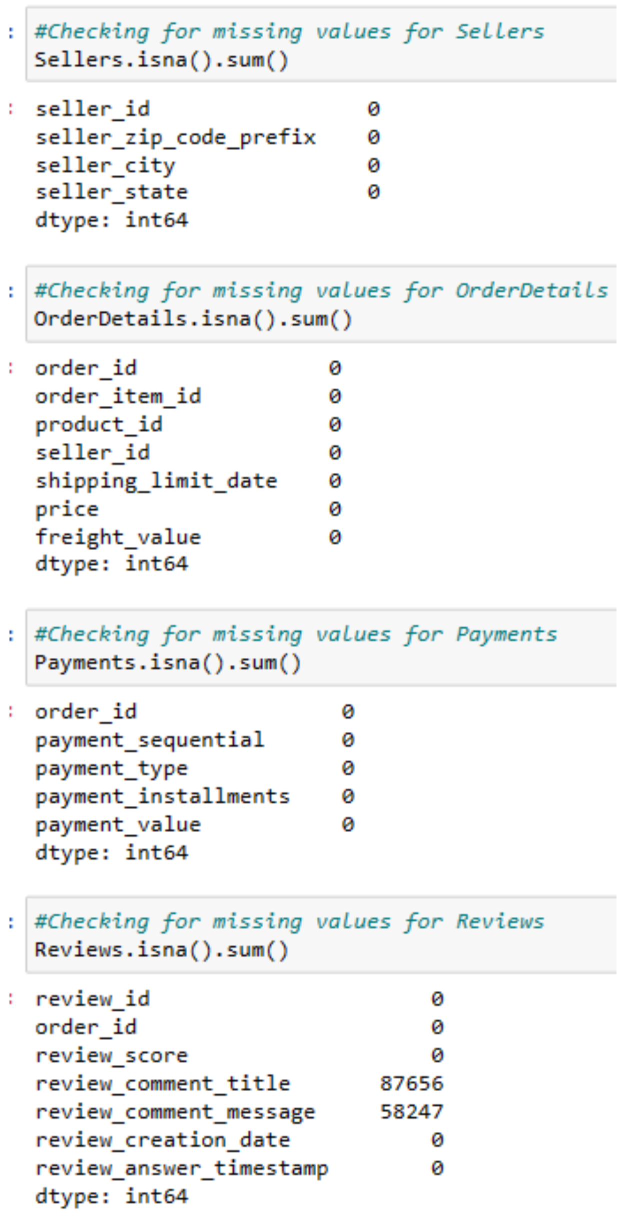

```
: #checking number of rows and unique values in dataset Customers
print("Total rows: {0}".format(len(Customers)))
print(list(Customers))
Customers.nunique()

Total rows: 99441
['customer_id', 'customer_unique_id', 'customer_zip_code_prefix', 'customer_city', 'customer_state']

: customer_id                 99441
customer_unique_id          96096
customer_zip_code_prefix    14994
customer_city                4119
customer_state                 27
dtype: int64
```

```
: #checking number of rows and unique values in dataset Orders
print("Total rows: {0}".format(len(Orders)))
print(list(Orders))
Orders.nunique()

Total rows: 99441
['order_id', 'customer_id', 'order_status', 'order_purchase_timestamp', 'order_approved_at', 'order_delivered_carrier_date', 'o
rder_delivered_customer_date', 'order_estimated_delivery_date']

: order_id                         99441
customer_id                      99441
order_status                         8
order_purchase_timestamp         98875
order_approved_at                90733
order_delivered_carrier_date     81018
order_delivered_customer_date    95664
order_estimated_delivery_date      459
dtype: int64
```

```
: #checking number of rows and unique values in dataset Products
print("Total rows: {0}".format(len(Products)))
print(list(Products))
Products.nunique()

Total rows: 32951
['product_id', 'product_category_name', 'product_name_lenght', 'product_description_lenght', 'product_photos_qty', 'product_wei
ght_g', 'product_length_cm', 'product_height_cm', 'product_width_cm']

: product_id                    32951
product_category_name            73
product_name_lenght              66
product_description_lenght     2960
product_photos_qty               19
product_weight_g               2204
product_length_cm                99
product_height_cm               102
```

Figure 3: Checking Number of Rows and Unique Values For Each Dataset

Each dataset's sum of rows and attributes are analysed as shown in Figure 2. This provides a basic overview of what each dataset contains as well as the relationships between datasets using IDs, followed by obtaining a list of unique values for each attribute inside each dataset as shown in Figure 3, which will aid in validating the number of data when importing into SQL.



```
#Checking for missing values for Sellers
Sellers.isna().sum()
```

```
seller_id                 0
seller_zip_code_prefix    0
seller_city               0
seller_state              0
dtype: int64
```

```
#Checking for missing values for OrderDetails
OrderDetails.isna().sum()
```

```
order_id               0
order_item_id          0
product_id             0
seller_id              0
shipping_limit_date    0
price                  0
freight_value          0
dtype: int64
```

```
#Checking for missing values for Payments
Payments.isna().sum()
```

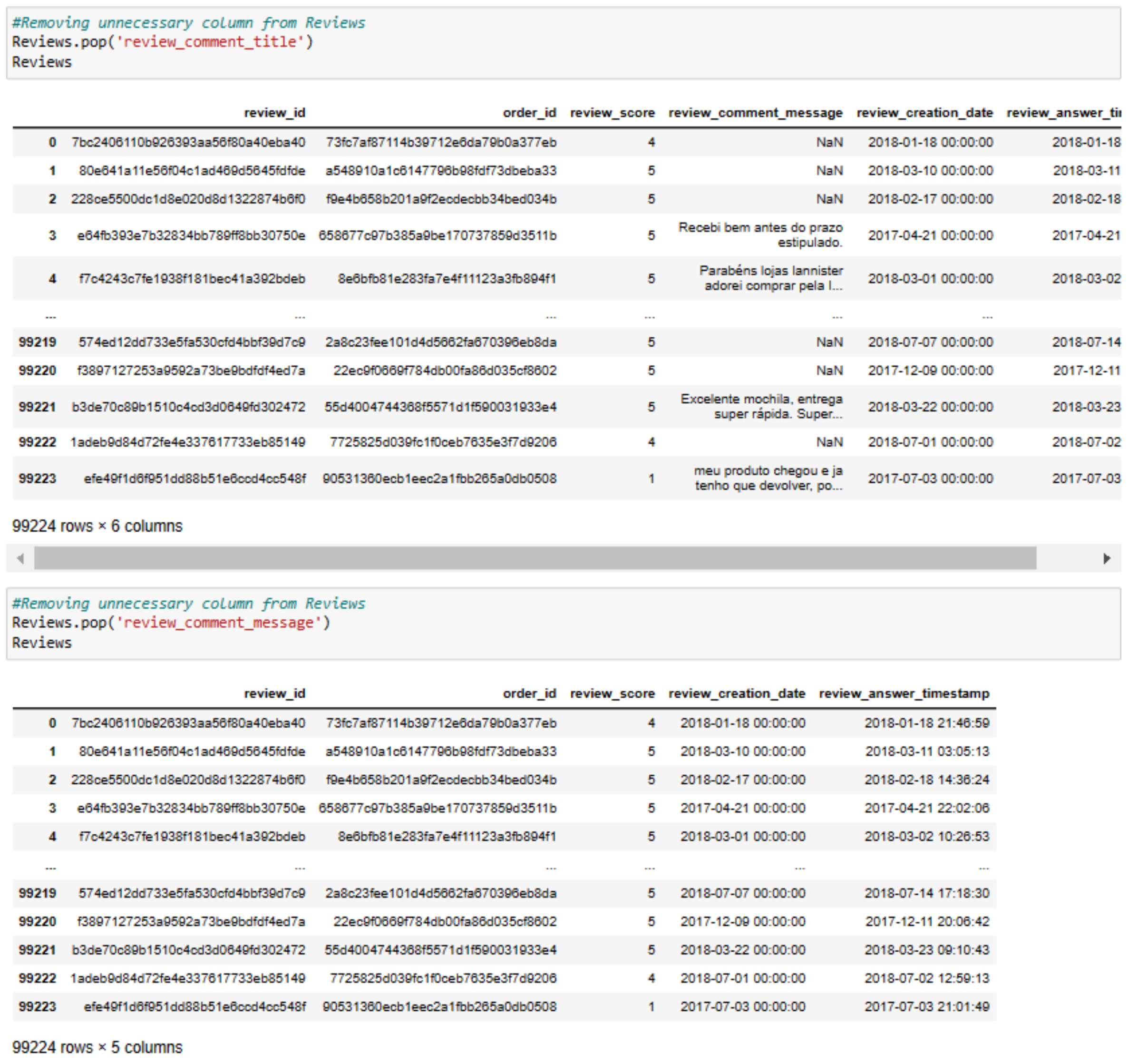
```
order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64
```

```
#Checking for missing values for Reviews
Reviews.isna().sum()
```

```
review_id                      0
order_id                       0
review_score                   0
review_comment_title       87656
review_comment_message     58247
review_creation_date           0
review_answer_timestamp        0
dtype: int64
```

Figure 4: Checking For Null Values

Checking for null values with the code ".isnull().sum," as shown in Figure 4, which indicate a lack of value in each dataset's attributes. Checking for nulls can assist decide if an attribute is useable or not based on the number of missing data values.



```
#Removing unnecessary column from Reviews
Reviews.pop('review_comment_title')
Reviews
```

	review_id	order_id	review_score	review_comment_message	review_creation_date	review_answer_ti
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	NaN	2018-01-18 00:00:00	2018-01-18
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	NaN	2018-03-10 00:00:00	2018-03-11
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	NaN	2018-02-17 00:00:00	2018-02-18
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	Recebi bem antes do prazo estipulado.	2017-04-21 00:00:00	2017-04-21
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	Parabéns lojas lannister adorei comprar pela l...	2018-03-01 00:00:00	2018-03-02
...	...	...	...	...	...	
99219	574ed12dd733e5fa530cfd4bbf39d7c9	2a8c23fee101d4d5662fa670396eb8da	5	NaN	2018-07-07 00:00:00	2018-07-14
99220	f3897127253a9592a73be9bdfdf4ed7a	22ec9f0669f784db00fa86d035cf8602	5	NaN	2017-12-09 00:00:00	2017-12-11
99221	b3de70c89b1510c4cd3d0649fd302472	55d4004744368f5571d1f590031933e4	5	Excelente mochila, entrega super rápida. Super...	2018-03-22 00:00:00	2018-03-23
99222	1adeb9d84d72fe4e337617733eb85149	7725825d039fc1f0ceb7635e3f7d9206	4	NaN	2018-07-01 00:00:00	2018-07-02
99223	efe49f1d6f951dd88b51e6ccd4cc548f	90531360ecb1eec2a1fbb265a0db0508	1	meu produto chegou e ja tenho que devolver, po...	2017-07-03 00:00:00	2017-07-03

99224 rows × 6 columns

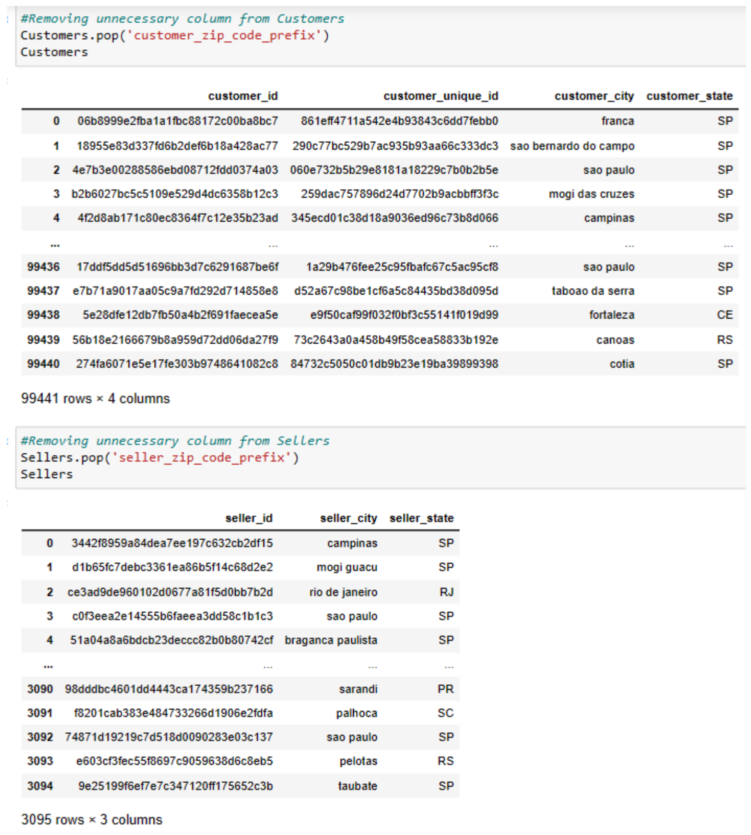
```
#Removing unnecessary column from Reviews
Reviews.pop('review_comment_message')
Reviews
```

	review_id	order_id	review_score	review_creation_date	review_answer_timestamp
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	2018-01-18 00:00:00	2018-01-18 21:46:59
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	2018-03-10 00:00:00	2018-03-11 03:05:13
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	2018-02-17 00:00:00	2018-02-18 14:36:24
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	2017-04-21 00:00:00	2017-04-21 22:02:06
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	2018-03-01 00:00:00	2018-03-02 10:26:53
...	...	...	...	...	...
99219	574ed12dd733e5fa530cfd4bbf39d7c9	2a8c23fee101d4d5662fa670396eb8da	5	2018-07-07 00:00:00	2018-07-14 17:18:30
99220	f3897127253a9592a73be9bdfdf4ed7a	22ec9f0669f784db00fa86d035cf8602	5	2017-12-09 00:00:00	2017-12-11 20:06:42
99221	b3de70c89b1510c4cd3d0649fd302472	55d4004744368f5571d1f590031933e4	5	2018-03-22 00:00:00	2018-03-23 09:10:43
99222	1adeb9d84d72fe4e337617733eb85149	7725825d039fc1f0ceb7635e3f7d9206	4	2018-07-01 00:00:00	2018-07-02 12:59:13
99223	efe49f1d6f951dd88b51e6ccd4cc548f	90531360ecb1eec2a1fbb265a0db0508	1	2017-07-03 00:00:00	2017-07-03 21:01:49

99224 rows × 5 columns

Figure 5: Removing Comment Title and Comment Message Attribute From Reviews Dataset

As shown in Figure 5, that the review comment title and message have missing values of 87656 and 58247, respectively. It is best to eliminate it with so many missing values, especially when the content of those two attributes represents public opinion.



```python
#Removing unnecessary column from Customers
Customers.pop('customer_zip_code_prefix')
Customers
```

	customer_id	customer_unique_id	customer_city	customer_state
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	franca	SP
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	sao bernardo do campo	SP
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	sao paulo	SP
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	mogi das cruzes	SP
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	campinas	SP
...	...	...	...	...
99436	17ddf5dd5d51696bb3d7c6291687be6f	1a29b476fee25c95fbafc67c5ac95cf8	sao paulo	SP
99437	e7b71a9017aa05c9a7fd292d714858e8	d52a67c98be1cf6a5c84435bd38d095d	taboao da serra	SP
99438	5e28dfe12db7fb50a4b2f691faecea5e	e9f50caf99f032f0bf3c55141f019d99	fortaleza	CE
99439	56b18e2166679b8a959d72dd06da27f9	73c2643a0a458b49f58cea58833b192e	canoas	RS
99440	274fa6071e5e17fe303b9748641082c8	84732c5050c01db9b23e19ba39899398	cotia	SP

99441 rows × 4 columns

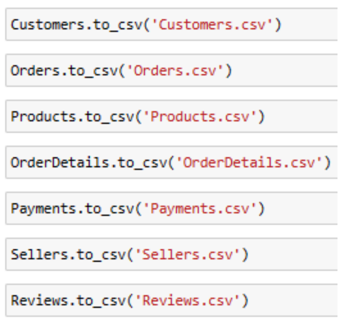
```python
#Removing unnecessary column from Sellers
Sellers.pop('seller_zip_code_prefix')
Sellers
```

	seller_id	seller_city	seller_state
0	3442f8959a84dea7ee197c632cb2df15	campinas	SP
1	d1b65fc7debc3361ea86b5f14c68d2e2	mogi guacu	SP
2	ce3ad9de960102d0677a81f5d0bb7b2d	rio de janeiro	RJ
3	c0f3eea2e14555b6faeea3dd58c1b1c3	sao paulo	SP
4	51a04a8a6bdcb23deccc82b0b80742cf	braganca paulista	SP
...	...	...	...
3090	98dddbc4601dd4443ca174359b237166	sarandi	PR
3091	f8201cab383e484733266d1906e2fdfa	palhoca	SC
3092	74871d19219c7d518d0090283e03c137	sao paulo	SP
3093	e603cf3fec55f8697c9059638d6c8eb5	pelotas	RS
3094	9e25199f6ef7e7c347120ff175652c3b	taubate	SP

3095 rows × 3 columns

Figure 6: Removing Zip Code Attribute From Customer's And Seller's Dataset

Following that, zip code was removed from both the Customers and Sellers datasets because the attribute state or city can be used to categorise location data if necessary, as shown in Figure 6.



```python
Customers.to_csv('Customers.csv')
```

```python
Orders.to_csv('Orders.csv')
```

```python
Products.to_csv('Products.csv')
```

```python
OrderDetails.to_csv('OrderDetails.csv')
```

```python
Payments.to_csv('Payments.csv')
```

```python
Sellers.to_csv('Sellers.csv')
```

```python
Reviews.to_csv('Reviews.csv')
```

Figure 7: Exporting Altered Dataset To .csv File

Finally, the altered dataset was exported into a.csv file as shown in Figure 7, which will be used to import data into MySql later on to run queries.

SCENARIO

The e-commerce company, Olist, is a Brazilian based online marketplace that connects small and medium-sized businesses to customers all over the country. The company has been in business for several years and has seen steady growth in sales and customer base. Based on the Marketing Magazine, Global e-commerce is estimated to make up 19% of global retail sales in 2022, growing to 25% by 2027. With the increasing competition in the e-commerce market, the company wants to take a closer look at its operations and identify areas where they can improve their service and product offerings to hold its position in the market.

As with other businesses, Olist has departments like those for human resources, information technology, accounting and finance, marketing, research and development (R&D), production, product management, purchasing, logistics and customer service. These divisions will be key in using the analysis from business insights and acting appropriately to improve corporate performance.

Olist wants to understand the behaviour of its customers and identify trends and patterns that can help them to better target their marketing efforts and increase customer loyalty. They also want to optimize their logistics operations to improve delivery times and reduce costs. Additionally, they are interested in understanding the performance of their product offerings and identifying which products are most popular and profitable. It also wants to make sure the sellers who are doing great job receive proper recognition too as they are the assets of the company.

To achieve these goals, Olist has decided to use the Brazilian Ecommerce Public Dataset they have collected over the past years. The dataset contains information on orders, customers, sellers and products from the company and covers the period from 2016 to 2018. The company plans to use the data to perform advanced analysis and gain insights that can help them to improve their operations and increase their competitiveness in the market.

BUSINESS INSIGHTS

An insight is a piece of information that is useful for business and helps to create or improve anything. When a business has access to important data that enables it to produce intelligence and comprehend what is occurring, why it is happening, and how it may be resolved, reversed, or improved, it is said to have insights. A business insight combines data and analysis to make sense of and deepen understanding of a situation, giving your company a competitive edge. This gives you a better understanding of the key mechanisms relating to your specific business and goes beyond a basic understanding of a problem. Olist may therefore possess the following business insights:

- Identifying the increment for customers over the years and total number of orders made by customers each year.
- identifying which regions are the most profitable.
- Identifying how fast a delivery is made after order is placed and average delivery time based on state
- Identifying customer behaviour over time and identifying patterns
- Identifying performance of sellers and the best-performing ones
- Identifying most expensive and cheapest products in each category
- Identifying product category performance and the most popular and profitable product categories
- Identifying number of different order status categories and their percentage
- Identifying sales patterns by months and by time of the day(Dawn/Morning/Afternoon/Night)
- Identifying products with review score of 5 and total number of purchase
- Identifying if there is any relationship between Freight value and and the total sales based on states.
- Identifying if there are any relationships between the order status and the rating for the orders
- Identifying which payment method is popular among customers
- Identifying the change in payment method over the years

By using the data and insights gained from the dataset, Olist can develop targeted strategies to improve their operations and increase their competitiveness in the market. This would help the company to improve customer satisfaction, increase sales and revenue, and ultimately improve the overall performance of the business.

TABLE RELATIONSHIPS

The cardinalities of the tables we have created in the SQL code:

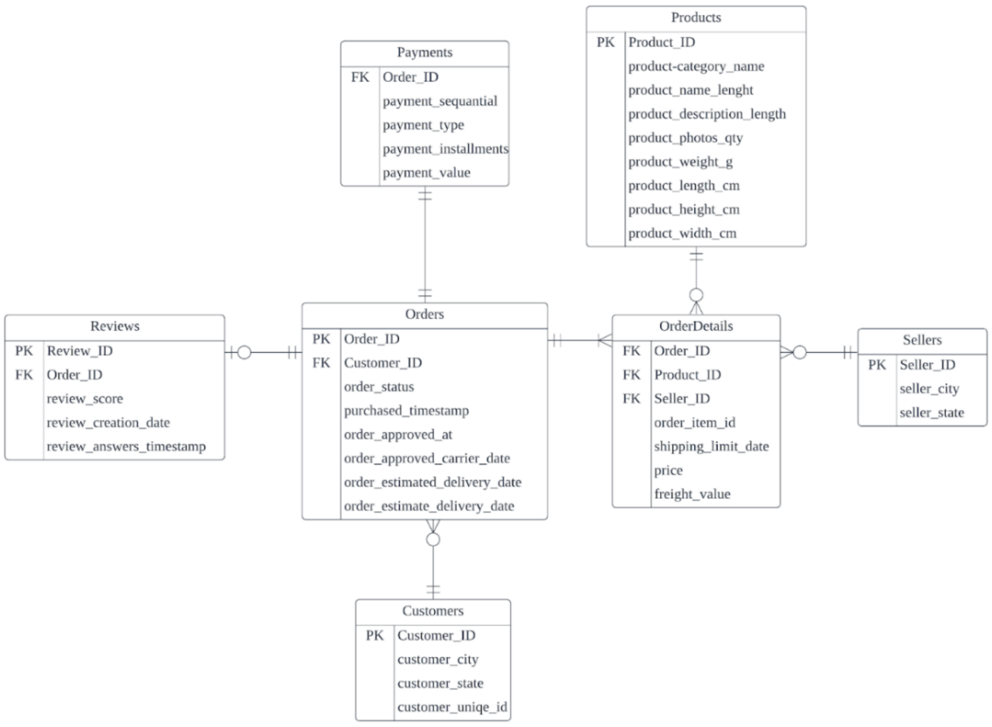
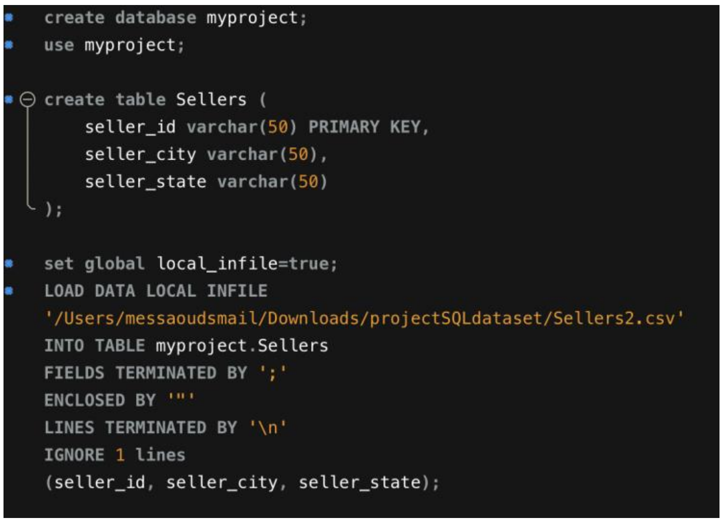


Diagram 1: ERD For Olist Brazil Dataset

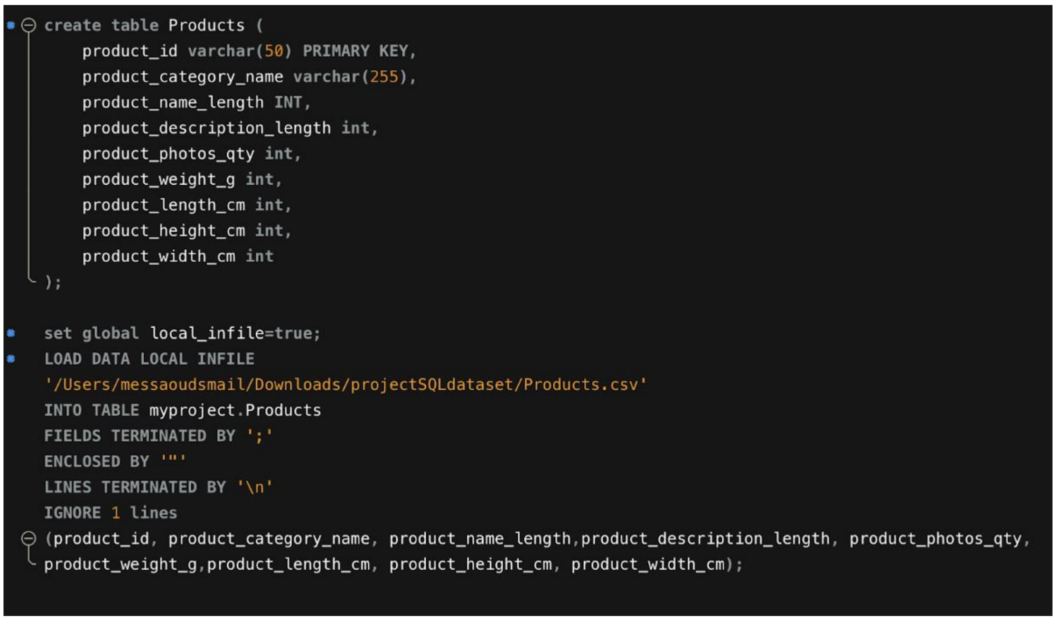
CREATING DATABASE/TABLE AND IMPORT DATA (MYSQL)



```
create database myproject;
use myproject;

create table Sellers (
    seller_id varchar(50) PRIMARY KEY,
    seller_city varchar(50),
    seller_state varchar(50)
);

set global local_infile=true;
LOAD DATA LOCAL INFILE
'/Users/messaoudsmail/Downloads/projectSQLdataset/Sellers2.csv'
INTO TABLE myproject.Sellers
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 lines
(seller_id, seller_city, seller_state);
```

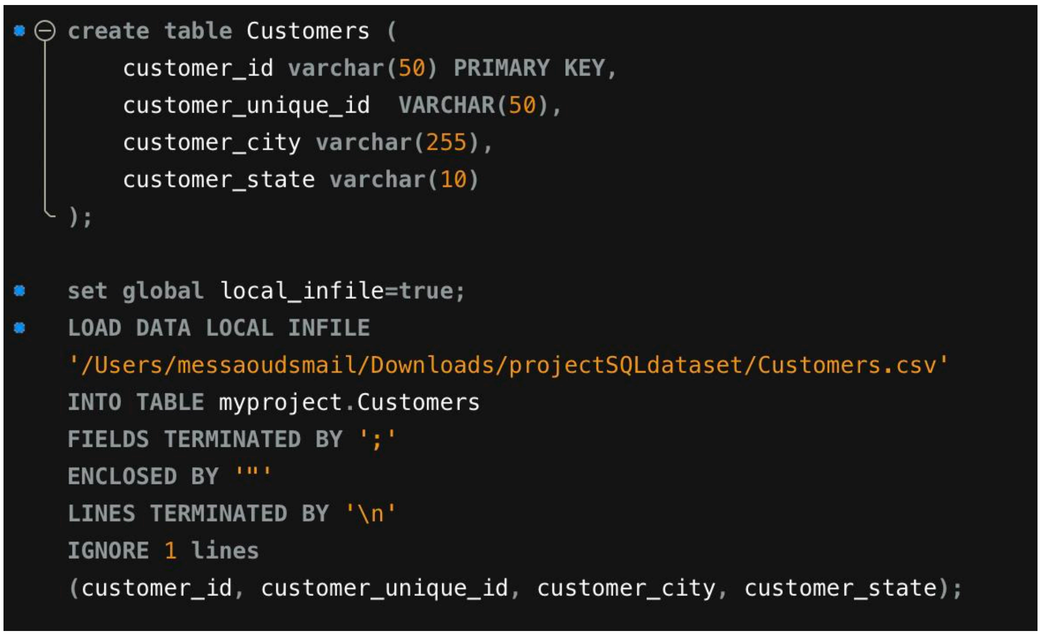
Figure 8: Creating Database, Sellers Table and Importing Data

Creating database called myproject followed by sellers table with the following attributes, types, and relation as shown in Figure 8 and importing the data for the table from directly using the file Sellers.csv which was exported after pre-processing in python. The process of creating table and importing data is similar for the rest of 6 tables (“Products”, “Customer”, “Orders”, “OrderDetails”, “Payments”, “Reviews”) with different Primary/Foreign key and attributes as shown below Figures 9, 10, 11, 12, 13 and 14.



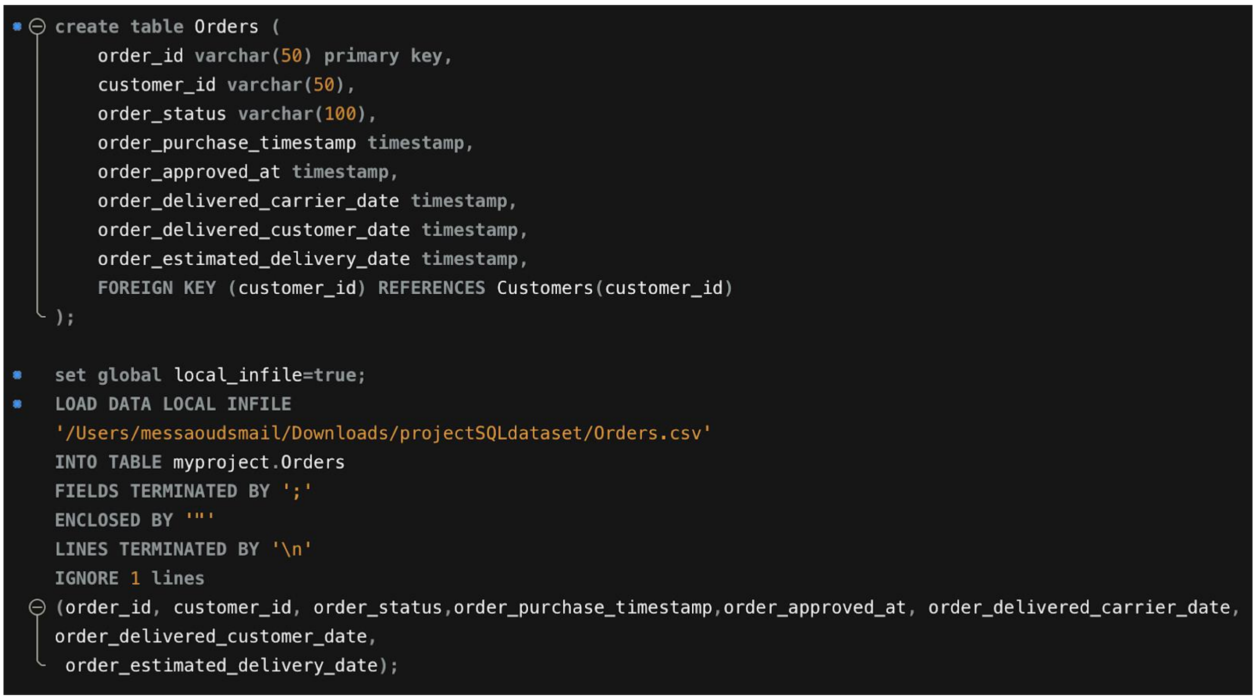
```sql
create table Products (
    product_id varchar(50) PRIMARY KEY,
    product_category_name varchar(255),
    product_name_length INT,
    product_description_length int,
    product_photos_qty int,
    product_weight_g int,
    product_length_cm int,
    product_height_cm int,
    product_width_cm int
);

set global local_infile=true;
LOAD DATA LOCAL INFILE
'/Users/messaoudsmail/Downloads/projectSQLdataset/Products.csv'
INTO TABLE myproject.Products
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 lines
(product_id, product_category_name, product_name_length,product_description_length, product_photos_qty,
product_weight_g,product_length_cm, product_height_cm, product_width_cm);
```

Figure 9: Creating Products Table and Importing Data



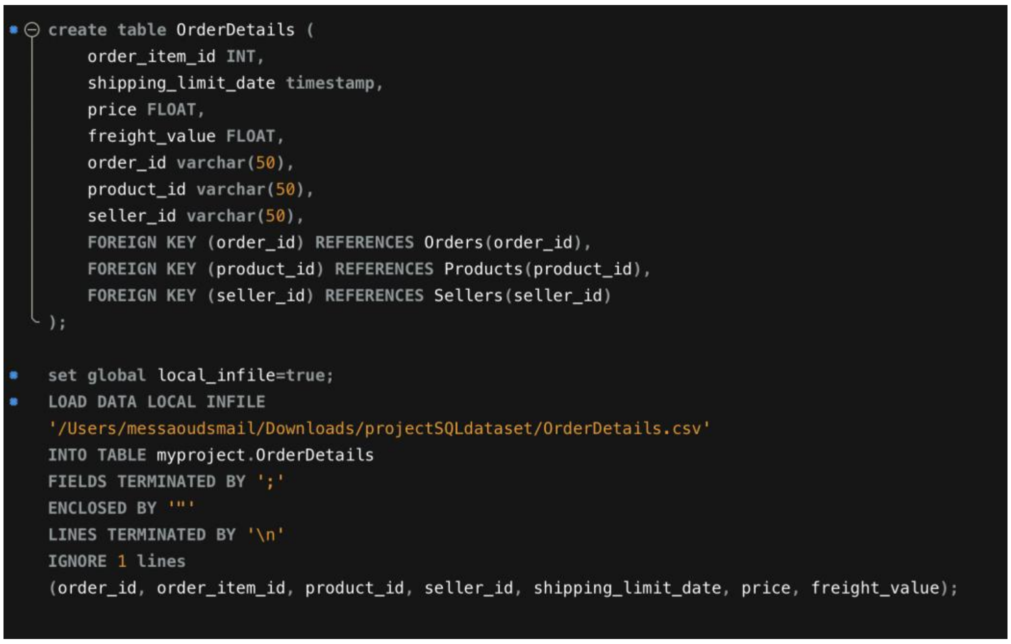
```sql
create table Customers (
    customer_id varchar(50) PRIMARY KEY,
    customer_unique_id  VARCHAR(50),
    customer_city varchar(255),
    customer_state varchar(10)
);

set global local_infile=true;
LOAD DATA LOCAL INFILE
'/Users/messaoudsmail/Downloads/projectSQLdataset/Customers.csv'
INTO TABLE myproject.Customers
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 lines
(customer_id, customer_unique_id, customer_city, customer_state);
```

Figure 10: Creating Customers Table and Importing Data



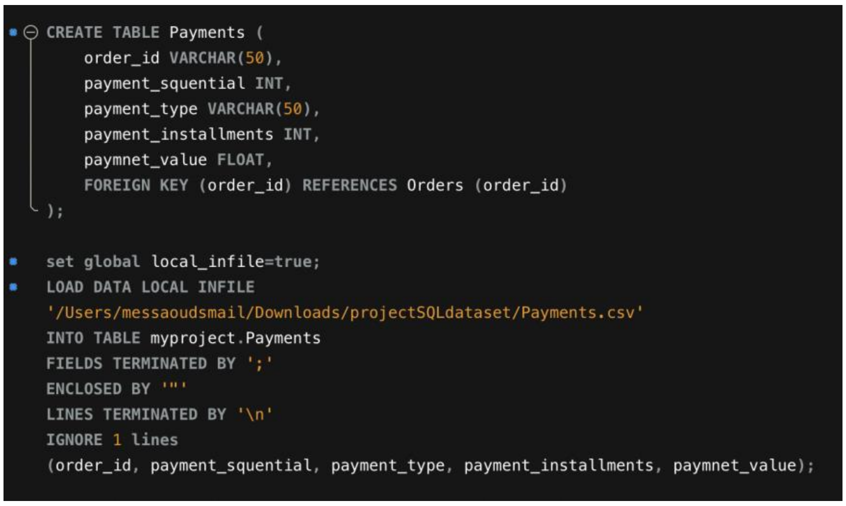
```sql
create table Orders (
    order_id varchar(50) primary key,
    customer_id varchar(50),
    order_status varchar(100),
    order_purchase_timestamp timestamp,
    order_approved_at timestamp,
    order_delivered_carrier_date timestamp,
    order_delivered_customer_date timestamp,
    order_estimated_delivery_date timestamp,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);

set global local_infile=true;
LOAD DATA LOCAL INFILE
'/Users/messaoudsmail/Downloads/projectSQLdataset/Orders.csv'
INTO TABLE myproject.Orders
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 lines
(order_id, customer_id, order_status,order_purchase_timestamp,order_approved_at, order_delivered_carrier_date,
order_delivered_customer_date,
 order_estimated_delivery_date);
```

Figure 11: Creating Orders Table and Importing Data



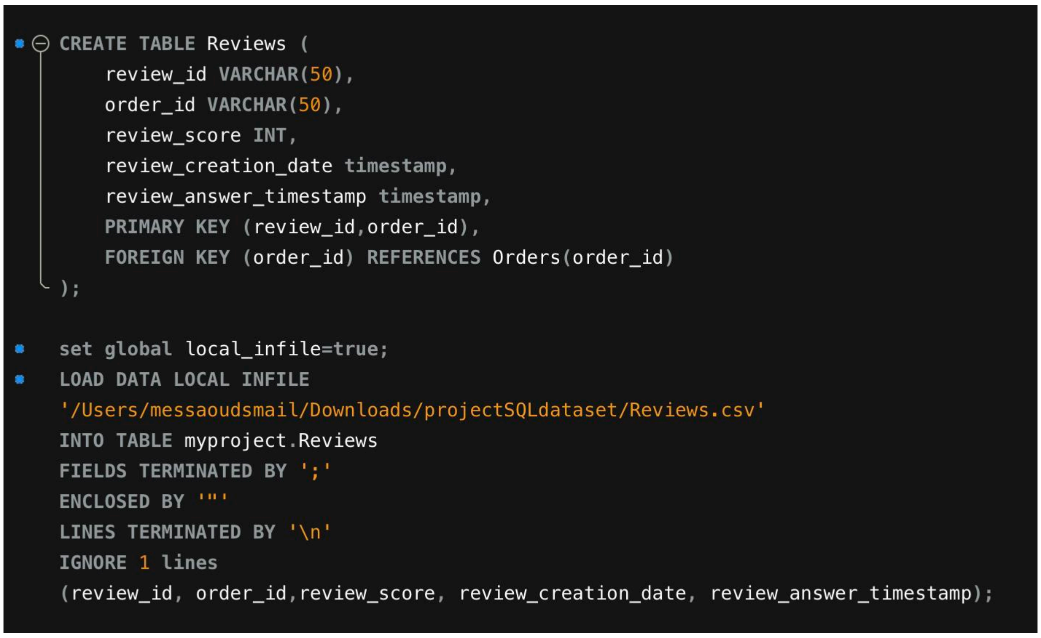
```sql
create table OrderDetails (
    order_item_id INT,
    shipping_limit_date timestamp,
    price FLOAT,
    freight_value FLOAT,
    order_id varchar(50),
    product_id varchar(50),
    seller_id varchar(50),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id),
    FOREIGN KEY (seller_id) REFERENCES Sellers(seller_id)
);

set global local_infile=true;
LOAD DATA LOCAL INFILE
'/Users/messaoudsmail/Downloads/projectSQLdataset/OrderDetails.csv'
INTO TABLE myproject.OrderDetails
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 lines
(order_id, order_item_id, product_id, seller_id, shipping_limit_date, price, freight_value);
```

Figure 12: Creating Orderdetails Table and Importing Data



```sql
CREATE TABLE Payments (
    order_id VARCHAR(50),
    payment_squential INT,
    payment_type VARCHAR(50),
    payment_installments INT,
    paymnet_value FLOAT,
    FOREIGN KEY (order_id) REFERENCES Orders (order_id)
);

set global local_infile=true;
LOAD DATA LOCAL INFILE
'/Users/messaoudsmail/Downloads/projectSQLdataset/Payments.csv'
INTO TABLE myproject.Payments
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 lines
(order_id, payment_squential, payment_type, payment_installments, paymnet_value);
```

Figure 13: Creating Payments Table and Importing Data



```sql
CREATE TABLE Reviews (
    review_id VARCHAR(50),
    order_id VARCHAR(50),
    review_score INT,
    review_creation_date timestamp,
    review_answer_timestamp timestamp,
    PRIMARY KEY (review_id,order_id),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);

set global local_infile=true;
LOAD DATA LOCAL INFILE
'/Users/messaoudsmail/Downloads/projectSQLdataset/Reviews.csv'
INTO TABLE myproject.Reviews
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 lines
(review_id, order_id,review_score, review_creation_date, review_answer_timestamp);
```

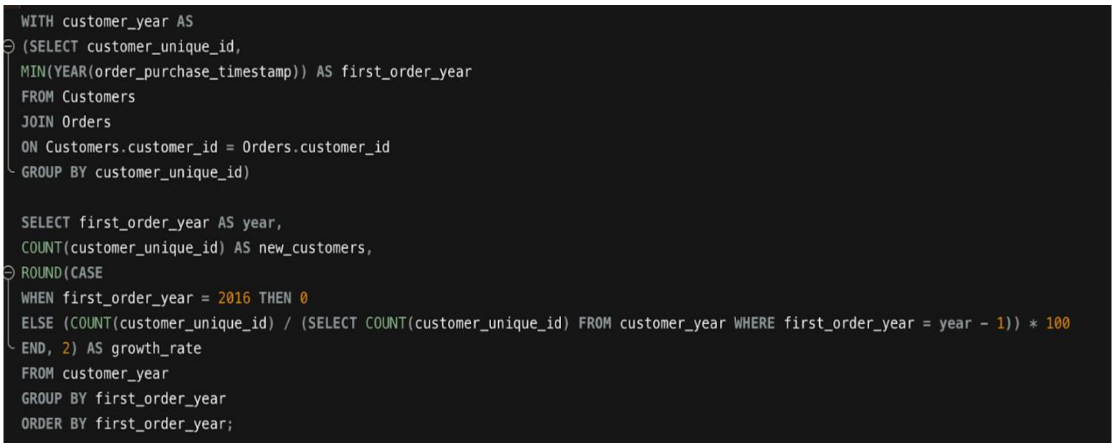
Figure 14: Creating Reviews Table and Importing Data

SQL QUERIES

According to the business insights recommended for Olist to analyse the results appropriately, certain SQL queries have been executed.

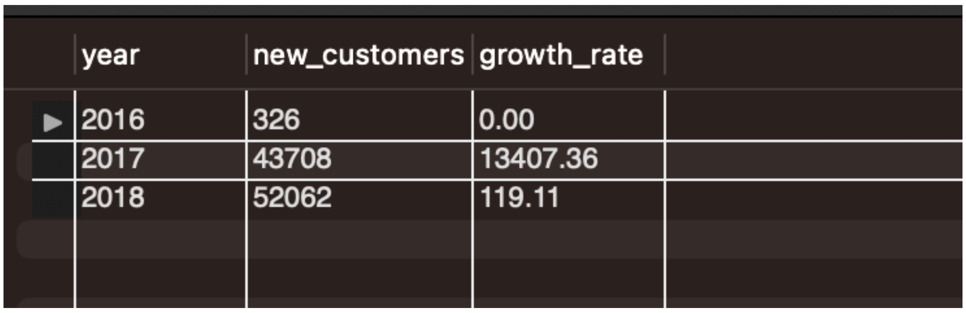
1. Assuming the company started in 2016 we want to analyse the as the following:

   a) Getting the number of new customers each year which will help in obtaining information about the growth percentage of the company each year based on the previous year.



```sql
WITH customer_year AS
(SELECT customer_unique_id,
MIN(YEAR(order_purchase_timestamp)) AS first_order_year
FROM Customers
JOIN Orders
ON Customers.customer_id = Orders.customer_id
GROUP BY customer_unique_id)

SELECT first_order_year AS year,
COUNT(customer_unique_id) AS new_customers,
ROUND(CASE
WHEN first_order_year = 2016 THEN 0
ELSE (COUNT(customer_unique_id) / (SELECT COUNT(customer_unique_id) FROM customer_year WHERE first_order_year = year - 1)) * 100
END, 2) AS growth_rate
FROM customer_year
GROUP BY first_order_year
ORDER BY first_order_year;
```

Figure 15: Query 1a Code

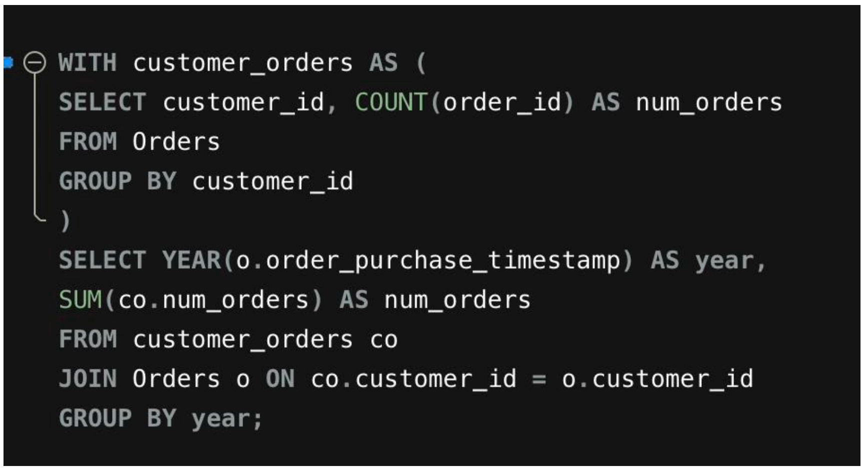


```
  year   new_customers  growth_rate
▶ 2016   326            0.00
  2017   43708          13407.36
  2018   52062          119.11
```

Figure 16: Query 1a Output

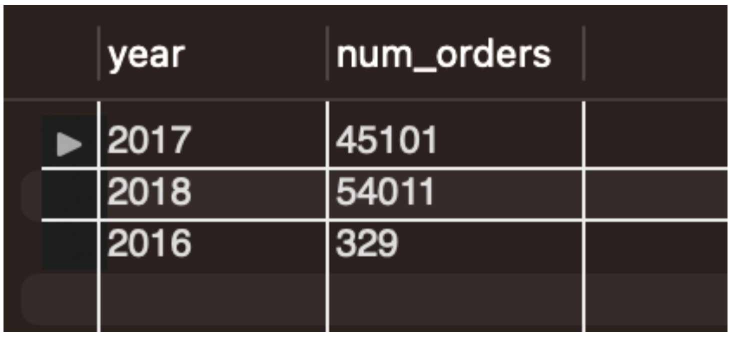
Figure 16 demonstrates that the growth rate of consumers from 2016 to 2017 was 13407.38%, whereas the reduction rate from 2017 to 2018 was much less, at 199.11%. As social media advertising enables marketers to hyper-target precise consumers and construct audience databases, it suggests that Olist's marketing team should start promoting the brand more to attract more customers.

b) Total number of orders made by customers each year



```sql
WITH customer_orders AS (
SELECT customer_id, COUNT(order_id) AS num_orders
FROM Orders
GROUP BY customer_id
)
SELECT YEAR(o.order_purchase_timestamp) AS year,
SUM(co.num_orders) AS num_orders
FROM customer_orders co
JOIN Orders o ON co.customer_id = o.customer_id
GROUP BY year;
```

Figure 17: Query 1b Code



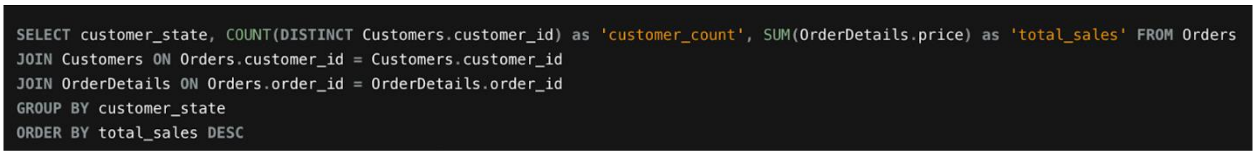
year	num_orders
2017	45101
2018	54011
2016	329

Figure 18: Query 1b Output

As shown in Figure 18, the number of orders customers from year 2016 (329) has increased in 2017 (45101) and continued to increase in 2018 (54011). This shows that customers of Olist are being active in the business as the years go and this should be maintained by the marketing and customer service team by providing good service.

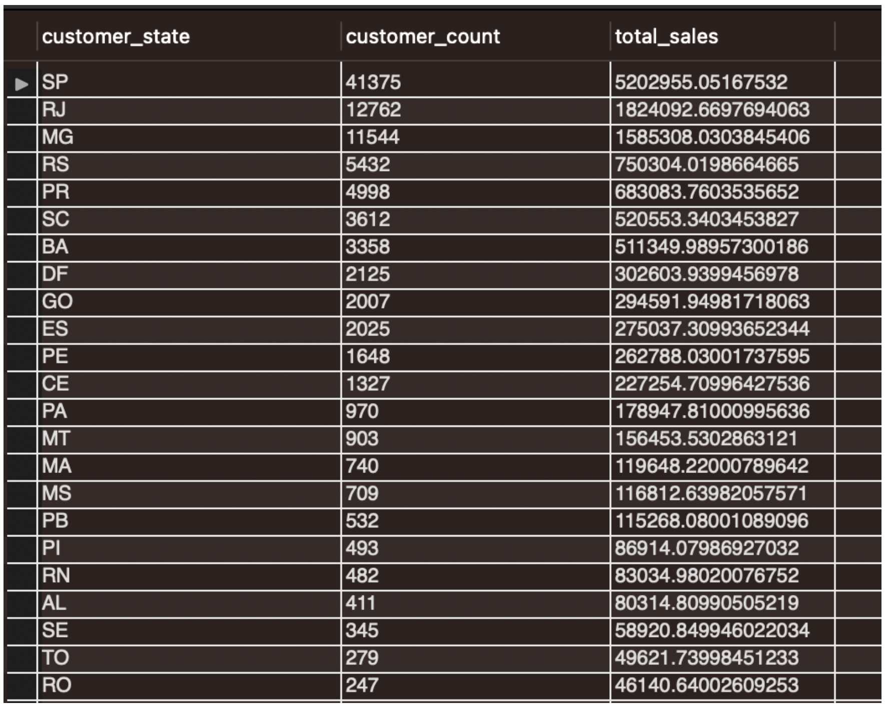
2. Understanding the customer demographics and identifying which regions are the most profitable.

To compare and contrast what distinguishes that location from other regions, whether it be due to business endeavours like advertising or any other aspect.



```sql
SELECT customer_state, COUNT(DISTINCT Customers.customer_id) as 'customer_count', SUM(OrderDetails.price) as 'total_sales' FROM Orders
JOIN Customers ON Orders.customer_id = Customers.customer_id
JOIN OrderDetails ON Orders.order_id = OrderDetails.order_id
GROUP BY customer_state
ORDER BY total_sales DESC
```

Figure 19: Query 2 Code



customer_state	customer_count	total_sales
SP	41375	5202955.05167532
RJ	12762	1824092.6697694063
MG	11544	1585308.0303845406
RS	5432	750304.0198664665
PR	4998	683083.7603535652
SC	3612	520553.3403453827
BA	3358	511349.98957300186
DF	2125	302603.9399456978
GO	2007	294591.94981718063
ES	2025	275037.30993652344
PE	1648	262788.03001737595
CE	1327	227254.70996427536
PA	970	178947.81000995636
MT	903	156453.5302863121
MA	740	119648.22000789642
MS	709	116812.63982057571
PB	532	115268.08001089096
PI	493	86914.07986927032
RN	482	83034.98020076752
AL	411	80314.80990505219
SE	345	58920.849946022034
TO	279	49621.73998451233
RO	247	46140.64002609253

Figure 20: Query 2 Output

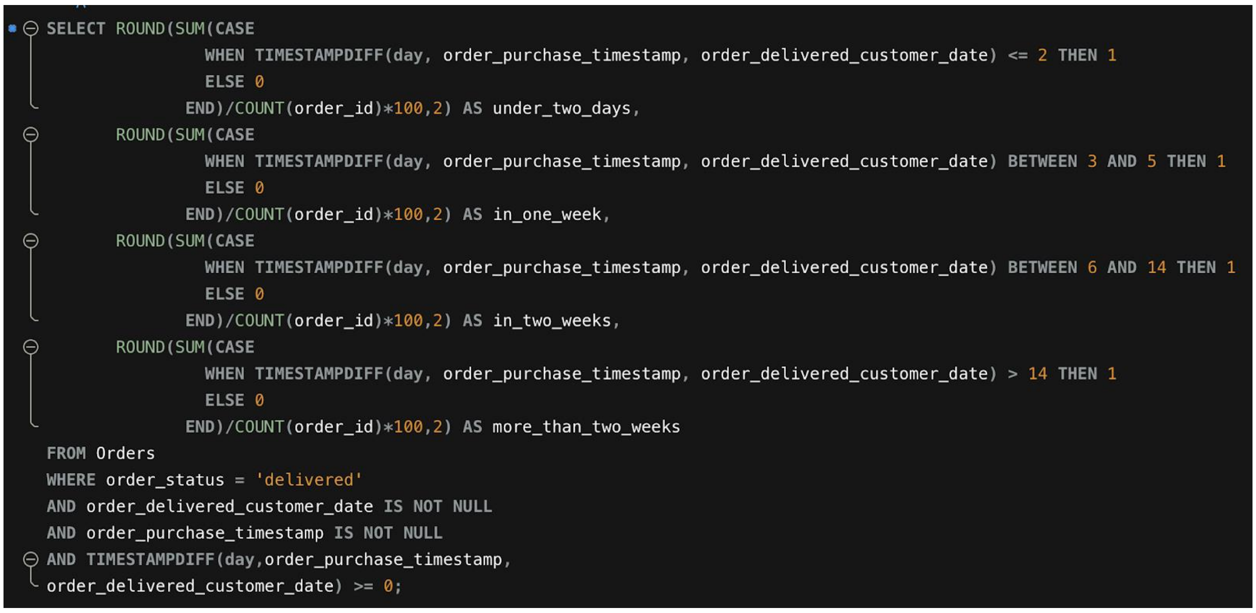
The query is used to identify the number of customers and total sales for each state. This information can be used to identify which states have the most customers and the highest sales and can be used to inform business decisions such as where to focus marketing efforts. As can be seen in Figure 20, less number customers cause less sales in the states directly, thus marketing team should advertise more on states with lesser customers and sales like SE, TO and RO, to attract more customers as years go.

3. Evaluating the delivery performance and identifying regions where delivery times can be improved.

According to meteorspace, if firms don't give their online clients fast delivery, more than 50% of them will cancel their orders or stop making purchases from the store. Thus, maintaining customers requires offering the greatest delivery service.

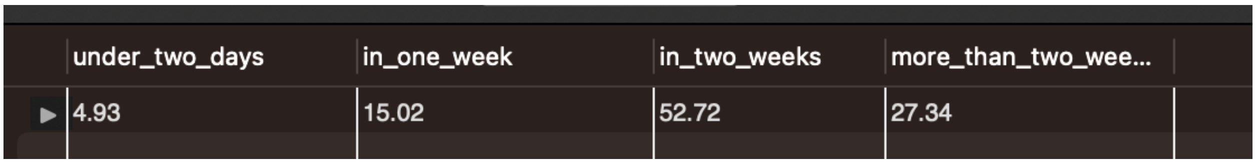
a) How fast are customers receiving their order?

One way to do this is to perform a query that calculates the percentage of orders arriving within 2 days, 1 week, 2 weeks, or more than 2 weeks after they are placed.



```
SELECT ROUND(SUM(CASE
                WHEN TIMESTAMPDIFF(day, order_purchase_timestamp, order_delivered_customer_date) <= 2 THEN 1
                ELSE 0
              END)/COUNT(order_id)*100,2) AS under_two_days,
       ROUND(SUM(CASE
                WHEN TIMESTAMPDIFF(day, order_purchase_timestamp, order_delivered_customer_date) BETWEEN 3 AND 5 THEN 1
                ELSE 0
              END)/COUNT(order_id)*100,2) AS in_one_week,
       ROUND(SUM(CASE
                WHEN TIMESTAMPDIFF(day, order_purchase_timestamp, order_delivered_customer_date) BETWEEN 6 AND 14 THEN 1
                ELSE 0
              END)/COUNT(order_id)*100,2) AS in_two_weeks,
       ROUND(SUM(CASE
                WHEN TIMESTAMPDIFF(day, order_purchase_timestamp, order_delivered_customer_date) > 14 THEN 1
                ELSE 0
              END)/COUNT(order_id)*100,2) AS more_than_two_weeks
FROM Orders
WHERE order_status = 'delivered'
AND order_delivered_customer_date IS NOT NULL
AND order_purchase_timestamp IS NOT NULL
AND TIMESTAMPDIFF(day,order_purchase_timestamp,
order_delivered_customer_date) >= 0;
```

Figure 21: Query 3a Code

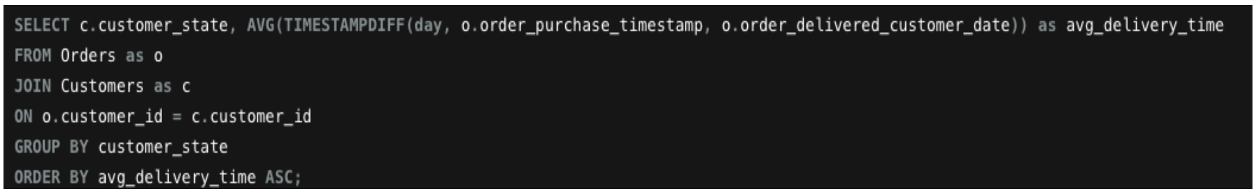


under_two_days	in_one_week	in_two_weeks	more_than_two_wee...
4.93	15.02	52.72	27.34

Figure 22: Query 3a Output

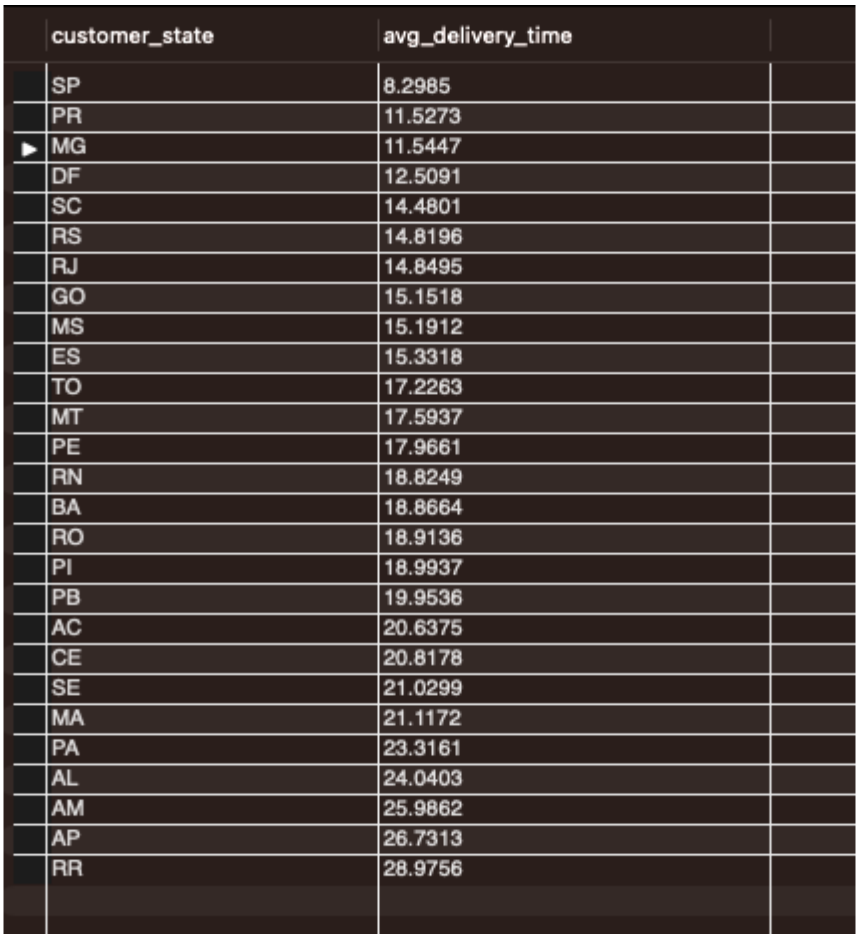
We can observe in Figure 22 that 27.34 orders are being received after 2 weeks of time. Due to a poor delivery experience, late deliveries reduce your customer retention rate and diminish your chances of keeping clients. Therefore, the logistics department should look at the causes of those orders taking longer to deliver and address it effectively in order to give speedy delivery services.

b) Determining the average delivery time based on State.



```
SELECT c.customer_state, AVG(TIMESTAMPDIFF(day, o.order_purchase_timestamp, o.order_delivered_customer_date)) as avg_delivery_time
FROM Orders as o
JOIN Customers as c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY avg_delivery_time ASC;
```

Figure 23: Query 3b Code



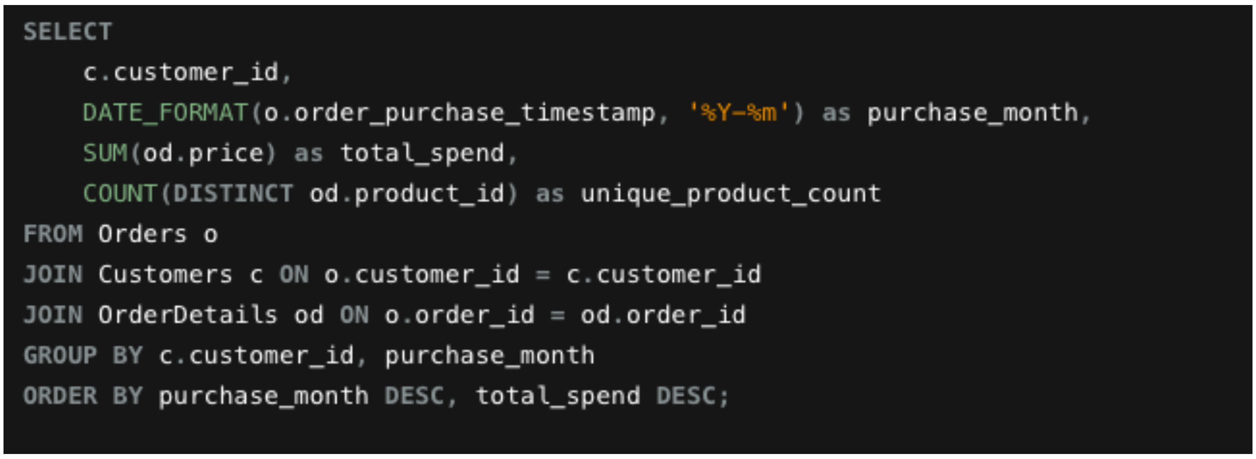
customer_state	avg_delivery_time
SP	8.2985
PR	11.5273
MG	11.5447
DF	12.5091
SC	14.4801
RS	14.8196
RJ	14.8495
GO	15.1518
MS	15.1912
ES	15.3318
TO	17.2263
MT	17.5937
PE	17.9661
RN	18.8249
BA	18.8664
RO	18.9136
PI	18.9937
PB	19.9536
AC	20.6375
CE	20.8178
SE	21.0299
MA	21.1172
PA	23.3161
AL	24.0403
AM	25.9862
AP	26.7313
RR	28.9756

Figure 24: Query 3b Output

The value of the avg_delivery_time is the average amount of time it takes for an order to be delivered to a customer, in days. This value can indicate how efficiently the delivery process is being managed and can help identify areas where improvements can be made in order to reduce delivery times and improve customer satisfaction. It also indicates the company's ability to meet the delivery expectations of their customers. Furthermore, we can see from the table the average delivery time from the fastest to the lowest in terms of days and based on this the company can determine which states need improvements in terms of delivery time.

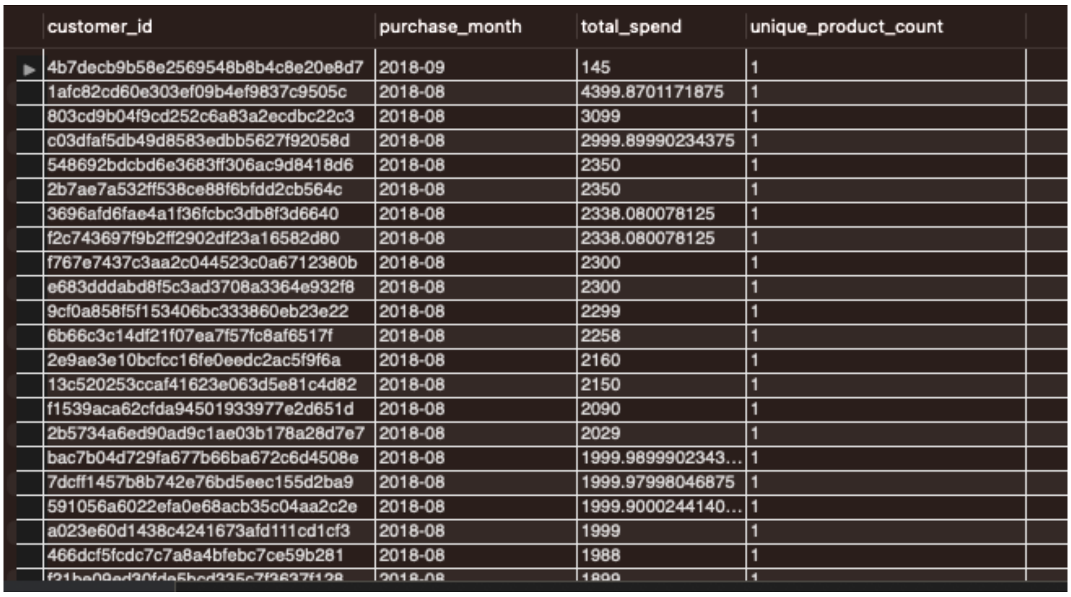
4. Analyzing the customer behaviour over time and identifying patterns that can be used to improve targeting.

Understanding consumer behaviour can be aided by looking at the sum of money spent by each client over time or by observing the product that is being purchased more frequently as it will help Olist understand what its customers want and need, so it can offer products and services that appeal to its target audiences.



```
SELECT
    c.customer_id,
    DATE_FORMAT(o.order_purchase_timestamp, '%Y-%m') as purchase_month,
    SUM(od.price) as total_spend,
    COUNT(DISTINCT od.product_id) as unique_product_count
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
JOIN OrderDetails od ON o.order_id = od.order_id
GROUP BY c.customer_id, purchase_month
ORDER BY purchase_month DESC, total_spend DESC;
```

Figure 25: Query 4 Code



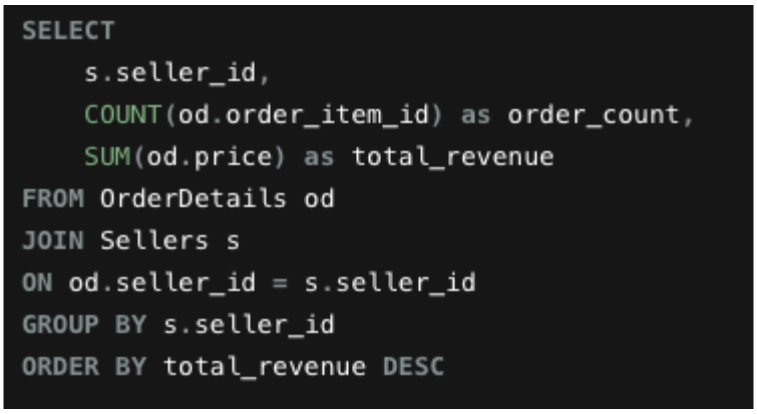
customer_id	purchase_month	total_spend	unique_product_count
4b7decb9b58e2569548b8b4c8e20e8d7	2018-09	145	1
1afc82cd60e303ef09b4ef9837c9505c	2018-08	4399.8701171875	1
803cd9b04f9cd252c6a83a2ecdbc22c3	2018-08	3099	1
c03dfaf5db49d8583edbb5627f92058d	2018-08	2999.89990234375	1
548692bdcbd6e3683ff306ac9d8418d6	2018-08	2350	1
2b7ae7a532ff538ce88f6bfdd2cb564c	2018-08	2350	1
3696afd6fae4a1f36fcbc3db8f3d6640	2018-08	2338.080078125	1
f2c743697f9b2ff2902df23a16582d80	2018-08	2338.080078125	1
f767e7437c3aa2c044523c0a6712380b	2018-08	2300	1
e683dddabd8f5c3ad3708a3364e932f8	2018-08	2300	1
9cf0a858f5f153406bc333860eb23e22	2018-08	2299	1
6b66c3c14df21f07ea7f57fc8af6517f	2018-08	2258	1
2e9ae3e10bcfcc16fe0eedc2ac5f9f6a	2018-08	2160	1
13c520253ccaf41623e063d5e81c4d82	2018-08	2150	1
f1539aca62cfda94501933977e2d651d	2018-08	2090	1
2b5734a6ed90ad9c1ae03b178a28d7e7	2018-08	2029	1
bac7b04d729fa677b66ba672c6d4508e	2018-08	1999.9899902343...	1
7dcff1457b8b742e76bd5eec155d2ba9	2018-08	1999.97998046875	1
591056a6022efa0e68acb35c04aa2c2e	2018-08	1999.9000244140...	1
a023e60d1438c4241673afd111cd1cf3	2018-08	1999	1
466dcf5fcdc7c7a8a4bfebc7ce59b281	2018-08	1988	1
f21be09ed30fde5bcd335c7f3637f128	2018-08	1899	1

Figure 26: Query 5 Output

The query will give the company a list of customers and their total spend per month, and the number of unique products they purchased. They can then use this information to identify patterns in customer behaviour over time, such as which months they tend to spend the most, which products are popular among certain customers, etc.

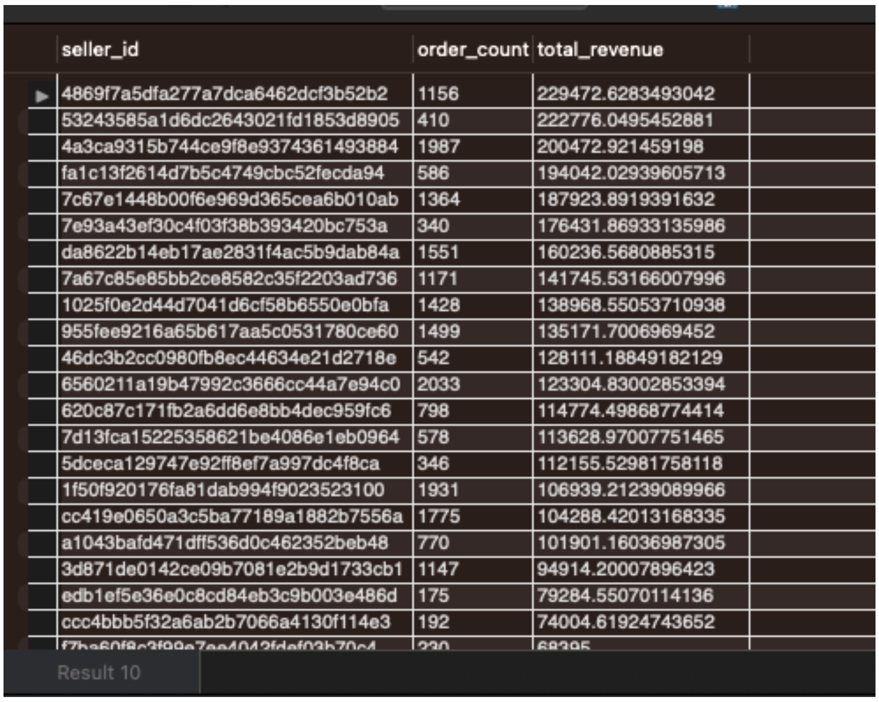
5. Evaluating the performance of sellers and identifying the best-performing ones.

   According to Greatplacetowork, acknowledging employees' contributions to the success of their teams and the organization as a whole helps them understand how much their employers regard them. This is especially important as organizations expand or undergo change. Olist should thus honor its top sellers and express its gratitude.



```
SELECT
    s.seller_id,
    COUNT(od.order_item_id) as order_count,
    SUM(od.price) as total_revenue
FROM OrderDetails od
JOIN Sellers s
ON od.seller_id = s.seller_id
GROUP BY s.seller_id
ORDER BY total_revenue DESC
```

Figure 27: Query 5 Code



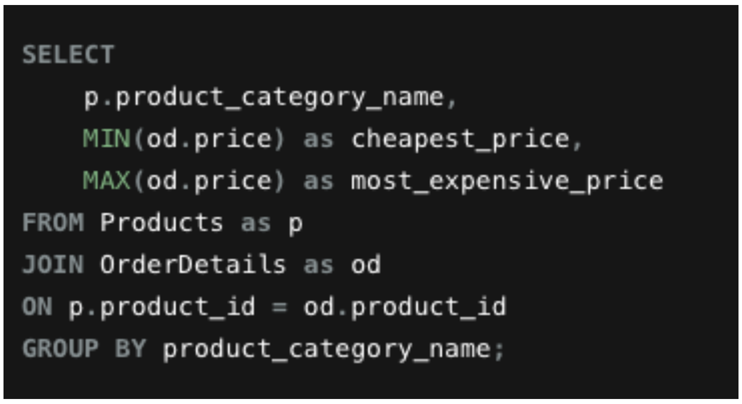
seller_id	order_count	total_revenue
4869f7a5dfa277a7dca6462dcf3b52b2	1156	229472.6283493042
53243585a1d6dc2643021fd1853d8905	410	222776.0495452881
4a3ca9315b744ce9f8e9374361493884	1987	200472.921459198
fa1c13f2614d7b5c4749cbc52fecda94	586	194042.02939605713
7c67e1448b00f6e969d365cea6b010ab	1364	187923.8919391632
7e93a43ef30c4f03f38b393420bc753a	340	176431.86933135986
da8622b14eb17ae2831f4ac5b9dab84a	1551	160236.5680885315
7a67c85e85bb2ce8582c35f2203ad736	1171	141745.53166007996
1025f0e2d44d7041d6cf58b6550e0bfa	1428	138968.55053710938
955fee9216a65b617aa5c0531780ce60	1499	135171.7006969452
46dc3b2cc0980fb8ec44634e21d2718e	542	128111.18849182129
6560211a19b47992c3666cc44a7e94c0	2033	123304.83002853394
620c87c171fb2a6dd6e8bb4dec959fc6	798	114774.49868774414
7d13fca15225358621be4086e1eb0964	578	113628.97007751465
5dceca129747e92ff8ef7a997dc4f8ca	346	112155.52981758118
1f50f920176fa81dab994f9023523100	1931	106939.21239089966
cc419e0650a3c5ba77189a1882b7556a	1775	104288.42013168335
a1043bafd471dff536d0c462352beb48	770	101901.16036987305
3d871de0142ce09b7081e2b9d1733cb1	1147	94914.20007896423
edb1ef5e36e0c8cd84eb3c9b003e486d	175	79284.55070114136
ccc4bbb5f32a6ab2b7066a4130f114e3	192	74004.61924743652
f7ba60f8c3f99e7ee4042fdef03b70c4	230	68395

Result 10

Figure 28: Query 5 Output

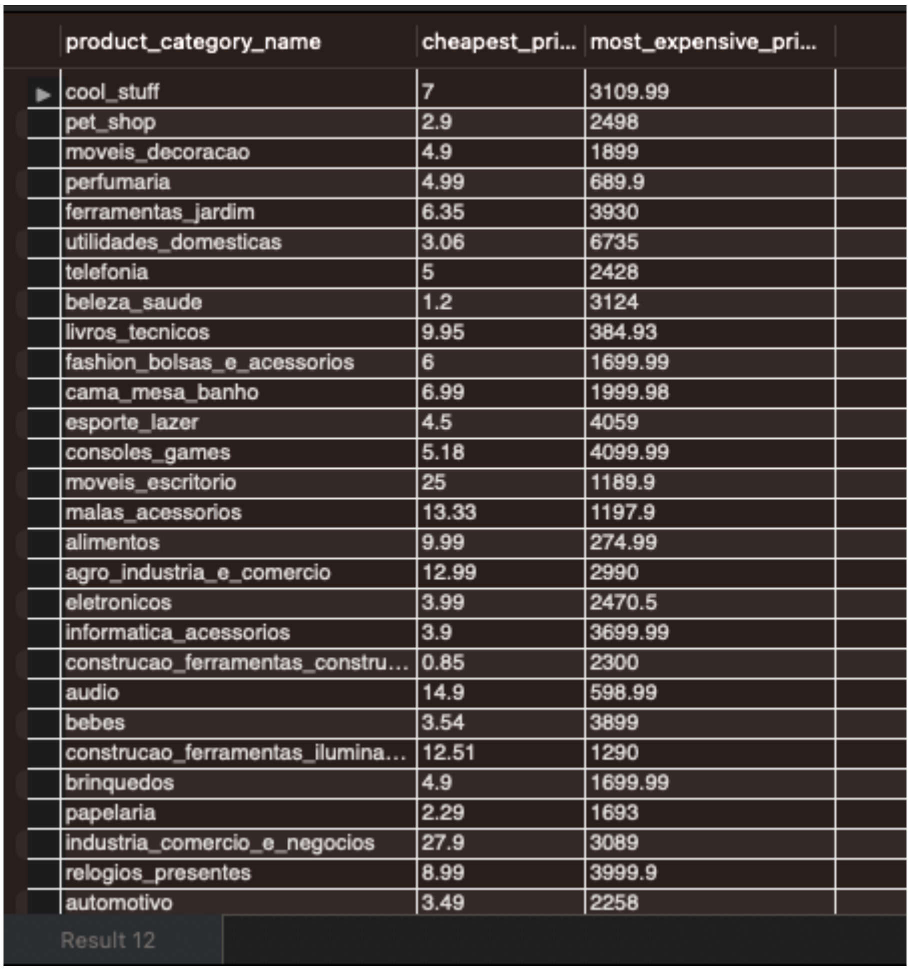
The query will return the seller_id, the number of orders (order_count) and the total revenue (total_revenue) for each seller, ordered by total revenue in descending order. This way the company can see which sellers have the highest revenue and are the best performers. Human resource departments should acknowledge top-performing sellers, such as the seller who has sold 1156 things. These sellers should be rewarded with money, promotions, or other forms of motivation so that they will keep up their efforts and bring in more customers for the company.

6. Knowing the most expensive and the cheapest products in each category.



```sql
SELECT
    p.product_category_name,
    MIN(od.price) as cheapest_price,
    MAX(od.price) as most_expensive_price
FROM Products as p
JOIN OrderDetails as od
ON p.product_id = od.product_id
GROUP BY product_category_name;
```

Figure 29: Query 6 Code



product_category_name	cheapest_pri...	most_expensive_pri...
cool_stuff	7	3109.99
pet_shop	2.9	2498
moveis_decoracao	4.9	1899
perfumaria	4.99	689.9
ferramentas_jardim	6.35	3930
utilidades_domesticas	3.06	6735
telefonia	5	2428
beleza_saude	1.2	3124
livros_tecnicos	9.95	384.93
fashion_bolsas_e_acessorios	6	1699.99
cama_mesa_banho	6.99	1999.98
esporte_lazer	4.5	4059
consoles_games	5.18	4099.99
moveis_escritorio	25	1189.9
malas_acessorios	13.33	1197.9
alimentos	9.99	274.99
agro_industria_e_comercio	12.99	2990
eletronicos	3.99	2470.5
informatica_acessorios	3.9	3699.99
construcao_ferramentas_constru...	0.85	2300
audio	14.9	598.99
bebes	3.54	3899
construcao_ferramentas_ilumina...	12.51	1290
brinquedos	4.9	1699.99
papelaria	2.29	1693
industria_comercio_e_negocios	27.9	3089
relogios_presentes	8.99	3999.9
automotivo	3.49	2258

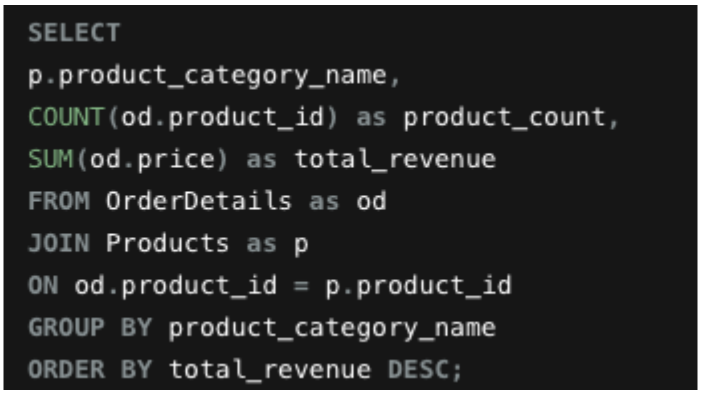
Result 12

Figure 30: Query 6 Output

The query will provide the information about the most expensive and the least expensive product for each product category. Furthermore, knowing the most expensive and cheapest items in a company's product line can help the company make strategic decisions about pricing, product development, and marketing. For example, if a company knows that its most expensive product is also its most profitable, it may choose to invest more resources into promoting and developing that product. On the other hand, if a company knows that its cheapest product is not selling well, it may choose to discontinue that product or lower its price to make it more competitive. This information can also help the company identify any potential gaps in its product line and develop new products to fill those gaps. Additionally, understanding the pricing strategy of competitors can also be useful to set the prices.

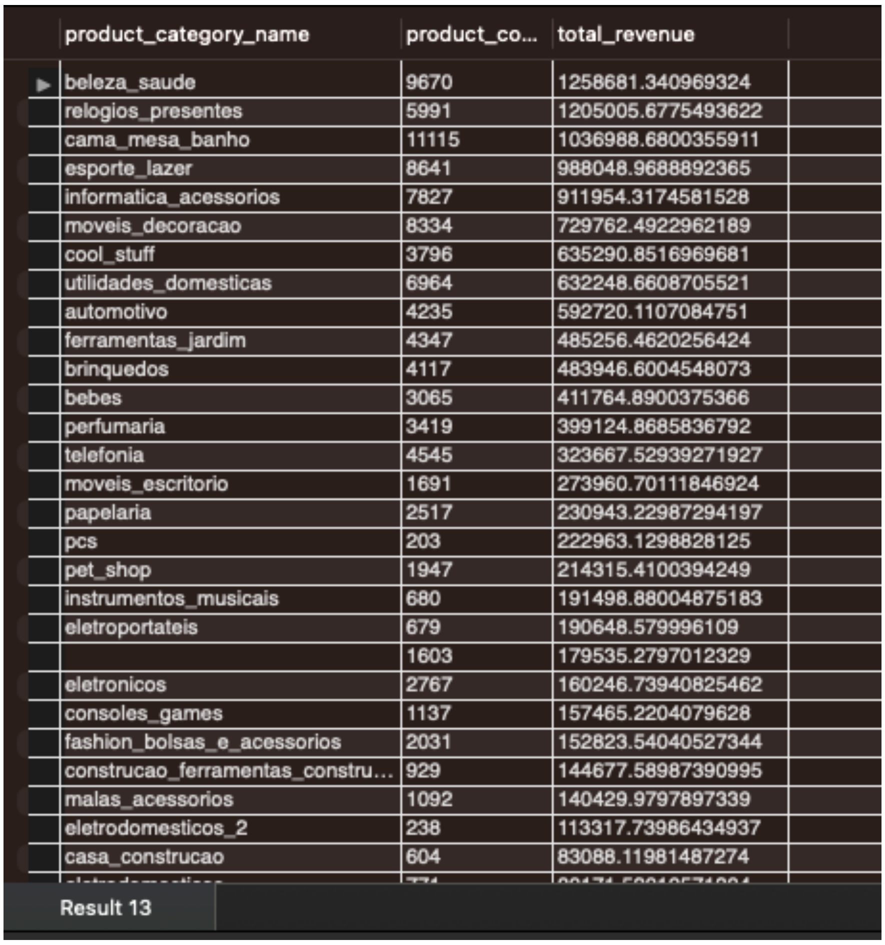
7. Analyzing the product category performance and identifying the most popular and profitable product categories.

To determine whether there is a demand for the product category or if there are any other factors contributing to the lower sales, and to work on developing strategies to boost the sales of such products.



```
SELECT
p.product_category_name,
COUNT(od.product_id) as product_count,
SUM(od.price) as total_revenue
FROM OrderDetails as od
JOIN Products as p
ON od.product_id = p.product_id
GROUP BY product_category_name
ORDER BY total_revenue DESC;
```

Figure 31: Query 7 Code



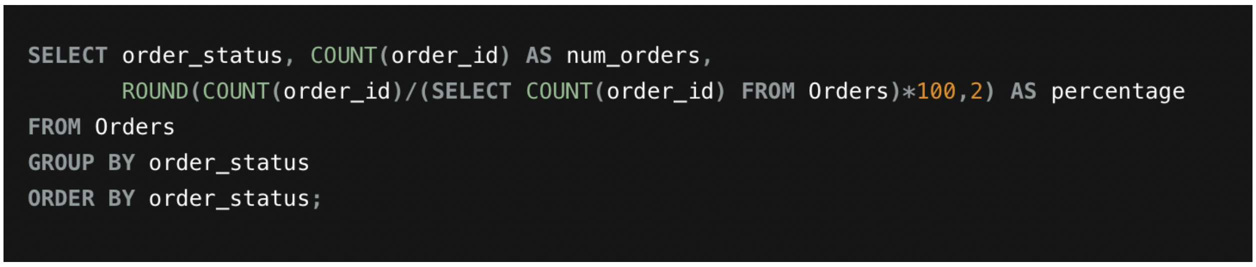
product_category_name	product_co...	total_revenue
beleza_saude	9670	1258681.340969324
relogios_presentes	5991	1205005.6775493622
cama_mesa_banho	11115	1036988.6800355911
esporte_lazer	8641	988048.9688892365
informatica_acessorios	7827	911954.3174581528
moveis_decoracao	8334	729762.4922962189
cool_stuff	3796	635290.8516969681
utilidades_domesticas	6964	632248.6608705521
automotivo	4235	592720.1107084751
ferramentas_jardim	4347	485256.4620256424
brinquedos	4117	483946.6004548073
bebes	3065	411764.8900375366
perfumaria	3419	399124.8685836792
telefonia	4545	323667.52939271927
moveis_escritorio	1691	273960.70111846924
papelaria	2517	230943.22987294197
pcs	203	222963.1298828125
pet_shop	1947	214315.4100394249
instrumentos_musicais	680	191498.88004875183
eletroportateis	679	190648.579996109
	1603	179535.2797012329
eletronicos	2767	160246.73940825462
consoles_games	1137	157465.2204079628
fashion_bolsas_e_acessorios	2031	152823.54040527344
construcao_ferramentas_constru...	929	144677.58987390995
malas_acessorios	1092	140429.9797897339
eletrodomesticos_2	238	113317.73986434937
casa_construcao	604	83088.11981487274

Figure 32: Query 7 Output

This information can help the company identify which product categories are the most popular and profitable, and can be used to develop strategies to boost sales for those products. Additionally, by understanding which product categories are not selling well, the company can determine whether there is a lack of demand for the product or if there are other factors contributing to the lower sales. This can help the company make informed decisions about inventory management, marketing efforts, and product development.

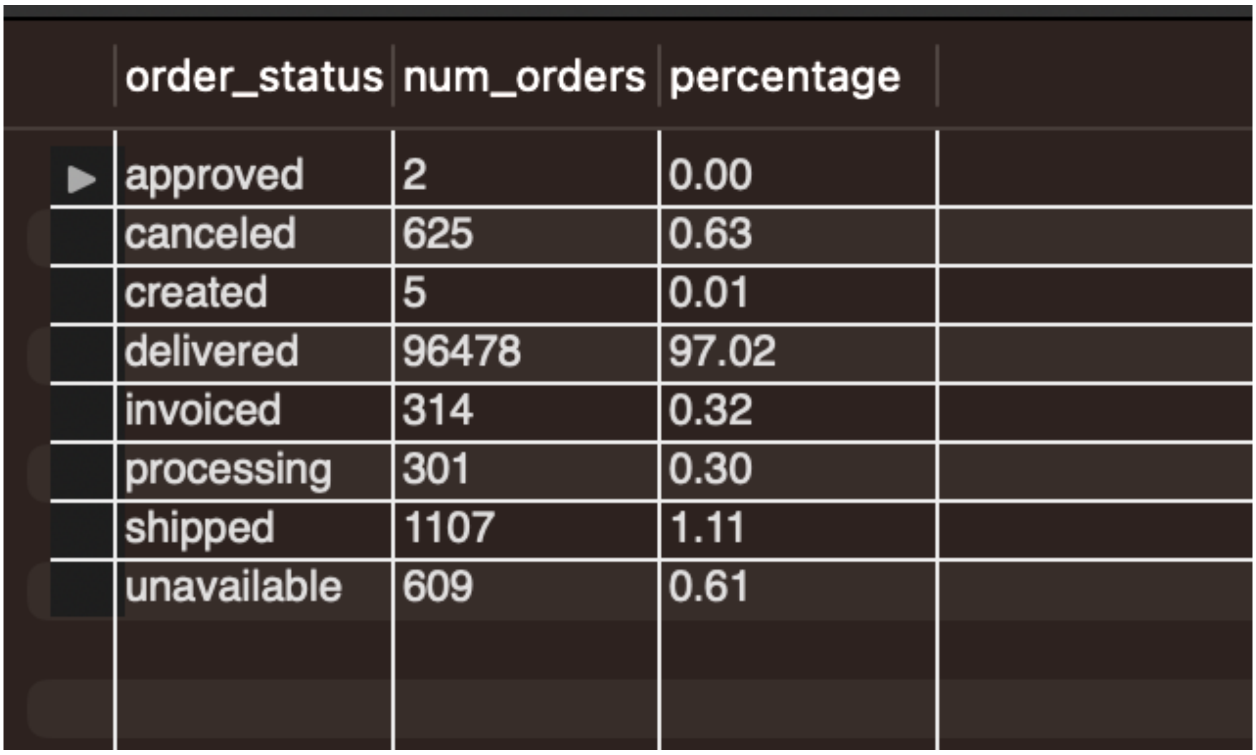
8. Checking the number of different order status categories and their percentage can give insight into the efficiency of the company order fulfilment process.

For example, if a high percentage of orders have a status of "delivered," it indicates that the company is fulfilling orders in a timely and efficient manner. If a large percentage of orders have a status of "cancelled," it could indicate that there are issues with the ordering process or with order fulfilment that need to be addressed. Additionally, this information can also be used to identify any bottlenecks or inefficiencies in the order fulfillment process, so that they can be addressed and improved.



```
SELECT order_status, COUNT(order_id) AS num_orders,
       ROUND(COUNT(order_id)/(SELECT COUNT(order_id) FROM Orders)*100,2) AS percentage
FROM Orders
GROUP BY order_status
ORDER BY order_status;
```

Figure 33: Query 8 Code



order_status	num_orders	percentage
approved	2	0.00
canceled	625	0.63
created	5	0.01
delivered	96478	97.02
invoiced	314	0.32
processing	301	0.30
shipped	1107	1.11
unavailable	609	0.61

Figure 34: Query 8 Output

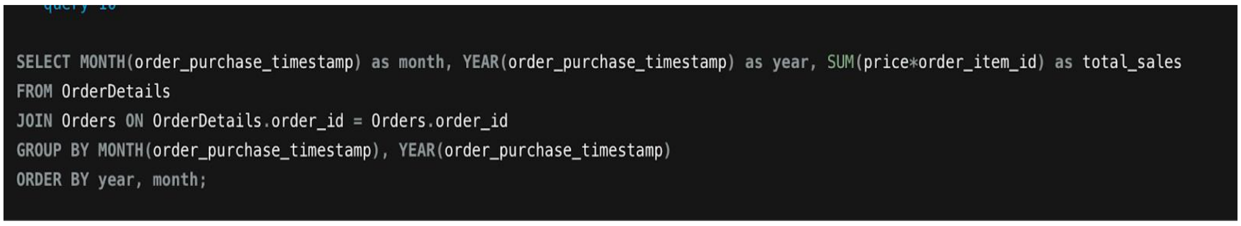
This query first gets the total number of orders by selecting the count of order_id from the Orders table and storing it in a subquery. Then, it joins the subquery with the Orders table to get the count of each order status category, and calculates the percentage using the COUNT(order_id) and the total number of orders obtained from the subquery.

We can see that the company has a high percentage for fulfilling orders. 97% of the orders are delivered. Although the proportion may appear small, customer satisfaction should be the top priority for any company since if it doesn't meet their needs, customers will eventually stop doing business with it. The customer service department should investigate what is causing 0.63% of orders to be cancelled.

9. Understanding the sales pattern over the months and the time of the day where the shopping is done.

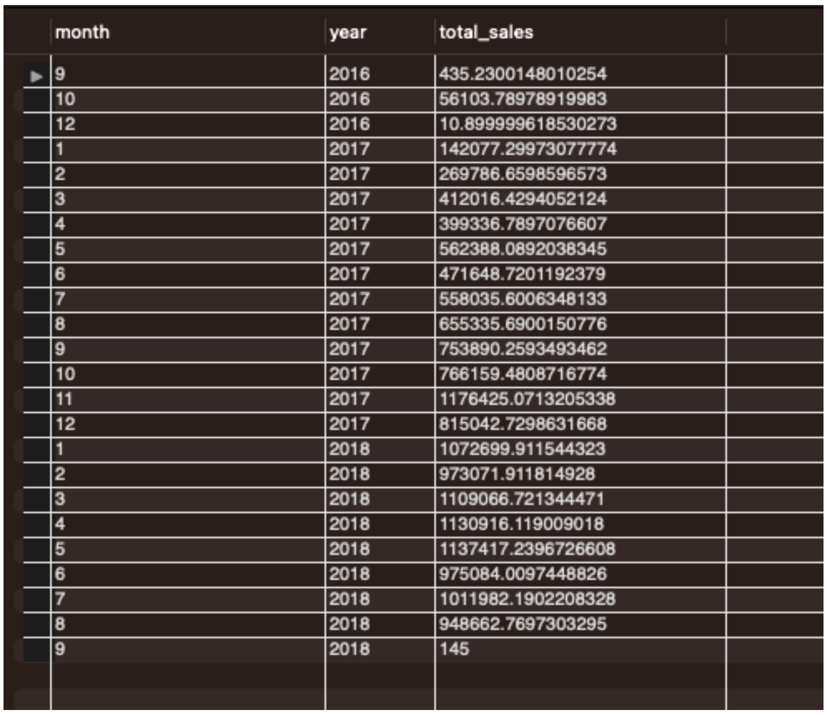
By knowing which month has the highest sales every year, to see if there is any festival causing more sales or any other factors affecting it to boost the sales in upcoming years.

a) the sales pattern over the months



```
SELECT MONTH(order_purchase_timestamp) as month, YEAR(order_purchase_timestamp) as year, SUM(price*order_item_id) as total_sales
FROM OrderDetails
JOIN Orders ON OrderDetails.order_id = Orders.order_id
GROUP BY MONTH(order_purchase_timestamp), YEAR(order_purchase_timestamp)
ORDER BY year, month;
```

Figure 35: Query 9a Code

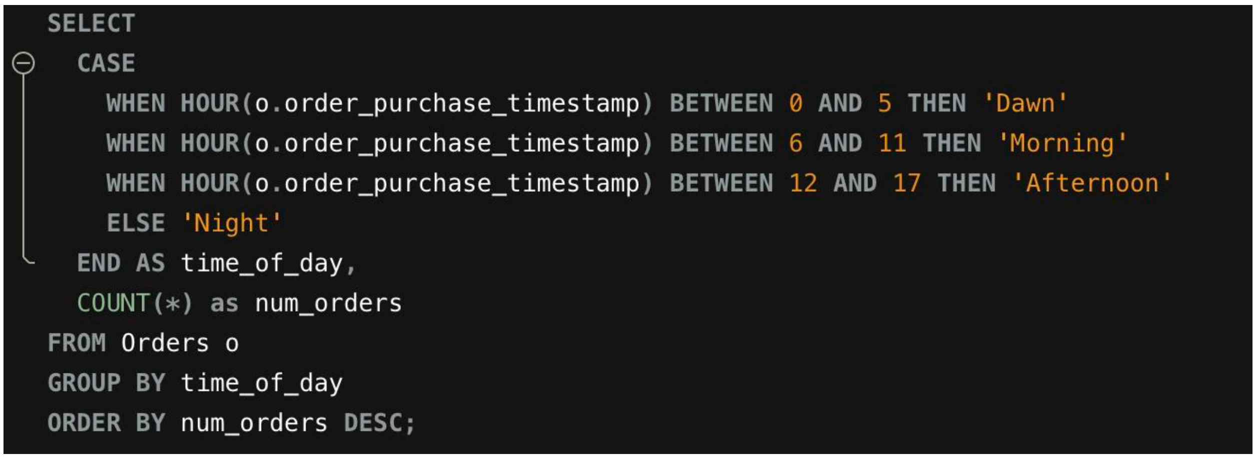


month	year	total_sales
9	2016	435.2300148010254
10	2016	56103.78978919983
12	2016	10.899999618530273
1	2017	142077.29973077774
2	2017	269786.6598596573
3	2017	412016.4294052124
4	2017	399336.7897076607
5	2017	562388.0892038345
6	2017	471648.7201192379
7	2017	558035.6006348133
8	2017	655335.6900150776
9	2017	753890.2593493462
10	2017	766159.4808716774
11	2017	1176425.0713205338
12	2017	815042.7298631668
1	2018	1072699.911544323
2	2018	973071.911814928
3	2018	1109066.721344471
4	2018	1130916.119009018
5	2018	1137417.2396726608
6	2018	975084.0097448826
7	2018	1011982.1902208328
8	2018	948662.7697303295
9	2018	145

Figure 36: Query 9a Output

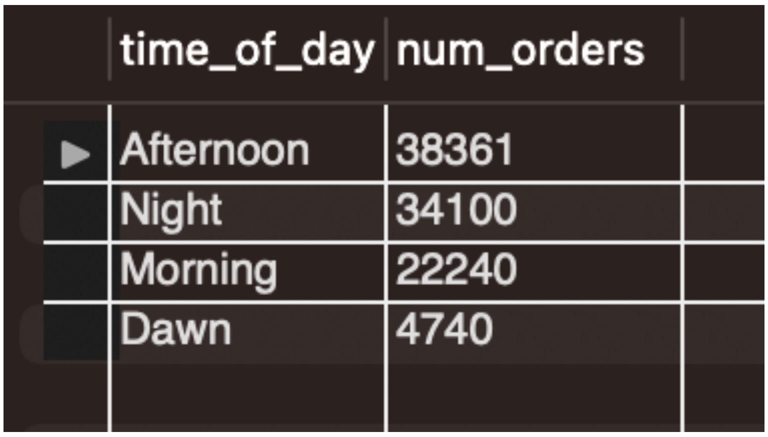
The query will group the sales data by month and year and will provide more detailed information on the sales patterns over time, as it will show how sales have changed from year to year for each month. This will allow the company to see if there are any annual patterns in sales, such as a peak in sales during a specific month each year, or if sales in a particular month have been steadily increasing or decreasing over time. Additionally, it will allow comparison of sales across different years for the same month, making it easier to identify any trends or patterns that may be specific to a certain year.

b) the time of the day where the shopping is done.



```
SELECT
  CASE
    WHEN HOUR(o.order_purchase_timestamp) BETWEEN 0 AND 5 THEN 'Dawn'
    WHEN HOUR(o.order_purchase_timestamp) BETWEEN 6 AND 11 THEN 'Morning'
    WHEN HOUR(o.order_purchase_timestamp) BETWEEN 12 AND 17 THEN 'Afternoon'
    ELSE 'Night'
  END AS time_of_day,
  COUNT(*) as num_orders
FROM Orders o
GROUP BY time_of_day
ORDER BY num_orders DESC;
```

Figure 37: Query 9b Code

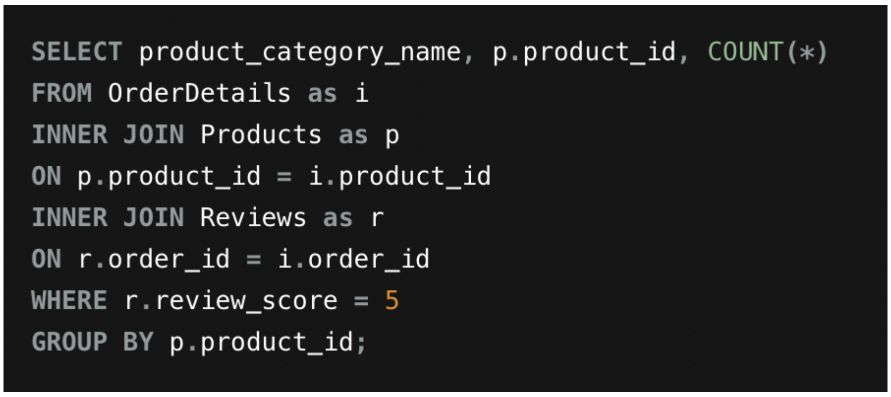


time_of_day	num_orders
Afternoon	38361
Night	34100
Morning	22240
Dawn	4740

Figure 38: Query 9b Output

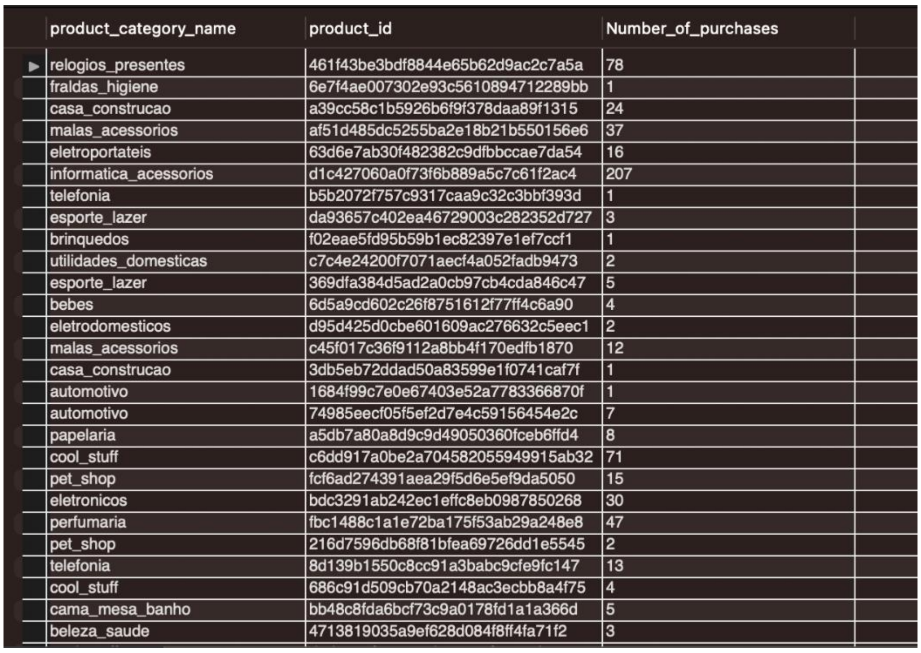
The query provides the number of orders in each segment of the day which help will help the company in determining their customer tendencies and patterns. Based on the results, it is clear that Olist customers are more likely to make purchases in the afternoon. This is advantageous for the business because it may run promotions in the afternoon that will increase sales. The business can also introduce new products during certain times in an effort to attract clients, who are more busy buying in the afternoon.

10. List products with review score of 5 and total number of purchase



```
SELECT product_category_name, p.product_id, COUNT(*)
FROM OrderDetails as i
INNER JOIN Products as p
ON p.product_id = i.product_id
INNER JOIN Reviews as r
ON r.order_id = i.order_id
WHERE r.review_score = 5
GROUP BY p.product_id;
```

Figure 39: Query 10 Code

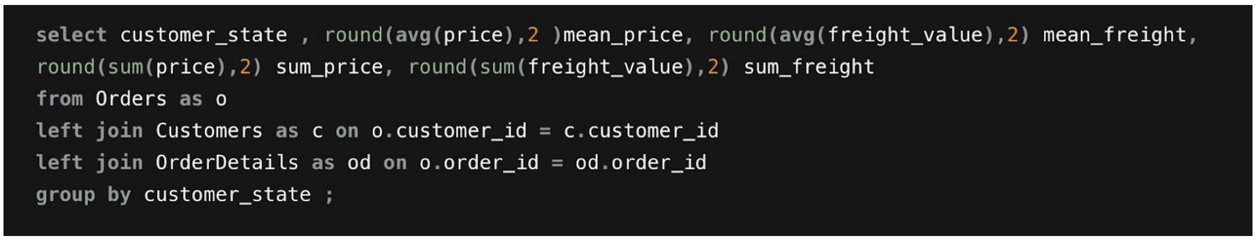


product_category_name	product_id	Number_of_purchases
relogios_presentes	461f43be3bdf8844e65b62d9ac2c7a5a	78
fraldas_higiene	6e7f4ae007302e93c5610894712289bb	1
casa_construcao	a39cc58c1b5926b6f9f378daa89f1315	24
malas_acessorios	af51d485dc5255ba2e18b21b550156e6	37
eletroportateis	63d6e7ab30f482382c9dfbbccae7da54	16
informatica_acessorios	d1c427060a0f73f6b889a5c7c61f2ac4	207
telefonia	b5b2072f757c9317caa9c32c3bbf393d	1
esporte_lazer	da93657c402ea46729003c282352d727	3
brinquedos	f02eae5fd95b59b1ec82397e1ef7ccf1	1
utilidades_domesticas	c7c4e24200f7071aecf4a052fadb9473	2
esporte_lazer	369dfa384d5ad2a0cb97cb4cda846c47	5
bebes	6d5a9cd602c26f8751612f77ff4c6a90	4
eletrodomesticos	d95d425d0cbe601609ac276632c5eec1	2
malas_acessorios	c45f017c36f9112a8bb4f170edfb1870	12
casa_construcao	3db5eb72ddad50a83599e1f0741caf7f	1
automotivo	1684f99c7e0e67403e52a7783366870f	1
automotivo	74985eecf05f5ef2d7e4c59156454e2c	7
papelaria	a5db7a80a8d9c9d49050360fceb6ffd4	8
cool_stuff	c6dd917a0be2a70458205594991 5ab32	71
pet_shop	fcf6ad274391aea29f5d6e5ef9da5050	15
eletronicos	bdc3291ab242ec1effc8eb0987850268	30
perfumaria	fbc1488c1a1e72ba175f53ab29a248e8	47
pet_shop	216d7596db68f81bfea69726dd1e5545	2
telefonia	8d139b1550c8cc91a3babc9cfe9fc147	13
cool_stuff	686c91d509cb70a2148ac3ecbb8a4f75	4
cama_mesa_banho	bb48c8fda6bcf73c9a0178fd1a1a366d	5
beleza_saude	4713819035a9ef628d084f8ff4fa71f2	3

Figure 40: Query 10 Output

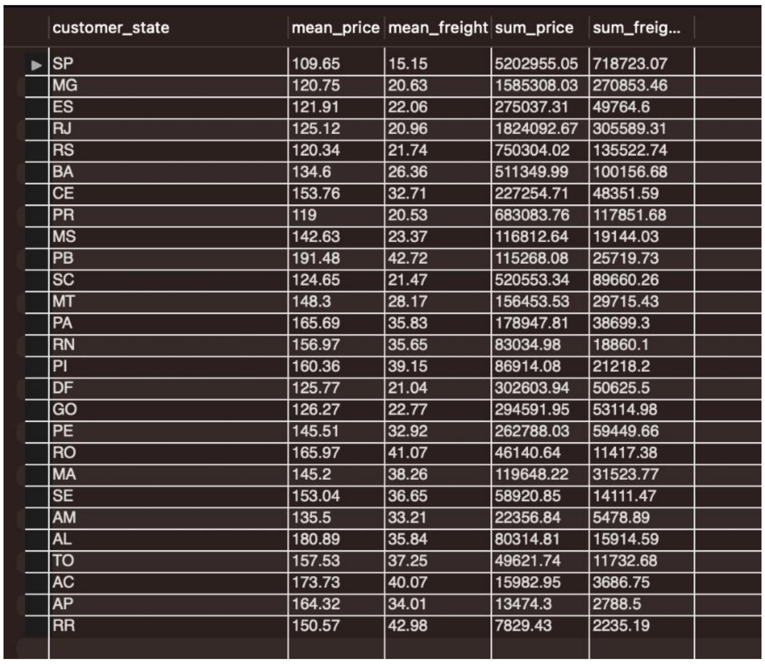
The query returns the product category name, product ID, and the number of purchases for each product where the corresponding order had a review score of 5. This query can give insight into which product categories and specific products are highly rated by customers. The customer service team should also investigate on the products that has received less than 3 stars on the reasons and try to solve those problems to increase customer satisfaction.

11. Identifying if there is a relationship between freight value and the total sales based on states.



```
select customer_state , round(avg(price),2 )mean_price, round(avg(freight_value),2) mean_freight,
round(sum(price),2) sum_price, round(sum(freight_value),2) sum_freight
from Orders as o
left join Customers as c on o.customer_id = c.customer_id
left join OrderDetails as od on o.order_id = od.order_id
group by customer_state ;
```

Figure 41: Query 11 Code

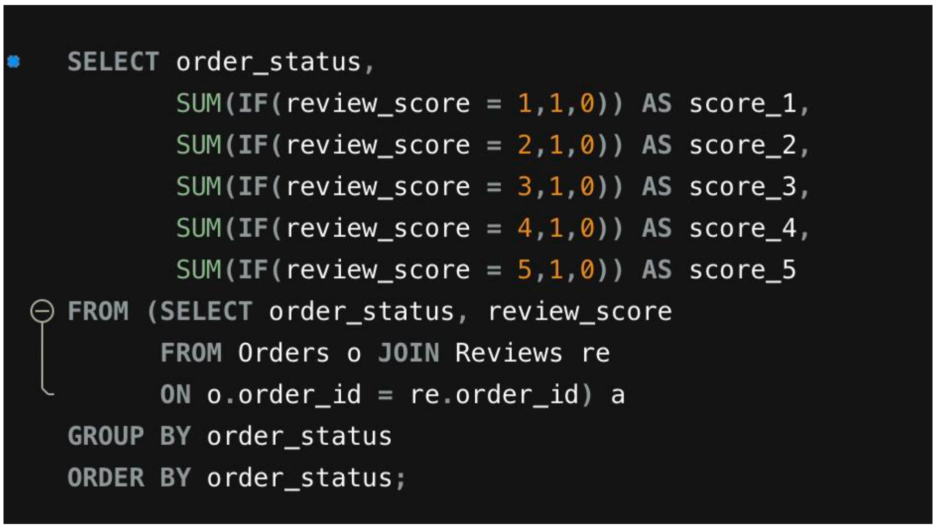


customer_state	mean_price	mean_freight	sum_price	sum_freig...
SP	109.65	15.15	5202955.05	718723.07
MG	120.75	20.63	1585308.03	270853.46
ES	121.91	22.06	275037.31	49764.6
RJ	125.12	20.96	1824092.67	305589.31
RS	120.34	21.74	750304.02	135522.74
BA	134.6	26.36	511349.99	100156.68
CE	153.76	32.71	227254.71	48351.59
PR	119	20.53	683083.76	117851.68
MS	142.63	23.37	116812.64	19144.03
PB	191.48	42.72	115268.08	25719.73
SC	124.65	21.47	520553.34	89660.26
MT	148.3	28.17	156453.53	29715.43
PA	165.69	35.83	178947.81	38699.3
RN	156.97	35.65	83034.98	18860.1
PI	160.36	39.15	86914.08	21218.2
DF	125.77	21.04	302603.94	50625.5
GO	126.27	22.77	294591.95	53114.98
PE	145.51	32.92	262788.03	59449.66
RO	165.97	41.07	46140.64	11417.38
MA	145.2	38.26	119648.22	31523.77
SE	153.04	36.65	58920.85	14111.47
AM	135.5	33.21	22356.84	5478.89
AL	180.89	35.84	80314.81	15914.59
TO	157.53	37.25	49621.74	11732.68
AC	173.73	40.07	15982.95	3686.75
AP	164.32	34.01	13474.3	2788.5
RR	150.57	42.98	7829.43	2235.19

Figure 42: Query 11 Output

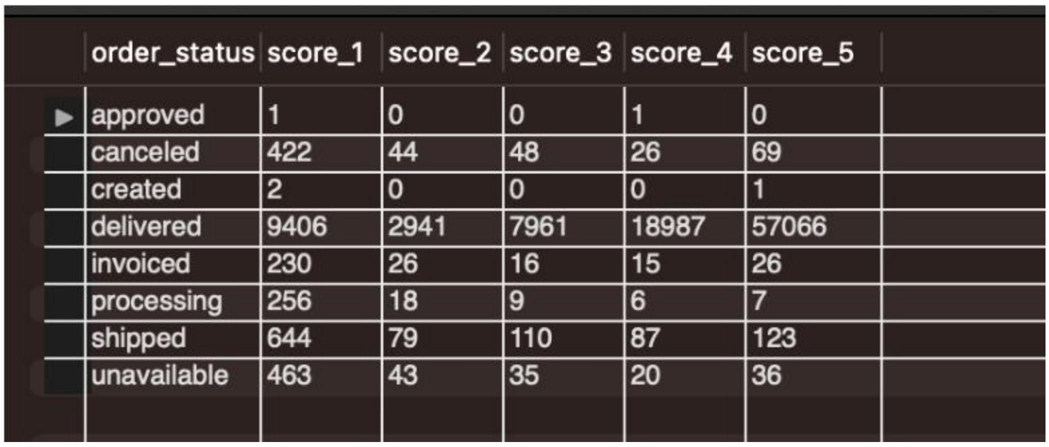
The query will provide insight into the average and total price and freight value for orders grouped by customer state. It will show the mean price and freight value for each state and the sum of the prices and freight values for all the orders in that state. This information can be used to analyse sales and shipping patterns by state and identify any potential issues or areas for improvement.

12. Identifying if there are any relationships between the order status and the rating for the orders (the Customer experience with each status).



```
SELECT order_status,
       SUM(IF(review_score = 1,1,0)) AS score_1,
       SUM(IF(review_score = 2,1,0)) AS score_2,
       SUM(IF(review_score = 3,1,0)) AS score_3,
       SUM(IF(review_score = 4,1,0)) AS score_4,
       SUM(IF(review_score = 5,1,0)) AS score_5
FROM (SELECT order_status, review_score
      FROM Orders o JOIN Reviews re
      ON o.order_id = re.order_id) a
GROUP BY order_status
ORDER BY order_status;
```

Figure 43: Query 12 Code

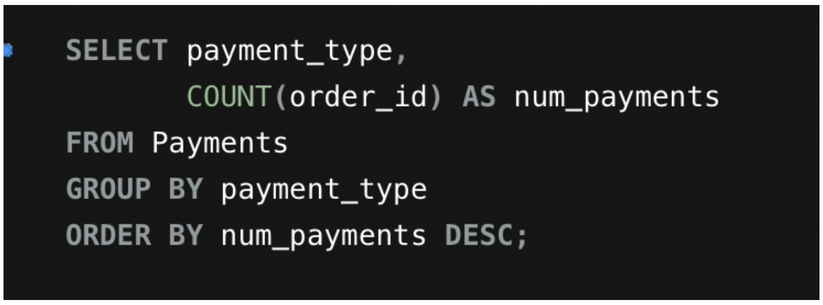


order_status	score_1	score_2	score_3	score_4	score_5
approved	1	0	0	1	0
canceled	422	44	48	26	69
created	2	0	0	0	1
delivered	9406	2941	7961	18987	57066
invoiced	230	26	16	15	26
processing	256	18	9	6	7
shipped	644	79	110	87	123
unavailable	463	43	35	20	36

Figure 44: Query 12 Output

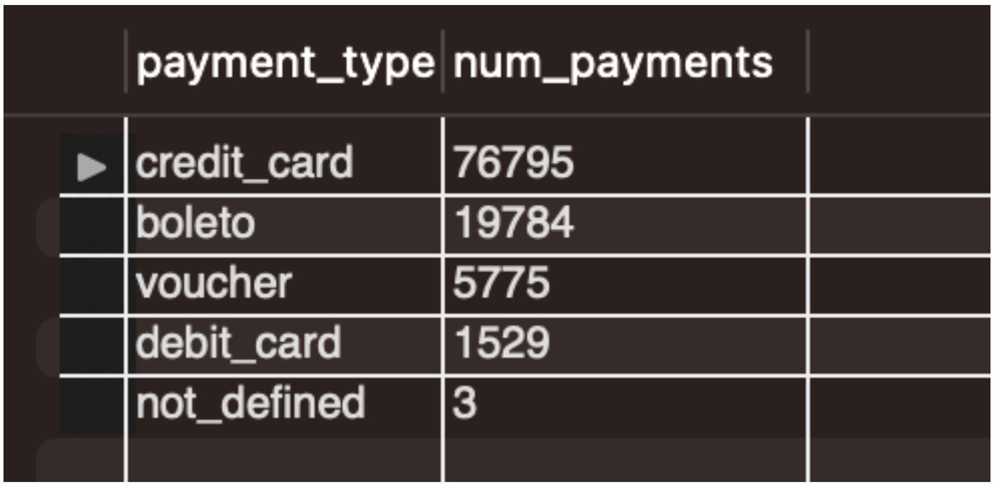
The query will give an insight on how customers rate their experiences with different order statuses. According to the results, Olist consumers give the poor order status a lower grade. For instance, 422 consumers provided 1 star for cancelled purchases while 463 customers gave 1 star for unavailable items. This indicates that the state of the order affects their star rating, so customer service and logistics should collaborate to effectively resolve this issue.

13. Which payment method is popular among the Customers?



```
SELECT payment_type,
       COUNT(order_id) AS num_payments
FROM Payments
GROUP BY payment_type
ORDER BY num_payments DESC;
```

Figure 45: Query 13 Code

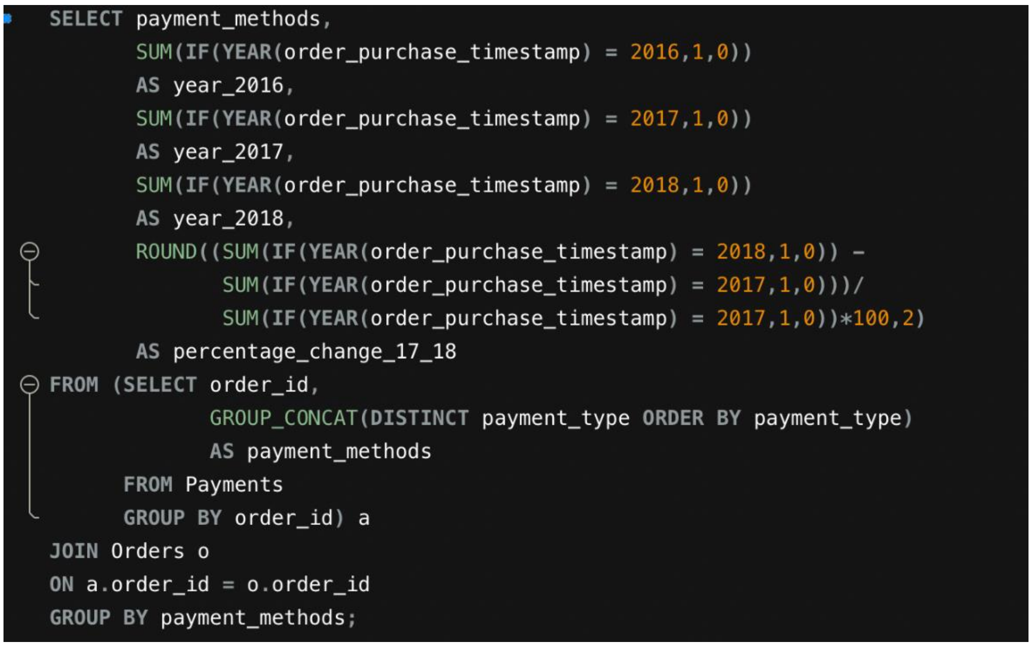


payment_type	num_payments
credit_card	76795
boleto	19784
voucher	5775
debit_card	1529
not_defined	3

Figure 46: Query 13 Output

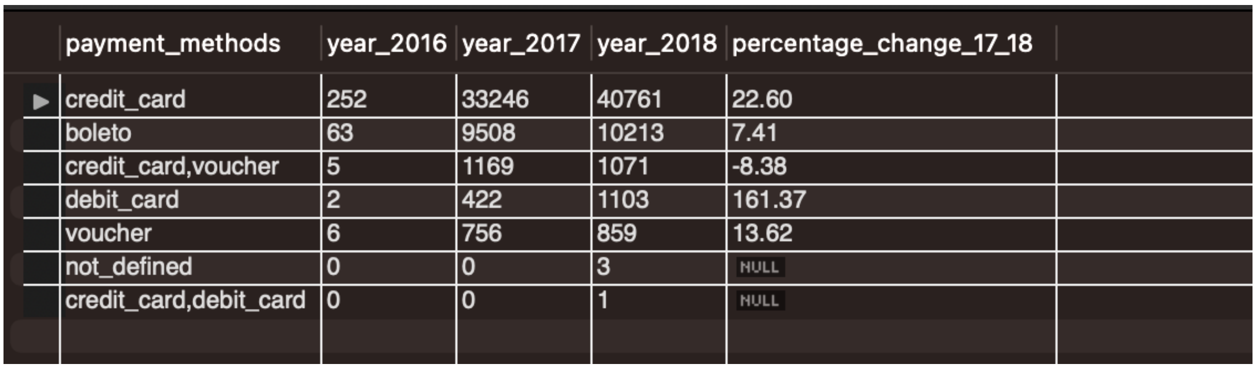
The query will give an insight on which payment method is popular among Olist customers. Since credit cards are the most popular mode of payment among Olist consumers, the business can attempt to offer perks to individuals who use credit cards, such as offers, discounts, or free shipping, to entice them to make additional purchases from the business. By offering unique discounts to clients who use a certain payment method, Olist can also encourage them to use more alternative payment methods, preventing a monopolisation of the market by a single payment method.

14. The change in payment types over the years.



```
SELECT payment_methods,
       SUM(IF(YEAR(order_purchase_timestamp) = 2016,1,0))
       AS year_2016,
       SUM(IF(YEAR(order_purchase_timestamp) = 2017,1,0))
       AS year_2017,
       SUM(IF(YEAR(order_purchase_timestamp) = 2018,1,0))
       AS year_2018,
       ROUND((SUM(IF(YEAR(order_purchase_timestamp) = 2018,1,0)) -
              SUM(IF(YEAR(order_purchase_timestamp) = 2017,1,0)))/
              SUM(IF(YEAR(order_purchase_timestamp) = 2017,1,0))*100,2)
       AS percentage_change_17_18
FROM (SELECT order_id,
             GROUP_CONCAT(DISTINCT payment_type ORDER BY payment_type)
             AS payment_methods
      FROM Payments
      GROUP BY order_id) a
JOIN Orders o
ON a.order_id = o.order_id
GROUP BY payment_methods;
```

Figure 47: Query 14 Code



payment_methods	year_2016	year_2017	year_2018	percentage_change_17_18
credit_card	252	33246	40761	22.60
boleto	63	9508	10213	7.41
credit_card,voucher	5	1169	1071	-8.38
debit_card	2	422	1103	161.37
voucher	6	756	859	13.62
not_defined	0	0	3	NULL
credit_card,debit_card	0	0	1	NULL

Figure 48: Query 14 Output

This query will provide insight into how the company's payment methods have changed over the years. By breaking down the number of orders by payment method and year, the company can see which payment methods are becoming more or less popular and make adjustments accordingly. The percentage change between 2017 and 2018 can also give the company an idea of how quickly a particular payment method is growing or declining. This information can be used to inform decisions about which payment methods to prioritize or phase out in the future. Additionally, knowing the most used payment types can help the company to negotiate better deals with payment providers.

CONCLUSION

Data analysis and business intelligence are all about making decisions, but not the decisions that were made in the past, which heavily relied on experience, gut instinct, and intuition, but the decisions that are made today, which heavily rely on data and computational / mathematical sciences. Managerial decision-making is undergoing a paradigm shift as a result of the need to make quicker and better decisions in today's fiercely competitive business environment, the availability of large and feature-rich data sources, and advanced computing resources (both on the hardware and software side). Thus, by doing continuous analysis on the data, Olist can get numerous advantages to stay up to the top in this competitive market by providing the best service and getting maximum customer satisfaction.