



Data Analytics

BUSINESS ANALYSIS & CO: **A Data-Driven evaluation of Olist's performances**

MOMBO Lionnel

December, 2025

Table of Contents

Introduction	3
Data and data sources	4
Data cleaning and Exploratory data analysis	5
Visualizations	8
Database type selection	10
Entities. ERD.....	11
API.....	14
GDPR	17

Introduction

Business Use Case:

Brazil represents the largest e-commerce market in Latin America with a population of 213 million inhabitants. an ultra-connected population where the smartphone is largely ingrained in consumption habits. In this immense country with a unique geography and an economic disparity between the different segments of the population Brazil presents unique logistical and financial challenges. Olist (is a Brazilian e-commerce company founded on July 15 2015 in Curitiba) attempts to meet this challenge. his business model is a marketplace focused on providing technology solutions for e-commerce, primarily for small and medium-sized enterprises (SMEs) in Brazil. In this project, i try to identify some key indicators of the company.

Goal:

The goal of my project is to:

- Understand the company, his market and his model.
- Understand some key indicators that may drive his growth.
- Give suggestions in how to improve the performance of OLIST

High-level plan:

- Research about project topic
- Data collection
- Project scope
- Exploratory data analysis in Python (data wrangling, data cleaning)
- Visualization insights in Tableau
- Selection and creation of a database using MySQL
- Adding data to database and create Entity Relationship Diagram
- Data manipulation in SQL
- Data manipulation in Python
- Exposing data via API

Data and data sources

I have found a dataset on <https://www.kaggle.com> based on a popular Brazilian e-commerce: Olist Store. It's a public dataset of orders made at Olist.

The dataset has information of more than 100k orders from 2016 to 2018 made at multiple marketplaces in Brazil.

Its features allow viewing an order from multiple dimensions: from order status, price, payment type, payment, freight, customer location, product attributes.

This is real commercial data, it has been anonymized.

This dataset, i used for my project, called “Brazilian E-Commerce Public Dataset by Olist” from Kaggle, is composed of 5 sheets in csv format:

- Olist_customers_dataset
- Olist_order_items_dataset
- Olist_order_payments_dataset
- Olist_orders_dataset
- Olist_sellers_dataset

Data cleaning and Exploratory data analysis

Data cleaning and Exploratory Data Analysis (EDA) was conducted on the 5 dataframes in Python.

The datasets has been integrated into Python using libraries like Pandas, Numpy ...

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

import scipy.stats as st

df_customers = pd.read_csv(r"C:\Users\mombo\OneDrive\Desktop\datas\Brazilian E-Commerce Public Dataset by Olist\olist_customers_dataset all.csv\olist_customers_dataset.csv")
df_order_items = pd.read_csv(r"C:\Users\mombo\OneDrive\Desktop\datas\Brazilian E-Commerce Public Dataset by Olist\olist_customers_dataset all.csv\olist_order_items_dataset.csv")
df_order_payments = pd.read_csv(r"C:\Users\mombo\OneDrive\Desktop\datas\Brazilian E-Commerce Public Dataset by Olist\olist_customers_dataset all.csv\olist_order_payments_dataset.csv")
df_orders = pd.read_csv(r"C:\Users\mombo\OneDrive\Desktop\datas\Brazilian E-Commerce Public Dataset by Olist\olist_customers_dataset all.csv\olist_orders_dataset.csv")
df_sellers = pd.read_csv(r"C:\Users\mombo\OneDrive\Desktop\datas\Brazilian E-Commerce Public Dataset by Olist\olist_customers_dataset all.csv\olist_sellers_dataset.csv")
```

Each dataframe was checked:

```
# df_customers
# df_order_items
# df_order_payments
# df_orders
# df_sellers
```

- Checking number of observations

```
df_customers.shape
✓ 0.0s
(99441, 5)
```

- Checking for informations

```
df_sellers.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3095 entries, 0 to 3094
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   seller_id        3095 non-null   object  
 1   seller_zip_code_prefix  3095 non-null   int64  
 2   seller_city       3095 non-null   object  
 3   seller_state      3095 non-null   object  
dtypes: int64(1), object(3)
memory usage: 96.8+ KB
```

- Checking for composition of column

```
df_sellers["seller_state"].value_counts().head(2)

✓ 0.0s

seller_state
SP    1849
PR     349
Name: count, dtype: int64
```

```
df_sellers["seller_state"].nunique()

✓ 0.0s

23
```

- Handling null values

```
df_sellers.isna().sum()
✓ 0.0s

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

- Checking for duplicates

```
df_sellers.duplicated().sum(),  
✓ 0.0s  
(np.int64(0),)
```

- Complete statistical column summary

```
df_sellers.describe()  
✓ 0.0s  


|       | seller_zip_code_prefix |
|-------|------------------------|
| count | 3095.000000            |
| mean  | 32291.059451           |
| std   | 32713.453830           |
| min   | 1001.000000            |
| 25%   | 7093.500000            |
| 50%   | 14940.000000           |
| 75%   | 64552.500000           |
| max   | 99730.000000           |

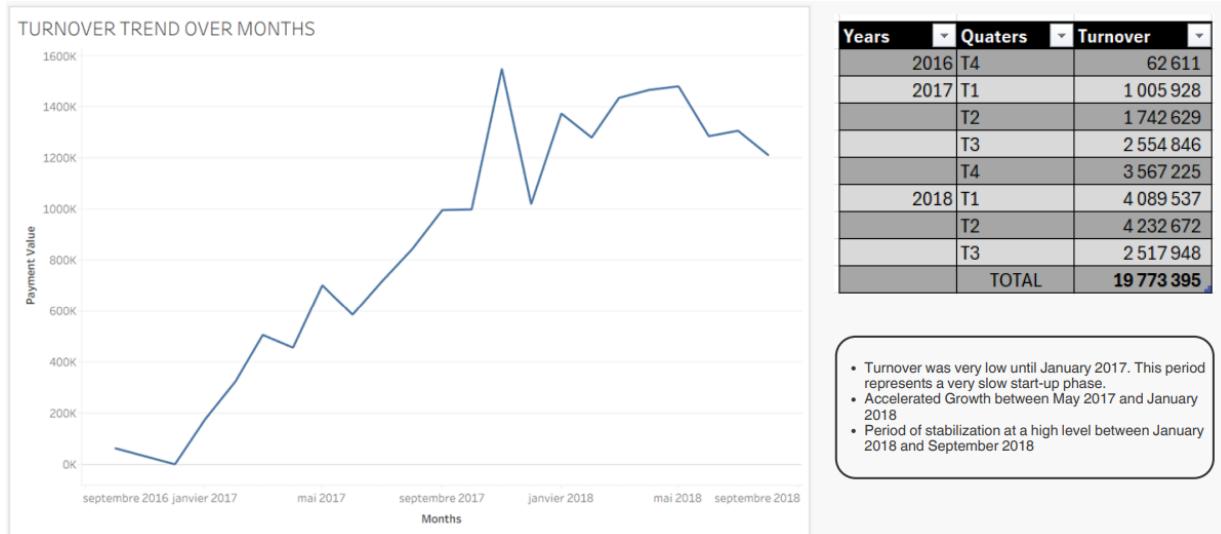

```

- Convert the specified columns to datetime format

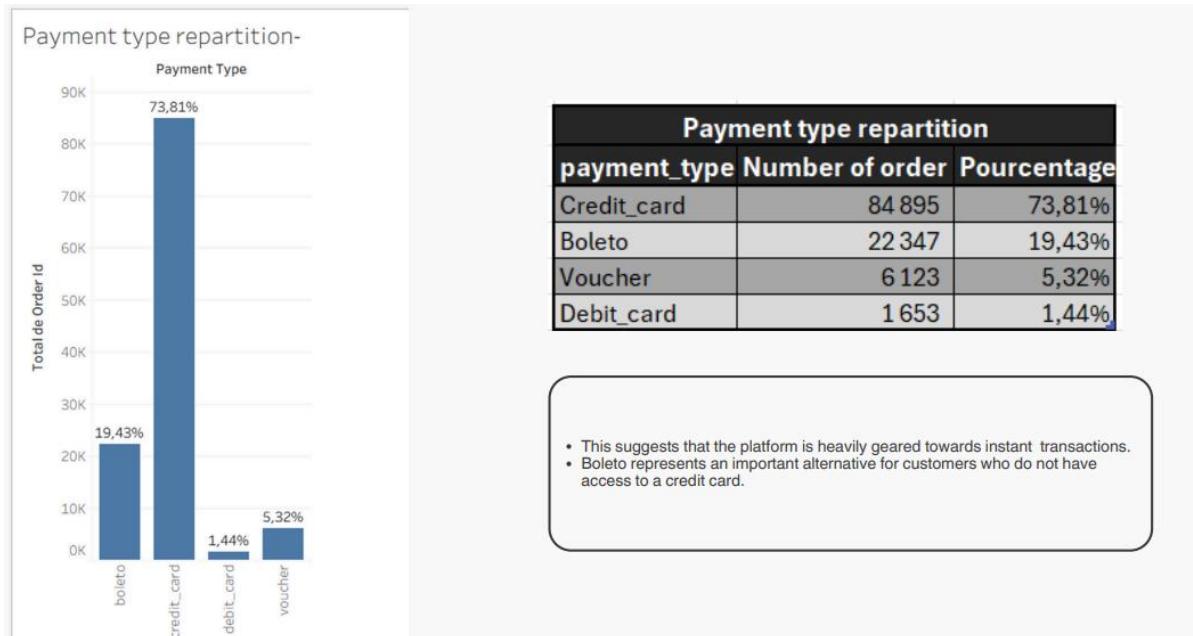
```
#specify date columns  
date_columns = ['order_purchase_timestamp', 'order_approved_at', 'order_delivered_carrier_date',  
|   |   |   |   'order_delivered_customer_date', 'order_estimated_delivery_date', 'shipping_limit_date']  
  
# Convert the specified columns to datetime format  
for col in date_columns:  
|   df_all[col] = pd.to_datetime(df_all[col])  
✓ 0.1s
```

Visualizations

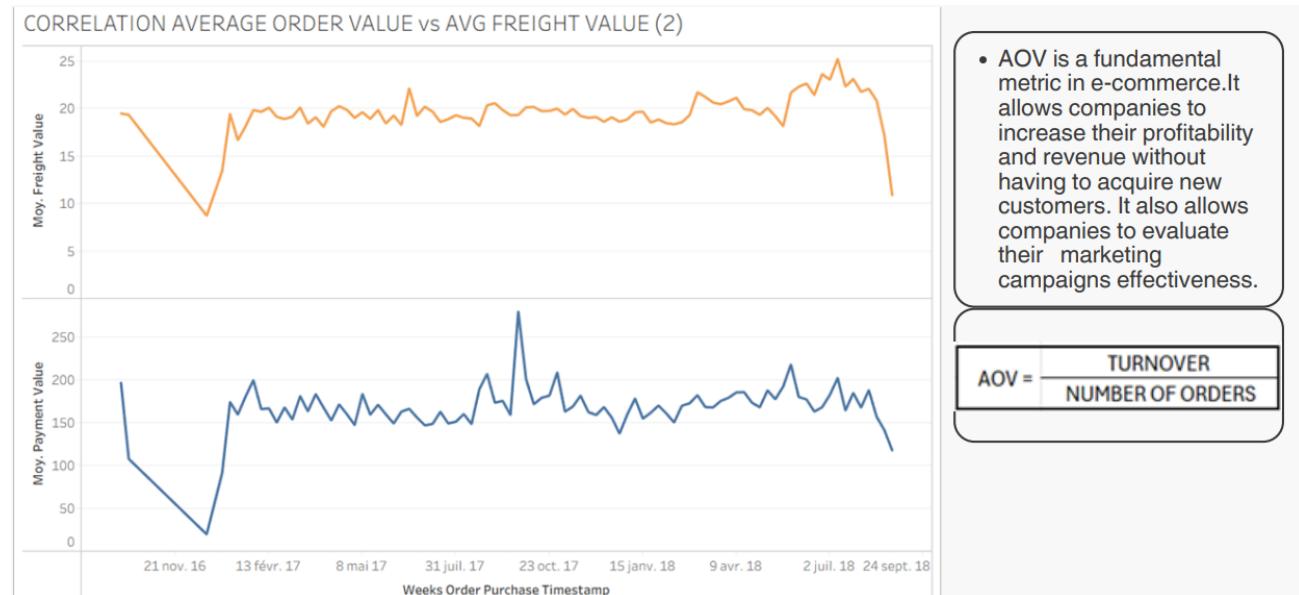
TURNOVER TREND OVER MONTHS



PAYMENT TYPE REPARTITION



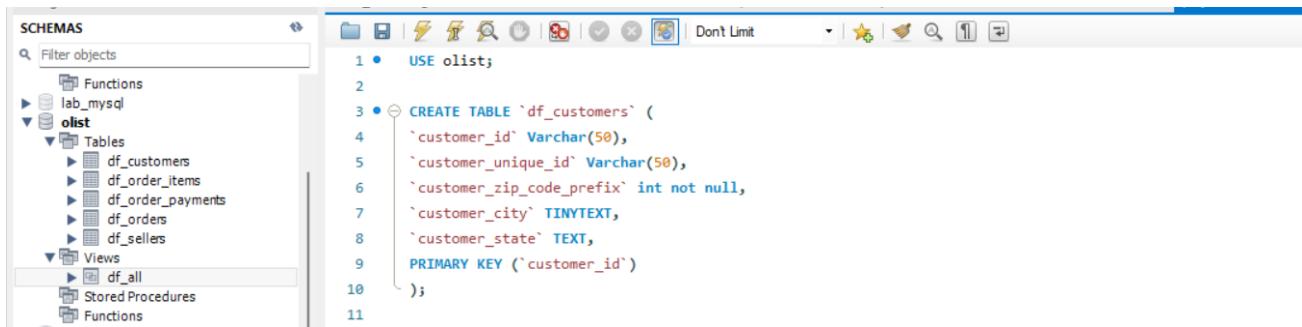
CORRELATION AVERAGE ORDER VALUE vs AVG FREIGHT VALUE



Database type selection

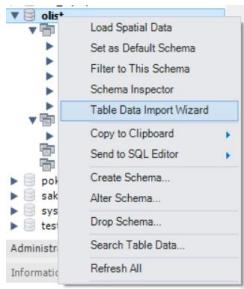
Once the data has been cleaned using Python, I sat up the database design phase, focusing on organization and normalization to ensure efficient data exploitation. I used MySql workbench.

To integrate data into MySql workbench, I created the database Olist with 5 tables.

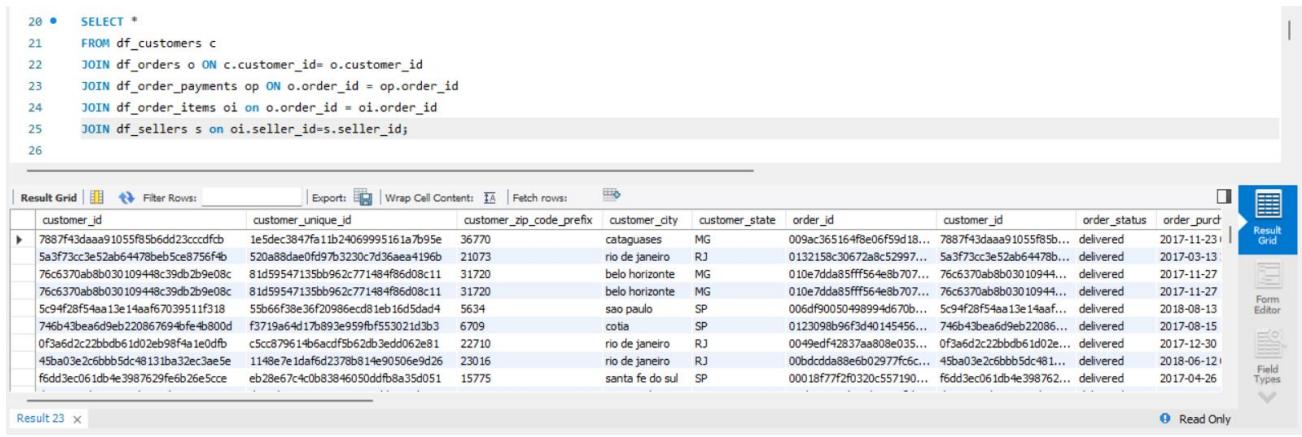


```
1 • USE olist;
2
3 • ◇ CREATE TABLE `df_customers` (
4   `customer_id` Varchar(50),
5   `customer_unique_id` Varchar(50),
6   `customer_zip_code_prefix` int not null,
7   `customer_city` TINYTEXT,
8   `customer_state` TEXT,
9   PRIMARY KEY (`customer_id`)
10 );
11
```

I imported data using the function Create table and the button Table Data Import Wizard.



Aggregation tables were created, as well as the relationships between the tables, using queries:

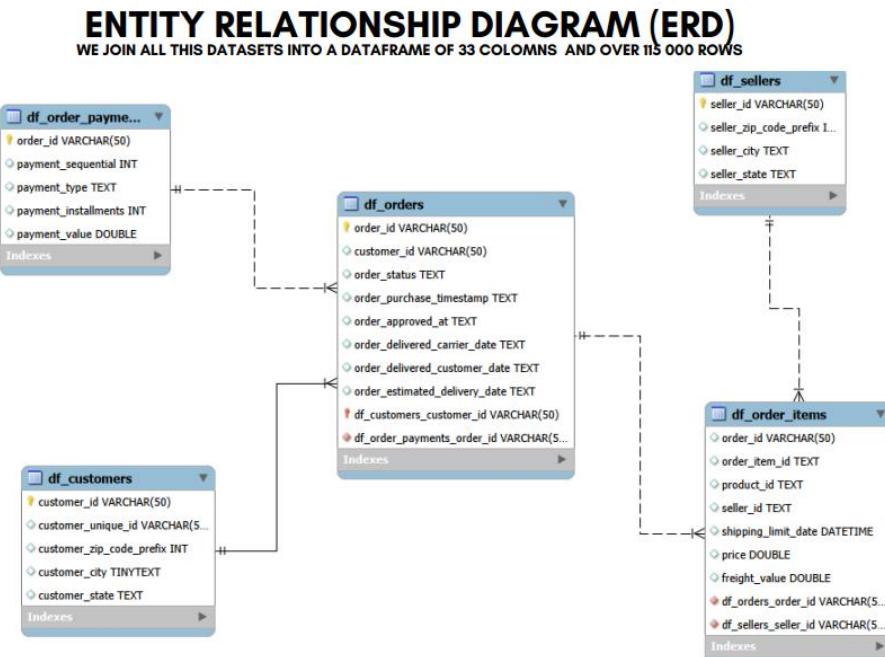


```
20 • SELECT *
21   FROM df_customers c
22   JOIN df_orders o ON c.customer_id= o.customer_id
23   JOIN df_order_payments op ON o.order_id = op.order_id
24   JOIN df_order_items oi ON o.order_id = oi.order_id
25   JOIN df_sellers s ON oi.seller_ids.seller_id;
```

customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state	order_id	customer_id	order_status	order_purch
7887f43daaa91055f85b6dd23ccdfcb	1e5de5c3847fa1b24069995161a7b95e	36770	cataguases	MG	009ac365164f8e06f59d18...	7887f43daaa91055f85b...	delivered	2017-11-23
5a3f73cc3e52ab64478beb5ce8756f4b	520a88daae0fd97b3230c7d36aae4196b	21073	rio de janeiro	RJ	0132158c30672a8c52997...	5a3f73cc3e52ab64478b...	delivered	2017-03-13
76c6370ab8b030109448c39db2b9e08c	81d59547135bb962c77148f86d08c11	31720	belo horizonte	MG	010e7da85ff554e8b707...	76c6370ab8b03010944...	delivered	2017-11-27
76c6370ab8b030109448c39db2b9e08c	81d59547135bb962c77148f86d08c11	31720	belo horizonte	MG	010e7da85ff554e8b707...	76c6370ab8b03010944...	delivered	2017-11-27
5c94f28f54ea13e14aa67039511f318	55666f36c36f20986ecdd1eb16d5dad4	5634	sao paulo	SP	006df900504989946708...	5c94f28f54ea13e14aa...	delivered	2018-08-13
746b43bead9deb220867694fe4b800d	f3719a64d17b893e959fb553021d3b3	6709	cotia	SP	0123098b96f3d40145456...	746b43bead9deb22086...	delivered	2017-08-15
0fa6a2c22bbdb51d02e98f4a1e0df8	c5c879614b65acd9fb62db3eddd062e81	22710	rio de janeiro	RJ	0049edf42837aa808e035...	0fa6a2c22bbdb51d02e...	delivered	2017-12-30
45ba03e2c6bb5dc48131ba32ec3ae5e	1148e7edaf6d2378b1490506e9d26	23016	rio de janeiro	RJ	00bcdcd48e6b02977fc6c...	45ba03e2c6bb5dc481...	delivered	2018-06-12
f6dd3ec061db4e3987629fe6b26e5cce	eb2ae67c40b83846050udfb8a35d051	15775	santa fe do sul	SP	00018f77f2f0320c557190...	f6dd3ec061db4e398762...	delivered	2017-04-26

Entities. ERD

The following image represents the Entities Relationship Diagram (ERD-MYSQL) of the Brazilian e-commerce company olist used in my project.



Database's Queries

In order to Understand some key factors that may drive his growth, I conducted certain queries.

- Creating view: Aggregating multiple columns from different table

```

72 • CREATE VIEW df_all AS
73   SELECT
74     o.order_id,
75     o.order_purchase_timestamp,
76     o.order_approved_at,
77     o.order_delivered_customer_date,
78     op.payment_type,
79     op.payment_value,
80     oi.product_id,
81     oi.price,
82     oi.freight_value,
83     s.seller_id,
84     s.seller_city
85   FROM df_customers c
86   JOIN df_orders o ON c.customer_id = o.customer_id
87   JOIN df_order_payments op ON o.order_id = op.order_id
88   JOIN df_order_items oi ON o.order_id = oi.order_id
89   JOIN df_sellers s ON oi.seller_id = s.seller_id;
  
```

order_id	order_purchase_timestamp	order_approved_at	order_delivered_customer_date	payment_type	payment_value	product_id	price	freight_value
008d65b666158b633f96054d31af43b	2017-10-13 19:57:48	2017-10-17 04:24:35	2017-11-07 20:38:45	boleto	130.88	a0450529bd974ebeaf93b1731184396a	94.9	35.98
009ac365164f8e06f59d18a08045f6c4	2017-11-23 00:03:52	2017-11-23 00:11:24	2017-11-29 16:33:25	voucher	8.75	35557c68a22ecbf066e25ca2ddc144	16.9	15.1
019bdbaeef5b512c9b1811976aa510a5	2018-05-08 11:08:03	2018-05-10 03:36:39	2018-05-18 20:30:44	boleto	108.64	eb6c2cede53034fc9ec47741b3232c9d	35	19.32
019bdbaeef5b512c9b1811976aa510a5	2018-05-08 11:08:03	2018-05-10 03:36:39	2018-05-18 20:30:44	boleto	108.64	eb6c2cede53034fc9ec47741b3232c9d	35	19.32
0132158c30672a8c52997a492613ec2	2017-03-13 23:23:23	2017-03-13 23:23:23	2017-03-15 09:27:54	credit_card	413.31	e9eebb8e8ba0fadbf9020f8ba1c003b48	399.9	13.41

- Calculate turnover by year

```

90
91
92 • SELECT
93     YEAR(order_purchase_timestamp) AS year,
94     ROUND(SUM(payment_value),2) AS turnover
95
96     FROM
97     df_all
98     GROUP BY
99     YEAR(order_purchase_timestamp)
100    ORDER BY
101        year;
102

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

year	turnover
2016	62611.27
2017	8870627.51
2018	10840155.92

Result 6 ×

- Calculate turnover by city (the 5 highest revenues per city)

```

108 • SELECT
109     seller_city,
110     ROUND(SUM(payment_value), 2) AS turnover
111
112     FROM
113     df_all
114     GROUP BY
115     seller_city
116     ORDER BY
117         turnover DESC
118     LIMIT 5;
119

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

seller_city	turnover
sao paulo	4143574.49
ibitinga	1037016.06
curitiba	631504.72
itaquaquecetuba	571007.34
guarulhos	483201.68

- Calculate AVERAGE ORDER VALUE (AOV) by quarter

```

130 • SELECT
131     CONCAT(YEAR(order_purchase_timestamp), '-Q', QUARTER(order_purchase_timestamp)) AS QUARTER,
132     FORMAT(SUM(payment_value) / COUNT(DISTINCT order_id), 2, 'fr_FR') AS AOV
133
134     FROM
135     df_all
136     GROUP BY
137     QUARTER
138     ORDER BY
139         QUARTER;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

QUARTER	AOV
2016-Q4	231,04
2017-Q1	203,84
2017-Q2	193,99
2017-Q3	209,17
2017-Q4	206,45
2018-Q1	198,25
2018-Q2	215,48
2018-Q3	201,32

- Calculate average delivery time by year

```
141 •  SELECT
142      YEAR(order_purchase_timestamp) AS year,
143      ROUND(AVG(DATEDIFF(order_delivered_customer_date, order_approved_at)), 2) AS avg_delivery_time_days
144  FROM
145      df_all
146  WHERE
147      order_delivered_customer_date IS NOT NULL
148  GROUP BY
149      year
150  ORDER BY
151      year;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	year	avg_delivery_time_days
▶	2016	19.25
	2017	12.38
	2018	11.47

API

In order to expose portion of data from the database I have created an API that allows users to retrieve specific data.

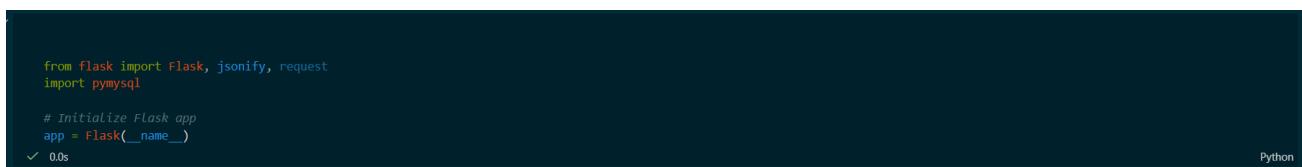
I used FLASK for creating this REST APIs. It's a good choice for transforming a Python script into a web service capable of receiving and sending data (often in JSON format).

- Installe Flask and Pymysql



```
pip install flask
pip install pymysql
```

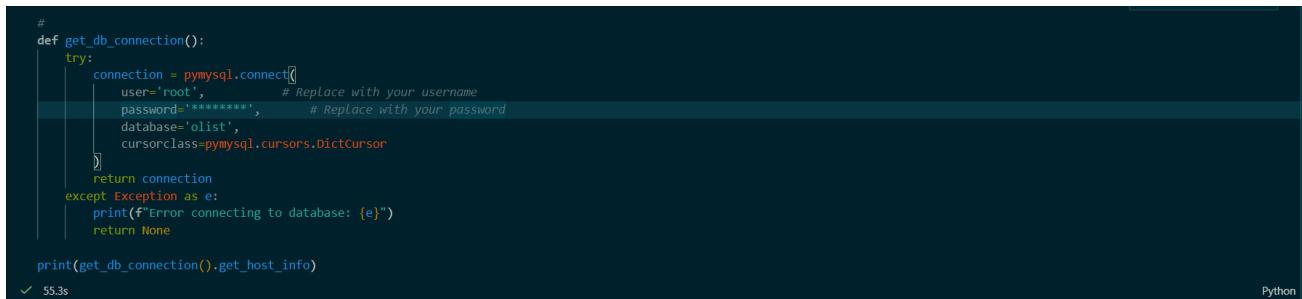
- Initialize Flask app



```
from flask import Flask, jsonify, request
import pymysql

# Initialize Flask app
app = Flask(__name__)
✓ 0.0s
```

- MySql database connector



```
# def get_db_connection():
try:
    connection = pymysql.connect(
        user='root',           # Replace with your username
        password='*****',      # Replace with your password
        database='olist',
        cursorclass=pymysql.cursors.DictCursor
    )
    return connection
except Exception as e:
    print(f"Error connecting to database: {e}")
    return None

print(get_db_connection().get_host_info)
✓ 55.3s
```

- Available Endpoints



```
# Define API endpoint to get information on customers

@app.route("/customers", methods=["GET"])
def get_customers():
    connection = get_db_connection()
    if not connection:
        return jsonify({"error": "Failed to connect to database."}), 500

    with connection.cursor() as cursor:
        query = "SELECT customer_id, customer_city, customer_zip_code_prefix, customer_state FROM olist.df_customers;"
        cursor.execute(query)
        customers = cursor.fetchall()

    connection.close()
    return jsonify(customers)
✓ 55.3s
```

```

# Define API endpoint to get information on order items

@app.route("/order_items", methods=["GET"])
def get_items():
    connection = get_db_connection()
    if not connection:
        return jsonify({"error": "Failed to connect to database."}), 500

    with connection.cursor() as cursor:
        query = "SELECT order_id, product_id, seller_id, price, freight_value FROM olist.df_order_items;"
        cursor.execute(query)
        items = cursor.fetchall()

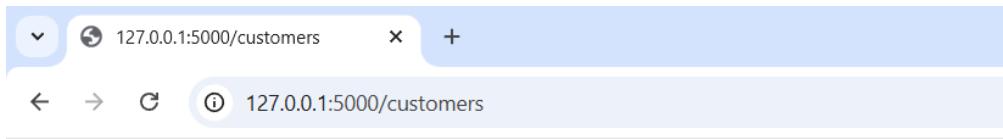
    connection.close()
    return jsonify(items)

55.3s

```

Python

• Result of Flask requests's



The screenshot shows a browser window with the URL `127.0.0.1:5000/customers`. The page content is a JSON array representing customer data:

```

[
  {
    "customer_city": "osasco",
    "customer_id": "00012a2ce6f8dcda20d059ce98491703",
    "customer_state": "SP",
    "customer_zip_code_prefix": 6273
  },
  {
    "customer_city": "itapecerica",
    "customer_id": "000161a058600d5901f007fab4c27140",
    "customer_state": "MG",
    "customer_zip_code_prefix": 35550
  },
  {
    "customer_city": "nova venecia",
    "customer_id": "0001fd6190edaaf884bcf3d49edf079",
    "customer_state": "ES",
    "customer_zip_code_prefix": 29830
  },
  {
    "customer_city": "mendonca",
    "customer_id": "0002414f95344307404f0ace7a26f1d5",
    "customer_state": "MG",
    "customer_zip_code_prefix": 39664
  },
  {
    "customer_city": "sao paulo",
    "customer_id": "000379cdec625522490c315e70c7a9fb",
    "customer_state": "SP",
    "customer_zip_code_prefix": 4841
  },
  {
    "customer_city": "valinhos",
    "customer_id": "0004164d20a9e969af783496f3408652",
    "customer_state": "SP",
    "customer_zip_code_prefix": 13272
  },
  {
    "customer_city": "niteroi",
    "customer_id": "000419c5494106c306a97b5635748086",
    "customer_state": "RJ",
    "customer_zip_code_prefix": 24220
  }
]

```

127.0.0.1:5000/order_items

Impression élégante □

```
[  
 {  
   "freight_value": 13.29,  
   "order_id": "00010242fe8c5a6d1ba2dd792cb16214",  
   "price": 58.9,  
   "product_id": "4244733e06e7ecb4970a6e2683c13e61",  
   "seller_id": "48436dade18ac8b2bce089ec2a041202"  
 },  
 {  
   "freight_value": 19.93,  
   "order_id": "00018f77f2f0320c557190d7a144bdd3",  
   "price": 239.9,  
   "product_id": "e5f2d52b802189ee658865ca93d83a8f",  
   "seller_id": "dd7ddc04e1b6c2c614352b383efe2d36"  
 },  
 {  
   "freight_value": 17.87,  
   "order_id": "000229ec398224ef6ca0657da4fc703e",  
   "price": 199.0,  
   "product_id": "c777355d18b72b67abbeef9df44fd0fd",  
   "seller_id": "5b51032eddd242adc84c38acab88f23d"  
 },  
 {  
   "freight_value": 12.79,  
   "order_id": "00024acbcdf0a6daa1e931b038114c75",  
   "price": 12.99,  
   "product_id": "7634da152a4610f1595efa32f14722fc",  
   "seller_id": "9d7a1d34a5052409006425275ba1c2b4"  
 },  
 {  
   "freight_value": 18.14,  
   "order_id": "00042b26cf59d7ce69dfabb4e55b4fd9",  
   "price": 199.9,  
   "product_id": "ac6c3623068f30de03045865e4e10089",  
   "seller_id": "df560393f3a51e74553ab94004ba5c87"  
 },  
 {  
   "freight_value": 12.69,  
   "order_id": "00048cc3ae777c65dbb7d2a0634bc1ea",  
   "price": 21.9,  
   "product_id": "ef92defde845ab8450f9d70c526ef70f",  
   "seller_id": "6426d21aca402a131fc0a5d0960a3c90"  
 },
```

GDPR

All datasets used in this project are exclusively public. The aggregated information ensures that no individual can be identified, maintaining strict data anonymity. As no personally identifiable information (PII) was utilized, the project strictly adheres to General Data Protection Regulation (GDPR) standards.

GITHUB:

https://github.com/mombolionnel-max/Project-_Olist_L.MOMBO.git

THANK YOU