

Final Project Documentation



Brazilian E-Commerce Public Dataset by Olist

By Derrick Ee, Nur Ain Binte Jamali, Randall Villasenor, Lee Kin Meng

Table of Contents

Introduction.....	3
Overview.....	3
Problem Statement.....	3
Timeline.....	4
Data Sources.....	4
Data Dictionary.....	5
Methodology/Approach.....	7
Tech Stack/Tools.....	7
ETL Process.....	7
Extraction.....	7
Transformations.....	7
Loading.....	8
Architecture Diagram.....	9
Database Schema/Entity-Relationship Diagram.....	10
Key Relationships between Tables.....	10
Data Validation.....	11
Challenges/Limitations.....	11
Challenges.....	11
Limitations.....	12
Findings/Analysis.....	12
Process.....	12
Key Results.....	13
Insights.....	13
City-Level Aggregation.....	14
Category-Level Aggregation.....	15
Business Recommendations.....	15
Simplifications and Focus.....	16
Conclusion.....	16
What's Next.....	17
Key Learnings.....	18
References.....	19

Introduction

Overview

The team embarked on this project to explore the Brazilian E-Commerce dataset by Olist — a rich yet unstructured dataset that reflects the real-world messiness of online retail. Our aim was to build a robust data pipeline that could transform raw transactional data into clear, actionable business insights.

We focused on cleaning, connecting, and preparing the data for visual storytelling through Power BI. With team members bringing diverse strengths — from ETL (Extract, Transform, Load) scripting to dashboard design — we collectively tackled a full ETL process, culminating in an interactive business intelligence dashboard hosted on Microsoft Fabric.

Problem Statement

E-commerce datasets, while valuable, are rarely clean or analysis-ready. For instance, the Olist dataset was scattered across multiple tables, riddled with nulls and inconsistencies, and required significant data wrangling before insights could be drawn.

Our problem was twofold:

- (1) Structure the data into a unified, analytics-ready model
- (2) Use it to explore key business questions, notably around seller performance and customer payment behavior

Objective

Our objective was to implement a business intelligence dashboard that evaluates seller performance and payment trends in the Olist marketplace, using data-driven Key Performance Indicators (KPIs) to support both strategic oversight and operational improvement.

Timeline

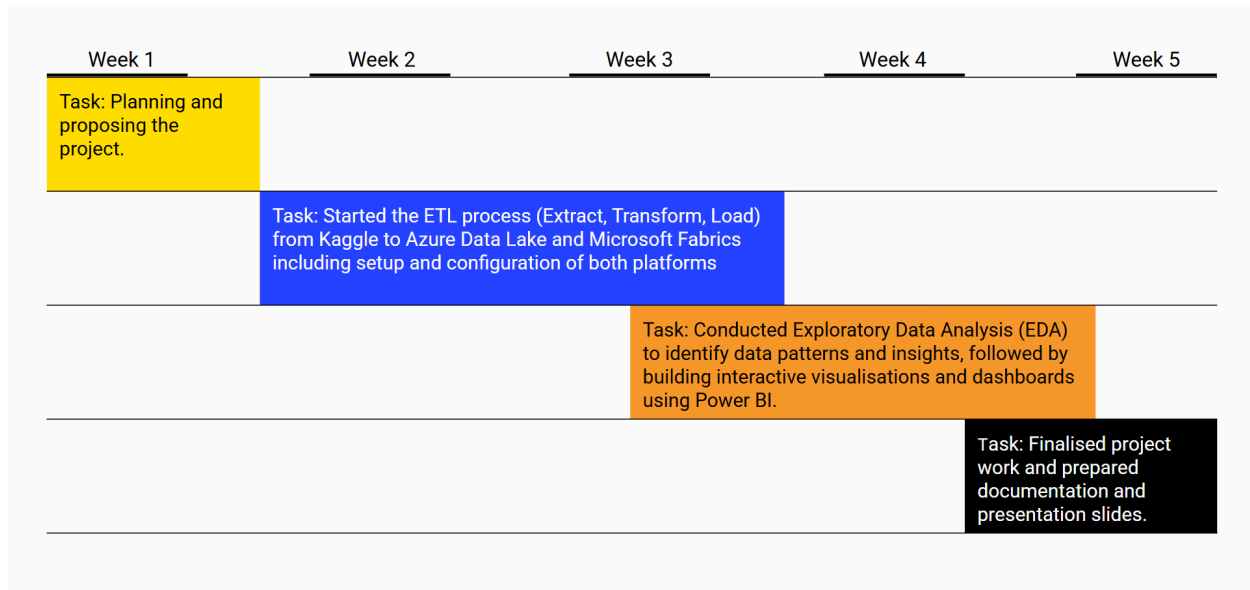


Figure 1: Project Timeline

Data Sources

The project utilises the following data sources:

1. Brazilian E-Commerce Public Dataset by Olist from Kaggle. This is the original source of the datasets.
2. Kaggle API is used to download the dataset files programmatically.
3. The raw data files are stored in Azure Data Lake Storage Gen2 under the "raw-data" container.

Data Dictionary

Figure 2 shows the field names and descriptions taken from the nine Comma Separated Values (CSV) files of the dataset.

Field Name	Description
customer_city	City of the customer.
customer_id	Unique identifier for a customer related to an order.
customer_state	State of the customer.
customer_unique_id	Unique identifier for a customer (may have multiple customer_ids).
customer_zip_code_prefix	First 5 digits of the customer's zip code.
freight_value	Item freight value.
geolocation_lat	Latitude coordinate for the zip code prefix.
geolocation_lng	Longitude coordinate for the zip code prefix.
geolocation_zip_code_prefix	First 5 digits of the zip code in the geolocation data.
order_approved_at	Timestamp of the order approval.
order_delivered_carrier_date	Timestamp of the order handover to the carrier.
order_delivered_customer_date	Timestamp of the actual order delivery to the customer.
order_estimated_delivery_date	The estimated delivery date calculated by Olist.
order_id	Unique identifier for an order.
order_item_id	Sequential number that identifies items within the same order.
order_purchase_timestamp	Timestamp of the order purchase.
order_status	The current status of the order (e.g., 'delivered', 'shipped').
payment_installments	Number of installments for the payment.
payment_sequential	A sequence of payments for the same order.
payment_type	Method of payment (e.g., 'boleto', 'credit_card').
payment_value	Total value of the payment transaction.

price	Item price.
product_category_name	Product category name in Portuguese.
product_category_name_english	Product category name in English (translation).
product_description_length	Number of characters in the product description.
product_height_cm	Product height in centimeters.
product_id	Unique identifier for a product.
product_length_cm	Product length in centimeters.
product_name_length	Number of characters in the product name.
product_photos_qty	Number of product photos.
product_weight_g	Product weight in grams.
product_width_cm	Product width in centimeters.
review_answer_timestamp	Timestamp of the review answer.
review_comment_message	Comment message in the review.
review_creation_date	Timestamp of the review creation.
review_id	Unique review identifier.
review_score	Score given by the customer in the review.
seller_city	City of the seller.
seller_id	Unique identifier for a seller.
seller_state	State of the seller.
seller_zip_code_prefix	First 5 digits of the seller's zip code.
shipping_limit_date	The date and time when the seller should ship the item.

Figure 2: Field names and their respective descriptions

Methodology/Approach

Tech Stack/Tools

- Programming language: Python is used for data extraction, cleaning, processing, and initial analysis.
- Libraries:
 - Data Manipulation/Analysis: pandas, numpy, scipy.stats, statsmodels.stats.proportion, scikit_posthocs, kruskal
 - Cloud Storage Interaction: azure-storage-file-datalake, azure-storage-blob, azure-core, azure-identity.
 - Database Connection: sqlalchemy, psycopg2-binary (mentioned as needed).
 - Data Extraction: kaggle, kagglehub
 - Data Serialisation: pyarrow, parquet
 - Visualisation: matplotlib.pyplot, seaborn, plotly.io, plotly.express, plotly.graph_objects, folium, wordcloud
 - Other Utilities: os, pathlib, io, IPython.display, re, warnings
- Database: PostgreSQL (accessed via sqlalchemy)
- Cloud Platform: Microsoft Azure (Data Lake Storage Gen2, Blob Storage).
- Data Platform: Microsoft Fabric (Lakehouse, Warehouse, Semantic model, Reports, Notebooks)

ETL Process

Extraction

- Data is extracted from Kaggle.
- The Kaggle API is used to download the dataset files programmatically.
- Downloaded raw data files are loaded into Azure Data Lake Storage Gen2 under the "raw-data" container. Standard Olist dataset filenames are renamed to simpler names during this upload.

Transformations

- Data cleaning and transformation steps are performed using Python and the pandas library.
- Sellers data: Checked for missing values and duplicates (seller_id). None were found. Data types verified.

- Order Items data: Loaded into DataFrame, structure and types checked. Basic statistics. No nulls were found during a check in PostgreSQL after cleaning.
- Orders data: Loaded into DataFrame, structure and types checked. Missing values were identified for order_approved_at, order_delivered_carrier_date, and order_delivered_customer_date.
- Payments data: Loaded into DataFrame, shape, columns, dtypes checked. No duplicate rows were found.
- Reviews data: Loaded into DataFrame, shape, columns, dtypes checked. No duplicate rows were found. The review_comment_title column was dropped.
- Products data: Loaded into DataFrame. Missing values in numerical columns were imputed with the column median. Rows with invalid measurements (≤ 0 for weight, length, height, width) were removed. Columns were converted to appropriate data types.
- Product Category Name Translation data: Loaded into DataFrame. Missing values checked (none > 0). Duplicates checked on product_category_name_english (none found).
- Customers data: Downloaded from Azure. A custom function, custom_title, was defined for applying Portuguese title case rules to text fields (though commented out in the excerpt), suggesting potential text cleaning for city/state names.
- Geolocation data: Imputation of missing geolocation_lat and geolocation_lng values with the median was performed *after* merging the datasets.

Loading

- Cleaned data is intended to be loaded into Azure Data Lake under the "transform-data" folder.
- The cleaned order_items dataset was saved locally as clean_order_items.csv.
- Cleaned payments and reviews DataFrames were converted to Parquet files (clean_payments.parquet, clean_reviews.parquet) and uploaded to Azure Blob Storage.
- The cleaned orders DataFrame was uploaded to a PostgreSQL database named 'DARK' as the table 'clean_orders'.

Architecture Diagram

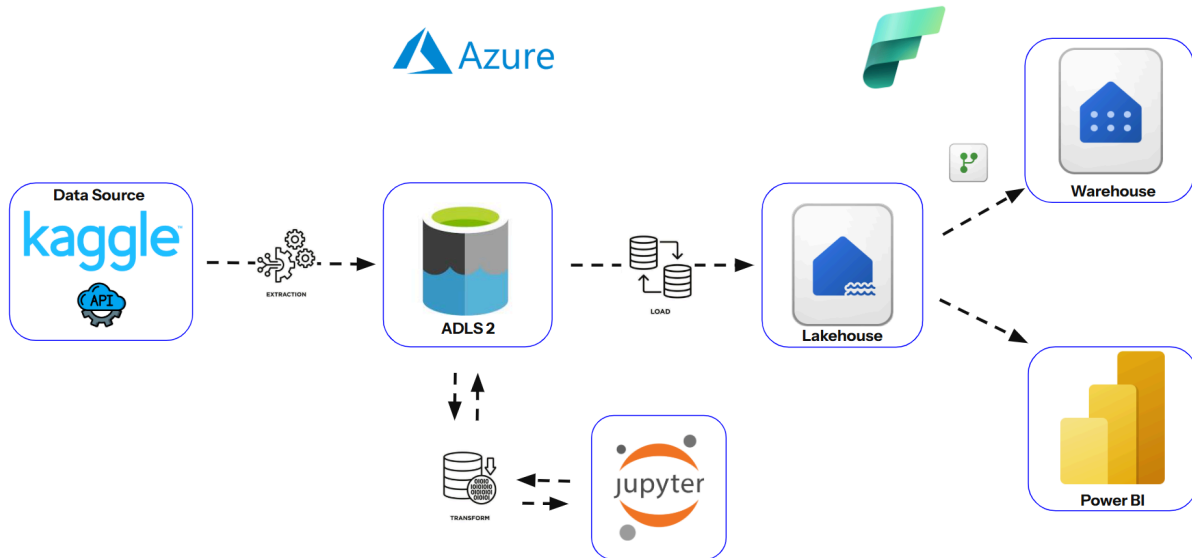


Figure 3: Architecture diagram

- Source: Kaggle (Brazilian E-Commerce Public Dataset).
- Extraction: Kaggle API is used to download data.
- Staging/Raw Layer: Data is loaded into the "raw-data" container in Azure Data Lake Storage Gen2.
- Transformation Layer: Python scripts using pandas notebooks perform cleaning, transformation, and merging. These scripts likely run on a platform capable of interacting with Azure.
- Cleaned/Transformed Layer: Cleaned data is saved back to Azure Data Lake under the "transform-data" folder in Parquet format. Some cleaned tables are also loaded into a PostgreSQL database.
- Consumption Layer: The Microsoft Fabric for analysis, including Lakehouse, Warehouse, Semantic Models, and Reports. PostgreSQL might also serve as a consumption layer.

Database Schema/Entity-Relationship Diagram

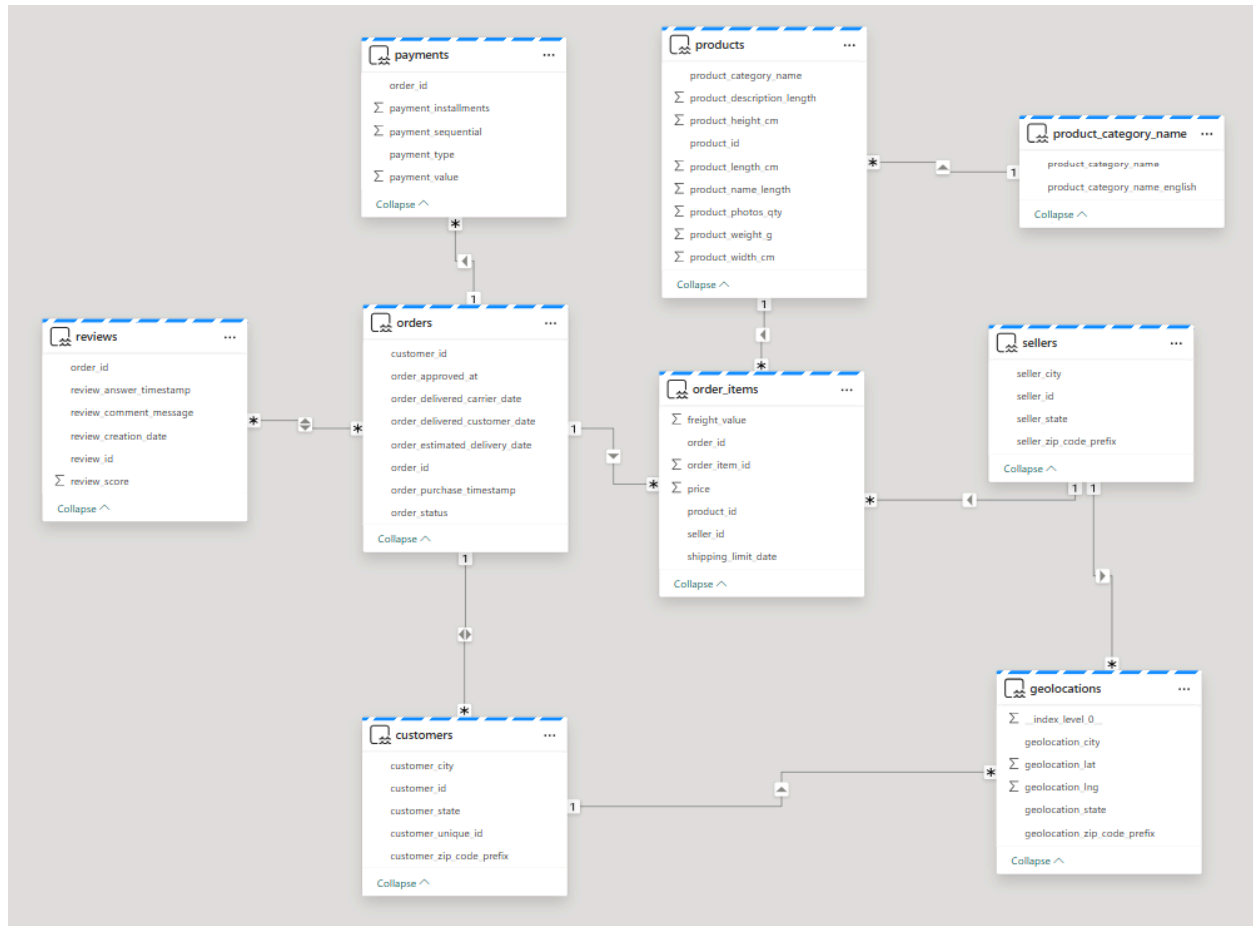


Figure 4: Entity-relationship diagram

Key Relationships between Tables

- orders and order_items are linked by order_id.
- orders and reviews are linked by order_id.
- orders and payments are linked by order_id.
- orders and customers are linked by customer_id.
- order_items and products are linked by product_id.
- order_items and sellers are linked by seller_id.
- customers and geolocation are likely linked by customer zip code prefix.
- sellers and geolocation are likely linked by seller zip code prefix.
- products and product_category_name_translation are linked by product_category_name.

Data Validation

- **Missing Value Analysis:** Checked for missing values in individual datasets like sellers, orders, and product categories. Missing numerical values in products were imputed with the median. After merging, missing values were re-checked across the combined dataset. Missing geolocation coordinates were imputed with the median in the merged data. A specific SQL query was used to check for NULLs in the cleaned order_items table in PostgreSQL.
- **Duplicate Check:** Checked for duplicate records based on key columns like seller_id, and checked for duplicates in the payments and reviews tables and product categories (product_category_name_english). No duplicates were found in these checks.
- **Data Type and Structure Check:** Used .info() and .dtypes to inspect column data types and overall DataFrame structure for several datasets. Conversions to appropriate types were performed, especially for numeric and potentially datetime columns.
- **Invalid Data Removal:** Rows with non-positive values for product measurements (weight, length, height, width) were removed.
- **Sample Data Review:** Used .head(), display(), and SQL queries to preview sample records and understand the data.

Challenges/Limitations

Challenges

1. **Package Installation and Dependency Conflicts:** The installation logs show various warnings and errors related to package versions, deprecation of older packages like the 'azure' meta-package, and issues during the build process (e.g., setuptools warnings about configuration outside pyproject.toml, problems finding packages like thriftpy2.transport.memory and thriftpy2.transport.sasl). These required addressing to set up the environment.
2. **Handling Missing Values Post-Merge:** After merging the various datasets, a significant number of missing values were introduced in several columns, including item/product/seller details (order_item_id, product_id, seller_id, shipping_limit_date, price, freight_value, product_category_name, product measurements), date fields (order_approved_at, order_delivered_carrier_date, order_delivered_customer_date), review details, payment details, and geolocation zip code. Imputing missing geolocation coordinates (geolocation_lat, geolocation_lng) with the median was implemented as a solution for those specific fields.
3. **Data Type Inconsistencies/Management:** Converting columns to appropriate data types (e.g., datetime, numeric) was necessary. The sources show timestamp columns initially being read as object type in some instances, requiring explicit conversion steps not fully detailed in the provided excerpts, although the orders information shows datetime64[ns] after cleaning.

Limitations

1. Scope of Analysis: The provided sources primarily focus on the ETL process and initial data exploration (EDA). They do not contain detailed findings, in-depth analysis, or business recommendations derived *from* the cleaned data. The "Findings/Analysis" section in the sources only covers basic statistics and distribution previews.
2. Imputation Strategy: For some fields, like missing numerical product attributes and geolocation coordinates, the median was used for imputation. This is a simplification and might not be the most suitable strategy, depending on the data distribution and the requirements of downstream analysis, potentially introducing bias.
3. Geolocation Precision: The geolocation data is linked by zip code prefix, not the full zip code, and there can be multiple latitude/longitude pairs for a single prefix. Imputing missing coordinates with the median of *all* geolocation data (or the existing coordinates in the merged data) is a broad approximation.

Findings/Analysis

Exploratory Data Analysis (EDA) steps were performed to understand the data after cleaning and merging.

Process

Several methods were used in the initial data exploration, as shown in Figure 5:

Method	Function
<code>head([n])</code>	Returns the first n rows.
<code>info([verbose, buf, max_cols, memory_usage, ...])</code>	Prints a concise summary of a DataFrame.
<code>describe([percentiles, include, exclude])</code>	Generates descriptive statistics.
<code>value_counts([subset, normalize, sort, ...])</code>	Returns a Series containing the frequency of each distinct row in the DataFrame.

Figure 5: Table describing the methods and functions used in the EDA

Only the first 1000 rows were loaded while analysing all nine datasets. Additionally, several visualisations, such as histograms, heatmaps, and scatterplots, were used.

Key Results

- Initial analysis of individual datasets revealed their shapes, column types, and basic statistics.
- Datasets like sellers and product categories were found to have no missing or duplicate values on key identifiers.
- Other datasets, like orders and reviews, contained missing values, particularly in date/time and comment fields.
- The distribution of order statuses was examined, showing a large proportion of 'delivered' orders.
- Relationships between geolocation features like zip code prefix, latitude, and longitude were visualised using scatter and density plots and correlation matrices. (Refer to Figures 6 and 7 in the Appendices section.)
- The merging process successfully combined data from multiple sources into a single DataFrame.

Insights

Based on the EDA and cleaning process:

- The data quality varies across datasets, requiring specific cleaning steps for each source.
- String cleaning, especially for city and state names, is crucial for standardising location data across datasets. Custom functions were needed for nuances, such as Portuguese capitalisation rules.
- Missing values are prevalent in date/time fields and review comments, requiring strategies like imputation or flagging, depending on the analysis goal.
- The final merged dataset provides a comprehensive view of orders, including items, payments, reviews, customer, seller, and product information, enabling holistic analysis.

Seller Reliability (Median-Based Composite Score)

Objective: Evaluate seller behavior through a composite reliability score, aggregating performance metrics such as customer reviews, fulfillment rate, and delivery punctuality.

Initially, we calculated the **Seller Reliability Score** using:

- **Average review scores**
- **A static 14- or 21-day benchmark** for determining delivery delays

However, in our latest iteration, we made two key improvements:

1. **Switched from AVERAGE to MEDIAN** for seller-level calculations
 - This minimizes distortion from extreme values and offers a fairer representation,

especially for sellers with fewer transactions or inconsistent scores.

2. **Used Olist's own** `estimated_delivery_date` to define delays

➤ Instead of assuming all orders should be fulfilled within a set timeframe, we now compare actual delivery dates (via `order_delivered_customer_date` or `order_delivered_carrier_date`) against what Olist promised.

This framework offers a fair and normalized view of seller performance across Olist's marketplace. While most sellers perform consistently well due to centralized processes, the score highlights subtle but meaningful differences when they occur.




We began by constructing a master table that brought together reviews, order items, sellers, and delivery data. From this unified base, we created the following seller-level KPIs:

- Median Review Score — Chosen over average to avoid distortion from extreme values. This ensures fairness, especially for cities or product categories with few sellers or outlier reviews.
- Fulfillment Rate — % of orders delivered by the seller.
- Delay Rate — % of orders delivered later than `expected_delivery_date`.
- Seller Reliability Score

Composite Score Formula:

DAX:

$(0.5 * (\text{Median Review Score} / 5)) +$
 $(0.3 * \text{Fulfillment Rate}) +$
 $(0.2 * (1 - \text{Delay Rate}))$

- Seller Tiering — Classifies each seller as  High (≥ 0.80),  Moderate (≥ 0.60),  Low (< 0.60) based on their reliability scores.

City-Level Aggregation

We further aggregated seller metrics by city, calculating the median values for:

- `city_med_review_score`
- `city_med_fulfillment_rate`
- `city_med_delay_rate`

We then applied the composite score logic above to compute:

- `avg_seller_reliability_by_city`

These values were used to assign each city a **service tier**, based on the overall performance of its sellers.

***Note: Whether we used MEDIANX or AVERAGEX, the results showed minimal variation between cities, due to highly standardized performance across Olist's seller network.*

This consistency validates our composite score approach — even if score differences are small, they're still meaningful when used in ranking and prioritizing seller interventions.

Category-Level Aggregation

We also joined product categories into the master table to analyse seller behavior by product type. Using a similar approach, we built:

- category_med_review_score
- category_med_fulfillment_rate
- category_med_delay_rate
- category_med_reliability_score

These were used to assign a Category Tier, helping the business identify which product areas face more operational challenges.

Business Recommendations

The seller performance analysis highlights a critical opportunity. While overall seller reliability across Olist's marketplace is high and consistent, a noticeable dip in review scores and a modest rise in delivery delays were observed during peak sales periods such as Q4 (e.g. Black Friday). This suggests that even reliable sellers face logistical strain during high-demand seasons. Furthermore, consistently low scores in early 2018, despite steady revenue, may indicate onboarding or operational readiness issues.

Based on these insights, the following business recommendations are proposed:

1. Maintain and Strengthen Seller Performance During Peak Seasons

- **Implement a Flash Sale Readiness Program:**
Equip sellers with Standard Operating Procedures (SOPs), inventory management support, and performance checklists ahead of Q4 campaigns.
- **Prioritise intervention for high-volume sellers in lower tiers:**
Provide coaching and logistical aid to sellers contributing significant revenue but showing signs of delivery delays or inconsistent reliability.

- **Preserve customer experience through proactive enablement:**
Shift from reactive penalty to early-stage support, especially for moderate-tier sellers who may struggle under Q4 pressures.

The payment trend analysis offers valuable insights into how customers structure their spending using credit card installments:

- 80.5% of the credit card value comes from installment payments, amounting to \$10.10M of the total Credit Card Value (\$12.54M).
- Most popular installment plans span 2–6 months (39,131 transactions), followed by full payments (25,457 transactions). Installment plans in the 7-9 months and 10-12 months range also have a notable number of transactions (6,538 and 5,484, respectively).
- Longer installment lengths correlate with higher average values
 - (e.g. 19–24 months → \$566 average)
 - (e.g. 13–18 months → \$412 average)
 - (e.g. 10-12 months → \$372 average).
- Weekday payments (total \$7.66M) are considerably higher compared to weekends (total \$2.44M).

Based on these insights, the following business recommendations are proposed:

2. Leverage Installment Payment Behavior

- **Promote flexible installment plans for higher-value products.**
- **Time promotions and marketing around weekday behavior patterns.**
- **Explore financing partnerships or optimised payment offers.**

Simplifications and Focus

The project focuses on the core ETL process for the Olist dataset. Simplifications include using median imputation for certain missing numerical values and dropping one column (review_comment_title) from the reviews dataset. The initial EDA presented in the sources also involves sampling the first 1000 rows for some datasets.

Conclusion

This project successfully established a data processing pipeline to extract, clean, transform, and integrate e-commerce data from Kaggle. By leveraging Python and Azure Data Lake Storage, the project prepared a comprehensive dataset that handled various data quality issues and standardised information from disparate sources. The cleaned and merged data is now ready for in-depth analysis and visualisation, potentially within a platform like Microsoft Fabric, to derive valuable business insights.

What's Next

The cleaned and integrated dataset is prepared for further analytical tasks. Potential next steps could include:

- In-depth Business Analysis: While initial analysis likely informed the existing reports, the integrated data in the Lakehouse/Warehouse allows for more granular and complex business analysis. This could include deeper customer segmentation beyond initial EDA, detailed product performance analysis across various dimensions (e.g., profitability, regional success), and intricate geographic sales analysis leveraging the loaded geolocation data.
- Advanced Analytics and Predictive Modelling: With a clean, integrated dataset and established data infrastructure (Lakehouse/Warehouse), the data is now ready for more sophisticated analytical techniques. This could involve:
 - Developing predictive models, such as forecasting sales, predicting delivery times, or identifying customers likely to churn.
 - Conducting customer lifetime value (CLTV) analysis.
 - Performing market basket analysis to understand product associations.
 - Implementing machine learning algorithms for tasks like recommending products or optimising pricing.
- Incorporating new data into future records:
 - Cost of Goods Sold (COGS) and Margin columns *would* be a significant enhancement from a business analysis perspective. It would allow for profitability analysis at various levels (product, seller, category, etc.), which goes beyond the current revenue-focused reporting and seller reliability analysis
 - More Detailed Customer Behavioral Data: The current data includes customer location and purchase history. Adding data on customer browsing activity (pages viewed, time on page, search queries), cart interactions (items added, removed), and wishlist activity could greatly enhance customer segmentation, personalised recommendations, and churn prediction models.
 - External Data: Incorporating external factors such as macroeconomic indicators (inflation, GDP growth), relevant holidays or events, and potentially even weather data could improve forecasting models (e.g., sales trend, delivery time predictions).
 - Detailed Logistics and Supply Chain Data: More granular data on inventory levels, warehouse locations (beyond seller zip code prefix), and carrier performance could refine delivery time predictions and optimise logistics.
 - Detailed Sellers Data: Join the existing dataset to a Marketing Funnel Dataset from sellers who filled in requests for contact to sell their products on the Olist Store.

Key Learnings

This project wasn't just about pipelines and visuals — it was about learning to navigate complexity as a team.

We learned how to:

1. Collaborate across a shared data model without overwriting one another's work
2. Handle messy, real-world data from ingestion to insight
3. Balance technical depth with storytelling clarity
4. And perhaps most importantly, we experienced firsthand how a clean data pipeline empowers meaningful decision-making.

References

- *Brazilian E-Commerce Public Dataset by Olist*. (2021). Kaggle.
<https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce>
- *Let's Talk E-Commerce: The Brazilian Edition*. (2025). Kaggle.
<https://www.kaggle.com/code/sergeistanislavovich/let-s-talk-e-commerce-the-brazilian-edition>
- *YvonneLipLim/JDE05_Final_Project: Generation SG Junior Data Engineer Programme Final Project*. (2025). GitHub.
https://github.com/YvonneLipLim/JDE05_Final_Project/tree/main
- *Marketing Funnel by Olist*. (2018). Kaggle.
<https://www.kaggle.com/olistbr/marketing-funnel-olist/home>
- *Olist Peak Time & Delivery Analysis*. (2022). Kaggle.
<https://www.kaggle.com/code/lipham/olist-peak-time-delivery-analysis>
- *Enhancing Customer Satisfaction*. (2024). Brazilian Journal of Technology.
<https://ojs.brazilianjournals.com.br/ojs/index.php/BJT/article/view/74444/51919>

Appendices

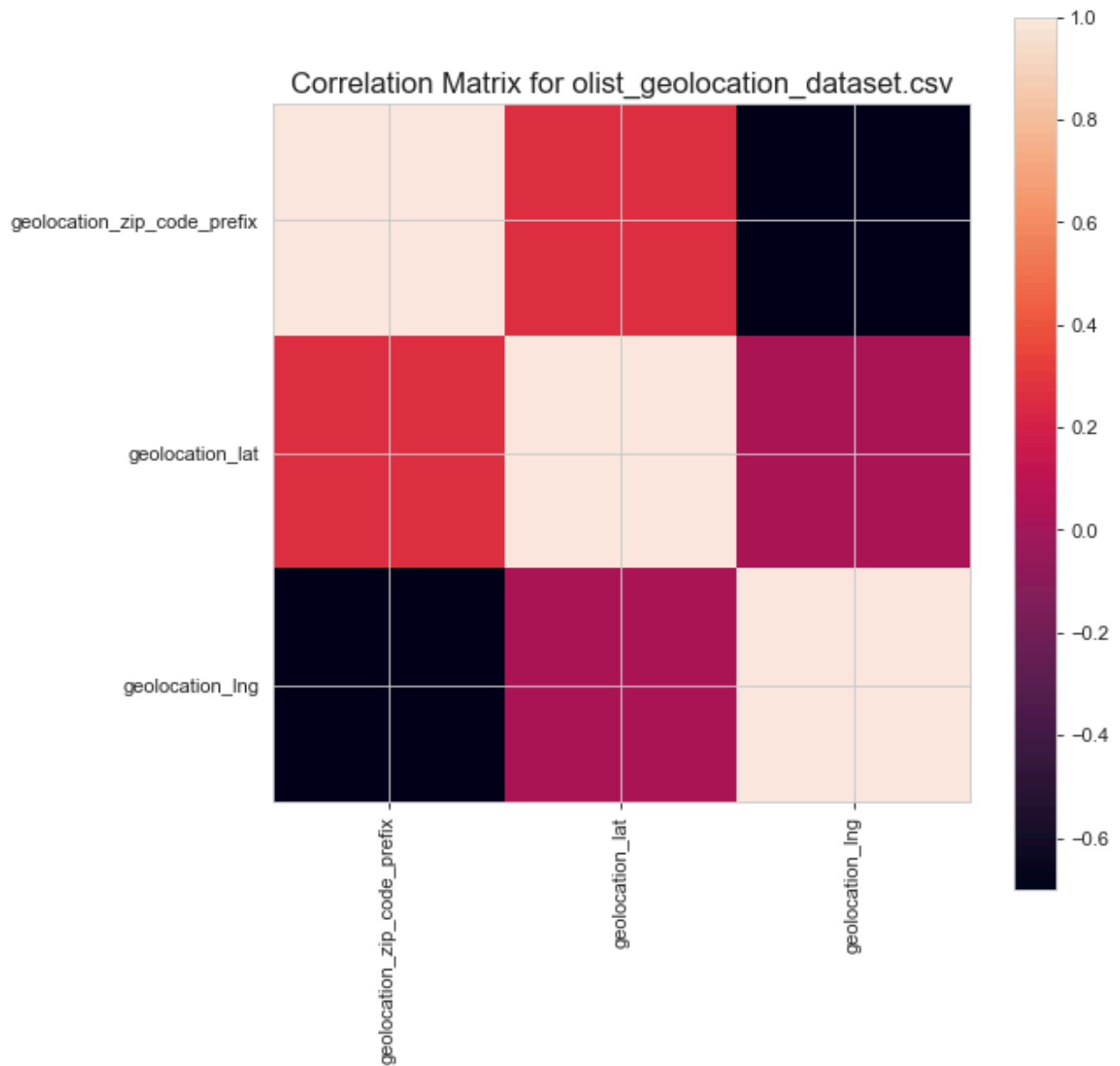


Figure 6: Correlation matrix for olist_geolocation_dataset.csv

Scatter and Density Plot

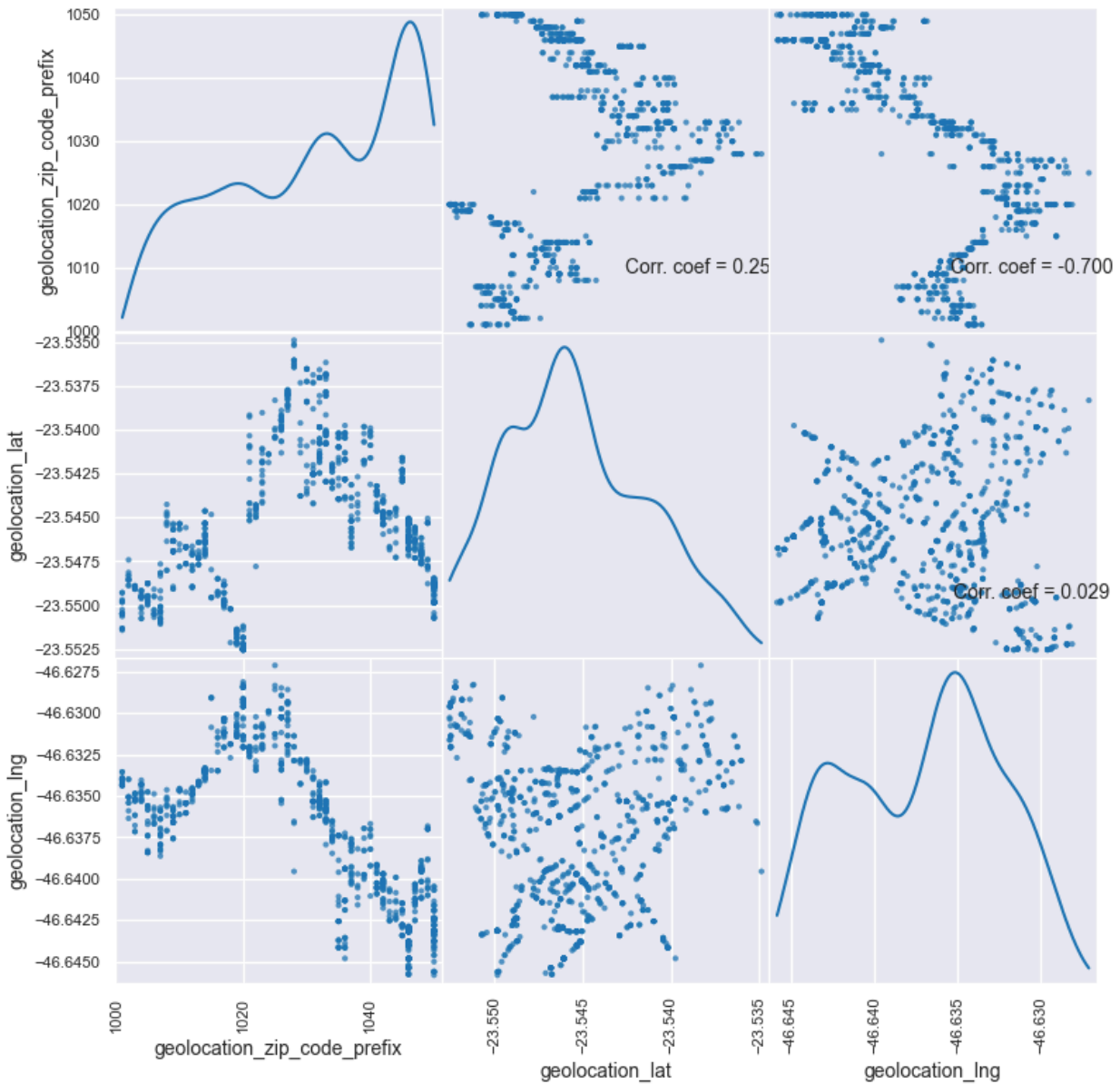


Figure 7: Scatter and density plot for `olist_geolocation_dataset.csv`