**An Exploratory Data Analysis (EDA) For Olist Store (Brazilian E-Commerce Dataset)**

**Summary**

1. Introduction

2. Data Overview

3. Data Description

4. Reading Data(.csv)

5. Data Cleaning: handling missing values and the duplication in data

6. Feedback Sentiment analysis

7. Clustering

8. Sales Prediction

9. Delivery performance

10. Conclusion/Recommendation

11. References

12. Appendix

**1. Introduction**

The report aims to perform an exploratory data analysis on client organisation Olist. A Brazilian ecommerce public dataset uploaded in year 2018 by Olist Store was used for examination to generate business insights and provide recommendations for improvement

**2. Data Overview**

Nine (9) csv files provided below were used for analysis.

1. olist\_customers\_dataset.csv (data)

2. olist\_geolocation\_dataset.csv(geo\_data)

3. olist\_order\_items\_dataset.csv(order\_itemdata)

4. olist\_order\_payments\_dataset.csv(pay\_data)

5. olist\_order\_reviews\_dataset.csv(rev\_data)

6. olist\_orders\_dataset.csv(orders)

7. olist\_products\_dataset.csv(order\_prddata)

8. olist\_sellers\_dataset.csv(order\_selldata)

9. product\_category\_name\_translation.csv(order\_prd\_catdata)

**3. Data Description**

For description of the columns and rows of the dataset used for exploration *see* appendix **1** for overview.

**4. Reading Data(.csv)**

To read the csv files perfectly, all the .csv files as mentioned in the data overview section in this report were carefully. Also, to avoid providing misinformation and providing the right insight, the number of columns and rows with the columns name of each .csv files were checked including the dtypes of each file.

**5. Data Cleaning:** handling missing values and the duplication in data

The datasets order\_payment, order\_items, geolocatuion, order\_review, sellers, products were thoroughly cleaned and transformed to produce quality visualizations. *See* **appendix 2** for various codes.

6. Feedback Sentiment analysis

7. Clustering

8. Sales Prediction

9. Delivery performance

**10. Conclusion/Recommendation**

Having critically examined the performance of Olist Store using various dataset, therefore, the following are recommended for the management of Olist Store to move their business forward:

* Increase customer satisfaction at least 2% / Increased 2% 2017 - 2018
* Maintain growth
* Surveys to extract more details about reviews (Cluster by product or delivery service)
* Advertising and Marketing during the night, overnight and morning.
* Offers and options for region 4 about payment installments
* Investigate the bad reviews without comments . Engage this relation – Post Purchase Service/ Customer Service Team.
* Delivery time had a proven effect on bad reviews.
* Delivery time distribution shows a lot of late delivery. It makes some improvement, but it’s much higher than E-Commerce norms. Average delivery time for sellers is about 4 days, and it should reduce to 1-2 days.

Average delivery time by carriers is very high with an average of 8 days. Although Brazil is a very big country, but the Olist must find a quicker way to send the items to its customers

* Offer discount for customers to use other payment type as it was observed that 78.34% of the customer used credit card.
* Investigate the regions 5,4 and 6 – To engage more sellers, as we can see there’s a potential growth and much more customers than sellers. Also Region 06 the most populated region
* Define a Target to engage more costumers in regions 8, 1 and 0. Increase Market-Share
* Investment in the Top 10 categories according Preference and Average Price.
* Produce a better Categories Catalogue following the Trends
* Open new categories relation to sustainability

11. References

[Brazilian Ebit Company E-commerce Overwier](https://company.ebit.com.br/webshoppers)

[Brazilian E-Commerce Public Dataset by Olist | Kaggle](https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce?select=olist_order_reviews_dataset.csv)

[Forecasting in Power BI](https://www.youtube.com/watch?v=mMd2rKK7dWc)

[Power BI Clustering](https://www.youtube.com/watch?v=LsFDWS77ER4)

[Open Spurce Brazilian Region Map Source](https://github.com/mapaslivres/municipios-br)

[Sentiment Analysis in Power BI](https://www.youtube.com/watch?v=mhe9Vs3jQes)

[Text Mining and Sentiment Analysis: Power BI Visualisations](https://www.red-gate.com/simple-talk/sql/bi/text-mining-and-sentiment-analysis-power-bi-visualizations/)

**12. Appendix**

**Appendix 1**

**Overview of dataset.csv**

Olist\_customers\_dataset.csv Table

Description automatically generated

Olist\_geo\_location

Table

Description automatically generated

Olist\_sellers\_dataset.csv

Table

Description automatically generated

olist\_order\_items\_dataset.csv

Text

Description automatically generated with medium confidence

Olist\_order\_payments\_dataset.csv

Table

Description automatically generated

Olist\_order\_reviews\_dataset.csv

Graphical user interface, application, table, Excel

Description automatically generated

Olist\_products\_dataset.csv

A picture containing text, indoor, screenshot

Description automatically generated

**Appendix 2**

**Codes Python and SQL**

Before any data cleansing occurred, the source data was visually checked for null values using a heat map created with the Seaborn Python library.

A screenshot of a computer

Description automatically generated

***Figure 1***

From the eight datasets provided by Olist via Kaggle, only three had NULL values (Order reviews, Orders, and Products).

Chart, bar chart

Description automatically generated

***Figure 2:*** Heatmap of the Sellers dataset with the yellow areas denoting NULL values.

Text

Description automatically generatedText

Description automatically generated***Figure 3:*** Order Review dataset NULL totals    Figure 4: Order dataset NULL totals

Text

Description automatically generated

***Figure*** ***5***: Product dataset NULL totals

The datasets were loaded into SSMS for cleansing. NULL values were replaced with appropriate dummy data using the CASE function, excess white spaces in review comments were removed using the TRIM function, and city names were edited using custom SQL functions such as removing special characters and capitalising city names. And finally, Power BI was used to remove duplicates.

Text

Description automatically generated

***Figure 6:*** SQL code to modify the city names of the Geolocation dataset.

Graphical user interface, application, Word

Description automatically generated

***Figure 7:*** Power BI used to remove duplicates.

**Other codes**

-- PYTHON code used to create heat maps to visually determine which datasets had nulls in them.

-------------------------------------------------------------------------------

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

customers = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\olist\_customers\_dataset.csv")

geolocation = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\olist\_geolocation\_dataset.csv")

order\_items = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\olist\_order\_items\_dataset.csv")

order\_payments = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\olist\_order\_payments\_dataset.csv")

order\_reviews = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\olist\_order\_reviews\_dataset.csv")

orders = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\olist\_orders\_dataset.csv")

products = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\olist\_products\_dataset.csv")

sellers = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\olist\_sellers\_dataset.csv")

product\_category\_name = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\product\_category\_name\_translation.csv")

municipios = pd.read\_csv(r"C:\Users\Asus\Desktop\Final Project\archive\municipios-2.csv")

sns.heatmap(order\_reviews.isnull(), yticklabels = False, cbar = False, cmap = 'viridis')

ax = plt.subplot()

plt.title('Order Reviews')

plt.setp(ax.get\_xticklabels(), rotation=30, ha='right')

plt.show()

print(order\_reviews.isnull().sum())

-------------------------------------------------------------------------------

-- Custom SQL function to remove extra characters

-------------------------------------------------------------------------------

create function [dbo].[RemoveExtraChars] ( @p\_OriginalString varchar(50) )

returns varchar(50) as

begin

declare @i int = 1; -- must start from 1, as SubString is 1-based

declare @OriginalString varchar(100) = @p\_OriginalString Collate SQL\_Latin1\_General\_CP1253\_CI\_AI;

declare @ModifiedString varchar(100) = '';

while @i <= Len(@OriginalString)

begin

if SubString(@OriginalString, @i, 1) like '[a-Z]'

begin

set @ModifiedString = @ModifiedString + SubString(@OriginalString, @i, 1);

end

set @i = @i + 1;

end

return @ModifiedString

-------------------------------------------------------------------------------

-- Custom SQL function to Capitalise the first letter of every word

-------------------------------------------------------------------------------

CREATE FUNCTION [dbo].[InitCap] ( @InputString varchar(4000) )

RETURNS VARCHAR(4000)

AS

BEGIN

DECLARE @Index INT

DECLARE @Char CHAR(1)

DECLARE @PrevChar CHAR(1)

DECLARE @OutputString VARCHAR(255)

SET @OutputString = LOWER(@InputString)

SET @Index = 1

WHILE @Index <= LEN(@InputString)

BEGIN

SET @Char = SUBSTRING(@InputString, @Index, 1)

SET @PrevChar = CASE WHEN @Index = 1 THEN ' '

ELSE SUBSTRING(@InputString, @Index - 1, 1)

END

IF @PrevChar IN (' ', ';', ':', '!', '?', ',', '.', '\_', '-', '/', '&', '''', '(')

BEGIN

IF @PrevChar != '''' OR UPPER(@Char) != 'S'

SET @OutputString = STUFF(@OutputString, @Index, 1, UPPER(@Char))

END

SET @Index = @Index + 1

END

RETURN @OutputString

-------------------------------------------------------------------------------

-- SQL code use to modify the geolocation\_city names in the geolocation\_dataset

-------------------------------------------------------------------------------

SELECT [geolocation\_zip\_code\_prefix]

,[geolocation\_lat]

,[geolocation\_lng]

, CASE

WHEN [geolocation\_city] = '\* cidade' THEN 'Cidade'

WHEN [geolocation\_city] = '...arraial do cabo' THEN 'Arraial Do Cabo'

WHEN [geolocation\_city] = '´teresopolis' THEN 'Teresopolis'

WHEN [geolocation\_city] = '4º centenario' THEN 'Quarto Centenario'

WHEN [geolocation\_city] = '4o. centenario' THEN 'Quarto Centenario'

WHEN [geolocation\_city] = 'rj' THEN 'Rio de Janeiro'

WHEN [geolocation\_city] = 'sp' THEN 'Sao Paulo'

WHEN [geolocation\_city] = 'bh' THEN 'Belo Horizonte'

ELSE [dbo].[InitCap]([dbo].[RemoveExtraChars]([geolocation\_city]))

END AS [geolocation\_city]

,[geolocation\_state]

FROM [Olist].[dbo].[olist\_geolocation\_dataset]

-------------------------------------------------------------------------------

-- SQL code use to modify the review comment titles and messages in the order\_reviews\_dataset

-------------------------------------------------------------------------------

SELECT [review\_id]

,[order\_id]

,[review\_score]

, CASE WHEN [review\_comment\_title] IS NULL THEN ' '

ELSE TRIM([review\_comment\_title]) END AS [review\_comment\_title]

, CASE WHEN [review\_comment\_message] IS NULL THEN ' '

ELSE TRIM([review\_comment\_message]) END AS [review\_comment\_message]

,[review\_creation\_date]

,[review\_answer\_timestamp]

FROM [Olist].[dbo].[olist\_order\_reviews\_dataset]

-------------------------------------------------------------------------------

-- SQL code use to modify the product details in the products\_dataset

-------------------------------------------------------------------------------

SELECT [product\_id]

, CASE

WHEN [product\_category\_name] IS NULL THEN ' '

ELSE [product\_category\_name]

END AS [product\_category\_name]

, CASE

WHEN [product\_name\_lenght] IS NULL THEN 0

ELSE [product\_name\_lenght]

END AS [product\_name\_lenght]

, CASE

WHEN [product\_description\_lenght] IS NULL THEN 0

ELSE [product\_description\_lenght]

END AS [product\_description\_lenght]

, CASE

WHEN [product\_photos\_qty] IS NULL THEN ' '

ELSE [product\_photos\_qty]

END AS [product\_photos\_qty]

, CASE

WHEN [product\_weight\_g] IS NULL THEN 0

ELSE [product\_weight\_g]

END AS [product\_weight\_g]

, CASE

WHEN [product\_weight\_g] IS NULL THEN 0

ELSE [product\_weight\_g]

END AS [product\_length\_cm]

, CASE

WHEN [product\_weight\_g] IS NULL THEN 0

ELSE [product\_weight\_g]

END AS [product\_height\_cm]

, CASE

WHEN [product\_weight\_g] IS NULL THEN 0

ELSE [product\_weight\_g]

END AS [product\_height\_cm]

FROM [Olist].[dbo].[olist\_products\_dataset]

**Python code for delivery time distribution.**  
import pandas as pd  
import matplotlib.pyplot as plt  
olist = pd.read\_csv('C:\\Users\\User\\Downloads\\Olist\\data12.csv')  
plt.hist(olist['Delivery\_hours\_to\_carriers'], bins = 35, density = True, stacked=True,  
    histtype = 'step' , range = (0, 1500), alpha = 0.5, label = 'Delivery\_to\_carriers')  
plt.hist(olist['Delivery\_hours\_for\_carriers'], bins = 35, density = True, stacked=True,  
    histtype = 'step', range = (0, 1500), alpha = 0.5, label = 'Delivery\_from\_carriers')  
plt.hist(olist['Total\_delivery\_hours'], bins = 35, density = True, stacked=True,  
    histtype = 'step', range = (0, 1500), alpha = 0.5, label='Total\_delivery\_time')  
plt.title('Delivery time distribution')  
plt.xlabel('Hours')  
plt.ylabel('Normalised Distribution')  
plt.legend()  
plt.show()

**SQL code for cleaning order dataset:**  
select [order\_id]  
,[customer\_id]  
    ,[order\_status]  
,cast([order\_purchase\_timestamp] as datetime) as order\_purchase\_timestamp  
,coalesce([order\_approved\_at], first\_value([order\_approved\_at])  
over (partition by approved\_group  
order by [order\_purchase\_timestamp]))as order\_approved\_at  
,coalesce(order\_delivered\_carrier\_date, first\_value(order\_delivered\_carrier\_date)  
over (partition by carrier\_group  
order by [order\_purchase\_timestamp]))as order\_delivered\_carrier\_date  
,coalesce(order\_delivered\_customer\_date, first\_value(order\_delivered\_customer\_date)  
over (partition by customer\_group  
order by [order\_purchase\_timestamp]))as order\_delivered\_customer\_date  
,cast([order\_estimated\_delivery\_date] as datetime)  
from (  
select [order\_id]  
,[customer\_id]  
    ,[order\_status]  
,[order\_purchase\_timestamp]  
,[order\_approved\_at]  
,[order\_delivered\_carrier\_date]  
,order\_delivered\_customer\_date  
,count (cast([order\_approved\_at] as datetime)) over (order by [order\_purchase\_timestamp]) as approved\_group  
,count (cast([order\_delivered\_carrier\_date] as datetime)) over (order by [order\_purchase\_timestamp]) as carrier\_group  
,count (cast([order\_delivered\_customer\_date] as datetime)) over (order by [order\_purchase\_timestamp]) as customer\_group  
,[order\_estimated\_delivery\_date]  
from [Olist].[dbo].[olist\_orders\_dataset]  
)sub  
order by order\_delivered\_customer\_date

/\*\*\*\*\*\* Script for import pandas as pddf = pd.read\_csv("./olist\_customers\_dataset.csv", header=None, skipfooter=5, engine="python")iterlist = df.iterrows()d = []for index, row in iterlist: item = {} item['id'] = row[0] \_, row = next(iterlist, ("end", "end")) item['id2'] = row[0] \_, row = next(iterlist) item["city"] = row[0] if item["city"] == "osasco": item["state"] = "SP"/\*\*\*\*\*\* Script for SelectTopNRows command from SSMS \*\*\*\*\*\*/

SELECT review\_score, COUNT(\*)

FROM [Olist\_Final\_Project].[dbo].[olist\_order\_reviews\_cleaned]

WHERE [review\_comment\_title] is NULL

GROUP BY review\_score

ORDER BY review\_score DESC;