UNIVERSITY OF ECONOMICS AND LAW

FACULTY OF INFORMATION SYSTEM

FINAL PROJECT

TOPIC: Enhance Delivery Performance

Subject: Artificial Intelligence in Business Analytics

Group:

Lecturer:

Ho Chi Minh,

Members of Group

No.	Student ID	Full name	Leader	Performance
1	K214162155	Nguyễn Phúc Thịnh	X	100%
2				100%
3				100%

Project Management

Task	Person in charge	Status
WEEK 1: IDENTIFY BUSINESS CASES		
Identify business cases, requirements, tools used, technology for analysis	All	100%
Learn about Delivery Performance	All	100%
Build a timeline for the implementation process	All	100%
Week 2: PERFORM DATA ANALYSIS, EXPLORE WITH PYTHON		
Review and proceed with week 1	Thành Tuân	100%
Describe and Cleaning dataset	Thịnh, Luân	100%
Simple Analysis & Performance Delivery Analysis	Thịnh, Luân	100%
WEEK 3: EXPLORE DATASET WITH CHAT GPT		
Review and proceed with week 2	Thành Tuân	100%
Explore Dataset by using ChatGPT	Phúc Thịnh	100%
Analytics to improve delivery performance	Luân, Tuân	100%
Summary of results and Proposal of Business Strategy	Thành Tuân	100%
WEEK 4: SUMMARY, PROPOSED APPROACH		
Comparison of analytics performance between Python and ChatGPT	All	100%
Propose a more optimal method	All	100%
Project summary	All	100%
WEEK 5: COMPLETE THE PROJECT		
Complete the project and present it	All	100%

Acknowledgments

Commitment

Table of Contens

Project Overview

Reasons

Analyzing a business's revenue and cost efficiency plays an important role in the business management process because it provides a detailed view of how resources are used and converted into revenue. In this way, businesses can evaluate business performance, identify strengths and weaknesses, and thereby make strategic decisions. Analytics help manage resources effectively, optimize costs and adapt to the market. Through this information, businesses can propose adjustments, purposeful operations and ensure sustainability in a volatile business context.

Objectives

Analyze the effectiveness of a business's revenue and costs to enhance strategic and financial management capabilities. By understanding cost structures and revenue streams, businesses can identify profit optimization opportunities, set competitive prices, and adapt quickly to market fluctuations. This analysis also aids in predicting financial trends, thereby helping businesses build long-term strategies and enhance competitive strength in the market.

Objects and scopes

Objects: The data from the sellers, geolocation, order_item, order, and customers tables will be used to analyze revenue and operational costs.

Scope:

Scope of Time: September 2016 to October 2018

Scope of Space: Olist dataset on E-commerce in Brazil

Tools and Programming Language

Python, Chat GPT, Orange

Structure of project

Chapter 1: Theoretical Basis

This chapter delves into the foundational theories and principles that underpin the subject matter. It likely covers theoretical frameworks, models, and concepts relevant to the study.

Chapter 2: Business Case

Focused on presenting a practical scenario or case study within a business context. It might detail real-world applications, challenges, and solutions related to the subject.

Chapter 3: Explore Dataset with Python

This chapter involves the exploration and analysis of a dataset using Python programming language. It would include techniques, tools, and methodologies to extract insights from the data.

Chapter 4: Explore Dataset by using ChatGPT

Explores the dataset employing ChatGPT, likely highlighting the use of natural language processing for analysis. It may showcase how ChatGPT can assist in data exploration, providing unique perspectives or insights.

Chapter 5: Comparison of analytics performance between Python and ChatGPT

A comparative analysis chapter, where the performances of Python (a traditional data analysis tool) and ChatGPT (a language-based model) are contrasted. It might discuss the strengths, weaknesses, and applications of each approach in handling the dataset.

Chapter 1: Theoretical Basis

1.1 What is Delivery Performance?

Delivery performance evaluates how well a business manages its delivery tasks, measuring success in meeting customer expectations with timely and accurate deliveries. For businesses dealing with goods, whether central to their operations or not, this aspect significantly impacts customer satisfaction and overall prosperity. Observing and assessing delivery performance helps pinpoint improvement areas, enabling the implementation of better service strategies.

A streamlined delivery performance optimizes resource use, reduces costs, and ensures customers are pleased with your services' reliability and swiftness. Examining delivery performance using specific KPIs and metrics empowers businesses to recognize strengths, weaknesses, and areas needing enhancement.

1. 2 Why it’s Important to Measure Delivery Performance

Customer Satisfaction

The cornerstone of any thriving business is a content and devoted customer base. Delivery performance holds a crucial role in shaping customer contentment. Timely and accurate order deliveries foster positive experiences, nurturing trust and loyalty. Content customers not only tend to make repeated purchases but also become enthusiastic advocates, spreading the word about your services, broadening your brand's reach, and impact.

Operational Efficiency

Assessing key performance indicators (KPIs) and metrics linked to delivery performance offers businesses valuable insights into their operational effectiveness. Pinpointing bottlenecks, inefficiencies, and areas for enhancement allows businesses to streamline their delivery processes. Streamlined operations result in reduced delivery times, heightened productivity, and decreased operational expenses. With efficient delivery operations, businesses can judiciously allocate resources, maximizing their time and budget.

Cost Optimization

Comprehending delivery costs is pivotal for resource optimization and overall profitability. Analyzing delivery performance metrics like average cost per delivery enables businesses to target areas where costs can be minimized. From fuel expenditures and vehicle maintenance to labor and routing inefficiencies, identifying chances to save on expenses aids businesses in operating in a financially sustainable manner, enhancing their financial outcomes.

Competitive Advantage

Consistently meeting and surpassing customer expectations distinguishes your business from competitors. As your reputation for dependable and effective delivery services strengthens, you gain a competitive edge, attracting new customers and retaining existing ones.

Continuous Improvement

Regularly gauging delivery performance enables businesses to stay adaptable and responsive to shifting market demands. By recognizing areas for enhancement, businesses can proactively introduce changes, fine-tune their services, and maintain a high delivery performance standard aligned with evolving customer requirements.

Chapter 2: Business Case

2.1 Problem

Analyzing distribution performance has become an integral and indispensable part of business strategies. The issue at hand is how to gain a deeper understanding of the distribution process's operations, from measuring performance to optimizing it. Companies need to focus on identifying which key performance indicators (KPIs) are most crucial and how to accurately measure them. Additionally, comprehending the entire picture of how goods or services are transported, from packaging to shipping and delivery, requires meticulous attention to detail.

Another challenge lies in handling the information gathered from analysis. How can this data be transformed into useful insights to improve performance and drive business decisions? This demands that companies not only know how to collect data but also possess the ability to creatively analyze and apply information derived from it.

Ultimately, the issue also involves adapting to change. Markets are constantly evolving, and customer demands are ever-changing. How can businesses maintain optimal distribution performance amidst this landscape? Continuously updating and adjusting strategies based on new information will become a crucial determinant of distribution success.

2.2 Expected goal

The main objective related to this topic is to enhance the overall efficiency of the business by deeply understanding and optimizing distribution performance. By delving into the intricacies of the distribution process, the goal is to identify key performance indicators that directly influence effectiveness (OTD, OFD, Delivery Time,...). This involves not only accurately measuring performance but also using that data to optimize operations.

Another objective is to utilize distribution performance analysis as a decision-making tool grounded in data. It involves transforming raw data into actionable insights to drive improvements, cost reduction, and enhance customer satisfaction. Ultimately, the aim is to build a flexible and continually adaptive system to respond to market changes, ensuring competitiveness and sustainable development.

In particular, the Python-based analysis will focus on impactful metrics influencing delivery performance, such as transportation time, On-Time Delivery (OTD), and Order Fill Rate (OFR), as follows:

- **The OTD (On-Time Delivery)** metric is one of the crucial indicators measuring a business's ability to deliver goods within the promised timeframe. It gauges the ratio of orders delivered on time as committed to customers. OTD plays a pivotal role in establishing a company's reliability in fulfilling delivery commitments. A high OTD rate often correlates with customer satisfaction, fostering an efficient and dependable delivery system.
- **The OFR (Order Fill Rate)** metric is another significant measure in delivery performance. It assesses the ratio of orders fulfilled with the correct quantity and quality of requested products. OFR reflects a company's capability to meet the entirety of a customer's needs within each order. A high OFR typically indicates efficiency in inventory management, ordering processes, and quick responsiveness to market demands.

Chapter 3: Explore Dataset with Python

3.1 Describe dataset

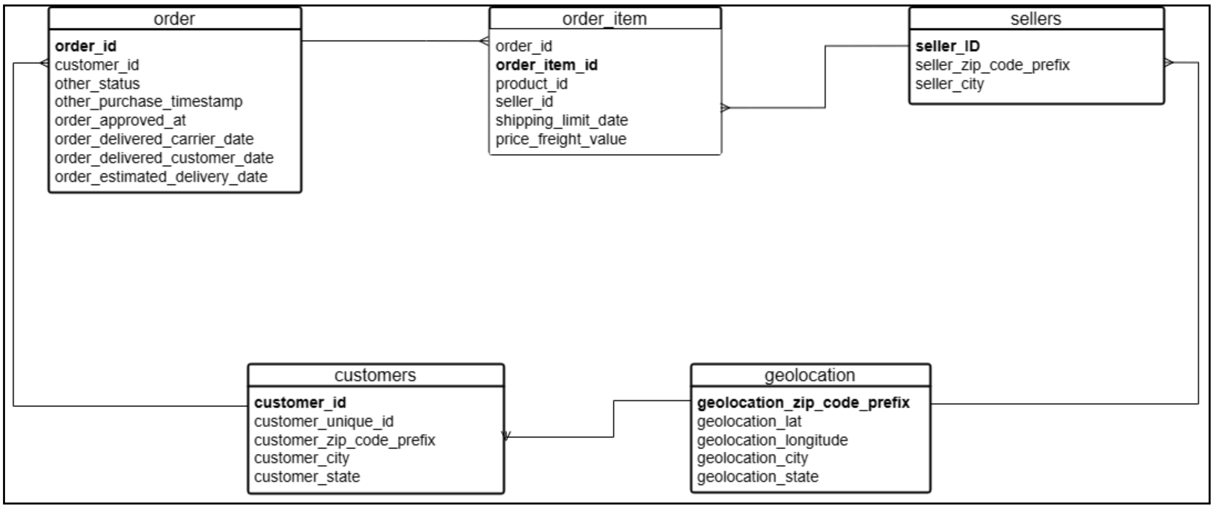


Figure 3.1 Relationships between data tables

The data set selected by the team to fit the business requirements includes 5 tables: order, order_item, seller, customers, geolocation.

Table 3.1 - order table

Columns	Contents
order_id	unique identifier of the order.
customer_id	key to the customer dataset. Each order has a unique customer_id.
order_status	Reference to the order status (delivered, shipped, etc).
order_purchase_timestamp	Shows the purchase timestamp.
order_approved_at	Shows the payment approval timestamp.
order_delivered_carrier_date	Shows the order posting timestamp. When it was handled to the logistic partner.
order_delivered_customer_date	Shows the actual order delivery date to the customer.
order_estimated_delivery_date	Shows the estimated delivery date that

	was informed to customer at the purchase moment.

This is the core dataset. From each order you might find all other information.

Table 3.2 - Table order_item

Columns	Contents
order_id	order unique identifier
order_item_id	sequential number identifying number of items included in the same order.
product_id	product unique identifier
seller_id	seller unique identifier
shipping_limit_date	Shows the seller shipping limit date for handling the order over to the logistic partner.
price	item price
freight_value	item freight value item (if an order has more than one item the freight value is splitted between items)

This dataset includes data about the items purchased within each order.

Table 3.3 - Table sellers

Columns	Contents
seller_id	seller unique identifier
seller_zip_code_prefix	first 5 digits of seller zip code
seller_city	seller city name
seller_state	seller state

This dataset includes data about the sellers that fulfilled orders made at Olist. Use it to find the seller location and to identify which seller fulfilled each product.

Table 3.4 - Table customers

Columns	Contents
customer_id	key to the orders dataset. Each order has a unique customer_id.
customer_unique_id	unique identifier of a customer.
customer_zip_code_prefix	first five digits of customer zip code
customer_city	customer city name
customer_state	customer state

This dataset has information about the customer and its location. Use it to identify unique customers in the orders dataset and to find the orders delivery location.

At our system each order is assigned to a unique customer_id. This means that the same customer will get different ids for different orders. The purpose of having a customer_unique_id on the dataset is to allow you to identify customers that made repurchases at the store. Otherwise you would find that each order had a different customer associated with.

Table 3.5 - Table geolocation

Columns	Contents
geolocation_zip_code_prefix	first 5 digits of zip code
geolocation_lat	latitude
geolocation_lng	longitude
geolocation_city	city name
geolocation_state	state

This dataset has information Brazilian zip codes and its lat/lng coordinates. Use it to plot maps and find distances between sellers and customers.

Table 3.6 - Table products

Columns	Contents
product_id	unique product identifier.
product_category_name	root category of product, in Portuguese.

product_name_lenght	number of characters extracted from the product name.
product_description_lenght	number of characters extracted from the product description.
product_photos_qty	number of product published photos
product_weight_g	product weight measured in grams.
product_length_cm	product length measured in centimeters.
product_height_cm	product height measured in centimeters.
product_width_cm	product width measured in centimeters.

This dataset includes data about the products sold by Olist.

3.2 Data cleaning

Customer Dataset

(99441, 5)

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	sao bernardo do campo	SP
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	sao paulo	SP
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775	mogi das cruzes	SP
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	SP

Figure 3.2 - Customer Dataset

The dataset consists of 5 columns containing information about ID, postal code, city code, and state code of each customer. The geographical location information can be utilized to categorize customers based on regions, cities, or states, enabling the identification of trends and distinctive characteristics within each group.

```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 5 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   customer_id               99441 non-null  object
 1   customer_unique_id        99441 non-null  object
 2   customer_zip_code_prefix  99441 non-null  int64
 3   customer_city             99441 non-null  object
 4   customer_state            99441 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.8+ MB
```

```
# check duplicated
print(f"Numbers of duplicated in customers: {customers.duplicated().sum()}")
```

```
Numbers of duplicated in customers: 0
```

Figure 3.3 - Info of Customers Dataset

The customers data table seems to be quite clean because out of the total 99441, there are no null values and there are no duplicate rows.

```
# Unique city
len(customers['customer_city'].unique())
```

```
4119
```

```
from collections import Counter

counter = Counter(customers['customer_city'])
most_common = counter.most_common(1)  # Lấy phần tử có tần suất cao nhất
print("Nơi khách hàng phân bố nhiều nhất:", most_common[0][0])
print(f"Số khách hàng ở tại thành phố {most_common[0][0]}:", most_common[0][1])
```

```
Nơi khách hàng phân bố nhiều nhất: sao paulo
Số khách hàng ở tại thành phố sao paulo: 15540
```

```
counter = Counter(customers['customer_city'])
least_common = counter.most_common()[:-2:-1]  # Lấy phần tử có tần suất cao nhất
print("Nơi khách hàng phân bố ít nhất:", least_common[0][0])
print(f"Số khách hàng ở tại thành phố {least_common[0][0]}:", least_common[0][1])
```

```
Nơi khách hàng phân bố ít nhất: eugenio de castro
Số khách hàng ở tại thành phố eugenio de castro: 1
```

Figure 3.4 - Distributing customers by city

Figure 3.4 tells us that there are 4,119 cities where this business's customers live. The place with the most customers is the city of "Sao Paulo" with a number of 15,540. On the contrary, the place with the fewest customers is the city of "Eugenio De Castro" with only 1 customer.

Geolocations Dataset

(1000163, 5)

	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
0	1037	-23.545621	-46.639292	sao paulo	SP
1	1046	-23.546081	-46.644820	sao paulo	SP
2	1046	-23.546129	-46.642951	sao paulo	SP
3	1041	-23.544392	-46.639499	sao paulo	SP
4	1035	-23.541578	-46.641607	sao paulo	SP

Figure 3.5 - Geolocations Dataset

The dataset comprises 5 columns, including geolocation codes and coordinates representing the location of each point of entry. With this table, we can establish connections with other tables using the geolocation code to precisely determine the location.

```
# Information of geolocation
geolocations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000163 entries, 0 to 1000162
Data columns (total 5 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   geolocation_zip_code_prefix  1000163 non-null  int64
 1   geolocation_lat              1000163 non-null  float64
 2   geolocation_lng              1000163 non-null  float64
 3   geolocation_city             1000163 non-null  object
 4   geolocation_state            1000163 non-null  object
dtypes: float64(2), int64(1), object(2)
memory usage: 38.2+ MB
```

```
# check duplicated
print(f"Numbers of duplicated in geolocations: {customers.duplicated().sum()}")
```

```
Numbers of duplicated in geolocations: 0
```

Figure 3.6 - Information of geolocation dataset

The geolocations data table is also very good when 1000163 out of 1000163 do not contain null values and there are no duplicate values.

Order Items Dataset

```
(112650, 7)
```

	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-09-19 09:45:35	58.90	13.29
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	2017-05-03 11:05:13	239.90	19.93
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	2018-01-18 14:48:30	199.00	17.87
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	2018-08-15 10:10:18	12.99	12.79
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	2017-02-13 13:57:51	199.90	18.14
5	00048cc3ae777c65dbb7d2a0634bc1ea	1	ef92defde845ab8450f9d70c526ef70f	6426d21aca402a131fc0a5d0960a3c90	2017-05-23 03:55:27	21.90	12.69

Figure 3.7 - Order Items Dataset

With this table, we will have information about the orders placed, the products included in each order, the seller's identity, prices, and shipping costs. From there, we can join it with other tables to find optimal strategies for operational efficiency, reducing shipping costs, and more.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   order_id             112650 non-null  object
 1   order_item_id        112650 non-null  int64
 2   product_id           112650 non-null  object
 3   seller_id            112650 non-null  object
 4   shipping_limit_date  112650 non-null  object
 5   price                112650 non-null  float64
 6   freight_value        112650 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 6.0+ MB
```

Figure 3.8 Information of order_item table

In this table there are a total of 112650 rows and do not contain null values. However, there is an absurd thing here: the data type of the "shipping_limit_date" column is object. We need to convert it to date.

```
# Change the datatype of time columns into Datetime
order_items['shipping_limit_date'] = pd.to_datetime(order_items.shipping_limit_date)
# Check the datatypes for each column
print(f'The datatypes for each column are:\n{order_items.dtypes}')
```

```
The datatypes for each column are:
order_id                       object
order_item_id                   int64
product_id                     object
seller_id                      object
shipping_limit_date    datetime64[ns]
price                         float64
freight_value                 float64
dtype: object
```

Figure 3.9 - Change type of “shipping_limit_date”

The data type of the “shipping_limit_date” column has been converted to the correct one.

```
# Unique order ID, productID
print(f"Unique Order ID: {len(order_items['order_id'].unique())}")
print(f"Unique productID: {len(order_items['product_id'].unique())}")
```

```
Unique Order ID: 98666
Unique productID: 32951
```

Figure 3.10 - Number of orders and products

The results of the above code show that there are a total of 98,666 orders, of which 32,951 products were purchased.

Orders Dataset

```
99441, 8)
```

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	orde
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-10-02 11:07:15	2017-10-04 19:55:00	2017-10-10 21:25:13	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	2018-07-26 03:24:27	2018-07-26 14:31:00	2018-08-07 15:27:45	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49	2018-08-08 08:55:23	2018-08-08 13:50:00	2018-08-17 18:06:29	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	2017-11-18 19:45:59	2017-11-22 13:39:59	2017-12-02 00:28:42	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-02-13 21:18:39	2018-02-13 22:20:29	2018-02-14 19:46:34	2018-02-16 18:17:02	

Figure 3.11 - Order dataset

The dataset provides information about each order, including the customer ID, along with timestamps for packaging, order dispatch, and delivery times. With such details, it is possible to calculate the average time it takes to package a product from the moment a customer places an order until the item is shipped. This information can be used to optimize packaging times for various products.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  object
 4   order_approved_at              99281 non-null  object
 5   order_delivered_carrier_date   97658 non-null  object
 6   order_delivered_customer_date  96476 non-null  object
 7   order_estimated_delivery_date  99441 non-null  object
dtypes: object(8)
memory usage: 6.1+ MB
```

Figure 3.12 - order.info()

The “order” table has a total of 99,441 rows and 8 columns. Surprisingly, this table has many null values appearing on the columns "order_approved_at", "order_delivered_carrier_date" and "order_delivered_customer_date". However, we probably won't delete them and just leave them as is. Another strange thing is that the data type of columns 3, 4, 5, 6, 7 must be datetime. Therefore we need to adjust:

```python
# Change the datatype of time columns into Datetime

orders['order_purchase_timestamp'] = pd.to_datetime(orders.order_purchase_timestamp)
orders['order_approved_at'] = pd.to_datetime(orders.order_approved_at)
orders['order_delivered_carrier_date'] = pd.to_datetime(orders.order_delivered_carrier_date)
orders['order_delivered_customer_date'] = pd.to_datetime(orders.order_delivered_customer_date)
orders['order_estimated_delivery_date'] = pd.to_datetime(orders.order_estimated_delivery_date)

orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  datetime64[ns]
 4   order_approved_at              99281 non-null  datetime64[ns]
 5   order_delivered_carrier_date   97658 non-null  datetime64[ns]
 6   order_delivered_customer_date  96476 non-null  datetime64[ns]
 7   order_estimated_delivery_date  99441 non-null  datetime64[ns]
dtypes: datetime64[ns](5), object(3)
memory usage: 6.1+ MB
```

Figure 3.13 - Change data type in order table

Now it looks a lot better!

Sellers Dataset

(3095, 4)

	seller_id	seller_zip_code_prefix	seller_city	seller_state
0	3442f8959a84dea7ee197c632cb2df15	13023	campinas	SP
1	d1b65fc7debc3361ea86b5f14c68d2e2	13844	mogi guacu	SP
2	ce3ad9de960102d0677a81f5d0bb7b2d	20031	rio de janeiro	RJ
3	c0f3eea2e14555b6faeea3dd58c1b1c3	4195	sao paulo	SP
4	51a04a8a6bdcb23deccc82b0b80742cf	12914	braganca paulista	SP

Figure 3.14 - Seller Data

With this table, we will have information about various sellers as well as geographic area codes. From this, it is possible to optimize the transportation process to reach customers. Additionally, we can identify which seller has the highest sales, enabling effective business decisions.

```
<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
```

Figure 3.15 - sellers.info()

The seller table has 3095 rows and does not contain null values.

Products dataset

	product_name_lenght	product_description_lenght	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm
count	32341.000000	32341.000000	32341.000000	32949.000000	32949.000000	32949.000000	32949.000000
mean	48.476949	771.495285	2.188986	2276.472488	30.815078	16.937661	23.196728
std	10.245741	635.115225	1.736766	4282.038731	16.914458	13.637554	12.079047
min	5.000000	4.000000	1.000000	0.000000	7.000000	2.000000	6.000000
25%	42.000000	339.000000	1.000000	300.000000	18.000000	8.000000	15.000000
50%	51.000000	595.000000	1.000000	700.000000	25.000000	13.000000	20.000000
75%	57.000000	972.000000	3.000000	1900.000000	38.000000	21.000000	30.000000
max	76.000000	3992.000000	20.000000	40425.000000	105.000000	105.000000	118.000000

Figure 3.16 - products description

The result of Figure 3.16 is some statistics about the product such as min, max, mean of height, length, width and volume of the product.

Products Categories dataset

```
# merge 2 dataframes
products = products.merge(product_categories, on='product_category_name', how='left')
products.head(3)
```

```
# Checking the missing values
products.isna().sum()
```

```
product_id                       0
product_category_name          610
product_name_lenght            610
product_description_lenght     610
product_photos_qty             610
product_weight_g                 2
product_length_cm                2
product_height_cm                2
product_width_cm                 2
product_category_name_english  623
dtype: int64
```

Figure 3.17 - Merge and Check Null product_category

For easier management, we will merge the "products" and "product_categories" tables into one table. After running the query to check, there are quite a few missing values here. In the "product_category_name" column, there are 610 values, but in the "product_category_name_english" column, there are 623 values. This indicates a shortfall of 13 values in the "product_category_name_english" column.

```python
# looking 13 missing data in product_category_name_english
products[products['product_category_name'].notna() & products['product_category_name_english'].isna()]
```

	product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm	product_category_name_english
1628	0105b5323d24fc655f73052694dbbb3a	pc_gamer	59.0	621.0	4.0	2839.0	19.0	16.0	18.0	NaN
5821	6fd83eb3e0799b775e4f946bd66657c0	portateis_cozinha_e_preparadores_de_alimentos	52.0	280.0	1.0	1200.0	25.0	33.0	25.0	NaN
7325	5d923ead886c44b86845f69e50520c3e	portateis_cozinha_e_preparadores_de_alimentos	58.0	284.0	1.0	1200.0	25.0	33.0	25.0	NaN
7478	6727051471a0fc4a0e7737b57bff2549	pc_gamer	60.0	1532.0	3.0	650.0	16.0	22.0	20.0	NaN
8819	bed164d9d628cf0593003389c535c6e0	portateis_cozinha_e_preparadores_de_alimentos	54.0	382.0	2.0	850.0	30.0	21.0	22.0	NaN
11039	1220978a08a6b29a202bc015b18250e9	portateis_cozinha_e_preparadores_de_alimentos	46.0	280.0	1.0	1200.0	25.0	33.0	25.0	NaN
14266	ae62bb0f95af63d64eae5f93dddea8d3	portateis_cozinha_e_preparadores_de_alimentos	59.0	927.0	1.0	10600.0	40.0	20.0	38.0	NaN
16182	1954739d84629e7323a4295812a3e0ec	portateis_cozinha_e_preparadores_de_alimentos	58.0	792.0	4.0	750.0	30.0	30.0	30.0	NaN
16930	dbe520fb381ad695a7e1f2807d20c765	pc_gamer	60.0	840.0	6.0	800.0	18.0	22.0	22.0	NaN
17800	c7a3f1a7f9eef146cc499368b578b884	portateis_cozinha_e_preparadores_de_alimentos	52.0	1372.0	5.0	7350.0	40.0	30.0	23.0	NaN
18610	7afdd65f79f63819ff5bee328843fa37	portateis_cozinha_e_preparadores_de_alimentos	48.0	305.0	1.0	750.0	20.0	20.0	20.0	NaN
26890	a4756663d007b0cd1af865754d08d968	portateis_cozinha_e_preparadores_de_alimentos	60.0	1304.0	4.0	650.0	22.0	6.0	14.0	NaN
29919	cb9d764f38ee4d0c00af64d5c388f837	portateis_cozinha_e_preparadores_de_alimentos	57.0	306.0	1.0	625.0	28.0	27.0	24.0	NaN

Figure 3.18 - 13 missing data

Thirteen missing values correspond to only two category names: "pc_gamer" and "portateis_cozinha_e_preparadores_de_alimentos". Therefore, we can easily search for and fill in the "product_category_name_english" column for these values.

```python
# filling the missing values in product_category_name_english where there are values in product_category_name
products['product_category_name_english'] = np.where(products["product_category_name"] == 'portateis_cozinha_e_preparadores_de_alimentos',
                                                     'kitchen_portable_and_food_preparers', products['product_category_name_english'])

products['product_category_name_english'] = np.where(products["product_category_name"] == 'pc_gamer',
                                                     'pc_gaming', products['product_category_name_english'])
```

```python
# filling the remain missing values in product_category_name and product_category_name_english with no_product_name
products['product_category_name'].fillna('not_defined_product', inplace= True)
products['product_category_name_english'].fillna('not_defined_product', inplace= True)

# filling with 0 for another missing numeric values
products.fillna(0, inplace=True)

# checking missing values again
products.isna().sum()
```

```
product_id                       0
product_category_name            0
product_name_lenght              0
product_description_lenght       0
product_photos_qty               0
product_weight_g                 0
product_length_cm                0
product_height_cm                0
product_width_cm                 0
product_category_name_english    0
dtype: int64
```

Figure 3.19 - Filling nulls

After replacement, the data has become cleaner and more organized.

3.3 Analysis

3.3.1 Simple Analysis

After understanding the structure and components of the dataset and cleaning them, we will proceed to analyze some simple aspects. This preliminary analysis will provide us with a more specific view and lay the foundation for an in-depth analysis of "Delivery Performance" in Chapter 5.

Firstly, we will explore the distribution of customers across different cities. The result of the following question is what we aim for: "How many customers are based on their city or states?" Python provides excellent support for us during the search for answers:

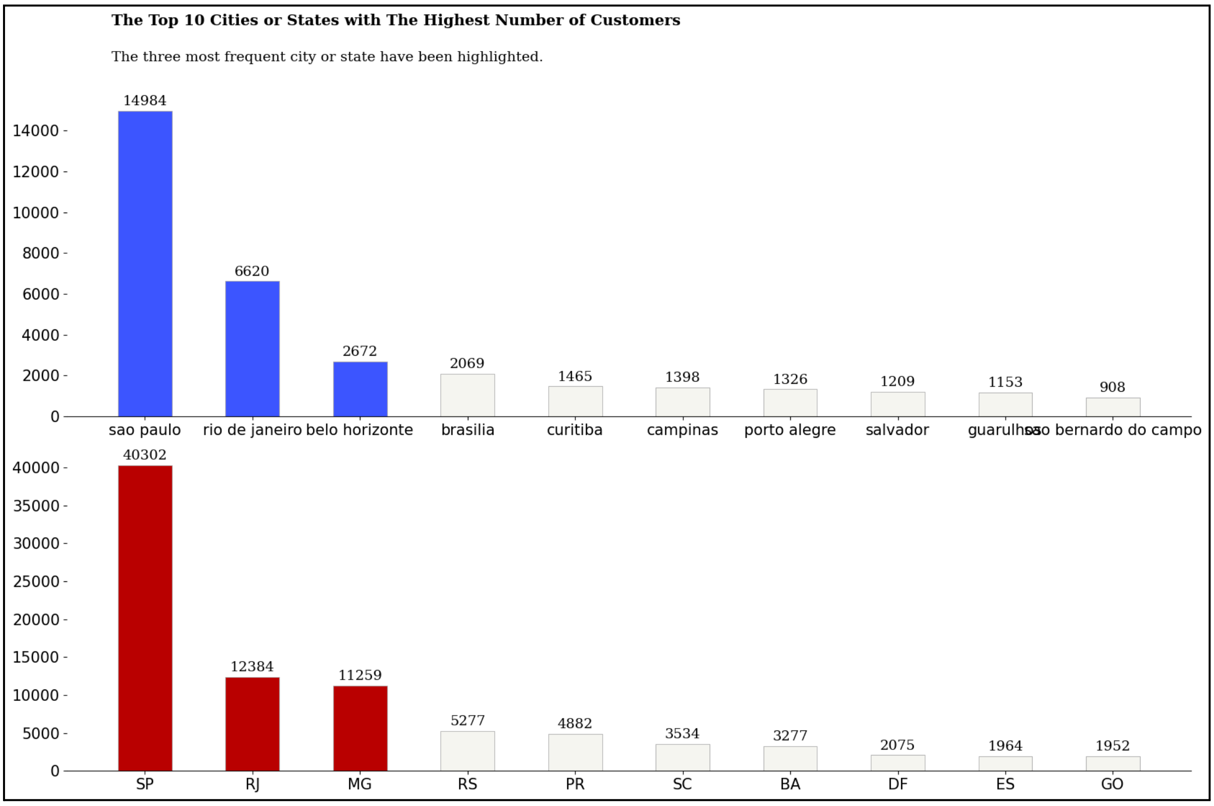


Figure 3.20 - How many Customers based on their city or states?

The results are quite clear. "Sao Paulo" is the city with the highest customer density, followed by the cities "Rio De Janeiro" and "Belo Horizonte". The states with a large number of customers are "SP" (Sao Paulo), "RJ" (Rio de Janeiro), and "MG" (Minas Gerais). It's noteworthy that these three states correspond to the three cities mentioned earlier in the same order.

Next, we will examine the distribution of sellers by region:

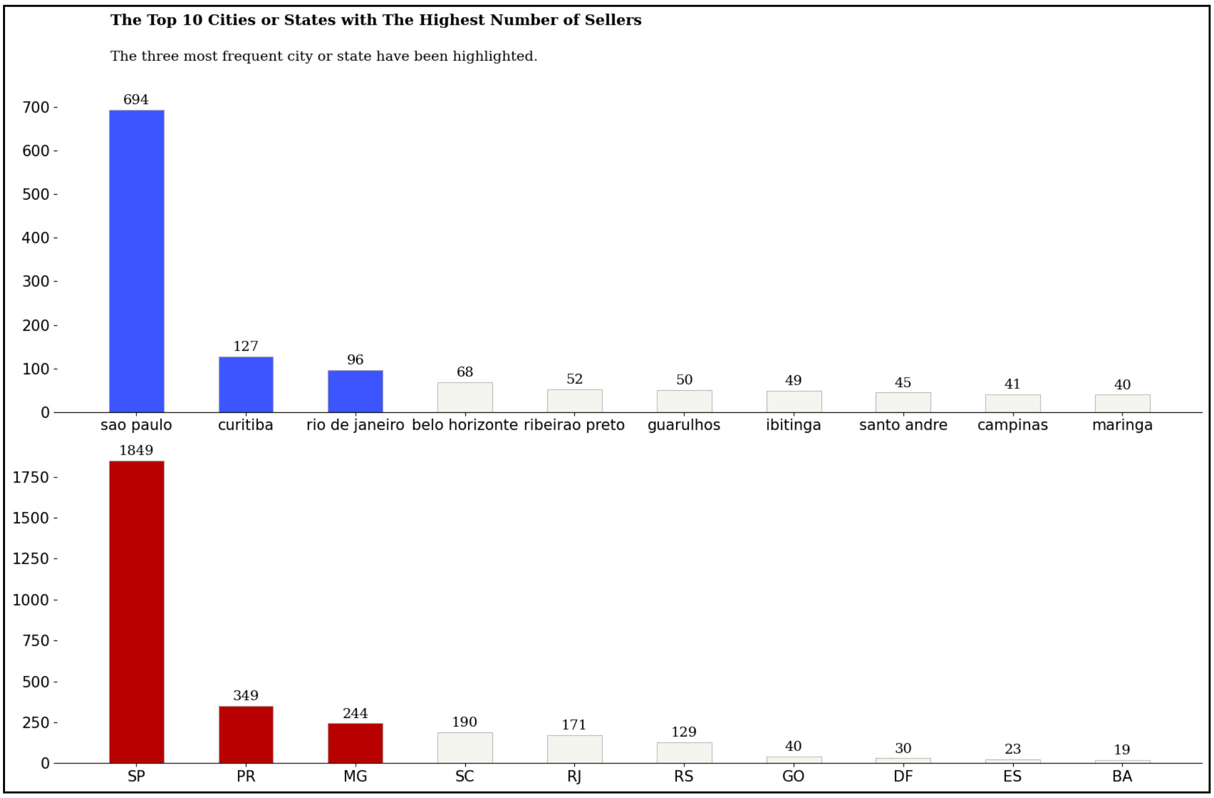


Figure 3.21 - How many sellers bases on their city or state?

Similar to customers, sellers are most densely distributed in the state of SP (Sao Paulo). However, there are some differences in the subsequent positions. The second position goes to the state of PR (Parana) with the city of Curitiba, followed by the state of MG (Minas Gerais) associated with the city of Rio de Janeiro.

An essential aspect to consider is the trend of purchasing over time. In this section, we will gain an overview of the business situation by asking: "How many orders occurred? (based on the time of purchase)"

```
1   print(orders['order_purchase_timestamp'].sort_values(ascending=False))

60938   2018-10-17 17:30:18
68373   2018-10-16 20:16:02
31891   2018-10-03 18:55:29
88500   2018-10-01 15:30:09
50387   2018-09-29 09:13:03
                ...
83078   2016-10-02 22:07:52
30710   2016-09-15 12:16:38
10071   2016-09-13 15:24:19
4396    2016-09-05 00:15:34
4541    2016-09-04 21:15:19
Name: order_purchase_timestamp, Length: 99441, dtype: datetime64[ns]
```
\

Figure 3.22 - Time of Purchase

Based on the information from Figure 3.21, we can see that details about purchase transactions are continuously recorded with precision down to the second. The dataset under analysis is stored from September 2016 to mid-October 2018.

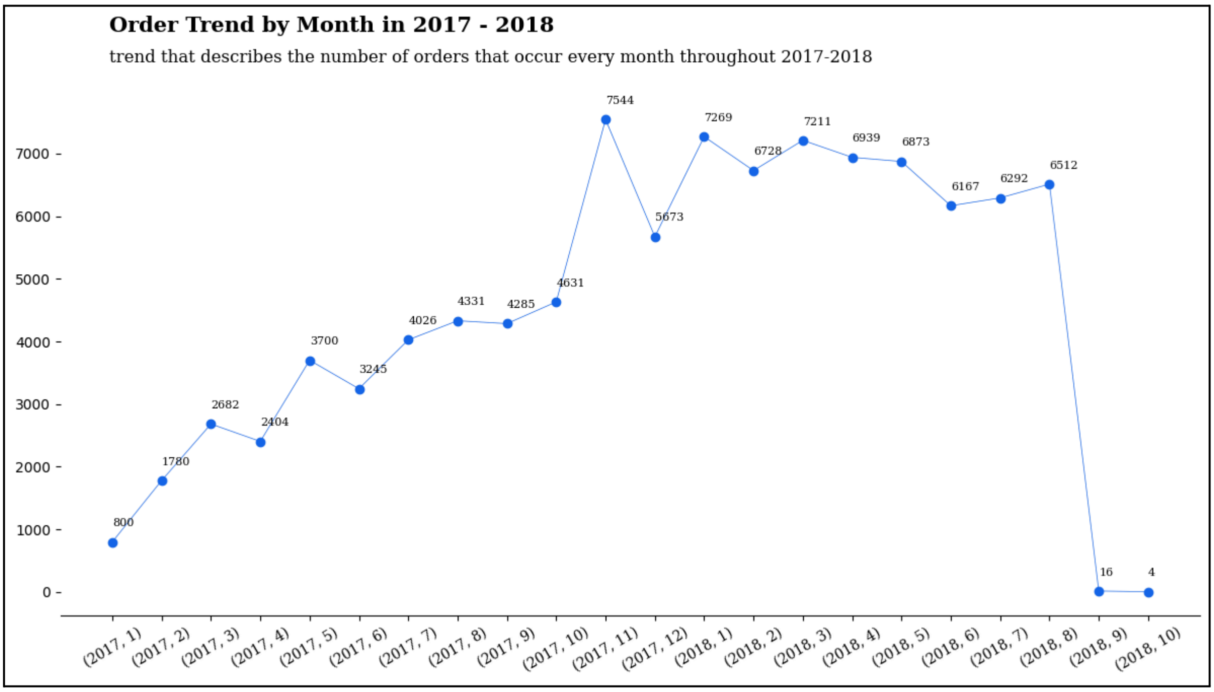


Figure 3.23 - Order Trend

With the above results, we can observe that the peak of sales over the two-year period (2017 and 2018) occurred in October 2017. This positive trend was sustained for approximately 9 months. However, starting from August 2018, for some reason, the sales frequency suddenly dropped below 1000 orders per month. The cause of this decline needs to be identified, and corrective actions should be taken immediately to avoid potential risks in the future.

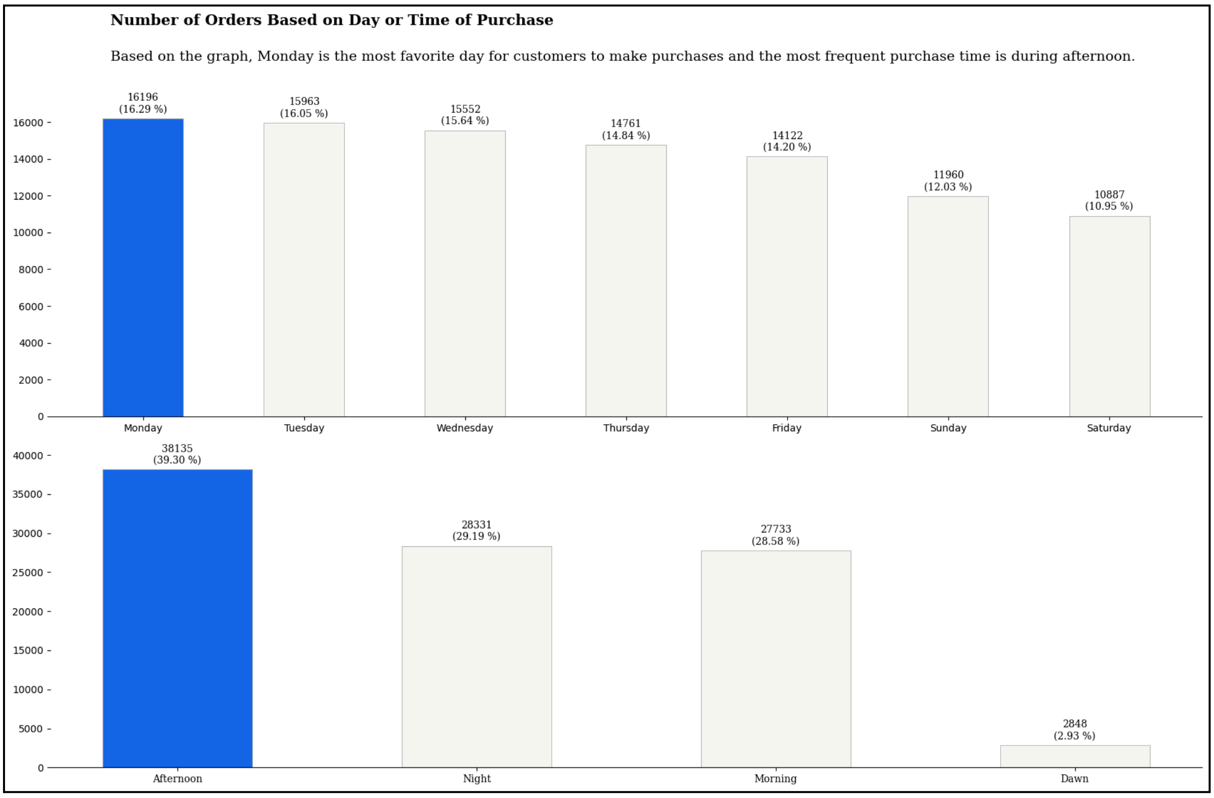


Figure 3.24 - Order Trend of Week & Day

Every customer tends to shop more at the beginning of the week and gradually decreases afterward. This trend could be due to people starting their workweek on Mondays and realizing the need for shopping to support their activities when supplies from the previous week run out. Alternatively, people often receive their salaries at the beginning of the week, so the shopping frequency during this time is higher. This aligns well with the fact that the afternoon is the time of day with the highest number of customers. In the morning, customers are likely focused on receiving information and their salaries, and they decide to shop in the afternoon.

Next, by using Python, we can quickly synthesize information to determine the total value of an order, the number of products in that order, and how many sales employees are involved in a given order. Here are the results:

	order_id	order_item_count	total_price_sum	unique_product	unique_seller
0	ca3625898fbd48669d50701aba51cd5f	10	1157.28	8	2
1	7d8f5bfd5aff648220374a2df62e84d5	7	582.94	7	1
2	77df84f9195be22a4e9cb72ca9e8b4c2	7	209.57	7	3
3	ad850e69fce9a512ada84086651a2e7d	7	1242.57	7	1
4	5efc0b7fe9df7f0c567404abaa4d25fc	6	469.20	6	1

Figure 3.25 - Summary Information Order

3.3.2 Performance Delivery Analysis

Order status

First, we'll explore the status of the orders through the question: "What's the status of the orders?"

```
# Count unique id & status
print(orders['order_id'].nunique())
print(orders['order_status'].nunique())
```

```
99441
8
```

Figure 3.26 - How many e-commerce order occured in Brazil?

There is a total of 99,441 orders with 8 different statuses

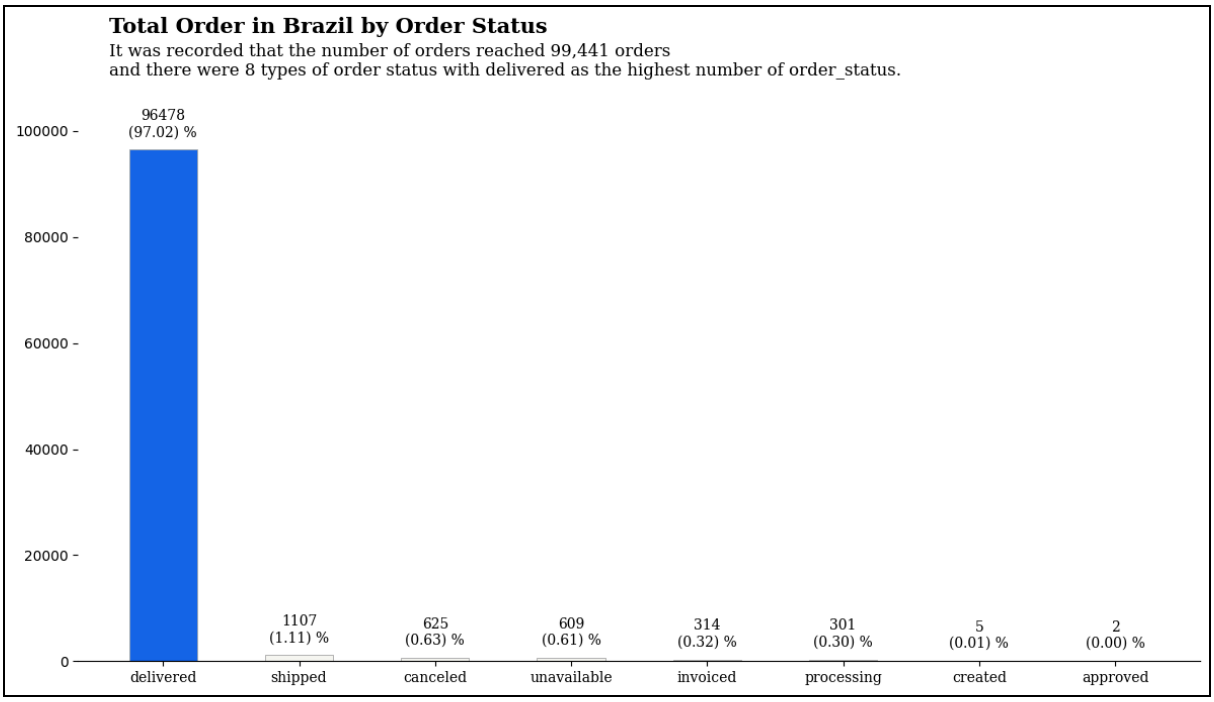


Figure 3.27 - Order Status

The statistics indicate that 97% of the total orders have been delivered ("delivered"). The remaining small portion is primarily in the "shipped" status, accounting for 1.11%. Other statuses make up a minority and include canceled, unavailable, invoiced, processing, created, and approved.

Recommendation: The results are quite positive, but there are still some points for the business to consider. First, to maximize revenue, the business should minimize the order cancellation rate. Therefore, it's essential to examine the reasons behind cancellations and address them promptly. Common reasons may include customers finding competitors offering lower prices or better customer service. Second, avoid letting orders linger in a pending processing state. Swiftly processing orders is also a factor that contributes to revenue improvement.

On-Time Delivery rate - OTD

OTD - On-Time Delivery is a critical metric in business, especially in the fields of transportation, manufacturing, and retail. It represents the percentage of orders or products delivered on schedule compared to the initially estimated time.

```python
# TỈ LỆ GIAO HÀNG ĐÚNG HẠN - OTD

# Merge relevant columns
merged_data = pd.merge(orders, customers, on='customer_id')

# Calculate On-Time Delivery Rates
merged_data['delivery_time'] = pd.to_datetime(merged_data['order_delivered_customer_date']) - pd.to_datetime(merged_data['order_estimated_delivery_date'])
merged_data['on_time'] = merged_data['delivery_time'] <= pd.Timedelta(0)
on_time_percentage = merged_data['on_time'].value_counts(normalize=True) * 100

# Visualize the Results
labels = ['On Time', 'Delayed']
colors = ['#1e90ff', '#b22222']  # Màu sắc tương ứng với các nhãn
plt.pie(on_time_percentage, labels=labels, autopct='%1.1f%%', colors=colors)
plt.title('On-Time Delivery Rate')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

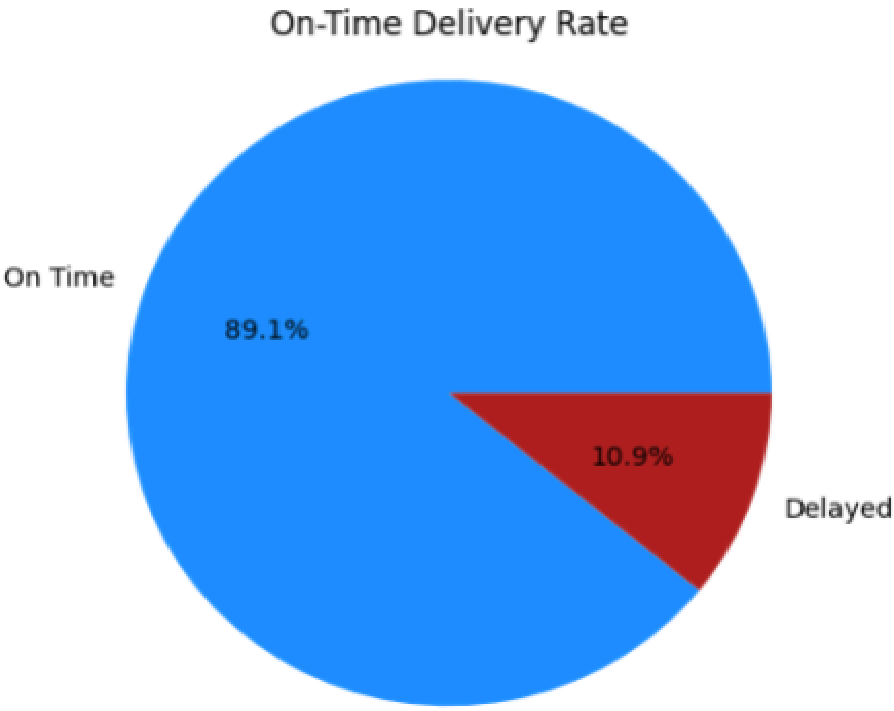


Figure 3.28 - OTD measure

In our case, the OTD has achieved a commendable result of 89.1%, a rather high figure. This indicates that our delivery timelines perform exceptionally well. With such strong performance, it implies a potential increase in customer trust. Moreover, the positive OTD reflects the efficiency of our supply chain.

Recommend: Despite a relatively high OTD of 89.1%, there is still a notable 10.9% of orders that are not delivered on time. This could be due to delays, non-deliveries, or challenges in the transportation process, potentially increasing delivery costs and diminishing overall business efficiency. Therefore, it is crucial for the business to pay attention to this 10.9% figure to address challenges and minimize delays in delivery. A low OTD may stem from issues in production, transportation, inventory management, or even planning processes. Hence, it is essential to adjust or optimize production and transportation processes to ensure higher performance.

Examining steps within the supply chain is necessary to identify areas for improvement. Additionally, businesses should enhance inventory management to ensure efficient inventory control, avoiding situations of stockouts or excessive inventory leading to delayed deliveries. Strengthening forecasting and planning is crucial, as accurate forecasts better prepare for demand and minimize the risk of delayed deliveries.

Investing in systems and technology is beneficial for improving data management, tracking goods, and streamlining processes. Training and developing employees are essential to ensure they are adequately trained and understand the work processes, thereby improving processes and minimizing errors.

Figure 3.29 - Delivery Time

- The average delivery time for the business is 12 days.
- Delivery times are concentrated largely within the range of 10 to 15 days, constituting 60% of the total orders. This indicates that the business is facing challenges in achieving short-term delivery goals.
- Some orders experience significantly longer delivery times, up to 200 days. Such prolonged delivery times may be attributed to reasons such as lost goods, transportation errors, or orders placed during peak periods.

This highlights that delivery time is a crucial factor affecting the customer experience. Extended delivery times can lead to customer disappointment and dissatisfaction with the business. Furthermore, delivery times can impact the revenue of the business. If delivery times are excessively long, customers may opt to purchase from a competitive rival with shorter delivery times.

Recommend:

The business needs to improve its transportation and delivery processes to shorten delivery times. This involves identifying the causes of orders with long delivery times and implementing timely corrective measures. Additionally, clear information regarding delivery times should be provided to customers to manage their expectations and prevent disappointment when delivery times deviate from their anticipated schedule.

For local orders, the business can utilize express delivery services to expedite delivery times. For orders outside the local area, collaboration with transportation providers offering fast delivery services is essential. The use of transportation management tools allows the business to track and monitor the delivery process, facilitating the identification and timely resolution of any arising issues.

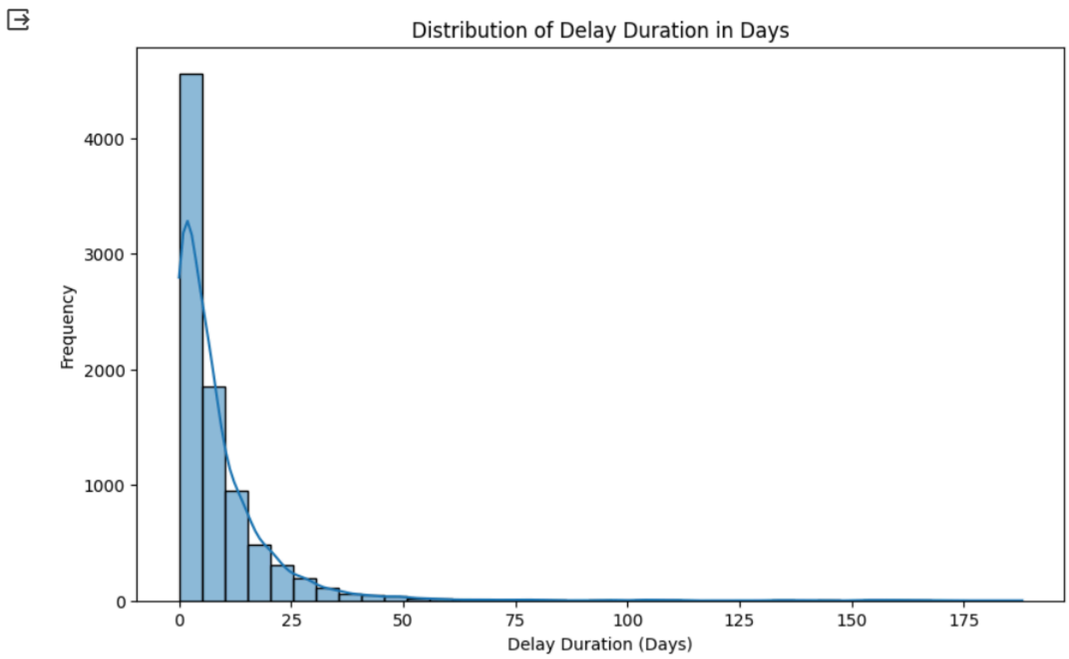


Figure 3.30 - Delay Duration

The number of days delayed for each order is primarily concentrated within the range of 1 day to approximately 5 days, with a broad distribution. Overall, this level of delay is generally acceptable.

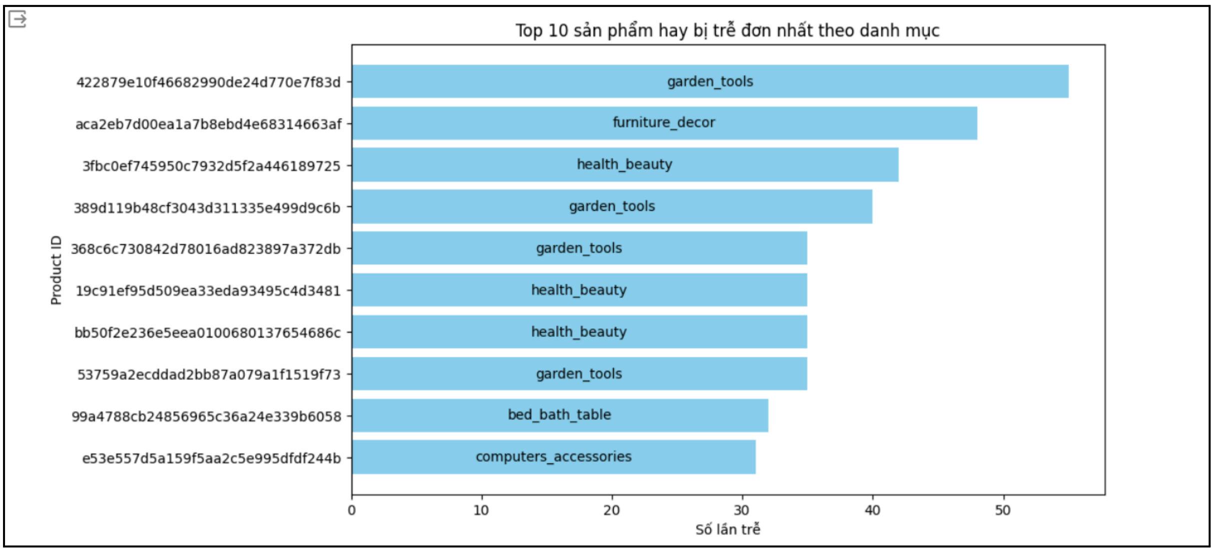


Figure 3.31 - Top products that are frequently delayed

According to the chart, the top 10 products most frequently experiencing delays fall into three main categories:

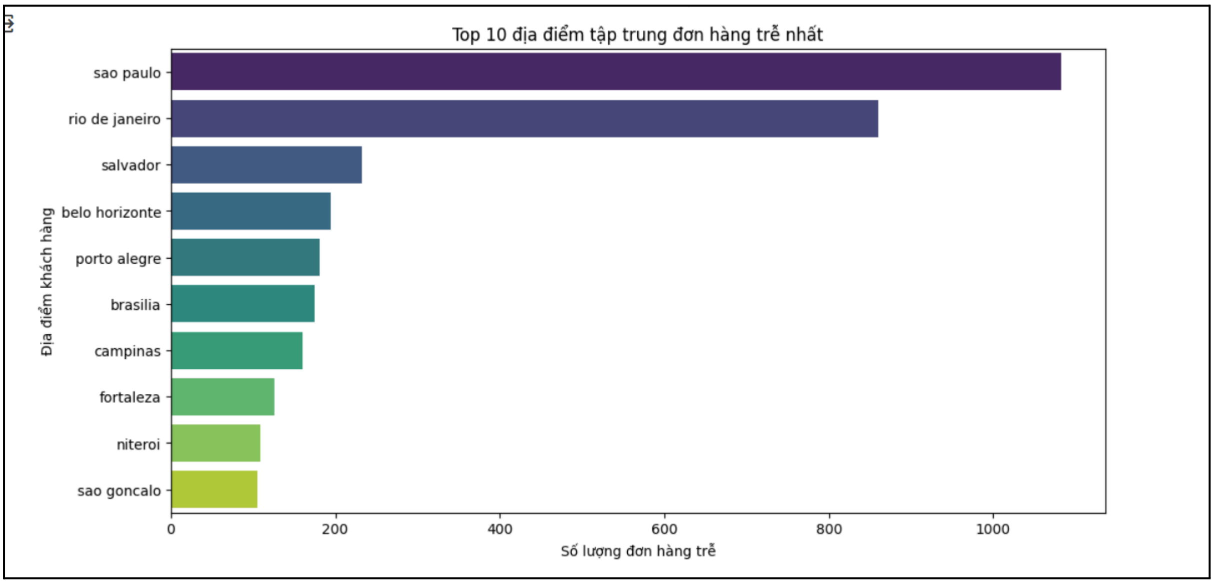
- Garden tools
- Furniture decor

- Health & beauty

- In which, the Garden tools category comprises 5 products within the top 10, accounting for 50%.
- Products in the Garden tools category typically have large, heavy, and cumbersome dimensions, making transportation challenging. Additionally, these products often have limited quantities, further complicating the search for suitable transportation providers.
- Products in the Furniture Decor category also have large, heavy, and cumbersome dimensions, similar to those in the Garden tools category. However, these products typically come in larger quantities, making it somewhat easier to find suitable transportation providers.
- "Products in the Health & Beauty category typically have compact dimensions, making them easy to transport. However, these products often come in limited quantities, posing challenges in finding suitable transportation providers.

From this, we can observe that products with large, heavy, and cumbersome dimensions tend to have a higher risk of order delays. Similarly, products with limited quantities also carry a higher risk of order delays.

Recommend: Regarding products with large, heavy, and cumbersome dimensions, involved parties need to coordinate closely to find suitable transportation providers, ensuring fast and accurate delivery times. For products with limited quantities, stakeholders should consider selecting reputable transportation services with schedules that align with customer needs. Additionally, it is essential for all parties involved to regularly update order status information for customers, allowing them to proactively arrange their delivery schedules.



The top 10 cities with the highest number of delayed deliveries in Brazil are all located in major urban areas, including São Paulo (SP), Rio de Janeiro (RJ), Salvador (BA), Belo Horizonte (MG), Porto Alegre (RS), Brasília (DF), Campinas (SP), Fortaleza (CE), Niterói (RJ), and São Gonçalo (RJ). These cities are characterized by dense populations, high urbanization rates, and a growing demand for online shopping. This places significant pressure on delivery businesses, resulting in a higher rate of delayed orders. Additionally, the average delivery distances for these delayed orders in these cities are relatively large, ranging from 208 km to 2308 km. This factor also contributes to the increased rate of delayed deliveries.

Based on the observations above, several insights can be drawn as follows:

(1) Delivery businesses need to have specific plans and strategies to meet the increasing demand for online shopping in large urban areas.

(2) Businesses should focus on optimizing the delivery process, minimizing delivery distances to reduce the rate of delayed orders.

(3) Companies need to enhance the quality of customer service to promptly address customer complaints regarding delayed deliveries.

To address the issue of delayed deliveries, delivery businesses can implement the following specific solutions:

(1) Utilize advanced technologies, such as artificial intelligence (AI), to optimize the delivery process. (2) Strengthen collaboration with transportation partners to ensure timely deliveries. (3) Expand warehouse networks to reduce delivery distances. (4) Establish an automated order sorting system to minimize errors during delivery. (5) Enhance the quality of training for delivery staff to ensure orders are delivered on time and intact. Resolving the issue of delayed deliveries is a significant challenge for delivery businesses. However, by implementing appropriate solutions, businesses can improve service quality and better meet customer demands.

Order Fill Rate (OFR)

Order Fill Rate (OFR) is a commonly used metric in retail management to measure the percentage of orders that are processed and delivered successfully compared to the total number of placed orders.

```python
# Kết hợp các bảng để tính OFR
merged_orf = pd.merge(orders, order_items, on='order_id', how='inner')

# Lọc các đơn hàng đã được giao đầy đủ
completed_orders = merged_orf[merged_orf['order_status'] == 'delivered']

# Tính tổng số lượng đơn hàng đã đặt
total_orders = orders['order_id'].nunique()

# Tính số lượng đơn hàng đã được giao đầy đủ
completed_order_count = completed_orders['order_id'].nunique()

# Tính OFR
order_fill_rate = (completed_order_count / total_orders) * 100

print(f"Order Fill Rate: {order_fill_rate:.2f}%")

# Trực quan hóa kết quả
sizes = [completed_order_count, total_orders - completed_order_count]
colors = ['#1e90ff', '#b22222']
#explode = (0.1, 0)  # Đẩy ra một phần tử để làm nổi bật

plt.pie(sizes, colors=colors, autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.title('Order Fill Rate')
plt.legend(labels=['Completed Orders', 'Uncompleted Orders'], loc='upper left')
plt.show()
```

Figure 3.32 OFR

Tỉ lệ hoàn thành đơn hàng trong trường hợp của chúng ta là khá cao. Tuy nhiên vẫn còn 3% chưa đạt. Do đó ta sẽ đi phân tích nguyên nhân tại sao lại xảy ra một lượng đơn hàng không hoàn thành.

Khi Order Fill Rate (OFR) thấp, điều này có thể tượng trưng cho nhiều vấn đề khác nhau. Thứ nhất có thể nói đến là khả năng cung ứng kém, OFR thấp có thể cho thấy rằng chuỗi cung ứng không linh hoạt, không đáp ứng được nhu cầu của khách hàng The completion rate of our orders is relatively high. However, there is still an unfulfilled rate of 3%. Therefore, we will analyze the reasons behind the occurrence of this portion of uncompleted orders.

When the Order Fill Rate (OFR) is low, it can indicate various issues. Firstly, inadequate supply capability may contribute to a low OFR, suggesting that the supply chain is inflexible and inefficient in meeting customer demands. There may be delays in order processing or problems in the transportation process. Secondly, poor service quality can lead to a low OFR, reflecting subpar product or service quality that prompts customers to cancel orders or request refunds. Additionally, a low OFR can impact the reliability of the business, eroding customer trust and affecting the brand image.

Next, let's clarify the reasons for the occurrence of Uncompleted Orders by examining the products that have the most significant impact.

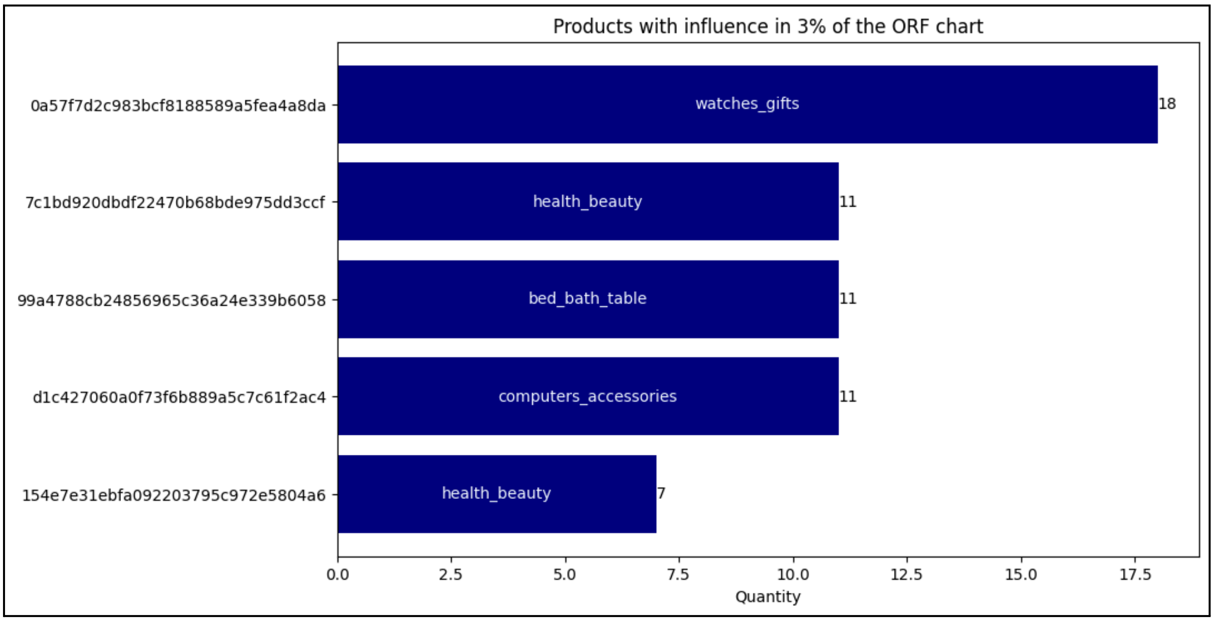


Figure 3.33 Products with influence in 3% of the ORF chart

It can be observed that the most unsuccessful products belong to the category of gift watches, followed by cosmetics and computers. These are indeed items that are easily supplied, but the reasons for their unfulfillment are not clear. The predicted reasons may fall into the following three points:

- Supply Issues: There may be a shortage of these products in the inventory, leading to an inability to meet customer demand. If these products contribute significantly to the low OFR, there may be issues in order processing or shipping these particular items.
- Product Quality: If these products frequently fail to meet requirements or have quality issues, customers may cancel orders or request returns. This can impact the overall OFR.
- Ineffective Product Management: If these products constitute a large portion of the low OFR, it may be necessary to review how products are managed, from the ordering process and warehouse tracking to the transportation process, to identify the root cause.

By identifying products that contribute significantly to the low OFR, businesses can focus on improving product quality, optimizing the shipping process, and ensuring product availability in the warehouse to meet customer demand.

What we suspect the most at this time is likely a timing issue.

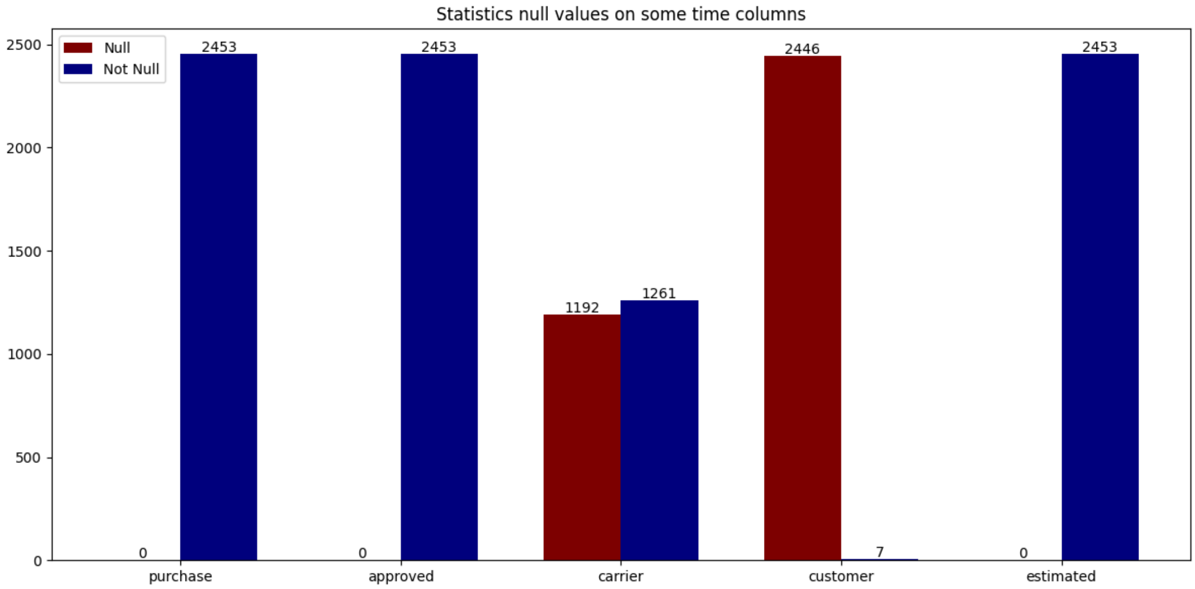


Figure 3.34 Statistic null values

The statistics reveal that null values are only present in the 'order_delivered_carrier_date' and 'order_delivered_customer_date' columns. Specifically:

'order_delivered_carrier_date': Displays the timestamp when the order is submitted for processing by the logistics partner.

'order_delivered_customer_date': Shows the actual delivery date of the order to the customer.

Regarding the 1192 null values in the 'carrier' column: These represent orders that have not been handed over to the logistics partner. Therefore, the business needs to investigate the reasons. There are even 7 non-null values in 'order_delivered_customer_date' after filtering by order status; these may likely be returned orders or instances where customers did not accept the delivery.

The next steps to be taken are:

(1) Identifying the products frequently associated with the 1192 null values in the 'carrier' column.

(2) Examining the order status within the 1261 non-null values in the 'carrier' column.

(3) Investigating the reasons behind the appearance of 7 non-null values in 'order_delivered_customer_date.

```python
# (1) những sản phẩm thường xuyên xuất hiện trong 1192 giá trị null tại cột carrier

# Lọc các dòng mà cột 'carrier' có giá trị null
null_carrier = incompleted_orders_df[incompleted_orders_df['order_delivered_carrier_date'].isnull()]

# Đếm số lần xuất hiện của từng sản phẩm và chọn ra 5 sản phẩm có tần suất xuất hiện cao nhất
top_5_products = null_carrier['product_id'].value_counts().head(5).reset_index()
top_5_products.columns = ['product_id', 'frequency']

# Kết hợp thông tin category từ bảng 'products'
top_5_products_with_category = pd.merge(top_5_products, products, on='product_id', how='left')

# Tạo một màu gradient trên số lượng sản phẩm
colors = plt.cm.plasma(np.linspace(0, 1, len(top_5_products_with_category)))

# Vẽ biểu đồ trực quan hóa kết quả
plt.figure(figsize=(10, 6))
bars = plt.barh(top_5_products_with_category['product_id'], top_5_products_with_category['frequency'], color=colors)
plt.xlabel('Quantity')
plt.title('Nulls in order_delivered_carrier_date')
plt.gca().invert_yaxis()  # Đảo ngược trục y để sản phẩm có số lần lớn nhất ở trên cùng

# Thêm nhãn
plt.bar_label(bars, labels=top_5_products_with_category['product_category_name_english'], label_type='center', color='#deb887')
plt.bar_label(bars, labels=top_5_products_with_category['frequency'].astype(str), label_type='edge', color='black')

plt.show()
```

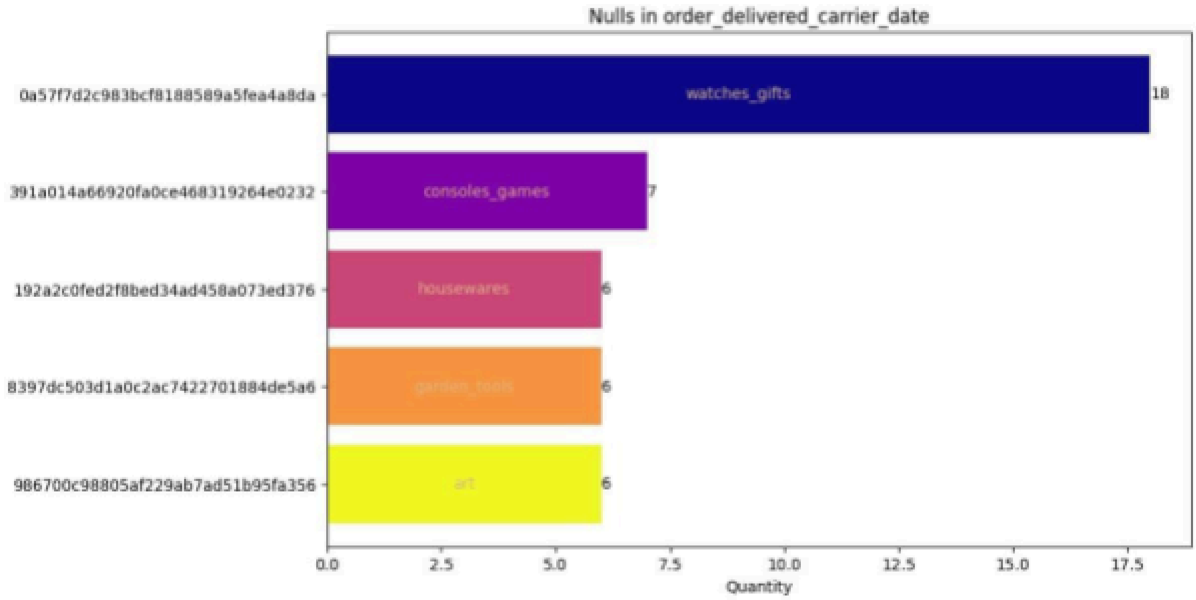


Figure 3.35 Nulls in order_delivered_carrier_date

Among the 1192 null values recorded in the 'order_delivered_carrier_date' column, these represent products that customers have ordered but have not been handed over to the carrier for delivery to customers. We have identified that the majority of these products fall under the category of 'watches_gifts.'

The fact that ordered products are not handed over to the carrier as described above may reflect some issues in the order processing. Products may not be available in the warehouse or may be in short supply due to inefficient inventory management. Alternatively, there could be problems in identifying and processing orders, leading to the improper transfer of products to the carrier. Additionally, products may not be in suitable condition for delivery due to incomplete packaging or failure to meet quality standards, resulting in the products not being dispatched.

Recommend: Optimizing warehouse management to ensure an adequate supply of products, improving the order processing to dispatch products to carriers promptly and accurately, reviewing the packaging process and product quality before delivery, and considering a reassessment of transportation partners to enhance reliability and delivery capabilities are significant enhancements to boost order fulfillment efficiency.

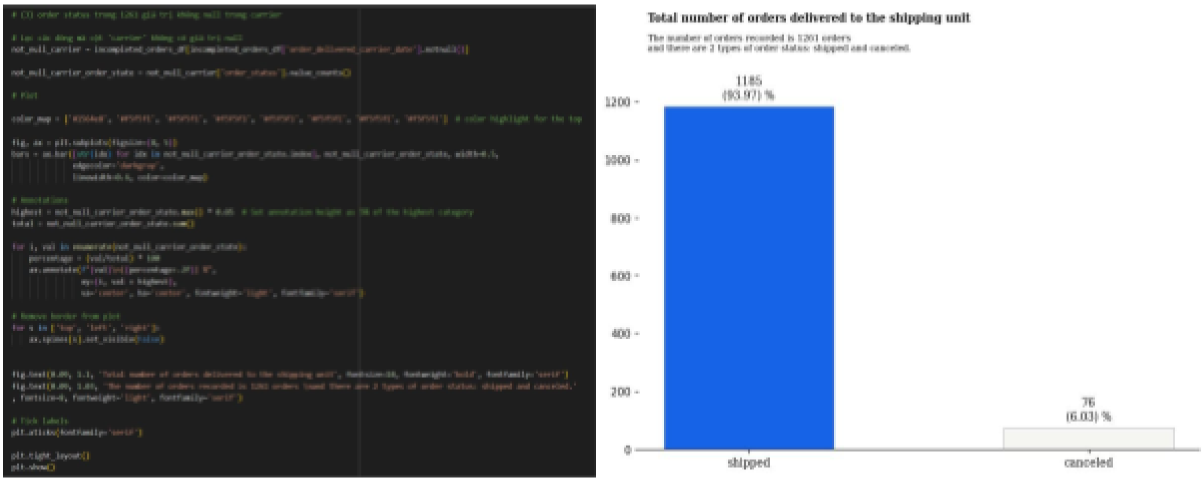


Figure 3.36 Not nulls in order_delivered_carrier_date

The results show that there are 1185 orders currently in the 'shipped' status, accounting for approximately 94%, while a small percentage, around 6%, consists of orders that have been handed over to the carrier but were subsequently canceled by the customers.

Recommend:

For orders currently in the shipping status, businesses need to closely monitor to avoid delayed deliveries or damage to products during transportation. It's crucial to ensure that the business has an effective tracking system to know the exact location of the orders in the delivery process. Updating transportation information will help address issues promptly if any incidents occur. Additionally, maintaining effective communication with customers by providing updates on the delivery process, sending notifications about order status, expected delivery times, and any changes in the schedule is essential. In the event of issues during delivery, such as delays, damages, or losses, the customer support team should be ready and capable of resolving them quickly.

For orders that have been sent to the carrier but are subsequently canceled by customers, businesses need to handle the situation responsibly to ensure customer satisfaction and optimize product inventory management. If the order has not yet left or is in transit, immediate contact with the carrier is necessary to request order cancellation or redirect it to the business's warehouse. Additionally, it's important to have clear and fair order cancellation policies for customers. Information about order cancellation and the refund process should be clearly communicated. Providing support to customers in canceling orders and processing refunds is crucial. The customer support team should handle this situation professionally and assist customers following established procedures. On the other hand, if the product has already left the business's warehouse, when the product is returned due to order cancellation, ensure to inspect the quality and condition of the product before issuing a refund to the customer. Finally, use insights from order cancellation cases to improve the order placement, processing, and delivery processes to minimize the order cancellation rate.

```python
# (2) nguyên nhân xuất hiện 7 giá trị không null trong order_delivered_customer_date

# Lọc những dòng có giá trị không null trong cột 'order_delivered_customer_date'
not_null_rows = incompleted_orders_df[incompleted_orders_df['order_delivered_customer_date'].notnull()]

# Hiển thị những dòng có giá trị không null
not_null_rows
```

	order_id	customer_id	order_status	order_purchase_timestamp
3267	1950d777989f6a877539f53795b4c3c3	1bccb206de9f0f25adc6871a1bcf77b2	canceled	2018-02-19 19:48:52
9893	dabf2b0e35b423f94618bf965fcb7514	5cdec0bb8cbdf53ffc8fdc212cd247c6	canceled	2016-10-09 00:56:52
9894	dabf2b0e35b423f94618bf965fcb7514	5cdec0bb8cbdf53ffc8fdc212cd247c6	canceled	2016-10-09 00:56:52
65923	770d331c84e5b214bd9dc70a10b829d0	6c57e6119369185e575b36712766b0ef	canceled	2016-10-07 14:52:30
67110	8beb59392e21af5eb9547ae1a9938d06	bf609b5741f71697f65ce3852c5d2623	canceled	2016-10-08 20:17:50
104879	65d1e226dfaeb8cdc42f665422522d14	70fc57eeae292675927697fe03ad3ff5	canceled	2016-10-03 21:01:41
106935	2c45c33d2f9cb8ff8b1c86cc28c11c30	de4caa97afa80c8eeac2ff4c8da5b72e	canceled	2016-10-09 15:39:56

7 rows × 24 columns

Figure 3.37 - 7 non-nulls in order_delivered_customer_date

It is easy to recognize that the 7 orders delivered to customers were filtered out with 'order_status != 'delivered,' yet they still remained in the canceled status. Therefore, we can infer that these are 7 orders that customers returned or refused to accept upon delivery by the shipper. The reasons could be products not meeting user requirements, lack of confirmation from users, difficulties in updating information, or inaccurate addresses. However, we still need to consider the business's service factors, especially regarding delivery times. If the delivery time is too long, users are likely to cancel their orders.

```python
# Tính thời gian đóng gói
not_null_rows['packing_time'] = (not_null_rows['order_approved_at'] - not_null_rows['order_purchase_timestamp'])

print("Thời gian đóng gói của 7 sản phẩm bị khách hàng cancel là:\n",not_null_rows['packing_time'])
```

```
Thời gian đóng gói của 7 sản phẩm bị khách hàng cancel là:
 3267     0 days 01:07:13
9893      0 days 12:40:06
9894      0 days 12:40:06
65923     0 days 00:14:40
67110     0 days 18:16:40
104879    0 days 13:17:16
106935    0 days 19:00:53
```

Figure 3.38 Packaging Time

Firstly, we need to assess the packaging time for these 7 orders. It is easy to recognize that the time from when the buyer placed the order until the order was packaged varies from 1 to 19 hours (within 1 day).

	product_id	product_category_name_english
0	e435ceb7ced9b8446defd858630454ed	health_beauty
1	473795a355d29305c3ea6b156833adf5	perfumery
2	7cd29da0653abeb444703cc5a957f479	health_beauty
3	eba7488e1c67729f045ab43fac426f2e	perfumery
4	c66def7098f4d87751f40c3a4855ced1	toys
5	9c7bdf67b06b419aefb93cfdfc96c55d	sports_leisure
6	e24f73b7631ee3fbb2ab700a9acaa258	fashion_bags_accessories

Figure 3.39 Products of 7 nulls

We believe that the packaging time ranging from 1 to 19 hours for products in categories such as 'health_beauty,' 'perfumery,' 'toys,' 'sports_leisure,' 'fashion_bags_accessories' is quite reasonable as they are completed within a day. Now, let's examine the next factor, which is the transportation time.

```python
# Thời gian giao hàng
print("Thời gian giao hàng của 7 sản phẩm bị khách hàng cancel là:\n",not_null_rows['delivery_time'])
```

```
Thời gian giao hàng của 7 sản phẩm bị khách hàng cancel là:
 3267     30 days 02:14:59
9893       7 days 13:40:07
9894       7 days 13:40:07
65923      7 days 00:14:41
67110     10 days 22:29:53
104879    35 days 13:56:53
106935    30 days 23:13:54
Name: delivery_time, dtype: timedelta64[ns]
```

Figure 3.40 Delivery Time

Recommend: The delivery time statistics indicate a considerable delay, with the fastest delivery taking up to 7 days to reach consumers. There are even 2 orders that took up to 30 days, equivalent to a month in the lunar calendar, for transportation. This is a noteworthy issue, as customers experiencing prolonged wait times may lead to order cancellations, posing a risk of future revenue loss for the business if slow delivery persists. Therefore, the business needs to closely manage the delivery department or implement stricter policies for carriers to address this issue. On the other hand, the business should handle the situation flexibly and professionally to ensure customer satisfaction and maintain the company's reputation.

Firstly, it is crucial to contact customers as soon as it is noticed that the delivery time will exceed the expected timeframe. Provide accurate and detailed information about the reasons for the delay, which could be related to transportation issues, inventory, or unforeseen circumstances. Secondly, propose solutions or options for customers, such as refunds, additional products, or discounts on future orders, as a way to compensate for the inconvenience. Thirdly, if possible, try to meet customer expectations by offering faster delivery options or customizing services to suit their needs. Finally, document the issues causing delays and conduct an analysis to improve processes. This includes reviewing the order placement, shipping, and inventory management processes to prevent similar issues in the future. In all situations, communication and flexibility are key to maintaining customer relationships. Importantly, the business should handle the situation professionally and support customers as best as possible.

Chapter 4. Explore Dataset by using ChatGPT

4.1 Describe dataset

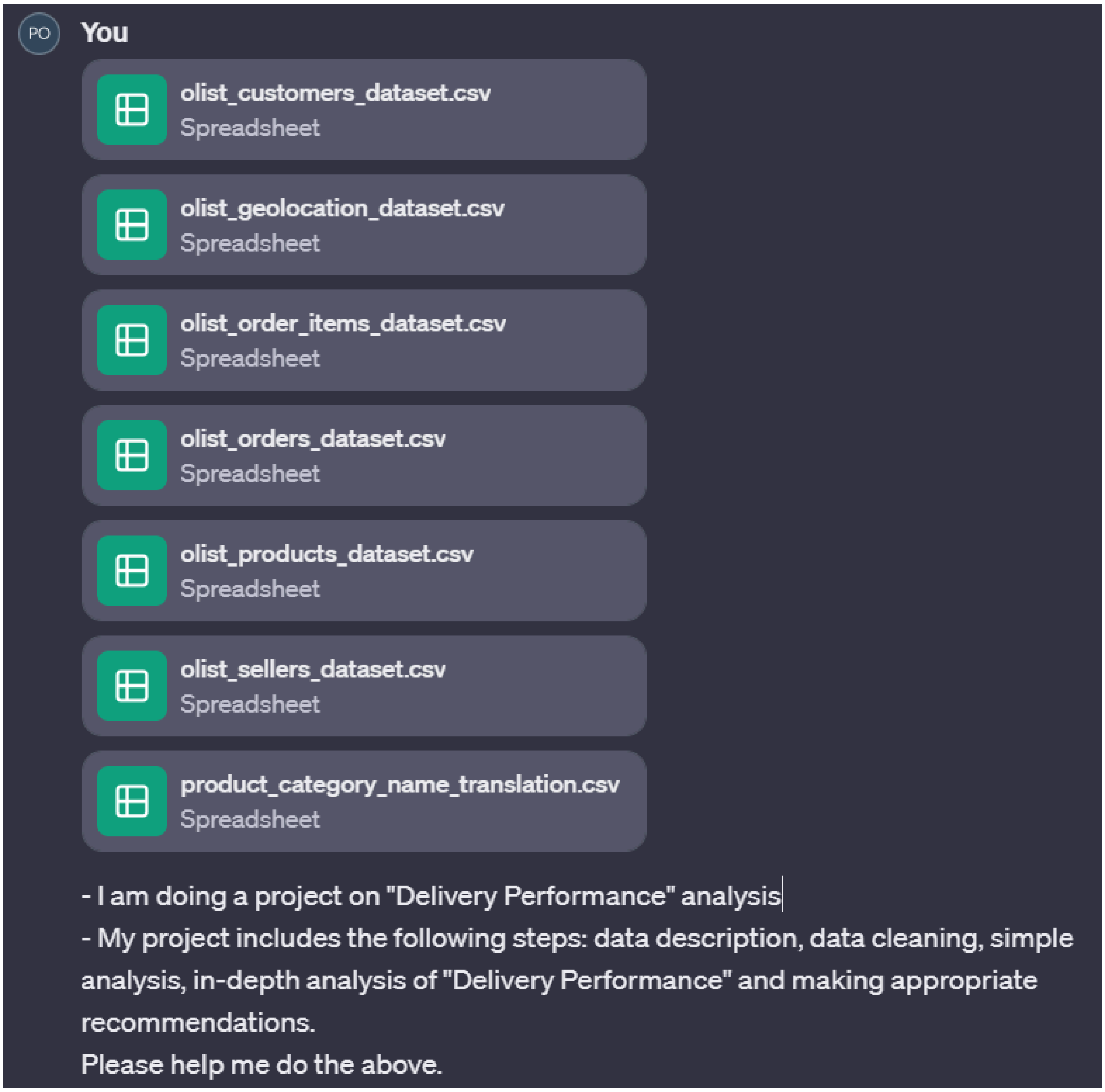


Figure 4.1 - Request

First, I provided ChatGPT with some requirements about my "Delivery Performance" problem.

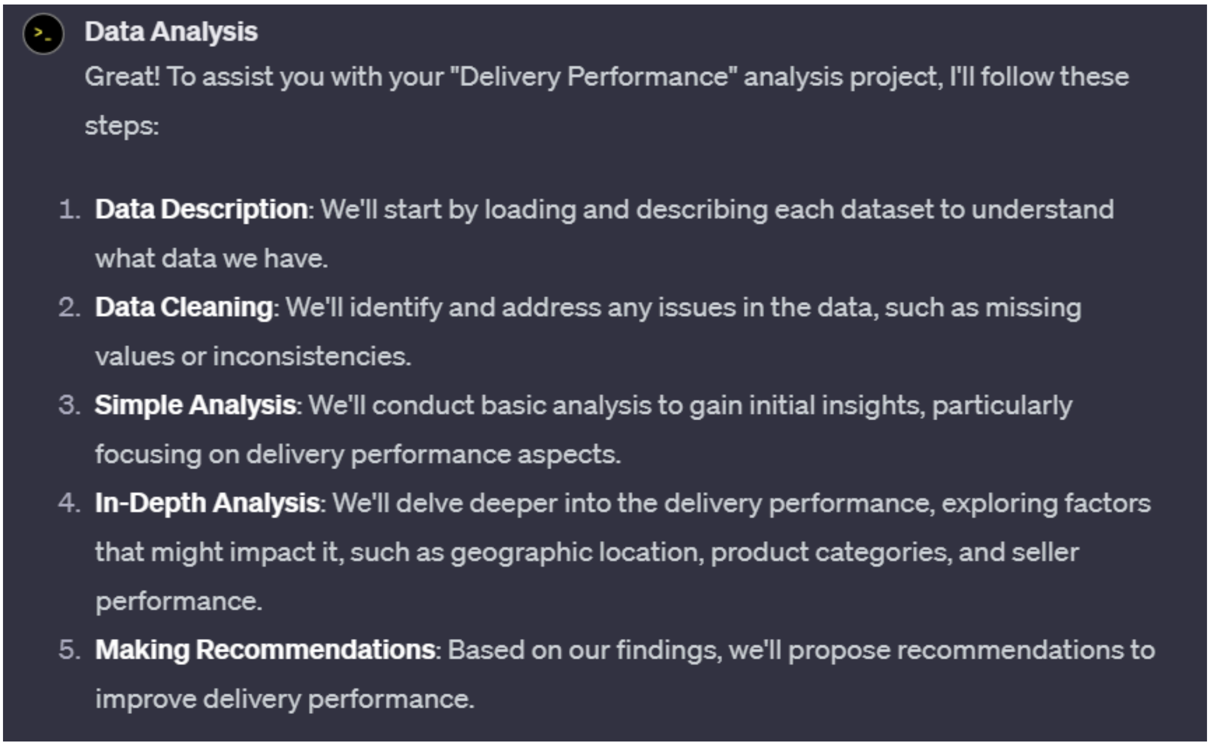


Figure 4.2 - Process Description

Next, ChatGTP understood the request by describing each of the steps I mentioned.

The data set was then described in quite some detail. For each data table, ChatGPT has described the number of rows, columns, names of each column and the meaning of the data of that table. This helps us understand the data set extremely quickly instead of searching online.

Let's begin with the first step, Data Description. We'll load each dataset and briefly describe its contents and structure.

Here's a brief overview of each dataset:

1. **Customers Dataset**
   - **Shape**: 99,441 rows × 5 columns
   - **Columns**: customer_id, customer_unique_id, customer_zip_code_prefix, customer_city, customer_state
   - **Description**: Contains information about customers, including their location details.
2. **Geolocation Dataset**
   - **Shape**: 1,000,163 rows × 5 columns
   - **Columns**: geolocation_zip_code_prefix, geolocation_lat, geolocation_lng, geolocation_city, geolocation_state
   - **Description**: Provides geolocation data for Brazilian zip codes.
3. **Order Items Dataset**
   - **Shape**: 112,650 rows × 7 columns
   - **Columns**: order_id, order_item_id, product_id, seller_id, shipping_limit_date, price, freight_value
   - **Description**: Details about each item in an order, including the product, seller, price, and shipping information.
4. **Orders Dataset**
   - **Shape**: 99,441 rows × 8 columns
   - **Columns**: order_id, customer_id, order_status, order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_date
   - **Description**: Contains information about the orders, such as status, purchase timestamp, and delivery dates.
5. **Products Dataset**
   - **Shape**: 32,951 rows × 9 columns
   - **Columns**: product_id, product_category_name, product_name_lenght, product_description_lenght, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm
   - **Description**: Includes details about the products, such as category, dimensions, and weight.
6. **Sellers Dataset**
   - **Shape**: 3,095 rows × 4 columns
   - **Columns**: seller_id, seller_zip_code_prefix, seller_city, seller_state
   - **Description**: Information about the sellers, including their location.
7. **Category Translation Dataset**
   - **Shape**: 71 rows × 2 columns
   - **Columns**: product_category_name, product_category_name_english
   - **Description**: Translation of product category names from Portuguese to English.

Figure 4.3 Data description

4.2 Data cleaning

Next, we'll move to the Data Cleaning step, where we'll check for any missing or inconsistent data in these datasets. Let's start with identifying missing values in each dataset. [>-]

Here's the summary of missing values in each dataset:

1. **Customers Dataset**: No missing values.
2. **Geolocation Dataset**: No missing values.
3. **Order Items Dataset**: No missing values.
4. **Orders Dataset**: Missing values in `order_approved_at` (160), `order_delivered_carrier_date` (1,783), and `order_delivered_customer_date` (2,965).
5. **Products Dataset**: Missing values in `product_category_name`, `product_name_lenght`, `product_description_lenght`, `product_photos_qty` (all 610 missing), and very few in `product_weight_g`, `product_length_cm`, `product_height_cm`, `product_width_cm` (2 each).
6. **Sellers Dataset**: No missing values.
7. **Category Translation Dataset**: No missing values.

To address these missing values:

- For the `Orders` dataset, the missing delivery dates might be due to uncompleted orders or delays in data entry. We'll need to decide how to handle these based on the analysis focus.
- For the `Products` dataset, missing product details could be imputed or these rows could be dropped, depending on their importance to our analysis.

Would you like to proceed with a particular method for handling these missing values, or should we move on to the simple analysis phase and revisit these issues later if necessary? [>-]

Figure 4.4 - Data cleaning 1

After completing the description, ChatGPT analyzed the missing value factor. It can scan each line in the dataset and count how many values are missing. The result is that tables such as Orders and Products are tables that contain missing values. The reason was also given that: For the Orders data set, a missing delivery date could be due to an incomplete order or a delay in data entry. ChatGPT was very careful because it also asked me if I needed to handle this issue or move on to the analysis step.

Figure 4.5 - Data cleaning 2

I asked ChatGPT to handle this missing value issue for me and they did a pretty good job.

4.3 Analysis

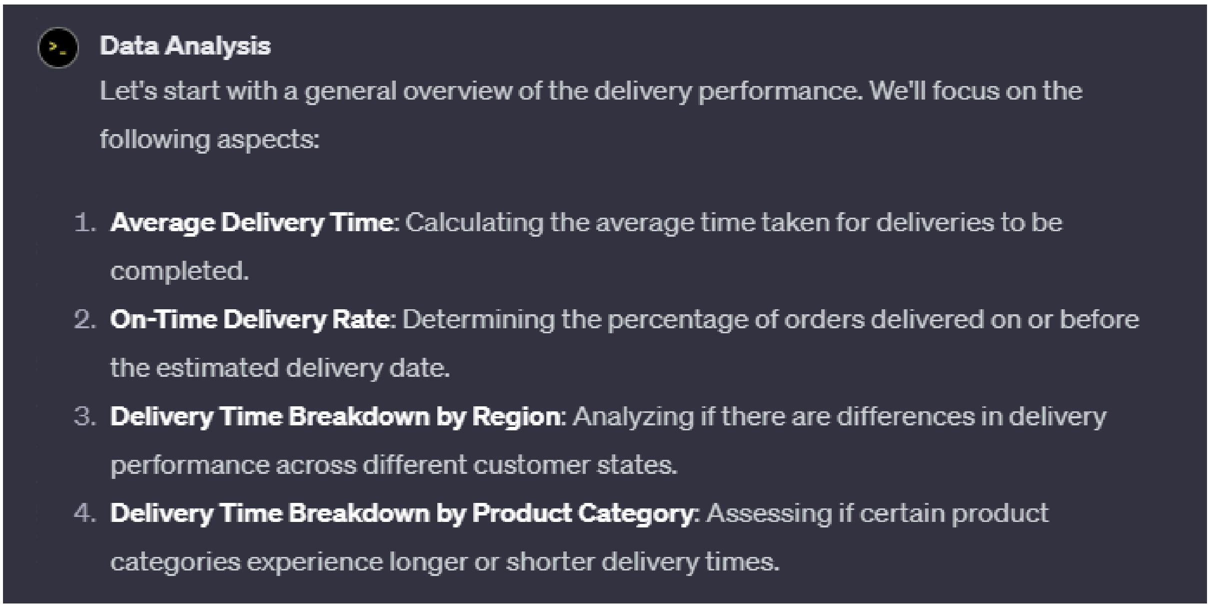


Figure 4.6 - Suggested analytical direction

Once the data was cleaned, ChatGPT suggested four areas that I should focus on. I find these 4 ideas quite good.

We will use the `orders_complete_df` for accurate delivery time calculations as it has complete delivery date information. Let's start with the first analysis: calculating the average delivery time.

The average delivery time, based on the `orders_complete_df` dataset, is approximately **12.56 days**. This is the time taken from the order purchase timestamp to the date when the customer received the order.

Next, let's calculate the On-Time Delivery Rate. For this, we'll compare the actual delivery date (`order_delivered_customer_date`) with the estimated delivery date (`order_estimated_delivery_date`) and determine the percentage of orders delivered on or before the estimated date. [-]

The On-Time Delivery Rate is approximately **91.89%**. This means that around 91.89% of the orders in the dataset were delivered on or before the estimated delivery date.

Now, let's analyze the delivery performance across different customer states. We'll examine if there are any notable differences in delivery times among regions. To do this, we'll need to merge the `orders_complete_df` dataset with the `customers_df` dataset to get customer location information. Let's proceed with this analysis. [-]

The average delivery time varies significantly across different customer states. Here's a summary:

- **Fastest Delivery**: São Paulo (SP) with an average of approximately **8.76 days**.
- **Slowest Delivery**: Roraima (RR) with an average of about **29.39 days**.

This data shows a clear geographic trend, where customers in states closer to the major logistics centers (like SP) tend to receive their orders faster than those in more remote areas.

Finally, let's analyze the delivery time breakdown by product category. For this, we will need to merge the `orders_complete_df` with the `products_df` and `order_items_df` to associate each order with its respective product category. Let's perform this analysis. [-]

1. **Fastest Delivery Categories:**
   - "Arts and Craftsmanship": Approximately 5.72 days.
   - "La Cuisine": Around 7.61 days.
   - "Books Imported": About 8.13 days.
2. **Additional Categories:**
   - "Fashion Children's Clothes": 8.82 days.
   - "Party Supplies": 9.33 days.
   - "Food": 9.56 days.

These categories represent the ones with the shortest average delivery times in the dataset. It's interesting to note that categories like arts, cuisine-related items, and imported books tend to have faster deliveries.

Now, let's take a look at some of the categories with longer delivery times to understand the other end of the spectrum. This could highlight potential areas for logistics improvements or challenges in delivering certain types of products. We'll examine the categories with the longest delivery times.

Here are some of the product categories with the longest average delivery times:

1. **Office Furniture**: Around 20.84 days.
2. **Christmas Supplies**: Approximately 15.74 days.
3. **Fashion Shoes**: About 15.44 days.
4. **Security and Services**: Roughly 15.14 days.
5. **Home Comfort 2**: Nearly 14.57 days.

These longer delivery times might be due to various factors like product size, packaging complexity, or limited availability. For example, items like office furniture are typically larger and require more careful handling and shipping, which could contribute to longer delivery times. Similarly, seasonal items like Christmas supplies might have specific shipping schedules.

This overview gives us a good starting point to understand the delivery performance in different dimensions. The next step is to delve deeper into specific areas of interest or concern, like exploring the reasons behind delayed deliveries in certain regions or categories, assessing the impact of factors like order size or shipping methods, or even evaluating the performance of individual sellers.

Based on these insights, we can make targeted recommendations to improve delivery performance. If you have specific areas you'd like to explore further or need recommendations based on these findings, please let me know! [-]

Figure 4.7 - Results of Analysis

ChatGPT analyzed the 4 aspects mentioned quite quickly:

(1) The average delivery time is about 12.56 days.

(2) The On-Time Delivery Rate is approximately 91.89%.

(3) Delivery time by region:

- Fastest delivery: São Paulo (SP) with an average time of about 8.76 days.
- Slowest delivery: Roraima (RR) with an average time of about 29.39 days.

(4) Delivery time by product:

- Fastest delivery category: "Arts and Craftsmanship": Approximately 5.72 days. "La Cuisine": Around 7.61 days. "Books Imported": About 8.13 days.
- Latest delivery category: Office Furniture: Around 20.84 days. Christmas Supplies: Approximately 15.74 days. Fashion Shoes: About 15.44 days. Security and Services: Roughly 15.14 days. Home Comfort 2: Nearly 14.57 days.

Not only does ChatGPT show the numbers, but it also explains them as if it were a real analyst:

- *"The On-Time Delivery Rate is approximately 91.89%. This means that around 91.89% of the orders in the dataset were delivered on or before the estimated delivery date."*

- "Fastest Delivery: São Paulo (SP) with an average of approximately 8.76 days.
Slowest Delivery: Roraima (RR) with an average of about 29.39 days.
This data shows a clear geographic trend, where customers in states closer to the major logistics centers (like SP) tend to receive their orders faster than those in more remote areas."

- "Office Furniture: Around 20.84 days.
Christmas Supplies: Approximately 15.74 days.
Fashion Shoes: About 15.44 days.
Security and Services: Roughly 15.14 days.
Home Comfort 2: Nearly 14.57 days.
These longer delivery times might be due to various factors like product size, packaging complexity, or limited availability. For example, items like office furniture are typically larger and require more careful handling and shipping, which could contribute to longer delivery times. Similarly, seasonal items like Christmas supplies might have specific shipping schedules."

Besides, ChatGPT can visualize the results after analysis.

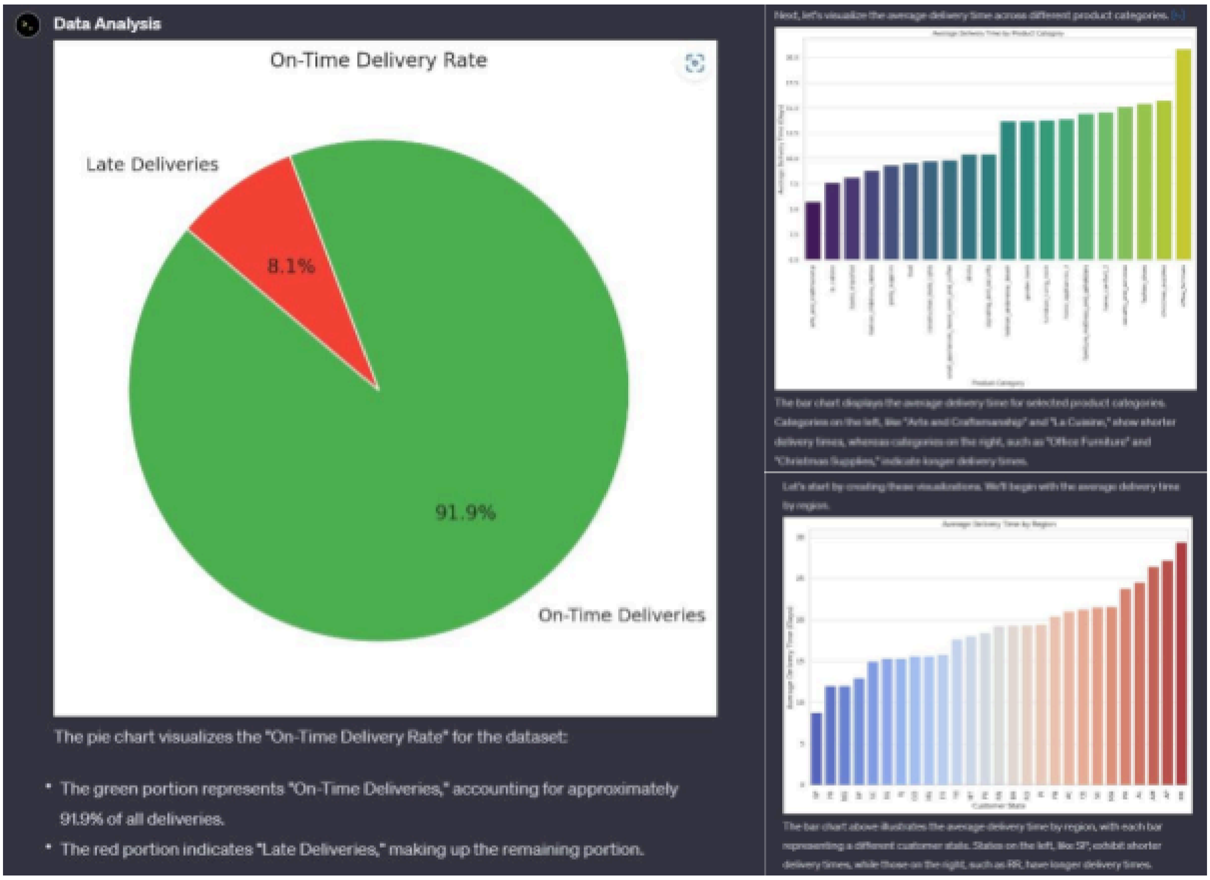
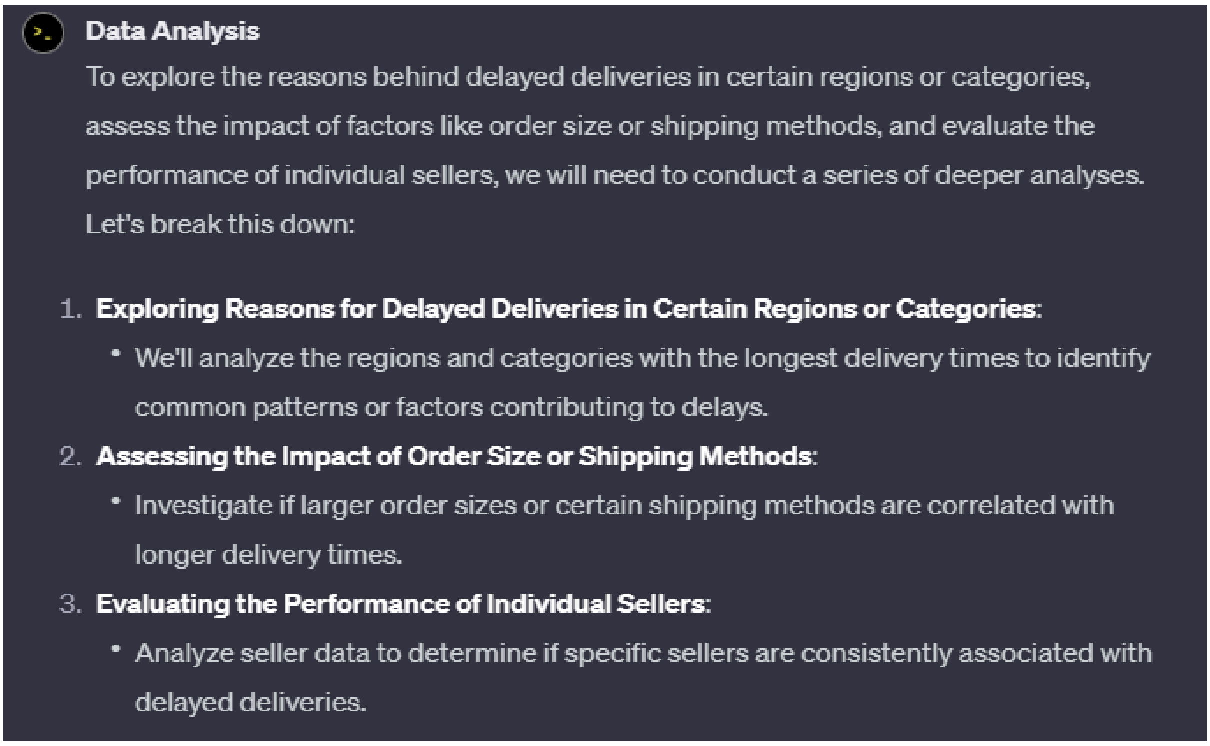


Figure 4.8 - Visualize results

ChatGPT then offered me an opportunity to explore the reasons behind delayed deliveries in certain regions or categories, assessing the impact of factors such as order size or shipping method. transfer and evaluate the performance of each seller. And ChatGPT does not disappoint me, it analyzes very well:



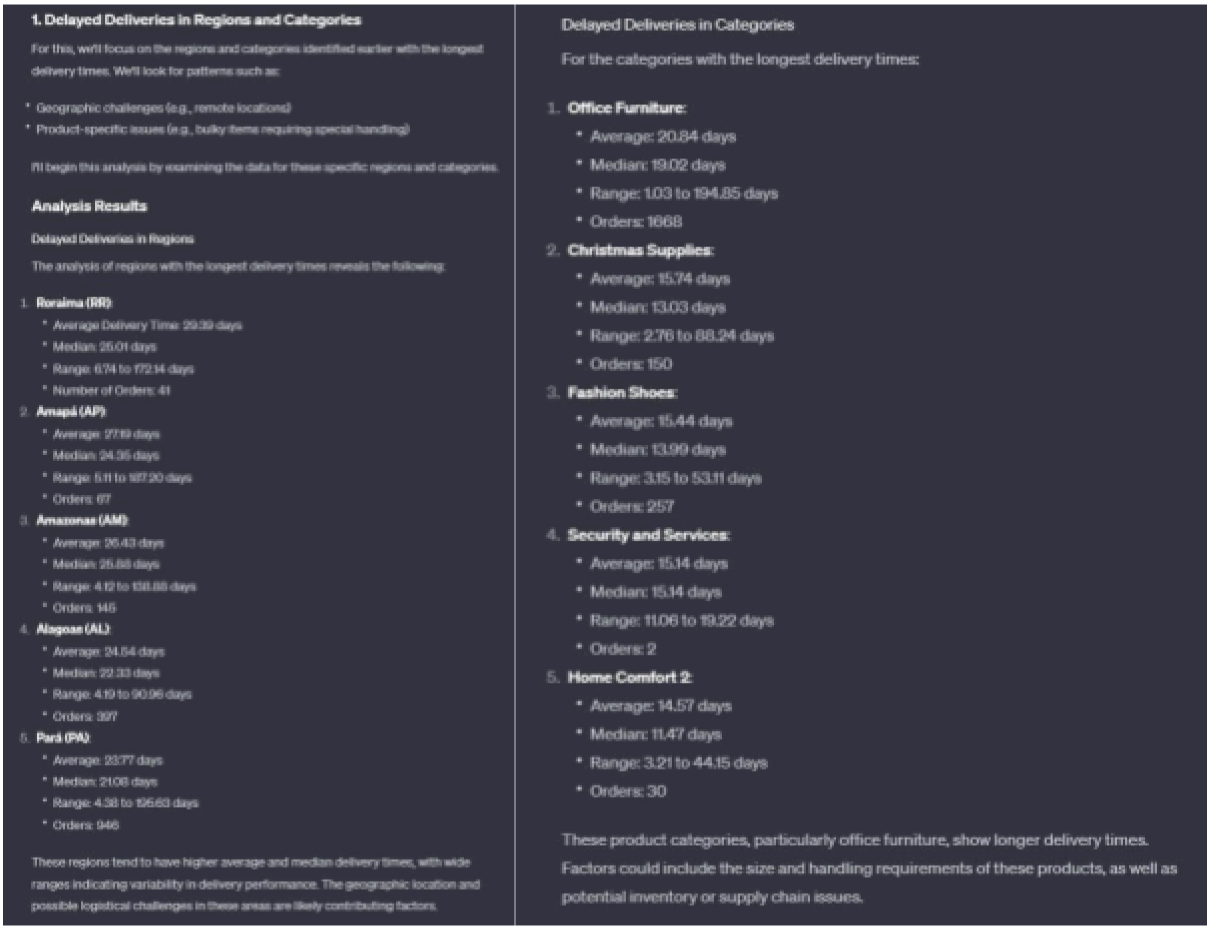
Data Analysis

To explore the reasons behind delayed deliveries in certain regions or categories, assess the impact of factors like order size or shipping methods, and evaluate the performance of individual sellers, we will need to conduct a series of deeper analyses. Let's break this down:

1. **Exploring Reasons for Delayed Deliveries in Certain Regions or Categories**:
   - We'll analyze the regions and categories with the longest delivery times to identify common patterns or factors contributing to delays.
2. **Assessing the Impact of Order Size or Shipping Methods**:
   - Investigate if larger order sizes or certain shipping methods are correlated with longer delivery times.
3. **Evaluating the Performance of Individual Sellers**:
   - Analyze seller data to determine if specific sellers are consistently associated with delayed deliveries.

Figure 4.9 Explore reasons of delayed deliveries

The analysis results are as follows:



1. Delayed Deliveries in Regions and Categories

For this, we'll focus on the regions and categories identified earlier with the longest delivery times. We'll look for patterns such as:

- Geographic challenges (e.g., remote locations)
- Product-specific issues (e.g., bulky items requiring special handling)

I'll begin this analysis by examining the data for these specific regions and categories.

Analysis Results

Delayed Deliveries in Regions

The analysis of regions with the longest delivery times reveals the following:

1. **Roraima (RR):**
   - Average Delivery Time: 29.39 days
   - Median: 25.01 days
   - Range: 6.74 to 172.14 days
   - Number of Orders: 41
2. **Amapá (AP):**
   - Average: 27.19 days
   - Median: 24.35 days
   - Range: 5.11 to 187.20 days
   - Orders: 67
3. **Amazonas (AM):**
   - Average: 26.43 days
   - Median: 25.88 days
   - Range: 4.12 to 138.88 days
   - Orders: 145
4. **Alagoas (AL):**
   - Average: 24.54 days
   - Median: 22.33 days
   - Range: 4.19 to 90.96 days
   - Orders: 397
5. **Pará (PA):**
   - Average: 23.77 days
   - Median: 21.08 days
   - Range: 4.38 to 195.63 days
   - Orders: 946

These regions tend to have higher average and median delivery times, with wide ranges indicating variability in delivery performance. The geographic location and possible logistical challenges in these areas are likely contributing factors.

Delayed Deliveries in Categories

For the categories with the longest delivery times:

1. **Office Furniture:**
   - Average: 20.84 days
   - Median: 19.02 days
   - Range: 1.03 to 194.85 days
   - Orders: 1668
2. **Christmas Supplies:**
   - Average: 15.74 days
   - Median: 13.03 days
   - Range: 2.76 to 88.24 days
   - Orders: 150
3. **Fashion Shoes:**
   - Average: 15.44 days
   - Median: 13.99 days
   - Range: 3.15 to 53.11 days
   - Orders: 257
4. **Security and Services:**
   - Average: 15.14 days
   - Median: 15.14 days
   - Range: 11.06 to 19.22 days
   - Orders: 2
5. **Home Comfort 2:**
   - Average: 14.57 days
   - Median: 11.47 days
   - Range: 3.21 to 44.15 days
   - Orders: 30

These product categories, particularly office furniture, show longer delivery times. Factors could include the size and handling requirements of these products, as well as potential inventory or supply chain issues.

Figure 4.10 - Result of Reason Analysis 1

With Delayed Deliveries in Regions, ChatGPT gave the following reasoning: The above regions tend to have higher mean and median delivery times, with a wide range indicating variation in delivery performance. Geographic location and possible logistical challenges in these areas may be contributing factors.

With Delayed Deliveries in Categories, ChatGPT gave the following reason: These product categories, especially office furniture, have longer delivery times. Factors may include the size and handling requirements of these products, as well as potential inventory or supply chain issues.

Seller Performance Analysis

Here are insights into the delivery performance of individual sellers:

Top Performing Sellers (Shortest Average Delivery Times):

1. Seller ID `139157dd4daa45c25b0807ffff348363`: Average delivery time ~ 1.21 days.
2. Seller ID `5e063e85d44b0f5c3e6ec3131103a57e`: Average delivery time ~ 1.29 days.
3. Seller ID `6561d6bf844e464b4019442692b40e02`: Average delivery time ~ 1.43 days.
4. Others in the top 10 range from ~1.80 to ~2.05 days.

Bottom Performing Sellers (Longest Average Delivery Times):

1. Seller ID `df683dfda87bf71ac3fc63063fba369d`: Average delivery time ~ 189.86 days.
2. Seller ID `8e670472e453ba34a379331513d6aab1`: Average delivery time ~ 86.00 days.
3. Seller ID `586a871d4f1221763fddb6ceefdeb95e`: Average delivery time ~ 68.62 days.
4. Others in the bottom 10 range from ~48.18 to ~66.75 days.

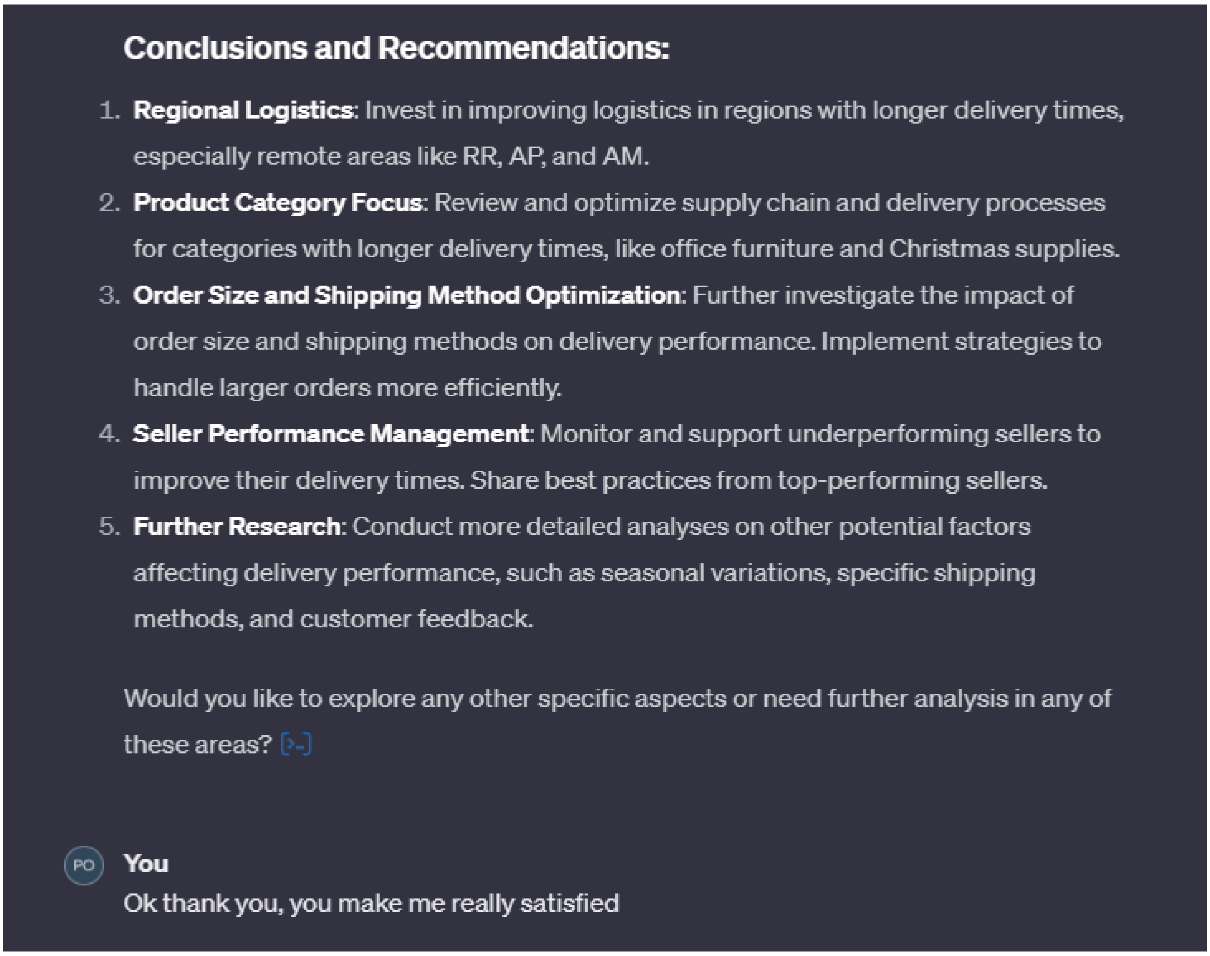
These results indicate significant variability in delivery performance among sellers. The top performers consistently achieve very short delivery times, while the bottom performers have much longer delivery times, with the most extreme case averaging nearly 190 days.

The visualizations above represent the performance of individual sellers:

1. **Left Chart (Top Performing Sellers)**: Shows sellers with the shortest average delivery times. The shorter the bar, the faster the delivery.
2. **Right Chart (Bottom Performing Sellers)**: Depicts sellers with the longest average delivery times. Longer bars indicate slower delivery performance.

Figure 4.11 - Result of Reason Analysis 2

The results showed the average sales delivery performance of each seller. The fastest person only needs 1.21 days to deliver the goods and the slowest person takes up to 2 months to deliver the goods. And the company's health is only as good as its fast delivery speed.



Conclusions and Recommendations:

1. **Regional Logistics**: Invest in improving logistics in regions with longer delivery times, especially remote areas like RR, AP, and AM.
2. **Product Category Focus**: Review and optimize supply chain and delivery processes for categories with longer delivery times, like office furniture and Christmas supplies.
3. **Order Size and Shipping Method Optimization**: Further investigate the impact of order size and shipping methods on delivery performance. Implement strategies to handle larger orders more efficiently.
4. **Seller Performance Management**: Monitor and support underperforming sellers to improve their delivery times. Share best practices from top-performing sellers.
5. **Further Research**: Conduct more detailed analyses on other potential factors affecting delivery performance, such as seasonal variations, specific shipping methods, and customer feedback.

Would you like to explore any other specific aspects or need further analysis in any of these areas?

PO

You

Ok thank you, you make me really satisfied

Figure 4.12 - Conclusion and Recommendations

Finally, ChatGPT has made some suggestions but in general these suggestions are only for reference because they are quite shallow.

Chapter 5. Comparison of analytics performance between Python and ChatGPT

5.1 Traditional Analytics with Python

In this project, the team performed delivery performance analysis using Python. In this way, the team made full use of supporting libraries such as Pandas, NumPy, Matplotlib, and Seaborn for the analysis process.

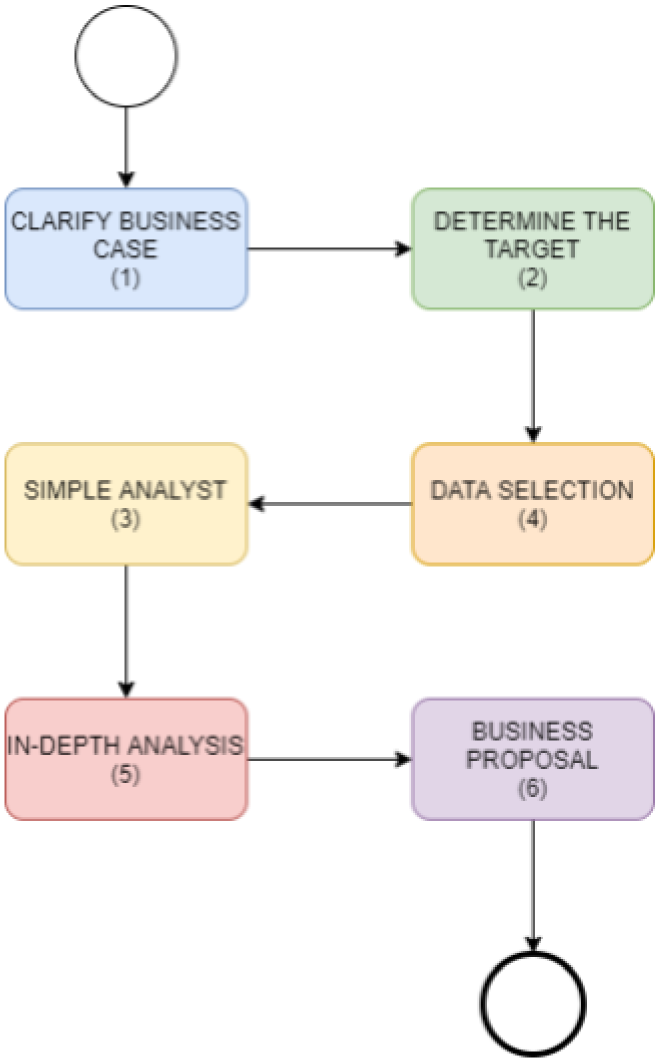


Figure 5.1 - Process Analyst

First, the team needs to build an analysis process from clarifying the business case, then determining the goals. Once the first two steps have been completed, the team needs to select relevant data to avoid unnecessary information redundancy. Finally, start analyzing from simple to in-depth to find insights. At the in-depth analysis step, the information considered key to the analysis is information about on-time delivery rate and successfully delivered order rate. When information is available, the group also provides assessments, comments and suggestions for businesses to improve transportation performance.

Building an analytical framework requires careful calculation and measurement, which is why it is quite time-consuming.

5.2 Analysis using ChatGPT

With ChatGPT4's Data Analyst package, analysis becomes quite simple and fast. We just need to provide the data sets we have and the requirements for the topic we are aiming for, it will automatically suggest analysis options from data cleaning to analysis and recommendations in business.

However, the output information that ChatGPT analyzes is quite simple and lacking in diversity. It only shows the most popular results without digging deep into our data set. This is quite understandable because ChatGPT's nature is to synthesize information and suggest things that are highly likely to be needed. But this is not really reasonable for a data analyst. When doing in-depth analysis, we need to synthesize information from many other aspects and evaluate each other so that the final result is truly good and useful.

To be more general, we will need to look at the following table:

Table 5.1 - Analysis performance comparison (Python/ChatGPT)

Feature	Python	Chat GPT4 (Data Analyst Pakage)
Accuracy	High	Lower
Flexible	High	Lower
Customize	High	Lower
Easy to use	Lower	High
Time	Could be long	Fast
Requires skills	High	Lower
Suitable for	Projects require high precision and high customization capabilities	Projects need quick, easy data analysis

5.3 Proposing a combined analysis method

Data analysis is becoming an indispensable part in many fields, from business to scientific research. To capture information from complex data, combining Python - a powerful programming language - with ChatGPT - an artificial intelligence tool capable of natural language processing - offers a powerful method strong and effective.

Python has become a popular tool in data analysis because of its flexibility and strong community support. Using libraries like Pandas, NumPy, and Matplotlib in Python helps perform complex analyzes from collecting and processing data to displaying results in an intuitive and easy-to-understand manner.

Combined with Python, ChatGPT - an artificial intelligence system based on deep learning models - can bring special benefits in analyzing and understanding data in natural language. ChatGPT is capable of handling requests, questions and linguistic interactions from users, helping to assist in better understanding the data content and providing detailed analysis based on specific requirements.

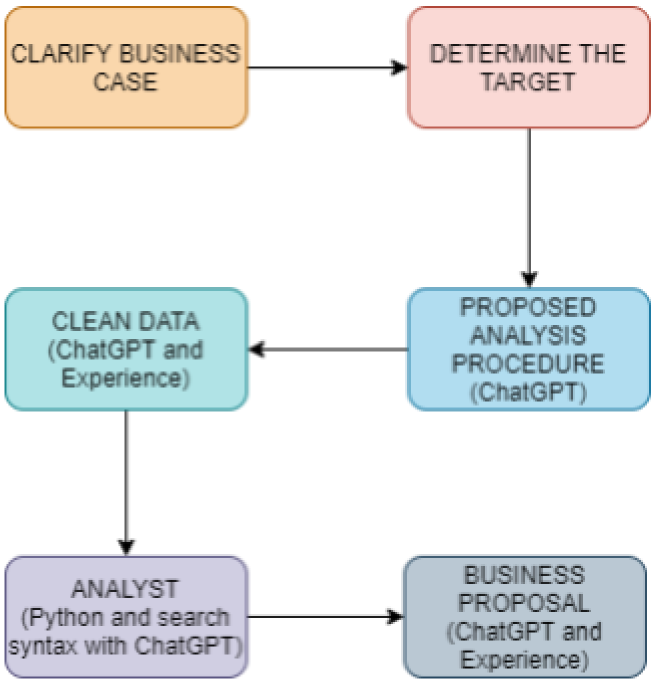


Figure 5.2 - Proposed analysis method

However, it should be noted that combining Python and ChatGPT also poses some challenges such as ensuring the accuracy of natural language understanding or optimizing the performance of the analysis process. This requires constant control and refinement during development and deployment.

REFERENCE

[1] ChatGPT (openai.com)
[2] 10 Ways to Improve Your Delivery Performance (linkedin.com)
[3] 8 ways to enhance your delivery process | Odoo
[4] How to Improve Delivery Efficiency (10 Ways to Improve… | FarEye
[5] Key Metrics to Improve Delivery Performance (linkedin.com)