DEPI

Digital Egypt Pioneers

2025

E-COMMERCE SHOPPING CART

Data Analysis Project

Project Profile

- **Project Sponsor:** Digital Egyptian Pioneer Initiative
- **Project Supervisor:** Eng. Dina Ezzat
- **Project Manager:** Abdelrhman El-Sayed Saleem
- **Technical Provider:** New Horizon Alex
- **Group Code:** ALX2_DAT1_S4
- **Track:** Google Data Analyst Specialist
- **Project Name:** E-commerce Shopping Cart
- **Due Date:** 11-5-2025

Team Members

Name	Intrast E-mail	Role
Abdelrhman El-Sayed Ahmedkamal El-Sayed	AbdelrhmanElSayedAhmedkamal @intrast.com.eg	Project Manager and Lead Data Analyst
Abdelrhman Mohamed Ahmed Abo El-Ela	AbdelrhmanMohamedAhmed @intrast.com.eg	Business Intelligence and Data Visualization Specialist
Mohamed Ahmed Anter Abd El-Hamid	MohamedAhmedAnter @intrast.com.eg	Data Preprocessing Engineer

Table of Contents

Table of Content

Business Story

In the fast-paced and competitive world of e-commerce, success is driven by data—the numbers behind the purchases, the insights within customer behaviors, and the trends shaping revenue growth. To stay ahead, we embarked on a comprehensive data analysis and mining journey, turning raw data into valuable business intelligence.

We examined our client's e-commerce performance from every angle: revenue growth, product demand, customer distribution, and order trends. Through this project, we transformed numbers into insights that empower smarter decision-making, optimize customer experience, and fuel sustainable business growth.

Business Objectives

- **Find What Drives Sales:** Use data to pinpoint which products, prices, or regions bring in the most revenue.
- **Understand Customers Behaviour:** Analyze purchase and engagement data to group customers by how and when they buy.
- **Track Product Success:** Monitor key metrics, like sales speed, returns, and satisfaction, to spot under-performing items.
- **Improve Customer Retention:** Use data to uncover why customers leave and tailor actions or offers to keep them coming back.
- **Streamline Operations:** Examine process and time-series data to uncover bottlenecks, forecast needs, and cut costs.

Data Exploration

1.Customers Sheet

customer_id	customer_name	gender	age	zip_code	city	state	Age Range
2	Zabrina Harrowsmith	Prefer not to say	50	8223	New Zacharyfort	South Australia	46-60
12	Fedora Dmych	Prefer not to say	78	6334	Taylorburgh	South Australia	60+
20	Gigi Kalaher	Prefer not to say	55	6318	East Audrey	South Australia	46-60
55	Giles Kyne	Prefer not to say	45	9871	Lake Phoenixside	South Australia	36-45
61	Lin Avraam	Prefer not to say	65	4358	Samuelland	South Australia	60+
64	Annabella Devote	Prefer not to say	75	7787	Sanfordborough	South Australia	60+
66	Vasili Shearston	Prefer not to say	34	964	South Elimouth	South Australia	26-35
73	Jehanna Water	Prefer not to say	48	5208	Monahanside	South Australia	46-60
75	Katinka Currier	Prefer not to say	75	9777	Smythland	South Australia	60+
103	Monica Teodorski	Prefer not to say	77	2603	South Nathan	South Australia	60+
113	Hew Cardinal	Prefer not to say	57	3303	North Ruby	South Australia	46-60
114	Cecile Jayume	Prefer not to say	30	9798	Williamsville	South Australia	26-35
122	Katey Dubbin	Prefer not to say	71	935	South Caleb	South Australia	60+
130	Jeniece Dumbare	Prefer not to say	71	6215	New Leahside	South Australia	60+
142	Valerie D'Errico	Prefer not to say	75	6043	Chelseaview	South Australia	60+
159	Marga Longhurst	Prefer not to say	73	482	Bellaburgh	South Australia	60+

Summary:

This sheet contains customer demographic information. It includes a unique Customer ID, customer names, gender distribution, age data, as well as location details such as zip code, city, and state. This dataset provides valuable insights into customer profiles and geographic distribution, which can be useful for market segmentation and targeted marketing analysis

Data Exploration

2.Products Sheet

product_ID	product_type	product_name	size	colour	price	quantity
423	Jacket	Denim	I	red	92.0	63
433	Jacket	Denim	I	yellow	92.0	44
438	Jacket	Denim	I	green	92.0	52
443	Jacket	Denim	I	blue	92.0	42
448	Jacket	Denim	I	indigo	92.0	67
453	Jacket	Denim	I	violet	143.5	50
458	Jacket	Puffer	I	red	110.0	44
463	Jacket	Puffer	I	orange	110.0	66
468	Jacket	Puffer	I	yellow	110.0	46
473	Jacket	Puffer	I	green	110.0	43
483	Jacket	Puffer	I	indigo	110.0	61
488	Jacket	Puffer	I	violet	110.0	49
493	Jacket	Windbreaker	I	red	109.0	40
498	Jacket	Windbreaker	I	orange	109.0	68
508	Jacket	Windbreaker	I	green	109.0	62
513	Jacket	Windbreaker	I	blue	109.0	41

Summary:

This sheet contains detailed information about the products offered. It includes a unique Product ID, the type and name of each product, as well as specifications such as size, color, price, and available quantity. This dataset helps in analyzing the product variety, pricing strategies, inventory levels, and product availability across different categories.

Data Exploration

3.Orders Sheet

order_id	customer_id	payment	order_date	delivery_date
1	64	30811	Monday, August 30, 2021	Friday, September 24, 2021
2	473	33692	Wednesday, February 3, 2021	Saturday, February 13, 2021
3	774	46763	Friday, October 8, 2021	Friday, June 11, 2021
4	433	39782	Friday, May 28, 2021	Wednesday, May 19, 2021
5	441	14719	Tuesday, March 23, 2021	Wednesday, March 24, 2021
6	800	16197	Thursday, September 9, 2021	Tuesday, October 5, 2021
7	626	37666	Monday, April 5, 2021	Sunday, April 11, 2021
8	58	28484	Monday, April 12, 2021	Saturday, May 1, 2021
9	852	12896	Saturday, May 1, 2021	Tuesday, May 11, 2021
10	659	21922	Friday, October 15, 2021	Saturday, October 16, 2021
11	785	36624	Tuesday, June 15, 2021	Wednesday, June 30, 2021
12	120	55507	Wednesday, June 30, 2021	Sunday, July 11, 2021
13	204	57810	Friday, October 15, 2021	Sunday, October 31, 2021
14	957	21270	Monday, March 29, 2021	Monday, April 12, 2021
15	468	15488	Wednesday, September 22, 2021	Friday, October 8, 2021
16	564	36479	Wednesday, July 7, 2021	Thursday, July 8, 2021

Summary:

This sheet provides information related to customer orders. It includes a unique Order ID, the Customer ID associated with each order, the payment method used, as well as the order and delivery dates. This dataset is crucial for tracking order fulfillment, analyzing payment preferences, and understanding delivery timelines.

Data Exploration

4.Sales Sheet

sales_id	order_id	product_id	price_per_unit	quantity	total_revenue
41	9	1061	93	2	186
49	12	632	93	2	186
81	17	1066	93	2	186
132	28	1053	93	2	186
177	37	645	93	2	186
219	47	664	93	2	186
237	51	1071	93	2	186
255	57	1078	93	2	186
263	60	1054	93	2	186
328	76	1083	93	2	186
449	100	1083	93	2	186
478	106	639	93	2	186
506	111	1056	93	2	186
576	123	1065	93	2	186
640	135	654	93	2	186
689	145	655	93	2	186

Summary:

This sheet captures detailed sales transaction data. It includes a unique Sales ID, associated Order ID, and references to Product ID for each sale. Additionally, it records the price per unit, the quantity sold, and the total revenue generated per transaction. This dataset is essential for analyzing sales performance, revenue generation, and understanding purchasing patterns.

Data Quality
“Problems & Solutions"

Customer sheet

1. Duplicate Customer IDs:

• **Issue**: Duplicate entries were found in the Customer ID column, which could result in inflated data and inaccuracies in analysis.

• **Action Taken**: Removed all duplicate rows based on the Customer ID column.

• **Reason**: Each customer should have a unique identifier. Duplicate removal ensures the integrity of the dataset, avoiding over representation or erroneous calculations.

2. Non-Standard Values in the Gender Column:

• **Issue:** The Gender column contained non-standard values such as "A gender," "Poly gender," and others.

• **Action Taken:** Replaced all non-standard values with "Prefer not to say," retaining only "Male" and "Female."

• **Reason**: Standardization reduces variability and ensures consistency in categorical data, which is critical for accurate segmentation and reporting. Using "Prefer not to say" adheres to inclusivity while addressing non-standard entries.

Data Quality
“Problems & Solutions"

Customer sheet

3. Missing Value in the Age Column:

• **Issue**: A missing value was identified in the Age column.

• **Action Taken:** Filled the missing value using the median of the column.

• **Reason**: The median is a robust measure of central tendency that is less affected by outliers compared to the mean. This ensures the imputation does not skew the data distribution.

4. Unnecessary Column: Home Address

• **Issue:** The Home Address column was present but irrelevant to the analysis and visualization goals.

• **Action Taken**: Removed the column.

• **Reason:** Reducing irrelevant data simplifies the dataset, enhances processing efficiency, and prevents clutter in visualization. This aligns with the principle of data minimization.

Data Quality
“Problems & Solutions"

Customers Sheet

5. Missing Value in the City Column

• **Issue**: A single missing value was found in the City column.

•**Action Taken:** Filled the missing value with "Unknown."

• **Reason**: Assigning "Unknown" maintains dataset completeness without making assumptions. This approach is transparent and ensures the data remains interpretable.

6. Inconsistent Formatting in the State Column

Issue: The State column had inconsistent text formatting.

• **Action Taken:** Applied proper case formatting (e.g., "new south wales" → "New South Wales").

• **Reason**: Standardizing text case improves readability and enhances the professional appearance of the data, which is essential for presentation and reporting.

Data Quality
“Problems & Solutions"

Customer Sheet

7. Redundant Country Column

• **Issue**: The Country column contained only a single unique value, "Australia," in all rows.

• **Action Taken:** Removed the column and included "Country: Australia" as a disclosure in the report.

• **Reason**: Columns with constant values do not provide analytical value. Removing them streamlines the dataset while ensuring the information is communicated effectively in the documentation.

Data Quality
“Problems & Solutions"

Products Sheet

1. Duplicate Product IDs

• **Issue**: Duplicate entries were present in the Product ID column, potentially leading to repeated or erroneous data.

• **Action Taken:** Removed all duplicate rows based on the Product ID column.

• **Reason:** Each product must have a unique identifier. Removing duplicates ensures data integrity and prevents incorrect aggregation or analysis.

2. Spelling Mistakes in the Product Type Category

• **Issue:** The Product Type column contained spelling mistakes and inconsistent formatting.

•**Action Taken:** Corrected the spelling errors and applied proper case formatting.

• **Reason:** Consistency in categorical data is essential for meaningful analysis and grouping. Proper case formatting enhances readability and professionalism in reporting.

Data Quality
“Problems & Solutions"

Products Sheet

3. Missing Values in the Product Name Column

• **Issue**: Missing values were found in the Product Name column.

• **Action Taken**: Filled the missing values using the mode (most frequent value) and applied proper case formatting.

• **Reason**: Using the mode ensures the imputed value aligns with the majority of the data, minimizing distortions. Proper case formatting improves readability.

4. Standardizing the Size Column

• **Issue:** The Size column contained non-standard values and inconsistent case (e.g., "Vs," "Vl," "Vm").

• **Action Taken:** Standardized all values to predefined categories (XS, S, M, L, XL) and applied uppercase formatting.

• **Reason:** Standardized sizes improve clarity and prevent mismatches during analysis. Uppercase formatting is a common convention for size□related data, ensuring uniformity.

Data Quality
“Problems & Solutions"

Products Sheet

5. Unstandardized Values in the Colour Column

• **Issue:** The Colour column contained inconsistent and unstandardized values, such as "Yellow Shadow" and "Green Dark."

• **Action Taken**: Standardized the values to primary color names (e.g., "Yellow," "Green") and applied proper case formatting.

• **Reason**: Simplifying color names into standard categories reduces ambiguity and facilitates accurate filtering and grouping in analysis.

6. Outliers in the Price Column

• **Issue**: Outliers, such as an excessively high value of "2016," were found in the Price column.

• **Action Taken**: Removed the outlier and replaced it with the median value for consistency.

• **Reason**: The median is a robust metric less affected by extreme values, ensuring the data remains representative of typical product prices.

Data Quality
“Problems & Solutions"

Products Sheet

7. Redundant Description Column

• **Issue**: The Description column repeated information already present in other columns.

• **Action Taken:** Removed the column.

• **Reason**: Redundant data increases the size and complexity of the dataset without adding value. Removing unnecessary columns ensures streamlined data for analysis and visualization.

Data Quality
“Problems & Solutions"

Order Sheet

1. Duplicate Order IDs

• **Issue**: Duplicate entries were identified in the Order ID column, which could lead to repeated transactions and inaccurate analysis.

• **Action Taken:** Removed all duplicate rows based on the Order ID column.

• **Reason**: Each order must have a unique identifier to ensure the accuracy of order-level analysis and prevent inflated results.

2. Missing Values in the Payments Column

• **Issue**: Missing values were found in the Payments column, which could affect financial analysis and reporting.

• **Action Taken**: Filled the missing values using the median of the column.

• **Reason:** The median is less sensitive to outliers and is a reliable measure to represent central tendency, ensuring data consistency without distortion.

Data Quality
“Problems & Solutions"

Order Sheet

3. Missing Value in the Order Date Column

• **Issue**: A missing value was present in the Order Date column, and the data type was not standardized.

• **Action Taken**: Filled the missing value using the median and converted the column data type to Date.

Reason:

- Using the median ensures the imputation maintains the temporal pattern of the dataset without introducing bias.
- Converting to the Date data type enables the use of date hierarchies (e.g., year, quarter, month) for drill-down and roll-up functionality during visualization, improving analytical depth.

Data Quality
“Problems & Solutions"

Sales Sheet

1.Duplicate Sales IDs

- **Issue**: Duplicate entries were found in the Sales ID column, leading to potential double-counting of transactions.
- **Action Taken:** Removed all duplicate rows based on the Sales ID column.
- **Reason**: Each sale must have a unique identifier to ensure the accuracy of sales data and avoid inflated results.

Data Quality
“Problems & Solutions"

Sales Sheet

2.Missing Values in Price Per Unit, Quantity, and Total Revenue Columns

- **Issue**: Missing values were identified in the Price Per Unit, Quantity, and Total Revenue columns, some of which were interdependent.
- **Action Taken**: • Where one value was missing, it was calculated using the equation:

• Price Per Unit × Quantity = Total Revenue.

• Rows with two missing values in the same record (e.g., both Price Per Unit and Quantity) were removed.

- **Justification**:

• The equation ensures that missing values are logically and accurately derived from existing data, maintaining dataset integrity.

• When two fields were missing simultaneously, it was impossible to accurately infer the third, so removing these rows avoided introducing potentially inaccurate or incomplete data.

Cleaning Code Snapshots

Importing Libraries

Importing libraries

```
import pandas as pd
import difflib
import re
import matplotlib.pyplot as plt
import seaborn as sns
```

[30]

Explanation:

- The initial step involves importing essential Python libraries for data cleaning and analysis.
- pandas is used for handling and manipulating structured data.
- difflib assists in comparing strings, useful for identifying similar or duplicate entries.
- re is used for working with regular expressions during text data cleaning.

Importing Files

Importing files

```
Customers_df=pd.read_csv("C:/Users/hp/OneDrive/Desktop/Data analysis projects/Depi project/Customers.csv")
Order_df=pd.read_csv("C:/Users/hp/OneDrive/Desktop/Data analysis projects/Depi project/Orders.csv")
Sales_df=pd.read_csv("C:/Users/hp/OneDrive/Desktop/Data analysis projects/Depi project/Sales.csv")
Product_df=pd.read_csv("C:/Users/hp/OneDrive/Desktop/Data analysis projects/Depi project/Products.csv")
```

[31] Python

Explanation:

This Code loads the project's datasets into pandas DataFrames using the read_csv() function.

Cleaning Code Snapshots

Customer Snapshots

Customer sheet

```
cutomer_bef=pd.read_csv("C:/Users/hp/OneDrive/Desktop/Data analysis projects/Depi project/Customers.csv")

print(Customers_df.head())
Customers_df.drop_duplicates(subset='customer_id',keep='first',inplace=True)

gender = {
    "Genderfluid" : "Prefer not to say",
    "Polygender" : "Prefer not to say",
    "Agender" : "Prefer not to say",
    "Bigender" : "Prefer not to say",
    "Non-binary" : "Prefer not to say",
    "Genderqueer" : "Prefer not to say"

}
# Handling different genders
Customers_df["gender"] = Customers_df["gender"].replace(gender)
```

Explanation:

- This Code performs initial cleaning steps on the customer dataset:
- The raw data is first read and stored in cutomer_bef for backup/reference purposes before any modifications.

- print(Customers_df.head()) is used to preview the first few records of the dataset.

- Duplicate records based on the customer_id field are removed, keeping only the first occurrence to ensure data integrity.

- A dictionary (gender) is defined to group non-standard or less common gender identities under a unified label: "Prefer not to say". This helps simplify gender classification and improve consistency in analysis.

- The gender column is then updated using .replace() to apply the standardized labels.

Cleaning Code Snapshots

Customer Snapshots

```
# Replacing Null Valued age to the median of the column
Customers_df["age"] = Customers_df["age"].fillna(Customers_df["age"].median())
# Removing address and Country  column
#Customers_df.drop(columns=['home_address'], inplace=True)
#Customers_df.drop(columns=["country"],inplace=True)
# Replacing null Values in City
Customers_df["city"] = Customers_df["city"].fillna("Unknown")
# Standardizing State column
Customers_df["state"]=Customers_df["state"].str.title()
```

Explanation:
This Code addresses missing values and performs standardization on certain columns:

- **Age Column:** Null values in the age column are replaced with the median age from the dataset. This ensures that the age data is complete and can be used in further analysis without distorting the dataset.
- **Address and Country Columns:** The home_address and country columns are commented out, meaning they were intended for removal, but this step is not executed in the current code.
- **City Column:** Missing values in the city column are replaced with the string "Unknown", ensuring no null entries remain in this critical field.
- **State Column:** The state column is standardized by capitalizing the first letter of each word using .str.title(), ensuring consistency in state naming conventions

Cleaning Code Snapshots

Order Snapshots

Order Sheet

```
order_bef=pd.read_csv("C:/Users/hp/OneDrive/Desktop/Data analysis projects/Depi project/Orders.csv")
order_bef["order_date"] = pd.to_datetime(order_bef["order_date"],errors='coerce')
order_bef["delivery_date"] = pd.to_datetime(order_bef["delivery_date"],errors='coerce')
#print(order_bef.isnull().sum())
# get info on the sheet
#print(Order_df.head())
#print(Order_df.isnull().sum())
# Droping the duplicated ids
Order_df.drop_duplicates(subset="order_id",keep='first',inplace=True)
# filling Null values with the median
Order_df["payment"] = Order_df["payment"].fillna(Order_df['payment'].median())
# Convert the date columns to date time
Order_df["order_date"] = pd.to_datetime(Order_df["order_date"],errors='coerce')
Order_df["delivery_date"] = pd.to_datetime(Order_df["delivery_date"],errors='coerce')
print(Order_df.info())
# replacing the null values with the median
Order_df["order_date"] = Order_df["order_date"].fillna(Order_df["order_date"].median())
Order_df["delivery_date"] = Order_df["delivery_date"].fillna(Order_df["delivery_date"].median())
```

Explanation:

This Code performs several data cleaning operations on the Order dataset:

- The raw data is first read and stored in order_bef for reference before any modifications.
- The order_date and delivery_date columns are converted to datetime format using pd.to_datetime(). Any invalid date values are coerced to NaT (Not a Time), ensuring proper date formatting.
- Duplicate order records based on order_id are dropped, keeping only the first occurrence to ensure each order is unique.
- Missing values in the payment column are replaced with the median value of the column, ensuring that payment data is complete and usable in further analysis
- The order_date and delivery_date columns are converted to datetime again and any remaining null values are replaced with the median date value. This prevents gaps in these important fields.

Cleaning Code Snapshots

Sales Snapshots

Sales Sheet

+ Code + Markdown

```
sales_bef=pd.read_csv("C:/Users/hp/OneDrive/Desktop/Data analysis projects/Depi project/Sales.csv")
sales_bef['price_per_unit'] = pd.to_numeric(sales_bef['price_per_unit'], errors='coerce')
# Sales sheet
Sales_DF = Sales_df.rename(columns={"total_price": "total_revenue"})

# Convert columns to numeric
Sales_DF['price_per_unit'] = pd.to_numeric(Sales_DF['price_per_unit'], errors='coerce')
Sales_DF['quantity'] = pd.to_numeric(Sales_DF['quantity'], errors='coerce')
Sales_DF['total_revenue'] = pd.to_numeric(Sales_DF['total_revenue'], errors='coerce')

# Drop when all are missing
Sales_DF = Sales_DF.dropna(subset=["price_per_unit", "quantity", "total_revenue"], how='all')
```

Explanation:
This Code performs several cleaning and data transformation tasks on the Sales dataset:

- The raw data is read into sales_bef as a reference before any changes are made.
- The price_per_unit column is converted to numeric values, with any errors in conversion being coerced into NaN (Not a Number).
- The Sales_df DataFrame is renamed, with the total_price column changed to total_revenue for consistency.
- Several columns (price_per_unit, quantity, and total_revenue) are explicitly converted to numeric values, ensuring the data is ready for mathematical operations.
- Rows with missing values in all three key columns (price_per_unit, quantity, and total_revenue) are removed.

Cleaning Code Snapshots

Sales Snapshots

```
# a loop that checks each row to fill the suitable value by using the equation
for index, row in Sales_DF.iterrows():
    if pd.isnull(row["price_per_unit"]) and not pd.isnull(row["quantity"]) and not pd.isnull(row["total_revenue"]):
        Sales_DF.at[index, "price_per_unit"] = row["total_revenue"] / row["quantity"]

    elif pd.isnull(row["quantity"]) and not pd.isnull(row["price_per_unit"]) and not pd.isnull(row["total_revenue"]):
        Sales_DF.at[index, "quantity"] = row["total_revenue"] / row["price_per_unit"]

    elif pd.isnull(row["total_revenue"]) and not pd.isnull(row["price_per_unit"]) and not pd.isnull(row["quantity"]):
        Sales_DF.at[index, "total_revenue"] = row["price_per_unit"] * row["quantity"]
# Drop if the row did not fullfill the conditions
Sales_DF = Sales_DF.dropna(subset=["price_per_unit", "quantity", "total_revenue"])

print(sales_bef[["price_per_unit","quantity","total_price"]].describe())
print("Before vs After Cleaning")
print(Sales_DF[["price_per_unit","quantity","total_revenue"]].describe())
```

A loop is used to fill missing values in each row based on logical conditions:

- If price_per_unit is missing, it is calculated using the equation: total_revenue / quantity.
- If quantity is missing, it is calculated using the equation: total_revenue / price_per_unit.
- If total_revenue is missing, it is calculated using the equation: price_per_unit * quantity.
- After these calculations, any remaining rows with missing values are dropped.
- Finally, a comparison is made before and after cleaning to show the summary statistics for the key columns (price_per_unit, quantity, total_revenue), providing insights into the effectiveness of the data cleaning process.

Cleaning Code Snapshots

Products Snapshots

Product Sheet

```
bef_clean=pd.read_csv("C:/Users/hp/OneDrive/Desktop/Data analysis projects/Depi project/Products.csv")
Product_df.drop_duplicates(subset=["product_ID"],keep="first",inplace=True)

# Valid Values
valid_types = ['Shirt', 'Jacket', 'Trousers']
def correct_product_type(product):

    # to lower
    product = product.lower()
    # mathces to the valid type
    matches = difflib.get_close_matches(product, valid_types, n=1, cutoff=0.6)
    return matches[0] if matches else product

# Apply the method the columns
Product_df['product_type'] = Product_df['product_type'].apply(correct_product_type)
```

```
# droping null values in the product name
Product_df.dropna(subset=["product_name"],inplace=True)
print(Product_df["product_name"].isnull().sum())

# valid values
accepted_sizes = {"xs", "s", "m", "l", "xl"}

def clean_size(size):
# to lower
    size = str(size).lower()
# split the string when face a non letter
    tokens = re.split(r'[^a-z]', size)
# check if its accepted
    for token in tokens:
        if token in accepted_sizes:
            return token
    return None
```

Cleaning Code Snapshots

Products Snapshots

```
# Apply the cleaning function to the 'size' column.
Product_df['size'] = Product_df['size'].apply(clean_size)

accepted_colors = {"red", "blue", "green", "yellow", "black", "white", "orange", "purple", "pink", "brown",
                   "grey","indigo","violet"}

def clean_color(value):

#to lower
    value = str(value).lower()

# Spliting.
    tokens = re.split(r'[^a-z]', value)

# Checking
    for token in tokens:
        if token in accepted_colors:
            return token
    return None

Product_df["colour"]=Product_df["colour"].apply(clean_color)
```

```
# Calculate IQR
Q1 = Product_df['price'].quantile(0.25)
Q3 = Product_df['price'].quantile(0.75)
IQR = Q3 - Q1

# caluculate bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# replace outliers with the median
median_price = Product_df['price'].median()
Product_df['price'] = Product_df['price'].clip(lower=lower_bound, upper=upper_bound, axis=0)
Product_df['price'] = Product_df['price'].fillna(median_price)

#Product_df.drop(columns="description",inplace=True)
```

Cleaning Code Snapshots

Products Snapshots

Explanation:

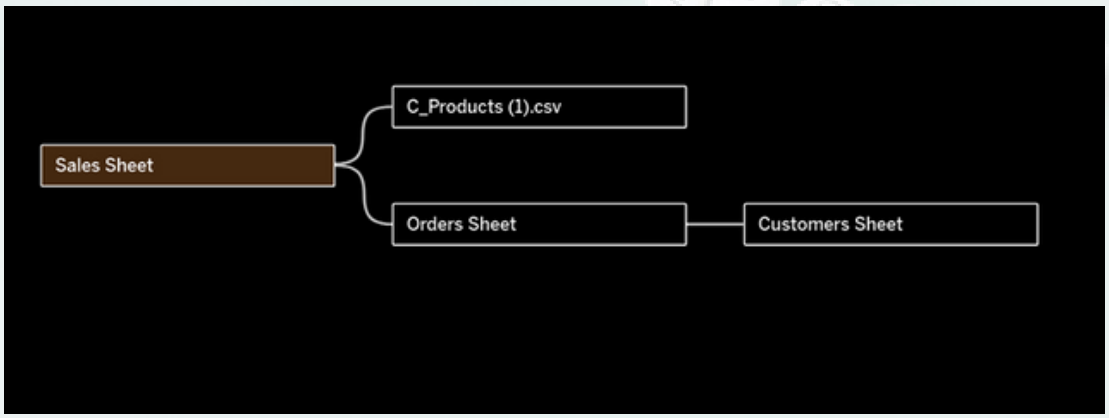
This Code focuses on cleaning and transforming the Products dataset:

- **Duplicate Removal:** The dataset is cleaned by removing duplicate product records based on product_ID, keeping only the first occurrence to ensure product uniqueness.
- **Product Type Standardization:** A function (correct_product_type) is used to standardize the product_type column by comparing each product type to a predefined list of valid types ('Shirt', 'Jacket', 'Trousers'). The function uses difflib.get_close_matches() to find the closest match and replace non-matching values with the closest valid type.
- **Product Name Cleaning:** Any rows with missing values in the product_name column are dropped to maintain complete product information.
- **Size Standardization:** A function (clean_size) is used to standardize the size column. The function converts all sizes to lowercase, splits the values based on non-alphabetical characters, and checks if they match any of the accepted sizes ("xs", "s", "m", "l", "xl"). Any unmatched sizes are replaced with None.
- **Color Standardization:** A function (clean_color) is applied to the colour column, similar to the clean_size function, to ensure only accepted color names (e.g., "red", "blue", "green") are retained. The function splits the color strings and matches each token with the predefined accepted colors.
- **Outlier Handling in Price:** The interquartile range (IQR) is calculated for the price column to identify outliers. Outliers beyond 1.5 times the IQR from the lower and upper quartiles are clipped to fall within the calculated bounds. Any remaining missing values in the price column are filled with the median price.

Tableau

Data Modeling Overview

Understanding Relationships Between **Customers**, **Orders**, **Sales**, and **Products**



Data Model Summary

1.Relationship: Orders Data and Customers Data

- **Type:** Many-to-One (Orders → Customers)
- **Field Used**: Customer_ID
- **Explanation:** Each customer can place multiple orders, but each order is associated with only one customer.

2.Relationship: Sales Data and Products Data

- **Type**: Many-to-One (Sales → Products)
- **Field Used**: Product_ID
- **Explanation:** Each sale involves one product, but a single product can appear in multiple sales transactions.

3.Relationship: Sales Data and Orders Data

- **Type**: Many-to-One (Sales → Orders)
- **Field Used:** Order_ID
- **Explanation:** Each sale is linked to one order, but a single order can contain multiple sales (e.g., multiple products purchased in one transaction).

Tableau

Analyzing data and extracting useful insights

How did our E-commerce business perform throughout the year 2021?

Executive Summary

Overview

This dashboard provides a high-level summary of key performance indicators (KPIs) for the e-commerce shopping cart. It highlights the overall business health by showing total revenue, customer count, product count, and total orders. Additionally, it offers visual insights into order distribution by state, top-selling products, category-wise quantity sold, and monthly revenue trends, helping executives quickly grasp sales performance across different dimensions

Tableau

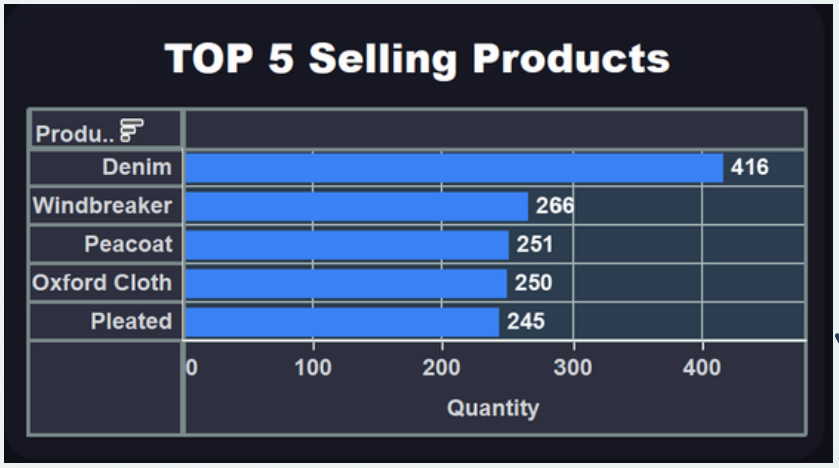
Analyzing data and extracting useful insights

Executive Summary

Bubble Chart: Orders vs States:
Question Answered:
What is the distribution of orders across different states?

Insights: The Bubble Chart illustrates the distribution of orders across different states. South Australia accounts for the highest percentage of orders at 148, while the Northern Territory shows the lowest percentage, with 106 of the total orders

Clustered Bar Chart: Top 5 Selling Products
Question Answered:
Which are the top 5 products by quantity sold, and how do they compare?



Insights: five most sold products, with Denim topping at 416 units sold and with pleated lowest at 245 units sold

Tableau

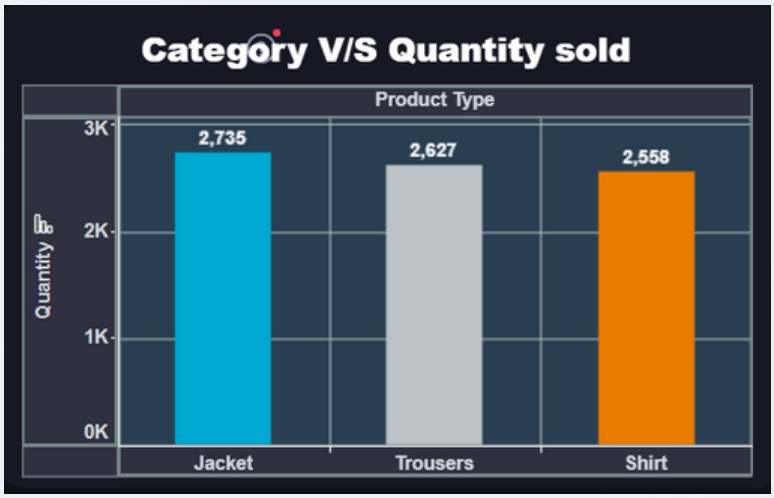
Analyzing data and extracting useful insights

Executive Summary

Clustered Column Chart: Product Category vs. Quantity Sold

Question Answered:

How does the quantity sold vary across different product categories?



Insights:

- Trousers and Jacket are the top-performing categories, with quantities sold being nearly equal, suggesting that these product types are highly popular among customers. This could be due to ongoing trends, strong consumer demand, or effective marketing strategies targeting these specific categories.
- Shirts, while still performing well, show slightly lower sales compared to Trousers and Jackets. This could indicate that shirts may be a more seasonal product or less favored in the current product mix.

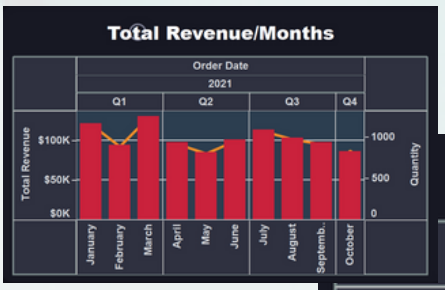
Tableau

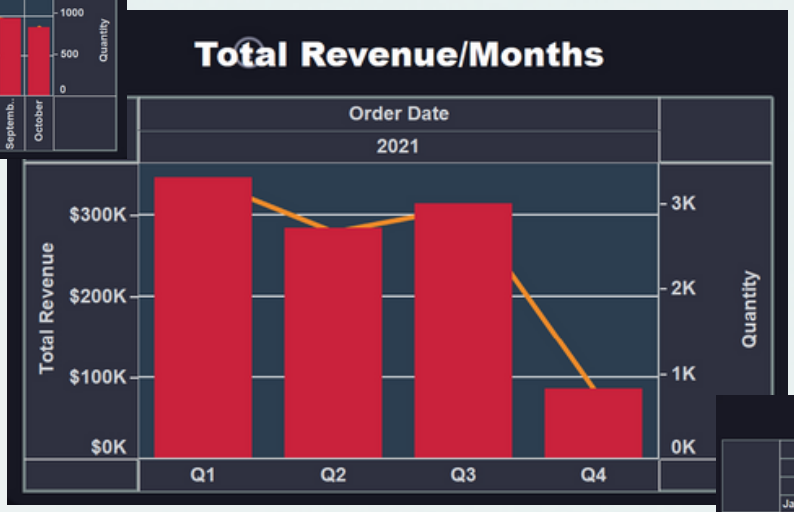
Analyzing data and extracting useful insights

Line and Stacked Column Chart: Total Revenue\Month

Question Answered:

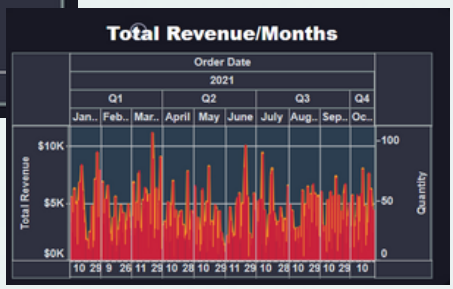
How does total revenue trend over months, and what are the details across the order date hierarchy?





Drilling by quarter and months

Drilling by Days



Insights:

- **March's Performance:** The significant sales in **March**, with both high total revenue and quantity sold, suggest that this month likely saw strong demand, possibly due to seasonal factors, promotions, or successful marketing campaigns. This could also be a result of external events or trends that made March a peak sales period.

- **Quarterly** revenue trends, peaking in Q1 and dropping in Q4, with quantity sold represented by a line.

- **May's Lower Performance:** The lower total revenue and quantity sold in **May** could be attributed to factors such as seasonality, a dip in consumer spending, or the absence of significant marketing activities or promotions during this month. It might also indicate that certain product categories performed poorly in that period.

Tableau

Analyzing data and extracting useful insights

Customer Insights

Overview

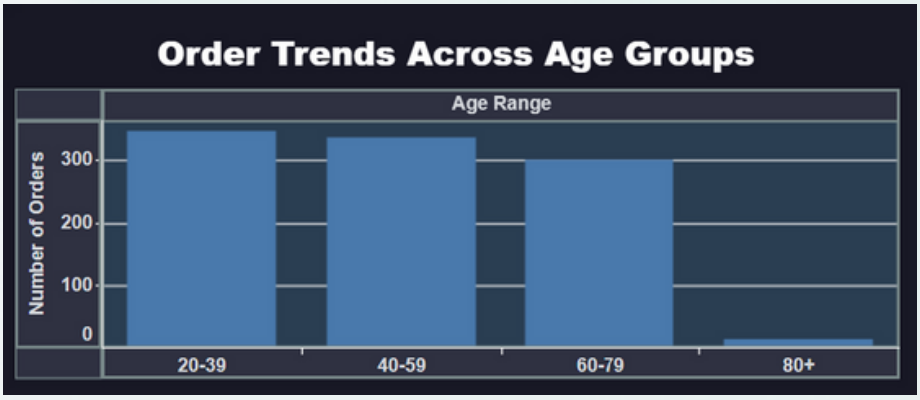
This dashboard focuses on customer demographics and behavior. It presents insights into order patterns by age, geographical distribution of customers, and revenue per city, offering a complete picture of who the customers are and how they contribute to sales.

Analyzing data and extracting useful insights

Customer Insights

Bar Chart:Order Trends Across Age Groups

Question Answered: **How does the quantity of orders vary across different age ranges?**



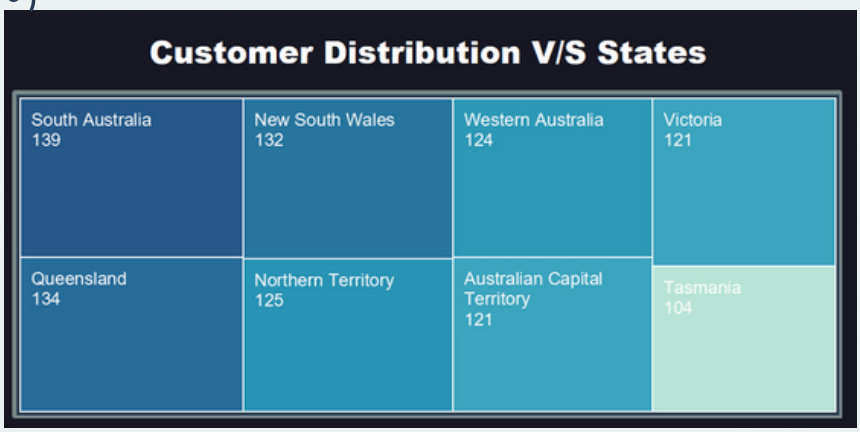
Insights:

- **Peak Orders in Younger Age Groups:** The age group of 20 stands out with the highest number of orders, reflecting a strong purchasing behavior among younger customers. This could be attributed to factors such as higher disposable income, more access to online shopping, or a greater interest in the types of products offered.
- **Declining Orders with Age:** As the age increases, the number of orders steadily decreases, with age 80 showing the lowest number of orders. This trend is typical, as older age groups often have different shopping behaviors and preferences. They may have less frequent purchasing habits, or may prefer in-store shopping over online purchases.

Treemap: Customer Distribution vs States

Question Answered:

How are customers distributed across different states?



Insights:

- **Dominance of South Australia:** South Australia stands out with the largest customer base, which could suggest that it is a key market for the business.
- **Lower Representation of Tasmania**: Tasmania, in contrast, has the smallest number of customers, which could be due to various reasons such as smaller population size, lower market penetration, or reduced availability of products and services in this region.

Hint: The color intensity is used to visually represent this, with the darkest color indicating South Australia (highest number of customers) and the lightest color representing Tasmania (lowest number of customers).

Tableau

Analyzing data and extracting useful insights

Customer Insights

Matrix: City VS. Number of Customers & Total Revenue

Question Answered:

What is the number of customers and total revenue generated in each city?

Customers Purchasing Power

City		
Port Hannahburgh	1	$7,420
Jordanside	1	$6,640
East Max	2	$5,769
West Finnberg	1	$5,657
South Clairetown	1	$5,581
O'keefeton	1	$5,455
West Kai	2	$5,224
Rubyfort	1	$5,119
Tillmanview	1	$4,969
Williammouth	2	$4,716
New Ava	3	$4,713
Lake Jesse	2	$4,583
Rodriguezbury	1	$4,569
Kohlerberg	1	$4,415
Claudiaview	1	$4,385
Port Annabelle	2	$4,284
Port Chelseaport	1	$4,213
Port Harrychester	1	$4,126
Jacobport	1	$4,095
Port Blake	1	$4,075
	0 200 400 600 Number of Custo...	0M 1M Total Revenue

Insights:

- Port Hannahburgh's High Revenue: Port Hannahburgh stands out as the top contributor to total revenue. This could suggest that it either has a large customer base, or the customers in this city are purchasing more premium products or making more frequent purchases.

- Imogenmouth's Low Revenue: On the other hand, Imogenmouth has the lowest total revenue, which could indicate a smaller customer base, lower average spend per customer, or other

Tableau

Analyzing data and extracting useful insights

Products Insights

Overview

This dashboard dives deeper into product-related data.
It analyzes the performance of different product types, product names, and color variants. It also presents sales insights geographically using a map. The goal is to provide actionable insights for product planning and inventory management.

Tableau

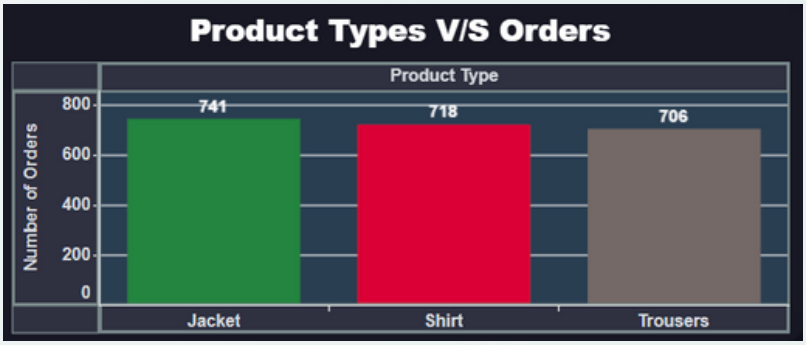
Analyzing data and extracting useful insights

Products Insights

Bar Chart: Product Types vs Orders

Question Answered:

How does the quantity of orders vary across different product types?



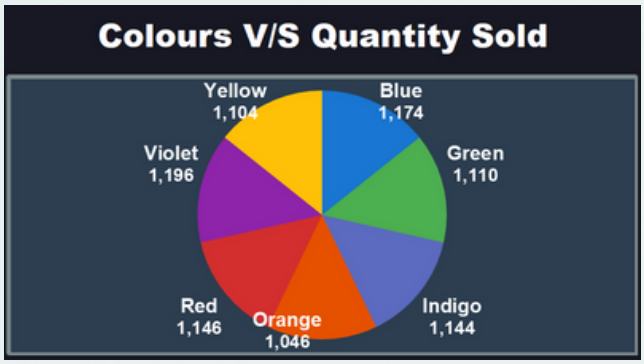
Insights

- **Jackets Leading in Sales:** Jackets have the highest sales, indicating a strong demand for this product category. This could reflect factors such as seasonal demand (e.g., colder weather) or successful marketing efforts that have particularly targeted jacket sales.
- **Shirts and Trousers Performing Nearly Equally:** The close performance of shirts and trousers, with only a slight difference in sales, suggests that these two product types are equally popular, or that they appeal to similar customer segments. This might indicate that both categories are staple items in the product mix

Pie Chart:Colours vs Quantity Sold

Question Answered:

How do product colors rank based on the quantity sold?



Insights

Color preferences of sold items, where Blue, Indigo, and Violet dominate.

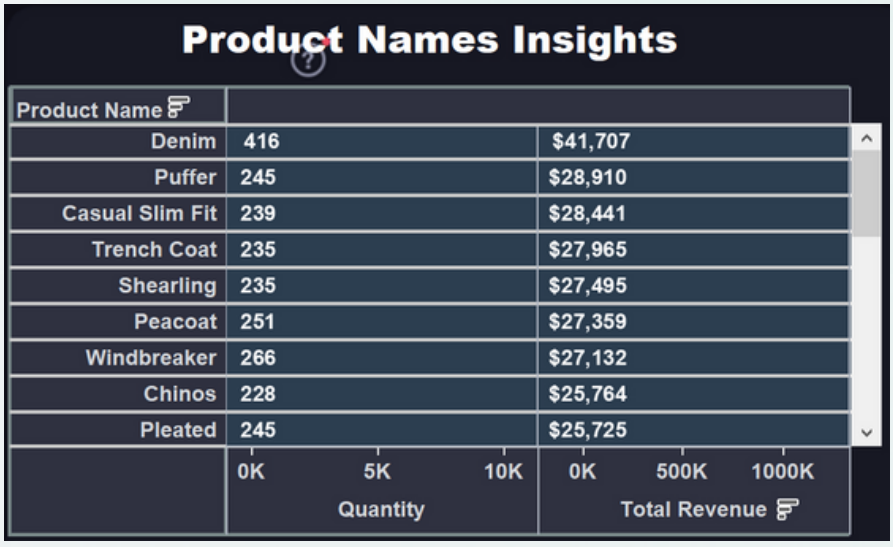
Analyzing data and extracting useful insights

Products Insights

Matrix: Product Names Insights

Question Answered:

What is the quantity sold and total revenue for each individual product?



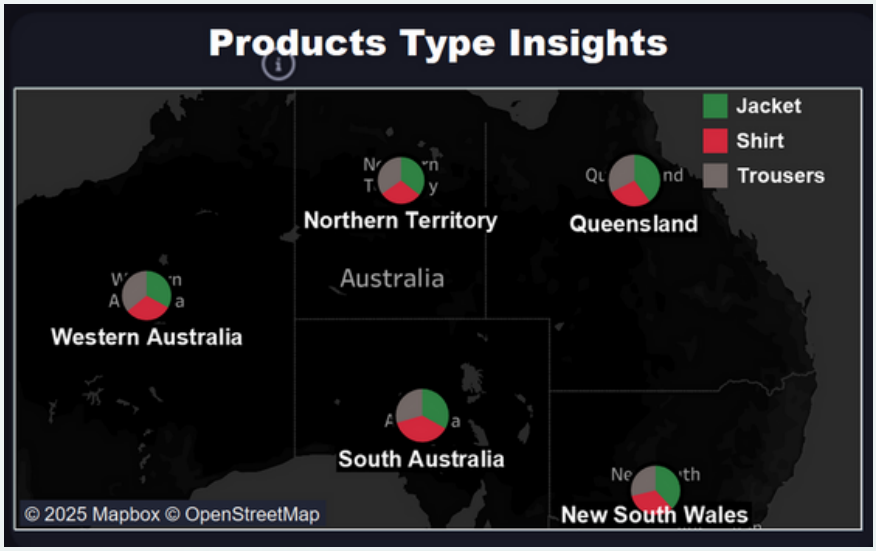
Insights:

- Denim's Strong Revenue Performance: The product Denim leads in total revenue, suggesting that it is a high-demand product with a strong customer preference.
- Wool's Underperformance in Revenue: On the other hand, Wool appears to have the lowest total revenue, indicating that this product is not performing as well as others.

Map: Products Type Insights:

Question Answered:

How is the distribution of product types spread across different states?



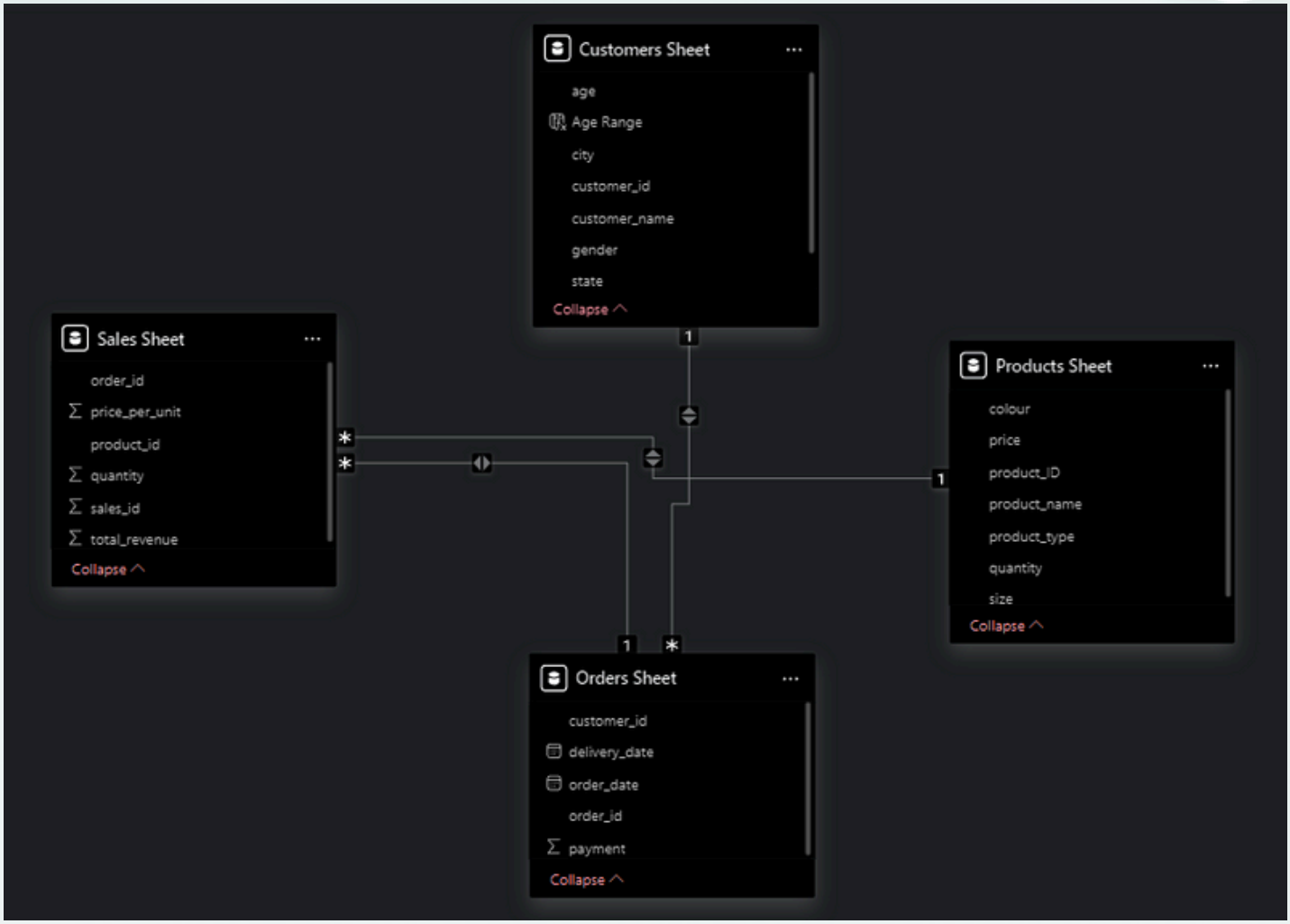
Insights:

Visual map showing product type distribution across states, enabling regional product planning.

Power BI

Analyzing data and extracting useful insights

Data Modelling



Power BI

Analyzing data and extracting useful insights

Executive Summary

Power BI

Analyzing data and extracting useful insights

Products Details

Power BI

Analyzing data and extracting useful insights

Customer Details

Customer Segmentation using RFM Model (AI)

customer_id	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Score
1	203	3	1641	2	5	3	253
7	156	1	1017	2	1	2	212
10	229	1	270	1	1	1	111
11	149	1	382	2	1	1	211
12	127	1	1551	3	1	3	313
13	26	1	332	5	1	1	511
15	61	2	2151	4	3	4	434
16	6	1	1046	5	1	2	512
18	240	2	2024	1	3	4	134
19	194	1	1172	2	1	2	212

Findings and Insights

- **Champions (R = 5 & F = 5 & M = 5)**
  75 customers (7.5% of 1,000) generate $239,188 (23.5% of total revenue). They bought most recently, purchase most often, and spend the most.
- **Loyal Customers (F ≥ 4 & M ≥ 4)**
  227 customers (22.7%) account for $485,404 (47.7% of revenue). They order frequently and contribute significant spend, though not all are in the top recency tier.
- **At-Risk Customers (R ≤ 2 & F ≥ 3 & M ≥ 3)**
  14 customers (1.4%) brought in $13,520 (1.3% of revenue) but haven't shopped in some time—high past value with lapsed engagement.
- **Dormant Customers (R ≤ 2 & F ≤ 2 & M ≤ 2)**
  383 customers (38.3%) generated $0 (0% of revenue). They have no purchase activity by the last order date, representing untapped potential.
- **Others**
  301 customers (30.1%) contribute $279,888 (27.5% of revenue). They show moderate recency, frequency, and spend—ripe for targeted growth.

Customer Segmentation using RFM Model (AI)

Recommendations

- **Champions**
  Invite to VIP “pre-launch” events and reward with exclusive bundles or bonus loyalty points to deepen engagement and encourage advocacy.
- **Loyal Customers**
  Launch a “Give $10, Get $10” referral program to turn frequent buyers into brand promoters.
- **At-Risk Customers**
  Send personalized win-back offers—e.g., limited-time discounts on their favorite categories—to rekindle their purchases.
- **Dormant Customers**
  Deploy a reactivation push: offer free shipping or a small discount plus a one-click survey to learn and remove barriers to purchase.
- **Others**
  Run category-specific micro-campaigns (like “Jackets Spotlight” or “Trousers Week”) based on past buys to nudge them toward more frequent orders.

Forecasting by linear regression using python

1. What are the expected revenue trends for the upcoming 6 months based on historical performance?

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Load datasets
df_orders = pd.read_csv("C_Orders.csv")
df_products = pd.read_csv("C_Products.csv")
df_sales = pd.read_csv("C_Sales.csv")
df_customers = pd.read_csv("C_Customers.csv")
```

```python
# Convert order_date to datetime
df_orders["order_date"] = pd.to_datetime(df_orders["order_date"], errors='coerce')

# ---- Sales Forecast ----
print("Processing sales forecast...")
df_sales = df_sales.merge(df_orders[['order_id', 'order_date']], on='order_id', how='left')
df_sales_monthly = df_sales.groupby(df_sales["order_date"].dt.to_period("M")).agg({"total_revenue": "sum"}).reset_index()
df_sales_monthly["month_num"] = np.arange(len(df_sales_monthly))

X_sales = df_sales_monthly[["month_num"]]
y_sales = df_sales_monthly["total_revenue"]

model_sales = LinearRegression()
model_sales.fit(X_sales, y_sales)
future_sales_months = np.arange(len(df_sales_monthly), len(df_sales_monthly) + 6).reshape(-1, 1)
future_sales = model_sales.predict(future_sales_months)
```

Explanation:

1. Importing Libraries

This section imports essential libraries:

- **pandas** for data manipulation,
- **numpy** for numerical operations,
- **matplotlib.pyplot** for data visualization,
- **LinearRegression** from **sklearn** to build a simple predictive model.

2. Loading Data and Date Conversion

- Order and sales data are loaded using **read_csv**.
- The **order_date** column is converted to datetime format to enable time-based aggregation.

Forecasting by Linear Regression using Python

3. Merging and Monthly Aggregation

- The sales data is merged with order dates using **order_id.**
- Revenue is aggregated by month.
- A numeric month index **month_num** is added to represent each time step sequentially for model training.

4. Training the Linear Regression Model

- Features (**X_sales**) and target (**y_sales**) are defined.
- A linear regression model is trained to fit the trend in monthly revenue over time.

5. Forecasting Future Revenue

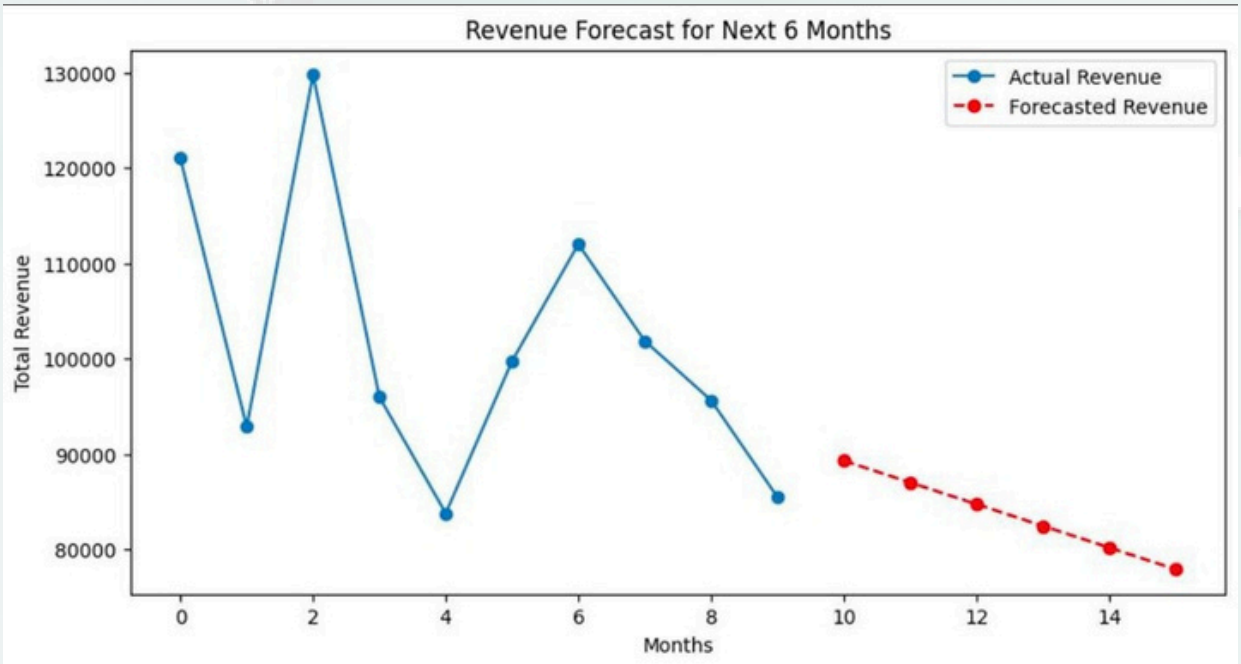
- Generates 6 future monthly indices.
- The trained model is used to predict revenue for the upcoming 6 months.

```
# Plot sales forecast
plt.figure(figsize=(10, 5))
plt.plot(df_sales_monthly["month_num"], df_sales_monthly["total_revenue"], marker='o', label="Actual Revenue")
plt.plot(future_sales_months, future_sales, marker='o', linestyle='dashed', color='red', label="Forecasted Revenue")
plt.xlabel("Months")
plt.ylabel("Total Revenue")
plt.title("Revenue Forecast for Next 6 Months")
plt.legend()
plt.show()
```

6. Visualization

- The actual historical revenue is plotted.
- Forecasted revenue is displayed as a red dashed line to distinguish it from past data.

Forecasting by Linear Regression using Python



Observation

The revenue forecast for the next six months reveals a **slight downward trend,** suggesting potential challenges in sustaining sales performance. The historical data illustrates fluctuations with observable peaks and declines, indicating possible seasonality or market dynamics. If no strategic actions are taken, this downward trend could persist. This forecast underscores the importance of proactive planning, including customer re-engagement strategies, promotional campaigns, or product diversification to stabilize or boost future revenue

```
# Sales Summary Table
sales_summary = pd.DataFrame({
    "Month": list(df_sales_monthly["order_date"].astype(str)) + [f"Forecast {i+1}" for i in range(6)],
    "Total Revenue": list(df_sales_monthly["total_revenue"]) + list(future_sales)
})
print("\nSales Summary:")
print(sales_summary.to_string(index=False))
```

```
Sales Summary:
      Month  Total Revenue
    2021-01  120999.000000
    2021-02   92826.000000
    2021-03  129775.000000
    2021-04   95965.000000
    2021-05   83766.000000
    2021-06   99719.000000
    2021-07  112001.000000
    2021-08  101852.000000
    2021-09   95622.000000
    2021-10   85475.000000
Forecast 1   89276.733333
Forecast 2   86999.775758
Forecast 3   84722.818182
Forecast 4   82445.860606
Forecast 5   80168.903030
Forecast 6   77891.945455
```

Forecasting by Linear Regression using Python

2. What is the projected customer growth over the upcoming 6 months, and how does it reflect past engagement trends?

```
# ---- Customer Trends Forecast ----
print("Processing customer trends forecast...")
df_customers_monthly = df_orders.groupby(df_orders["order_date"].dt.to_period("M"))["customer_id"].nunique().reset_index()
df_customers_monthly["month_num"] = np.arange(len(df_customers_monthly))

X_customers = df_customers_monthly[["month_num"]]
y_customers = df_customers_monthly["customer_id"]

model_customers = LinearRegression()
model_customers.fit(X_customers, y_customers)
future_customers_months = np.arange(len(df_customers_monthly), len(df_customers_monthly) + 6).reshape(-1, 1)
future_customers = model_customers.predict(future_customers_months)
```

Explanation

1. Grouping Monthly Customers

- The **order_date** is grouped by month using **.dt.to_period("M").**
- For each month, the number of unique customers **(nunique())** is calculated.
- This results in a monthly trend of customer activity.

2. Adding a Month Index

Adds a numerical representation of months for modeling (e.g., 0, 1, 2, ...) using **["month_num"]**

3. Preparing Features and Target

X_customers: Independent variable (month number).

y_customers: Dependent variable (number of customers).

Forecasting by Linear Regression using Python

4. Fitting the Linear Regression Model

A linear regression model is trained to learn the trend in customer growth over time using **model_customers = LinearRegression()**
model_customers.fit(X_customers, y_customers)

5. Creating Future Data Points

Predicts the number of customers for the next 6 months by extrapolating the trend Using **future_customers_months = np.arange(len(df_customers_monthly), len(df_customers_monthly) + 6).reshape(-1, 1)**
future_customers = model_customers.predict(future_customers_months)

```
# Plot customer forecast
plt.figure(figsize=(10, 5))
plt.plot(df_customers_monthly["month_num"], df_customers_monthly["customer_id"], marker='o', label="Actual Customers")
plt.plot(future_customers_months, future_customers, marker='o', linestyle='dashed', color='red', label="Forecasted Customers")
plt.xlabel("Months")
plt.ylabel("Number of Customers")
plt.title("Customer Trend Forecast for Next 6 Months")
plt.legend()
plt.show()
```

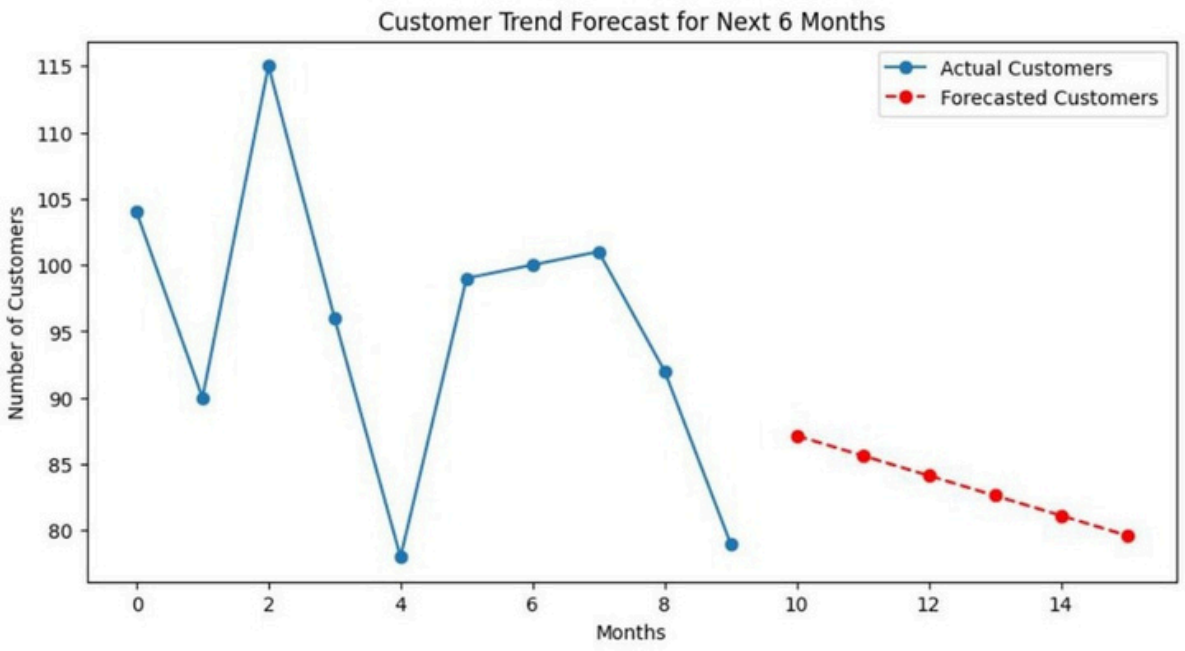
Explanation

6. Visualization

The chart displays:

- Actual customer data (solid line with markers).
- Forecasted customer trend (dashed red line).
- X-axis: Time in months.
- Y-axis: Number of unique customers.
- Helps visualize both historical customer trends and future expectations.

Forecasting by Linear Regression using Python



Observation

Decreasing Trend: The forecast predicts a consistent decrease in the number of unique customers per month, potentially dropping below 80 in six months.

Actionable Implications:

- Evaluate potential reasons for decline: seasonality, competition, reduced engagement.
- Consider implementing customer retention campaigns or targeted marketing strategies.

Model Limitations:

- The current forecast assumes a linear relationship, which may oversimplify real-world dynamics.
- Incorporating external factors or using more advanced models (e.g., ARIMA, Prophet) may improve forecast accuracy.

Forecasting by Linear Regression using Python

```
[ ] # Customer Trends Summary Table
    customer_summary = pd.DataFrame({
        "Month": list(df_customers_monthly["order_date"].astype(str)) + [f"Forecast {i+1}" for i in range(6)],
        "Number of Customers": list(df_customers_monthly["customer_id"]) + list(future_customers)
    })
    print("\nCustomer Trends Summary:")
    print(customer_summary.to_string(index=False))
```

```
Customer Trends Summary:
      Month  Number of Customers
    2021-01           104.000000
    2021-02            90.000000
    2021-03           115.000000
    2021-04            96.000000
    2021-05            78.000000
    2021-06            99.000000
    2021-07           100.000000
    2021-08           101.000000
    2021-09            92.000000
    2021-10            79.000000
Forecast 1            87.133333
Forecast 2            85.630303
Forecast 3            84.127273
Forecast 4            82.624242
Forecast 5            81.121212
Forecast 6            79.618182
```

Conclusion

The client's platform served 1,000 customers with 1,000 orders across 1,008 SKUs in three core categories—jackets (2,735 units), trousers (2,627) and shirts (2,558)—delivering $1.018 M in revenue through October 2021. Performance peaked early and mid-year (Q1: $343.6 K; Q2: $279.5 K; Q3: $309.5 K), with the $85.5 K Q4 figure reflecting an October cutoff rather than a true year-end decline. Demand skewed heavily toward 20–59 year-olds (685 orders), tapering among 60–79 (301) and 80+ (14), while South Australia led in customer count (139) and Tasmania trailed (104). Color preferences favored cooler tones —violet (1,196 units), blue (1,174), red (1,146), indigo (1,144) and green (1,110)—offering clear direction for your next assortment.

Recommendations

- **Align inventory with demand:** Review your jackets, trousers, and shirts to spot the top sellers for restocking and the slow movers for clearance or bundling. Calculate each key SKU's average weekly sales, add a two-week buffer based on sales swings and supplier lead times, and set that as your reorder point. Check a simple weekly dashboard to flag any item nearing its reorder level so you can replenish before it runs out.
- **Segment customer campaigns:** Send simple, targeted emails or social ads for 20–59 year-olds. Offer chatbot or live-chat help for the 60+ group.
- **Geo-target promotions:** Roll out state-specific deals in Tasmania and other low-penetration areas. Allocate ad spend per capita to get the best ROI in high-density markets.
- **Refresh color lines quickly:** Feature violet, blue and red in your next launch. Release small test batches of new colors, then scale up only the top sellers.
- **Bundle and cross-sell:** Pair bestselling items (e.g., denim with windbreakers) to raise average order value without steep discounts.
- **Gather fast customer feedback:** Add a one-click survey on your site or in emails to capture real-time preferences on products, colors and offers.
- **Implement a simple loyalty program:** Give repeat buyers points or minor discounts to boost retention and lifetime value.

Tools Used

- Kaggle 

- Python Language 
- Google Colab 
- Tableau 
- Microsoft Power BI 

- DEPI Technical Material 

- ChatGpt 
- YouTube