Dataset Analysis of an Online Retailer Based in Brazil Using Python

Objective:

The primary objective of this project is to analyze the sales data of an online retailer based in Brazil to identify key insights into product demand and sales distribution across various categories. We will utilize Pareto analysis to determine which product categories contribute the most to total sales revenue, aiding in inventory management, marketing strategies, and business decision-making.

Summary:

The analysis involves multiple steps, including data collection, cleaning, and visualization. We focused on understanding the demand for product categories and the distribution of sales among them. By leveraging Pareto analysis, we aimed to pinpoint the categories that significantly contribute to the retailer's sales, following the 80/20 rule (Pareto Principle).

Data Collection:

1. **Data Sources:**
    - Order Items Data: Contains order IDs, product IDs, and prices.
    - Products Data: Contains product IDs and category names.
    - Category Translation Data: Provides English translations for product category names.
2. **Data Import:**
    - Used the Pandas library to import CSV files directly from GitHub.
3. **Data Preprocessing:**
    - Merged the datasets on the product_id column to form a complete dataset.
    - Addressed missing values by categorizing them as 'Unknown.'
    - Extracted relevant columns like order_id, product_id, price, and product_category_name.

Pareto Analysis:

The Pareto analysis was conducted to identify the most profitable product categories based on sales revenue, following these steps:

1. **Grouping and Sorting:**
    - Grouped the data by product_category and summed up the total price for each category.
    - Sorted the categories by descending order of total sales (price).
2. **Cumulative Percentage Calculation:**
    - Calculated the cumulative percentage of sales for each category.
    - Used the cumulative percentage to determine the impact of each category on overall sales.

3. **Visualization:**
    - Employed Seaborn and Matplotlib libraries to create bar and line plots, representing sales distribution and cumulative percentage.
    - Plotted Pareto charts to visualize the "vital few" categories that contribute to most sales.

Results:

Analysis 1: Product Categories Based on Demand

- **Demand Distribution:**
    - A bar plot was created to display the demand across various product categories. The categories were sorted by the number of orders to highlight the most popular products.
    - Categories such as "bed_bath_table," "health_beauty," and "sports_leisure" demonstrated high demand, reflecting consumer preferences.

Analysis 2: Product Categories Based on Sales - Pareto Analysis

- **Top Contributors to Sales:**
    - The Pareto analysis revealed that a small number of product categories accounted for a large portion of sales revenue.
    - The top 20% of categories, including "bed_bath_table," "health_beauty," and "computers_accessories," contributed significantly to the overall sales.
- **Pareto Chart Visualization:**
    - The Pareto chart visualized the distribution of sales, highlighting that the majority of sales revenue comes from a limited number of categories, following the 80/20 rule.
    - The cumulative line indicated that a small set of categories is responsible for a significant portion of sales, allowing the business to focus on these key categories for strategic planning.
- **Top 40 Categories Analysis:**
    - A variation of the Pareto chart was plotted, focusing on the top 40 categories, with the rest combined into a single "Others" category.
    - This approach emphasized the categories that require attention for maximizing sales opportunities while maintaining operational efficiency.

Conclusion:

1. **Key Findings:**
   - The analysis identified that only a handful of product categories contribute disproportionately to sales, emphasizing the Pareto Principle's applicability in retail sales analysis.
   - Categories like "bed_bath_table," "health_beauty," and "computers_accessories" are the primary revenue drivers, suggesting that the retailer should focus on these categories for promotions, stock management, and marketing efforts.

Code:

```
#Getting the data and cleaning it


import  pandas as pd

import numpy as np


#Read the csv file

order_df =
pd.read_csv("https://raw.githubusercontent.com/swapnilsaurav/OnlineRetail/master/order_items.csv")


#Display all the column names

print(list(order_df.columns))


#Required columns

order_df = order_df[['order_id','product_id','price']]

#print(order_df)


prod_df =
pd.read_csv("https://raw.githubusercontent.com/swapnilsaurav/OnlineRetail/master/products.csv")

#Display all the column names

print(list(prod_df.columns))


#Required columns

prod_df = prod_df[['product_id','product_category_name']]

#print(prod_df)


#Read category translation file

cat_df =
pd.read_csv("https://raw.githubusercontent.com/swapnilsaurav/OnlineRetail/master/product_category_name.csv")


#Display all the column names
```

```
print(list(cat_df.columns))

#Output : ['1 product_category_name', '2 product_category_name_english']


#Let's Rename the column names

cat_df = cat_df.rename(columns={'1 product_category_name':'product_category_name','2
product_category_name_english':'product_category'})

print(list(cat_df.columns))


#Final dataset - merge tab1 and tab2

data = pd.merge(order_df,prod_df, on='product_id',how='left')


#Now merge with category to get English category

data = pd.merge(data, cat_df, on='product_category_name',how='left')


#Check for Missing Data Percentage List

for col in data.columns:

   pct_missing = np.mean(data[col].isnull())

   print('{} - {}%'.format(col, round(pct_missing*100)))

print("\n")


#product_category_name - 1% - lets create a new category called Unknown

data['product_category'] = data['product_category'].fillna(("Unknown"))


#Check if all rows have been accounted for

#if not then merge didnt happen correctly


print("Number of rows: \n\n order_items[{}], \n\n
MergedData[{}]".format(order_df.count(),data.count()))


#Note : Number of rows in order_items and MergedData should be same


#if you want to push the content to a csv file and
```

```
#perform manual test, then uncomment the below line
#data.to_csv("TestingMerge.csv")

#We are now ready to perform the Pareto analysis

#Code 2 : Analyzing using Pareto

import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.ticker import PercentFormatter
df=data[['price','product_category']]
df.set_index(data['product_category'])

#Initially test with small dataset to see what you get
#df = df.head(100) #review with the smaller dataset
#print(df)

#Analysis 1 : What is the most in demand product category?
sns.countplot(df['product_category'], order=df['product_category'].value_counts().index)
plt.title('Product Categories based on Demand'.title(),fontsize=20)
plt.ylabel('count'.title(),fontsize=14)
plt.xlabel('product category'.title(),fontsize=14)
plt.xticks(rotation=90, fontsize=10)
plt.yticks(fontsize=12)
plt.show()

#2 : Which categories generates high sales-Pareto
#Sort the values in the descending order
quant_variable = df['price']
by_variable = df['product_category']
```

```
column = 'price'
group_by = 'product_category'

df = df.groupby(group_by)[column].sum().reset_index()
df = df.sort_values(by=column,ascending=False)
df["cumpercentage"] = df[column].cumsum()/df[column].sum()*100
fig, ax = plt.subplots(figsize=(20,5))
ax.bar(df[group_by], df[column], color="C0")
ax2 = ax.twinx()
ax2.plot(df[group_by], df["cumpercentage"], color="C1",
marker="D", ms=7)
ax2.yaxis.set_major_formatter(PercentFormatter())
ax.tick_params(axis="x", rotation=90)
ax.tick_params(axis="y", colors="C0")
ax2.tick_params(axis="y", colors="C1")
plt.title('Product Categories based on Sales'.title(),fontsize=20)

plt.show()

#Variation 2
#Plotting above graph with only top 40 categories, rest as other categories
total=quant_variable.sum()

df = df.groupby(group_by)[column].sum().reset_index()
df = df.sort_values(by=column,ascending=False)
df["cumpercentage"] = df[column].cumsum()/df[column].sum()*100
threshold = df[column].cumsum() /5 #20%

df_above_threshold = df[df['cumpercentage']< threshold]
df=df_above_threshold
df_below_threshold = df[df['cumpercentage'] >= threshold]
```

```
sum = total - df[column].sum()

restbarcumsum = 100 - df_above_threshold['cumpercentage'].max()

rest = pd.Series(['OTHERS', sum, restbarcumsum], index=[group_by,column, 'cumpercentage'])

df = df._append(rest,ignore_index=True)

df.index = df[group_by]

df = df.sort_values(by='cumpercentage',ascending=True)

fig, ax = plt.subplots()

ax.bar(df.index, df[column], color="C0")

ax2 = ax.twinx()

ax2.plot(df.index, df["cumpercentage"], color="C1", marker="D", ms=7)

ax2.yaxis.set_major_formatter(PercentFormatter())

ax.tick_params(axis="x", colors="C0", labelrotation=90)

ax.tick_params(axis="y", colors="C0")

ax2.tick_params(axis="y", colors="C1")

plt.title('Product Categories based on Sales-2'.title(),fontsize=20)

plt.show()
```