Data Analysis Project (Product Sales)

The code presented below is a skeletal structure for a data analysis project on analysis of n number of product sales of m number of distinguished products from a given dataset. This is to be encoded in Jupyter Notebook (PyTorch works as well) and various Python tools such as Pandas, Matplotlib, Numpy, Itertools, etc has been used to pre-process, clean and ready the data for fervent analysis. This code provides key insights such as Sales Graphs, Most Sold Products, Cities with Most Sales, Products often Bought Together, etc to help out in the analysis even more.

The code has been sectioned out for convenience purposes, with proper titles.

Data Preparation

# Data Pre-Processing :

Mainly consists of importing the dataset and creating different data-frames for it according to need (here on the basis of monthly sales).

## Importing Data :

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**from** itertools **import** combinations

**from** collections **import** Counter

*# import data to separate dataframe*

df1 **=** pd**.**read\_csv("Sales\_January\_2019.csv")

df2 **=** pd**.**read\_csv("Sales\_February\_2019.csv")

df3 **=** pd**.**read\_csv("Sales\_March\_2019.csv")

df4 **=** pd**.**read\_csv("Sales\_April\_2019.csv")

df5 **=** pd**.**read\_csv("Sales\_May\_2019.csv")

df6 **=** pd**.**read\_csv("Sales\_June\_2019.csv")

df7 **=** pd**.**read\_csv("Sales\_July\_2019.csv")

df8 **=** pd**.**read\_csv("Sales\_August\_2019.csv")

df9 **=** pd**.**read\_csv("Sales\_September\_2019.csv")

df10 **=** pd**.**read\_csv("Sales\_October\_2019.csv")

df11 **=** pd**.**read\_csv("Sales\_November\_2019.csv")

df12 **=** pd**.**read\_csv("Sales\_December\_2019.csv")

*# concatenate dataframe*

df **=** pd**.**concat([df1, df2, df3, df4, df5, df6, df7, df8, df9, df10, df11, df12])

# Data Profiling :

Is done to see the specifics of the dataset we are working with (number of rows, columns, datatype, unique vales, frequency, etc).

## Data Shape & Specifics :

*# check number of rows and columns*

print(f"Total rows and columns dataset: {df**.**shape[0]} rows and {df**.**shape[1]} columns")

*# check top 5 dataset*

print('-'**\***60)

print("Top 5 dataset: ")

df**.**head()

*# check data type*

df**.**info()

*# check count, unique value, top frequency, and frequency*

df[["Product", "Price Each", "Quantity Ordered"]]**.**describe()

*# check missing value*

print("Number of missing value:")

print("-"**\***30)

df**.**isnull()**.**sum()

Number of missing value:

*# check distinct value*

print("Number of distinct value: ")

print("-"**\***30)

df**.**nunique()

*# check for duplicate data*

print("Number of duplicate data:")

print("-"**\***30)

df**.**duplicated()**.**sum()

# Data Cleaning :

Concerned with cleansing the dataset of erroneous data like duplicates, missing values, etc.

## Removing Erroneous Data :

*# removing missing value*

df **=** df**.**dropna(how**=**"all")

*# removing duplicate data*

df **=** df**.**drop\_duplicates()

*# check*

print("Number of missing value: ")

print(df**.**isnull()**.**sum())

print("\nNumber of duplicate data: ", df**.**duplicated()**.**sum())

df **=** df[df['Order Date']**.**str[0:2]**!=**'Or']

## Changing Datatype :

*# Change the data type*

df["Quantity Ordered"] **=** pd**.**to\_numeric(df["Quantity Ordered"])

df["Price Each"] **=** pd**.**to\_numeric(df["Price Each"])

df["Order Date"] **=** pd**.**to\_datetime(df["Order Date"])

## Changing Data Format :

*# create city column from purchase address*

df["City"] **=** df["Purchase Address"]**.**apply(**lambda** x: x**.**split(',')[1])

*# create a month and our column from purchase date*

df["Month"] **=** df["Order Date"]**.**dt**.**month

df["Hour"] **=** df["Order Date"]**.**dt**.**hour

*# create total sales column*

df["Total Sales"] **=** df["Quantity Ordered"] **\*** df["Price Each"]

*# drop column order date and purchase address*

df**.**drop(columns**=**["Order Date", "Purchase Address"], inplace**=True**)

*# check dataset*

df**.**head()

## Data Correlation Plot :

*# check for data correlation*

plt**.**figure(figsize**=**(10,6))

corr **=** df**.**corr(method**=**"pearson")

sns**.**heatmap(corr, annot**=True**, cmap**=**"mako")

plt**.**title("Feature Correlation")

Gathering Key Insights

# 1. Month with Highest Sales :

## Determination :

df\_month **=** pd**.**DataFrame(df**.**groupby("Month")['Total Sales']**.**sum())**.**reset\_index()

df\_month

## Bar Graph :

fig, ax **=** plt**.**subplots(figsize**=**(12,8))

bars **=** ax**.**bar(

x **=** df\_month['Month'],

height **=** df\_month["Total Sales"]

)

plt**.**title("Total Sales per Month", pad**=**20, fontsize**=**20)

plt**.**xlabel("Month")

plt**.**ylabel("Total Sales")

labels, locations **=** plt**.**yticks()

plt**.**yticks(labels, (labels**/**1)**.**astype(int))

**for** bar **in** bars:

ax**.**text(

bar**.**get\_x() **+** bar**.**get\_width() **/** 2,

bar**.**get\_height() **+** 0.8,

round(bar**.**get\_height(), 1),

horizontalalignment**=**'center'

)

plt**.**show()

# 2. City with Highest Sales :

## Determination :

df\_city **=** pd**.**DataFrame(df**.**groupby("City")["Total Sales"]**.**sum())**.**reset\_index()**.**sort\_values(by**=**'Total Sales', ascending**=False**)

df\_city

## Bar Graph :

fig, ax **=** plt**.**subplots(figsize**=**(12,8))

bars **=** ax**.**bar(

x **=** df\_city['City'],

height **=** df\_city["Total Sales"]

)

plt**.**title("Total Sales per City", pad**=**20, fontsize**=**20)

plt**.**xlabel("City")

plt**.**ylabel("Total Sales")

labels, locations **=** plt**.**yticks()

plt**.**yticks(labels, (labels**/**1)**.**astype(int))

**for** bar **in** bars:

ax**.**text(

bar**.**get\_x() **+** bar**.**get\_width() **/** 2,

bar**.**get\_height() **+** 0.8,

round(bar**.**get\_height(), 1),

horizontalalignment**=**'center'

)

plt**.**show()

# 3. Sales Trends in Top 5 Cities :

## Determination :

*# create variable top city*

top\_city **=** (

df**.**groupby("City")["Total Sales"]

**.**sum()

**.**reset\_index()

**.**sort\_values(by**=**"Total Sales", ascending**=False**)

**.**head()

)

*# create column top\_city*

df["Top City"] **=** df["City"]**.**apply(**lambda** x: x **if**(x **in** top\_city["City"]**.**to\_list())**else** 'other')

## Trend Multi-line Chart :

*# plot multiline chart*

df**.**groupby(['Month', 'Top City'])['Total Sales']**.**sum()**.**unstack()**.**plot(marker**=**'.', cmap**=**"mako")

plt**.**ylabel("Total Sales")

plt**.**xlabel("Month")

plt**.**legend(loc**=**"upper center", bbox\_to\_anchor**=**(1.2, 1))

plt**.**gcf()**.**set\_size\_inches(12,6)

plt**.**show()

# 4. Hourly Orders (for time-based targeted ads)

## Determination :

df\_hour **=** df**.**groupby("Hour")**.**count()

df\_hour

## Trend Chart :

plt**.**figure(figsize**=**(10,6))

plt**.**plot(df**.**groupby("Hour")**.**count())

plt**.**title("Number of Orders by Hour", pad**=**20, fontsize**=**20)

plt**.**ylabel("Number of Order")

plt**.**xlabel("Hour")

plt**.**xticks(range(0,24))

plt**.**annotate("10.929 Orders", xy**=**(10, 10929),

xytext**=**(4, 10929), color**=**'black',

arrowprops**=**dict(arrowstyle**=**'-',

connectionstyle**=**"angle",

color**=**"black")

)

plt**.**annotate("10.905 Orders", xy**=**(21, 10905),

xytext**=**(24, 10905), color**=**'black',

arrowprops**=**dict(arrowstyle**=**'-',

connectionstyle**=**"angle",

color**=**"black")

)

plt**.**show()

# 5. Products often Bought Together (for personalized bundle recommendations)

## Determination :

df\_product **=** df[df['Order ID']**.**duplicated(keep**=False**)]

df\_product['Grouped'] **=** df\_product**.**groupby('Order ID')['Product']**.**transform(**lambda** x: ','**.**join(x))

df\_product\_sold **=** df\_product[['Order ID', 'Grouped']]**.**drop\_duplicates()

count **=** Counter()

**for** row **in** df\_product\_sold['Grouped']:

row\_list **=** row**.**split(',')

count**.**update(Counter(combinations(row\_list, 2)))

**for** key,value **in** count**.**most\_common(10):

print(key, value)

# 6. Most Popular Product :

## Determination :

df\_product **=** df**.**groupby('Product')['Quantity Ordered']**.**sum()**.**reset\_index()

df\_product

## Bar Graph :

fig, ax **=** plt**.**subplots(figsize**=**(12,8))

bars **=** ax**.**bar(

x **=** df\_product['Product'],

height **=** df\_product["Quantity Ordered"]

)

plt**.**title("Total Sales per City", pad**=**20, fontsize**=**20)

plt**.**xlabel("City")

plt**.**ylabel("Total Sales")

labels, locations **=** plt**.**yticks()

plt**.**yticks(labels, (labels**/**1)**.**astype(int))

plt**.**xticks(rotation**=**"vertical")

**for** bar **in** bars:

ax**.**text(

bar**.**get\_x() **+** bar**.**get\_width() **/** 2,

bar**.**get\_height() **+** 0.8,

round(bar**.**get\_height(), 1),

horizontalalignment**=**'center'

)

plt**.**show()

Conclusion

Using the key data insights and accompanying graphs gathered utilizing the above code, it will be way easier for the analyst to draw conclusions and produce a comprehensive report on the sales data given by the company/employer and provide consultation rather than trying to comprehend the raw data (most probably given in csv format). The conclusions drawn from the polished data points, plots and insights provided by this code have been astutely summarized in a follow-up document.

END