Data Analytics Problems on Sales_Data

1. Import necessary libraries for sales Data Analysis

In [3]:
```python
import pandas as pd
import os
```

2.Concatenate each month sale Data (https://github.com/svkarthik86/Assignment/tree/main/Sales_Data) into one dataframe and save the dataframe to annual_sale.csv

In [4]:
```python
os.listdir('C:\\Users\\user\Sales_Data')
```

Out[4]:
```
['annual_sale.csv',
 'Sales_April_2019.csv',
 'Sales_August_2019.csv',
 'Sales_December_2019.csv',
 'Sales_February_2019.csv',
 'Sales_January_2019.csv',
 'Sales_July_2019.csv',
 'Sales_June_2019.csv',
 'Sales_March_2019.csv',
 'Sales_May_2019.csv',
 'Sales_November_2019.csv',
 'Sales_October_2019.csv',
 'Sales_September_2019.csv']
```

In [24]:
```python
path=r"C:\Users\user\Sales_Data\\"
files=[s for s in os.listdir('C:\\Users\\user\Sales_Data')]
sales_data=pd.DataFrame()
for s in files:
    annual_sale=pd.read_csv(path+s)
    annual_sale=pd.concat([sales_data,annual_sale])
    annual_sale.to_csv("annual_sale.csv",index=False)
annual_sale
```

Out[24]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301
...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
11682	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
11683	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

11686 rows × 6 columns

3.Read the annual_sale.csv fom current working directory and store it as annual_sale dataframe , and display the first five row of the dataframe

In [25]:
```python
annual_sale.head(5)
```

Out[25]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301

4.show the metadata information of the annual_sale data frame and check if data is missing or not, if yes How many data are missing.

In [26]:
```python
print(annual_sale.info())
annual_sale.isna().sum().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11686 entries, 0 to 11685
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Order ID          11646 non-null  object
 1   Product           11646 non-null  object
 2   Quantity Ordered  11646 non-null  object
 3   Price Each        11646 non-null  object
 4   Order Date        11646 non-null  object
 5   Purchase Address  11646 non-null  object
dtypes: object(6)
memory usage: 547.9+ KB
None
```

Out[26]:
```
240
```

5.Clean up the data!

5.1 Verify all the column names are in a valid format, if any space between the column name then rename the column names

example: Product ID as Product_ID

In [28]:
```python
annual_sale.columns=[i.replace(" ","_") for i in annual_sale.columns]
annual_sale
```

Out[28]:

	Order_ID	Product	Quantity_Ordered	Price_Each	Order_Date	Purchase_Address
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301
...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
11682	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
11683	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

11686 rows × 6 columns

5.2 check the isnan is present in dataframe, if there is nan is present remove the nan using dropna() function

In [29]:
```python
annual_sale.dropna(inplace=True)
```

Out[29]:

	Order_ID	Product	Quantity_Ordered	Price_Each	Order_Date	Purchase_Address
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301
...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
11682	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
11683	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

11646 rows × 6 columns

5.3 Find The duplicated data present in the data frame, and remove the duplicated data from the dataframe

In [33]:
```python
annual_sale=annual_sale[~(annual_sale.Price_Each=="Price Each")]
annual_sale
```

Out[33]:

	Order_ID	Product	Quantity_Ordered	Price_Each	Order_Date	Purchase_Address
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301
...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
11682	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
11683	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

11629 rows × 6 columns

In [34]:
```python
annual_sale.drop_duplicates(inplace=True)
annual_sale
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_1580\572220041.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  annual_sale.drop_duplicates(inplace=True)
```

Out[34]:

	Order_ID	Product	Quantity_Ordered	Price_Each	Order_Date	Purchase_Address
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301
...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
11682	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
11683	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

11611 rows × 6 columns

6.memory_usage

check memory_usage of Product column , type cast the Product column as "category" type and then check memmory_usage compare the memory utialization and how much percentage effectively reduce the storage space?

In [35]:
```python
annual_sale.Product.memory_usage()
```

Out[35]:
```
185776
```

In [36]:
```python
annual_sale.Product=annual_sale.Product.astype("category")
annual_sale.Product.memory_usage()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_1580\1369973099.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  annual_sale.Product=annual_sale.Product.astype("category")
```

Out[36]:
```
105207
```

7.Create and add a new column

Add month column to annual_sale DataFrame object from 'Order Date' column using Series.str method

annual_sale["month"]

In [44]:
```python
annual_sale['month']=annual_sale.Order_Date.str[:2]
annual_sale
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_1580\2994047926.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  annual_sale['month']=annual_sale.Order_Date.str[:2]
```

Out[44]:

	Order_ID	Product	Quantity_Ordered	Price_Each	Order_Date	Purchase_Address	month	(month, date)
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001	09	09
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301	09	09
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001	09	09
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101	09	09
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301	09	09
...	...	...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001	09	NaN
11682	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016	09	NaN
11683	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016	09	NaN
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016	09	NaN
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016	09	NaN

11611 rows × 8 columns

7.1 converts the datatype of month column as int using astype('int32')

In [45]:
```python
annual_sale.month=annual_sale.month.astype('int32')
annual_sale
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_1580\2799170011.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  annual_sale.month=annual_sale.month.astype('int32')
```

Out[45]:

	Order_ID	Product	Quantity_Ordered	Price_Each	Order_Date	Purchase_Address	month	(month, date)
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001	9	09
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301	9	09
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001	9	09
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101	9	09
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301	9	09
...	...	...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001	9	NaN
11682	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016	9	NaN
11683	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016	9	NaN
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016	9	NaN
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016	9	NaN

11611 rows × 8 columns

7.2 Add sale column to annual_sale DataFrame object using following calculation

sales = Quantity_Ordered * Price_Each

In [46]:
```python
annual_sale['sale']=annual_sale.Quantity_Ordered.astype(float)*annual_sale.Price_Each.astype(float)
annual_sale
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_1580\3649044914.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  annual_sale['sale']=annual_sale.Quantity_Ordered.astype(float)*annual_sale.Price_Each.astype(float)
```

Out[46]:

	Order_ID	Product	Quantity_Ordered	Price_Each	Order_Date	Purchase_Address	month	(month, date)	sale
0	248151	AA Batteries (4-pack)	4	3.84	09/17/19 14:44	380 North St, Los Angeles, CA 90001	9	09	15.36
1	248152	USB-C Charging Cable	2	11.95	09/29/19 10:19	511 8th St, Austin, TX 73301	9	09	23.90
2	248153	USB-C Charging Cable	1	11.95	09/16/19 17:48	151 Johnson St, Los Angeles, CA 90001	9	09	11.95
3	248154	27in FHD Monitor	1	149.99	09/27/19 07:52	355 Hickory St, Seattle, WA 98101	9	09	149.99
4	248155	USB-C Charging Cable	1	11.95	09/01/19 19:03	125 5th St, Atlanta, GA 30301	9	09	11.95
...	...	...	...	...	...	...	...	...	...
11681	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001	9	NaN	8.97
11682	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016	9	NaN	700.00
11683	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016	9	NaN	700.00
11684	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016	9	NaN	379.99
11685	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016	9	NaN	11.95

11611 rows × 9 columns

8. Find out the day,in which sales is high? using gruop by agg function

In [47]:
```python
annual_sale.groupby(annual_sale.Order_Date.str[:8])["sale"].sum().sort_values(ascending=False)
```

Out[47]:
```
Order_Date
09/09/19    79444.04
09/14/19    79399.76
09/25/19    78245.17
09/13/19    78224.28
09/11/19    77830.23
09/26/19    77607.77
09/18/19    76551.31
09/12/19    74776.11
09/10/19    74257.35
09/15/19    73242.32
09/29/19    73135.43
09/02/19    72260.43
09/30/19    72105.25
09/28/19    71946.63
09/04/19    71784.82
09/07/19    70665.44
09/06/19    70087.10
09/23/19    70082.03
09/05/19    67822.56
09/24/19    66751.42
09/27/19    66319.23
09/01/19    65816.65
09/08/19    63948.30
09/19/19    63822.47
09/20/19    63013.83
09/21/19    62876.26
09/17/19    62443.57
09/03/19    62059.34
09/16/19    57940.24
09/22/19    47909.61
10/01/19     3353.31
Name: sale, dtype: float64
```

In [48]:
```python
annual_sale.groupby(annual_sale.Order_Date.str[:8])["sale"].sum().sort_values(ascending=False)[:1]
```

Out[48]:
```
Order_Date
09/09/19    79444.04
Name: sale, dtype: float64
```

9. What Product is most frequently purchased over the all period

In [49]:
```python
annual_sale.Product.value_counts()[:1]
```

Out[49]:
```
USB-C Charging Cable    1451
Name: Product, dtype: int64
```

10. List out the Product price above 200$

In [50]:
```python
annual_sale[annual_sale.Price_Each.astype(float)>200].loc[:,["Product","Price_Each"]]
```

Out[50]:

	Product	Price_Each
5	34in Ultrawide Monitor	379.99
8	Vareebadd Phone	400.0
16	Google Phone	600.0
25	iPhone	700.0
33	34in Ultrawide Monitor	379.99
...	...	...
11672	ThinkPad Laptop	999.99
11677	Google Phone	600
11682	iPhone	700
11683	iPhone	700
11684	34in Ultrawide Monitor	379.99

2525 rows × 2 columns

In [ ]: