

Essentials of Data Science(EDS)

Theory Assignment

Submitted by :

Name: Ritkriti Singh

Batch: CS62

Roll No: 44

Assignment: Theory Activity No. 1

This is for all division students. You have to complete this assignment as per the instructions of your theory teacher.

Formulate 20 problem statements for a given dataset using Numpy and Pandas and Apply Numpy and pandas methods to find the solution for the formulated problem statements.

Each one will take a real-life dataset.

Submission:

Given Dataset : Sales Dataset (Link - [C:\Users\user\Downloads\archive \(3\).zip](C:\Users\user\Downloads\archive (3).zip))

AutoSave

OFF

sales_data_sample.csv - Read... - Saved to this PC

Search

FileHomeInsertPage LayoutFormulasDataReviewViewDeveloperHelp

PasteClipboard

Font

Alignment

Number

Styles

Conditional Formatting

Format as Table

Cell Styles

Insert

Delete

Format

Cells

Sum

Sort & Filter

Find & Select

Comments

Share

F4

07-01-2003 00:00:00

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	ORDERNU	QUANTITY	PRICEEACI	ORDERLIN	SALES	ORDERDA	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCTI	MSRP	PRODUCTI	CUSTOME	PHONE	ADDRESSL	ADDRESSL	CITY	STATE	POSTALCC	COUN
2	10107	30	95.7	2	2871	2/24/2003	Shipped	1	2	2003	Motorcycl	95 S10_1678	Land of Tc	2.13E+09	897 Long Airport Ave	NYC	NY			10022	USA
3	10121	34	81.35	5	2765.9	#####	Shipped	2	5	2003	Motorcycl	95 S10_1678	Reims Col	26.47.155	59 rue de l'Abbaye	Reims				51100	France
4	10134	41	94.74	2	3884.34	#####	Shipped	3	7	2003	Motorcycl	95 S10_1678	Lyon Souv	+33 1 46 61 27	rue du Colonel P	Paris				75508	France
5	10145	45	83.26	6	3746.7	8/25/2003	Shipped	3	8	2003	Motorcycl	95 S10_1678	Toys4Grov	6.27E+09	78934 Hillside Dr.	Pasadena	CA			90003	USA
6	10159	49	100	14	5205.27	#####	Shipped	4	10	2003	Motorcycl	95 S10_1678	Corporate	6.51E+09	7734 Strong St.	San Franci	CA				USA
7	10168	36	96.66	1	3479.76	10/28/200	Shipped	4	10	2003	Motorcycl	95 S10_1678	Technics	6.51E+09	9408 Furth Circle	Burlingam	CA			94217	USA
8	10180	29	86.13	9	2497.77	#####	Shipped	4	11	2003	Motorcycl	95 S10_1678	Daedalus	20.16.155	184, chaussée de Tour	Lille				59000	France
9	10188	48	100	1	5512.32	11/18/200	Shipped	4	11	2003	Motorcycl	95 S10_1678	Herkku Gi	+47 2267 3	Drammen 121, PR 74	Bergen		N 5804	Norw		
10	10201	22	98.57	2	2168.54	#####	Shipped	4	12	2003	Motorcycl	95 S10_1678	Mini Whe	6.51E+09	5557 North Pendale	San Franci	CA				USA
11	10211	41	100	14	4708.44	1/15/2004	Shipped	1	1	2004	Motorcycl	95 S10_1678	Auto Cané (1)	47.55.€ 25,	rue Lauriston	Paris				75016	France
12	10223	37	100	1	3965.66	2/20/2004	Shipped	1	2	2004	Motorcycl	95 S10_1678	Australian	03 9520 45	636 St Kilc Level 3	Melbourn	Victoria			3004	Austr
13	10237	23	100	7	2333.12	#####	Shipped	2	4	2004	Motorcycl	95 S10_1678	Vitachrom	2.13E+09	2678 King Suite 101	NYC	NY			10022	USA
14	10251	28	100	2	3188.64	5/18/2004	Shipped	2	5	2004	Motorcycl	95 S10_1678	Tekni Coll	2.02E+09	7476 Moss Rd.	Newark	NJ			94019	USA
15	10263	34	100	2	3676.76	6/28/2004	Shipped	2	6	2004	Motorcycl	95 S10_1678	Gift Depo	2.04E+09	25593 South Bay Ln.	Bridgewater	CT			97562	USA
16	10275	45	92.83	1	4177.35	7/23/2004	Shipped	3	7	2004	Motorcycl	95 S10_1678	La Rochell	40.67.855	€ 67, rue des Cinquant	Nantes				44000	France
17	10285	36	100	6	4099.68	8/27/2004	Shipped	3	8	2004	Motorcycl	95 S10_1678	Marta's Re	6.18E+09	39323 Spinnaker Dr.	Cambridge	MA			51247	USA
18	10299	23	100	9	2597.39	9/30/2004	Shipped	3	9	2004	Motorcycl	95 S10_1678	Toys of Fi	90-224 85	Keskuskatu 45	Helsinki				21240	Finlan
19	10309	41	100	5	4394.38	10/15/200	Shipped	4	10	2004	Motorcycl	95 S10_1678	Baane Mir	07-98 955	Erling Skakkes gate 7	Stavern				4110	Norw
20	10318	46	94.74	1	4358.04	#####	Shipped	4	11	2004	Motorcycl	95 S10_1678	Diecast Cl	2.16E+09	7586 Pompton St.	Allentown	PA			70267	USA
21	10329	42	100	1	4396.14	11/15/200	Shipped	4	11	2004	Motorcycl	95 S10_1678	Land of Tc	2.13E+09	897 Long Airport Ave	NYC	NY			10022	USA

sales_data_sample

+

Ready

Accessibility: Unavailable

ENG IN

19:46

30-04-2022

100%

```
import pandas as pd
import numpy as np

df = pd.read_csv('/content/sample_data/sales_data_sample.csv', encoding='latin1')
df
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1	2	2003	...
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	Shipped	2	5	2003	...
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	Shipped	3	7	2003	...
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3	8	2003	...
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	Shipped	4	10	2003	...

Problem Statements and solutions :

1. Calculate the total revenue generated from all sales.

```
df['Revenue'] = df['QUANTITYORDERED'] * df['PRICEEACH']
total_revenue = df['Revenue'].sum()
print(f"Total Revenue: ${total_revenue}")
```

Total Revenue: \$8290886.789999999

2. Determine the average revenue per order.

```
df['Revenue'] = df['QUANTITYORDERED'] * df['PRICEEACH']
```

3. Identify the top 5 products that generated the highest revenue.

```
product_revenue = df.groupby('PRODUCTLINE')['Revenue'].sum()
top_5_products = product_revenue.sort_values(ascending=False).head(5)
print("Top 5 Products by Revenue:")
print(top_5_products)
```

```
Top 5 Products by Revenue:
PRODUCTLINE
Classic Cars      2968546.40
Vintage Cars      1644212.05
Motorcycles        971086.29
Trucks and Buses   947355.18
Planes             877942.21
Name: Revenue, dtype: float64
```

4. Analyze the total revenue generated each month.

```
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])
df['Month'] = df['ORDERDATE'].dt.month
monthly_revenue = df.groupby('Month')['Revenue'].sum()
print("Total Revenue Generated Each Month:")
print(monthly_revenue)
```

Total Revenue Generated Each Month:

Month

1	659582.29
2	668328.15
3	626186.73
4	560334.46
5	756812.91
6	384743.59
7	420973.34
8	552132.91
9	474900.12
10	919036.70
11	1744682.45
12	523173.14

Name: Revenue, dtype: float64

5. Determine which month had the highest sales revenue.

```
highest_revenue_month = monthly_revenue.idxmax()
highest_revenue = monthly_revenue.max()

print(f"The month with the highest sales revenue was {highest_revenue_month}, with a total revenue of ${highest_revenue:.2f}")
```

The month with the highest sales revenue was 11, with a total revenue of \$1744682.45

6. Identify the day of the week with the highest average sales revenue.

```
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])

df['Revenue'] = df['QUANTITYORDERED'] * df['PRICEEACH']

df['DayOfWeek'] = df['ORDERDATE'].dt.day_name()

average_revenue_by_day = df.groupby('DayOfWeek')['Revenue'].mean()

highest_revenue_day = average_revenue_by_day.idxmax()

print(f"The day of the week with the highest average sales revenue is {highest_revenue_day}.")
```

The day of the week with the highest average sales revenue is Tuesday.

7. Identify the hour of the day when most orders are placed.

```
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])
df['Hour'] = df['ORDERDATE'].dt.hour
orders_by_hour = df.groupby('Hour')['ORDERNUMBER'].count()
most_orders_hour = orders_by_hour.idxmax()
print(f"The hour with the most orders is: {most_orders_hour}")
```

The hour with the most orders is: 0

8. Find the product with the highest quantity sold.

```
product_quantities = df.groupby('PRODUCTCODE')['QUANTITYORDERED'].sum()
highest_quantity_product = product_quantities.idxmax()
print(f"The product with the highest quantity sold is: {highest_quantity_product}")
```

The product with the highest quantity sold is: S18_3232

9. Analyze the relationship between product price and quantity sold.

```
product_quantities = df.groupby('PRODUCTCODE')['QUANTITYORDERED'].sum().reset_index()
product_data = pd.merge(product_quantities, df[['PRODUCTCODE', 'PRICEEACH']], on='PRODUCTCODE', how='left')
product_data = product_data.drop_duplicates(subset=['PRODUCTCODE'])
correlation = product_data['PRICEEACH'].corr(product_data['QUANTITYORDERED'])
print(f"Correlation coefficient: {correlation}")
```

Correlation coefficient: 0.10636620722949874

10. Identify which products have the most consistent revenue generation over time.

```
[15] df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])
df['Month'] = df['ORDERDATE'].dt.month
df['Revenue'] = df['QUANTITYORDERED'] * df['PRICEEACH']
product_monthly_revenue = df.groupby(['PRODUCTCODE', 'Month'])['Revenue'].sum().reset_index()
```

11. Identify the top 5 cities with the highest number of orders.

```
city_orders = df.groupby('CITY')['ORDERNUMBER'].nunique().reset_index()
top_5_cities = city_orders.sort_values(by=['ORDERNUMBER'], ascending=False).head(5)
print(top_5_cities)
```

	CITY	ORDERNUMBER
34	Madrid	31
61	San Rafael	17
42	NYC	16
63	Singapore	9
52	Paris	9

12. Calculate the average quantity ordered for each product.

```
average_quantity_by_product = df.groupby('PRODUCTCODE')['QUANTITYORDERED'].mean().reset_index()
print(average_quantity_by_product)
```

	PRODUCTCODE	QUANTITYORDERED
0	S10_1678	36.307692
1	S10_1949	34.321429
2	S10_2016	35.692308
3	S10_4698	35.423077
4	S10_4757	35.259259
..
104	S700_3505	35.269231
105	S700_3962	32.769231
106	S700_4002	38.111111
107	S72_1253	34.074074
108	S72_3212	35.653846

13. Analyze daily sales revenue to identify trends.

```
import matplotlib.pyplot as plt

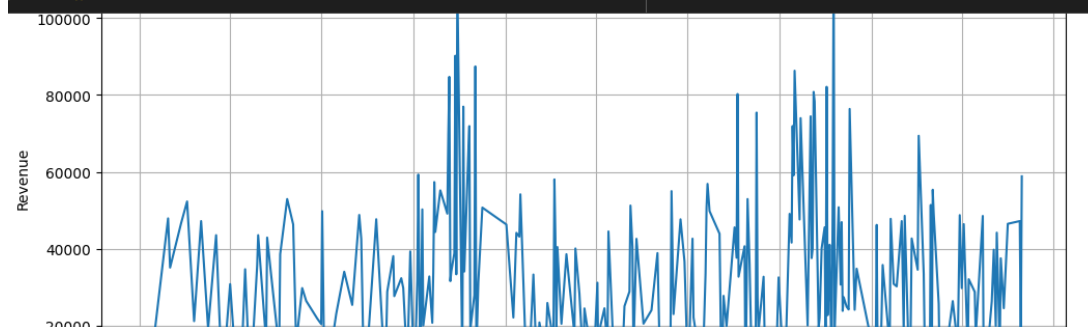
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])

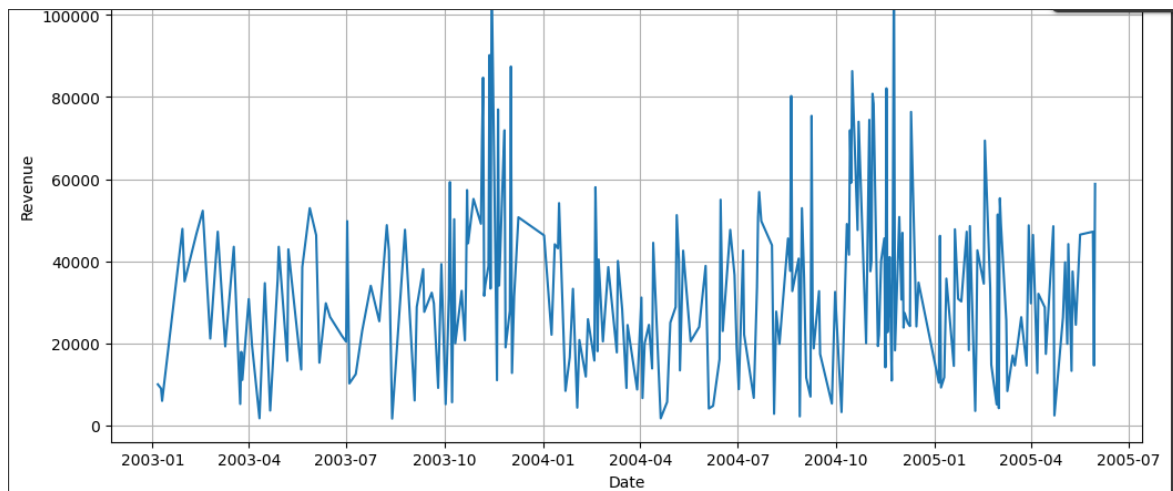
df['Date'] = df['ORDERDATE'].dt.date

daily_revenue = df.groupby('Date')['Revenue'].sum().reset_index()

plt.figure(figsize=(12, 6))
plt.plot(daily_revenue['Date'], daily_revenue['Revenue'])
plt.title('Daily Sales Revenue Trend')
plt.xlabel('Date')
plt.ylabel('Revenue')
plt.grid(True)
plt.show()

daily_revenue['MovingAverage'] = daily_revenue['Revenue'].rolling(window=7).mean() # 7-day moving average
plt.figure(figsize=(12, 6))
plt.plot(daily_revenue['Date'], daily_revenue['Revenue'], label='Daily Revenue')
plt.plot(daily_revenue['Date'], daily_revenue['MovingAverage'], label='7-Day Moving Average', color='red')
plt.title('Daily Sales Revenue Trend with Moving Average')
plt.show()
```





14. Find the most frequent purchase address.

```
address_counts = df['ADDRESSLINE1'].value_counts()
most_frequent_address = address_counts.index[0]

print(f"The most frequent purchase address is: {most_frequent_address}")

The most frequent purchase address is: C/ Moralarzal, 86
```

15. Calculate the total number of units sold across all products.

```
total_units_sold = df['QUANTITYORDERED'].sum()
print(f"Total units sold across all products: {total_units_sold}")

Total units sold across all products: 99067
```

16. Determine the average price for each product.

```
average_price_by_product = df.groupby('PRODUCTCODE')['PRICEEACH'].mean()
print(average_price_by_product)
```

PRODUCTCODE	PRICEEACH
S10_1678	92.607692
S10_1949	100.000000
S10_2016	94.365769
S10_4698	98.593846
S10_4757	94.190000
...	...
S700_3505	92.877308
S700_3962	91.812308
S700_4002	73.621111
S72_1253	55.897037
S72_3212	63.865769

Name: PRICEEACH, Length: 109, dtype: float64

17. Identify pairs of products that are frequently sold together.

```

unique_orders = df['ORDERNUMBER'].unique()

pair_counts = {}

# 3. Iterate through each order
for order in unique_orders:
    # Get the products in the current order
    products_in_order = df.loc[df['ORDERNUMBER'] == order, 'PRODUCTCODE'].tolist()

    # If there's more than one product in the order
    if len(products_in_order) > 1:
        # Create all possible pairs of products within the order
        for i in range(len(products_in_order)):
            for j in range(i + 1, len(products_in_order)):
                pair = tuple(sorted([products_in_order[i], products_in_order[j]]))

                # Update the pair count in the dictionary
                pair_counts[pair] = pair_counts.get(pair, 0) + 1

# 4. Convert the dictionary to a Pandas DataFrame for easier handling
pair_counts_df = pd.DataFrame(list(pair_counts.items()), columns=['Product Pair', 'Count'])

# 5. Sort the DataFrame by count in descending order
frequent_pairs = pair_counts_df.sort_values(by=['Count'], ascending=False)

# 4. Convert the dictionary to a Pandas DataFrame for easier handling
pair_counts_df = pd.DataFrame(list(pair_counts.items()), columns=['Product Pair', 'Count'])

# 5. Sort the DataFrame by count in descending order
frequent_pairs = pair_counts_df.sort_values(by=['Count'], ascending=False)

# 6. Display the top frequently sold together pairs (e.g., top 10)
print(frequent_pairs.head(10))

```

	Product Pair	Count
428	(S700_2047, S72_1253)	26
792	(S24_2841, S24_3420)	26
1126	(S18_2319, S18_3232)	26
292	(S24_3949, S700_4002)	26
311	(S50_1341, S700_1691)	26
530	(S18_2957, S18_3136)	26
457	(S10_1949, S18_1097)	25
924	(S700_3962, S72_3212)	25
477	(S10_4962, S18_4600)	25
1147	(S24_2840, S32_2509)	25

18. Calculate each product's contribution to total revenue as a percentage.

```
df['Revenue'] = df['QUANTITYORDERED'] * df['PRICEEACH']
total_revenue = df['Revenue'].sum()
print(f"Total Revenue: ${total_revenue}")
product_revenue = df.groupby('PRODUCTCODE')['Revenue'].sum()
product_contribution = (product_revenue / total_revenue) * 100
print("Product Contribution to Total Revenue (%):")
print(product_contribution)
```

```
Total Revenue: $8290886.789999999
Product Contribution to Total Revenue (%):
PRODUCTCODE
S10_1678      1.069133
S10_1949      1.159104
S10_2016      1.050801
S10_4698      1.096533
S10_4757      1.078773
...
S700_3505     1.025599
S700_3962     0.936506
S700_4002     0.909412
S72_1253      0.623116
S72_3212      0.707085
Name: Revenue, Length: 109, dtype: float64
```

19. Determine the total number of orders placed.

```
total_orders = len(df['ORDERNUMBER'].unique())
print(f"Total number of orders placed: {total_orders}")

Total number of orders placed: 307
```

20. Calculate average revenue generated per city.


```
city_revenue = df.groupby('CITY')['Revenue'].sum()
city_orders = df.groupby('CITY')['ORDERNUMBER'].nunique()
average_revenue_per_city = city_revenue / city_orders
print("Average Revenue per City:")
print(average_revenue_per_city)
```

Average Revenue per City:

CITY

Aarhus 38739.395000

Allentown 24094.567500

Barcelona 24642.333333

Bergamo 42381.213333

Bergen 28779.703333

...

Toulouse 19628.426667

Tsawassen 35746.165000

Vancouver 30298.955000

Versailles 26682.485000

White Plains 36825.240000

Length: 73, dtype: float64