



EDS Theory Activity

Name :- Sneha Chaurasia

Roll No :- CS7-44

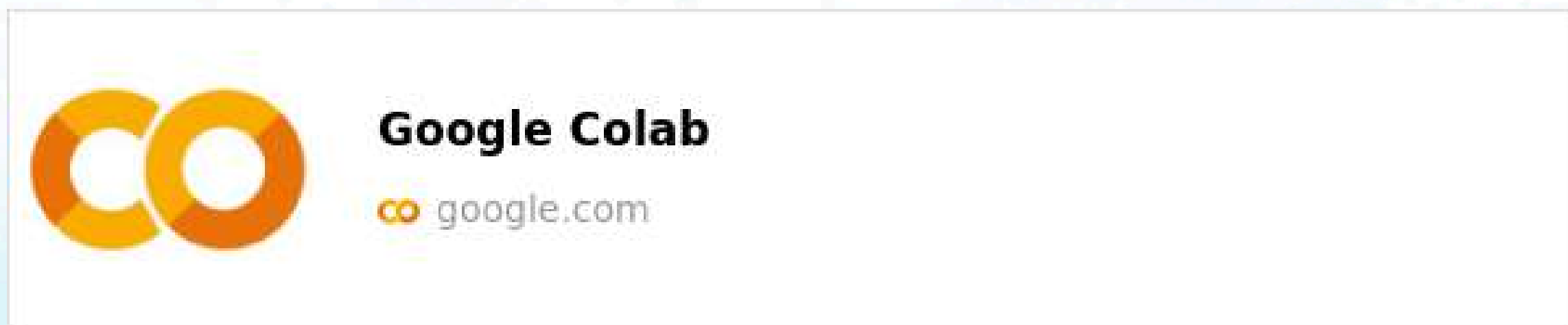
PRN :- 202401110046

Divison :- CS7

Batch :-CS72



You can check the 20 problem statements by clicking on the given link



[https://colab.research.google.com/drive/
1qvjQTxCNDkfOcYH_UHZ5M5a2eGfv
Zplh?usp=sharing](https://colab.research.google.com/drive/1qvjQTxCNDkfOcYH_UHZ5M5a2eGfvZplh?usp=sharing)

This is the dataset that i have imported from the kaggle.

✓
2s



```
import kagglehub
```

```
# Download latest version
```

```
path = kagglehub.dataset_download("kyanyoga/sample-sales-data")
```

```
print("Path to dataset files:", path)
```



```
Path to dataset files: /kaggle/input/sample-sales-data
```

Sales Dataset

1. Product with maximum revenue and the total revenue

```
[ ] import pandas as pd

# 1. Load the CSV file (adjust path if needed)
df = pd.read_csv('/root/.cache/kagglehub/datasets/kyanyoga/sample-sales-data/versions/1/sales_data_sample.csv', encoding='latin1')

# 2. Group by 'PRODUCTCODE' and sum the 'SALES'
product_sales = df.groupby('PRODUCTCODE')['SALES'].sum()

# 3. Find the PRODUCTCODE with the maximum revenue
max_revenue_product = product_sales.idxmax()
max_revenue_amount = product_sales.max()

# 4. Print the result
print(f"The product with the maximum revenue is '{max_revenue_product}' with total sales of ${max_revenue_amount:.2f}.")
```

➞ The product with the maximum revenue is 'S18_3232' with total sales of \$288245.42.

2. Find the total sales made in the USA

```
import pandas as pd

# Define the path to your dataset
file_path = '/root/.cache/kagglehub/datasets/kyanyoga/sample-sales-data/versions/1/sales_data_sample.csv'

# Load the dataset
df = pd.read_csv(file_path, encoding='latin1') # Using latin1 because we know the file isn't UTF-8

# Filter for USA sales
usa_sales = df[df['COUNTRY'] == 'USA']

# Calculate total sales
total_sales_usa = usa_sales['SALES'].sum()

print(f"Total Sales made in USA: ${total_sales_usa:,.2f}")
```

➡ Total Sales made in USA: \$3,627,982.83

3. Top 5 customers with the highest total sales

```
import pandas as pd

# Correct file path
file_path = ('/root/.cache/kagglehub/datasets/kyanyoga/sample-sales-data/versions/1/sales_data_sample.csv')

# Load dataset
df = pd.read_csv(file_path ,encoding='latin1')

# Group by CUSTOMERNAME and sum the SALES
customer_sales = df.groupby('CUSTOMERNAME')['SALES'].sum()

# Sort in descending order
top_5_customers = customer_sales.sort_values(ascending=False).head(5)

# Display the results
print("Top 5 Customers with Highest Total Sales:\n")
print(top_5_customers)
```

Output:-

```
➞ Top 5 Customers with Highest Total Sales:

CUSTOMERNAME
Euro Shopping Channel      912294.11
Mini Gifts Distributors Ltd. 654858.06
Australian Collectors, Co.  200995.41
Muscle Machine Inc          197736.94
La Rochelle Gifts          180124.90
Name: SALES, dtype: float64
```


4. Convert ORDERDATE to datetime format and extract year, month, and day as new columns

```
[ ] import pandas as pd

# Define the path to your dataset
file_path = '/root/.cache/kagglehub/datasets/kyanyoga/sample-sales-data/versions/1/sales_data_sample.csv'

# Load the dataset
df = pd.read_csv(file_path, encoding='latin1') # Using latin1 to handle special characters

# Convert 'ORDERDATE' to datetime format
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'], errors='coerce')

# Extract year, month, and day into new columns
df['ORDER_YEAR'] = df['ORDERDATE'].dt.year
df['ORDER_MONTH'] = df['ORDERDATE'].dt.month
df['ORDER_DAY'] = df['ORDERDATE'].dt.day

# Show the updated dataframe (first few rows)
print(df[['ORDERDATE', 'ORDER_YEAR', 'ORDER_MONTH', 'ORDER_DAY']].head())
```



| | ORDERDATE | ORDER_YEAR | ORDER_MONTH | ORDER_DAY |
|---|------------|------------|-------------|-----------|
| 0 | 2003-02-24 | 2003 | 2 | 24 |
| 1 | 2003-05-07 | 2003 | 5 | 7 |
| 2 | 2003-07-01 | 2003 | 7 | 1 |
| 3 | 2003-08-25 | 2003 | 8 | 25 |
| 4 | 2003-10-10 | 2003 | 10 | 10 |

5. Sorting the dataset by SALES in descending order and finding the top 10 highest sale transactions

```
[ ] import pandas as pd
import os

# Path to the directory
folder_path = '/kaggle/input/sample-sales-data'

# List files in the directory
files = os.listdir(folder_path)
print(files) # See available files

# Find the CSV file
csv_file = [file for file in files if file.endswith('.csv')][0]

# Full path to the CSV file
file_path = os.path.join(folder_path, csv_file)


# Load the dataset
# Specify the encoding as 'latin1' to handle the special characters
data = pd.read_csv(file_path, encoding='latin1')

# Check the column names first
print(data.columns)

# Assuming the sales column is named 'SALES'
# Sort the data by 'SALES' in descending order
sorted_data = data.sort_values(by='SALES', ascending=False)

# Get the top 10 highest sale transactions
top_10_sales = sorted_data.head(10)

# Display the result
print(top_10_sales)
```



Output:-

```
['sales_data_sample.csv']
```

```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',  
      'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',  
      'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',  
      'ADDRESSLINE1', 'ADDRESSLINE2', 'CITY', 'STATE', 'POSTALCODE',  
      'COUNTRY', 'TERRITORY', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME',  
      'DEALSIZE'],  
      dtype='object')
```

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES |
|------|-------------|-----------------|-----------|-----------------|---------|
| 598 | 10407 | 76 | 100.0 | 2 | 14082.8 |
| 744 | 10322 | 50 | 100.0 | 6 | 12536.5 |
| 53 | 10424 | 50 | 100.0 | 6 | 12001.0 |
| 1062 | 10412 | 60 | 100.0 | 9 | 11887.8 |
| 104 | 10403 | 66 | 100.0 | 9 | 11886.6 |
| 1995 | 10405 | 76 | 100.0 | 3 | 11739.7 |
| 44 | 10312 | 48 | 100.0 | 3 | 11623.7 |
| 1133 | 10333 | 46 | 100.0 | 2 | 11336.7 |
| 188 | 10127 | 46 | 100.0 | 2 | 11279.2 |
| 30 | 10150 | 45 | 100.0 | 8 | 10993.5 |

| | ORDERDATE | STATUS | QTR_ID | MONTH_ID | YEAR_ID | ... | \ |
|------|-----------------|------------|--------|----------|---------|-----|---|
| 598 | 4/22/2005 0:00 | On Hold | 2 | 4 | 2005 | ... | |
| 744 | 11/4/2004 0:00 | Shipped | 4 | 11 | 2004 | ... | |
| 53 | 5/31/2005 0:00 | In Process | 2 | 5 | 2005 | ... | |
| 1062 | 5/3/2005 0:00 | Shipped | 2 | 5 | 2005 | ... | |
| 104 | 4/8/2005 0:00 | Shipped | 2 | 4 | 2005 | ... | |
| 1995 | 4/14/2005 0:00 | Shipped | 2 | 4 | 2005 | ... | |
| 44 | 10/21/2004 0:00 | Shipped | 4 | 10 | 2004 | ... | |
| 1133 | 11/18/2004 0:00 | Shipped | 4 | 11 | 2004 | ... | |
| 188 | 6/3/2003 0:00 | Shipped | 2 | 6 | 2003 | ... | |
| 30 | 9/19/2003 0:00 | Shipped | 3 | 9 | 2003 | ... | |

| | ADDRESSLINE1 | ADDRESSLINE2 | CITY | STATE | \ |
|------|--------------------------------------|--------------|---------------|-------|---|
| 598 | 3086 Ingle Ln. | NaN | San Jose | CA | |
| 744 | 2304 Long Airport Avenue | NaN | Nashua | NH | |
| 53 | C/ Moralzarzal, 86 | NaN | Madrid | NaN | |
| 1062 | C/ Moralzarzal, 86 | NaN | Madrid | NaN | |
| 104 | Berkeley Gardens 12 Brewery | NaN | Liverpool | NaN | |
| 1995 | 24, place Kluber | NaN | Strasbourg | NaN | |
| 44 | 5677 Strong St. | NaN | San Rafael | CA | |
| 1133 | 5557 North Pendale Street | NaN | San Francisco | CA | |
| 188 | 4092 Furth Circle | Suite 400 | NYC | NY | |
| 30 | Bronz Sok., Bronz Apt. 3/6 Tesvikiye | NaN | Singapore | NaN | |

| | POSTALCODE | COUNTRY | TERRITORY | CONTACTLASTNAME | CONTACTFIRSTNAME | DEALSIZE |
|------|------------|-----------|-----------|-----------------|------------------|----------|
| 598 | 94217 | USA | NaN | Frick | Sue | Large |
| 744 | 62005 | USA | NaN | Young | Valarie | Large |
| 53 | 28034 | Spain | EMEA | Freyre | Diego | Large |
| 1062 | 28034 | Spain | EMEA | Freyre | Diego | Large |
| 104 | WX1 6LT | UK | EMEA | Devon | Elizabeth | Large |
| 1995 | 67000 | France | EMEA | Citeaux | Frederique | Large |
| 44 | 97562 | USA | NaN | Nelson | Valarie | Large |
| 1133 | NaN | USA | NaN | Murphy | Julie | Large |
| 188 | 10022 | USA | NaN | Young | Jeff | Large |
| 30 | 79903 | Singapore | Japan | Natividad | Eric | Large |

[10 rows x 25 columns]

6. Identify and count the different order statuses (STATUS column).

```
import pandas as pd
import os

folder_path = '/kaggle/input/sample-sales-data'

files = os.listdir(folder_path)

csv_file = [file for file in files if file.endswith('.csv')][0]

file_path = os.path.join(folder_path, csv_file)

data = pd.read_csv(file_path, encoding='latin1')

print(data.columns)
status_counts = data['STATUS'].value_counts()

print(status_counts)
```

Output :-

```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
      'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',
      'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',
      'ADDRESSLINE1', 'ADDRESSLINE2', 'CITY', 'STATE', 'POSTALCODE',
      'COUNTRY', 'TERRITORY', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME',
      'DEALSIZE'],
      dtype='object')
```

```
STATUS
Shipped      2617
Cancelled      60
Resolved      47
On Hold       44
In Process    41
Disputed      14
Name: count, dtype: int64
```

7. Group data by YEAR_ID and calculate the total sales per year.

```
import pandas as pd
import os

folder_path = '/kaggle/input/sample-sales-data'

files = os.listdir(folder_path)

csv_file = [file for file in files if file.endswith('.csv')][0]

file_path = os.path.join(folder_path, csv_file)

data = pd.read_csv(file_path, encoding='latin1')
print(data.columns)

# Group by 'YEAR_ID' and sum up the 'SALES'
sales_per_year = data.groupby('YEAR_ID')['SALES'].sum()

# Display the total sales per year
print(sales_per_year)
```

Output :-

```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
      'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',
      'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',
      'ADDRESSLINE1', 'ADDRESSLINE2', 'CITY', 'STATE', 'POSTALCODE',
      'COUNTRY', 'TERRITORY', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME',
      'DEALSIZE'],
      dtype='object')
YEAR_ID
2003    3516979.54
2004    4724162.60
2005    1791486.71
Name: SALES, dtype: float64
```

8. Find the product line with the maximum average sale value.

```
[ ] import pandas as pd
import os

folder_path = '/kaggle/input/sample-sales-data'

# List files in the directory
files = os.listdir(folder_path)

csv_file = [file for file in files if file.endswith('.csv')][0]

file_path = os.path.join(folder_path, csv_file)

data = pd.read_csv(file_path, encoding='latin1')

product_line_avg_sales = data.groupby('PRODUCTLINE')['SALES'].mean()

# Find the product line with the maximum average sales
max_avg_sales_product_line = product_line_avg_sales.idxmax()
max_avg_sales_value = product_line_avg_sales.max()

# Display the result
print(f"The product line with the maximum average sale value is '{max_avg_sales_product_line}' with an average sale value of ${max_avg_sales_value:.2f}")
```

➡ The product line with the maximum average sale value is 'Classic Cars' with an average sale value of \$4053.38.

9. Identify Sales>3000 and countrywise

```
import pandas as pd
import os

folder_path = '/kaggle/input/sample-sales-data'

files = os.listdir(folder_path)

csv_file = [file for file in files if file.endswith('.csv')][0]

file_path = os.path.join(folder_path, csv_file)

# Load the dataset with 'latin1' encoding
data = pd.read_csv(file_path, encoding='latin1')

# Filter for sales greater than 3000
filtered_data = data[data['SALES'] > 3000]

# Group by country and sum sales
country_sales = filtered_data.groupby('COUNTRY')['SALES'].sum()

# Display the result
print("Sales greater than 3000, grouped by country:")
print(country_sales)
```

Output :-

→ Sales greater than 3000, grouped by country:

| COUNTRY | |
|-------------|------------|
| Australia | 441497.04 |
| Austria | 158106.18 |
| Belgium | 71531.37 |
| Canada | 146495.37 |
| Denmark | 189937.37 |
| Finland | 241754.28 |
| France | 822611.09 |
| Germany | 163647.85 |
| Ireland | 42262.31 |
| Italy | 241699.31 |
| Japan | 126567.38 |
| Norway | 231544.93 |
| Philippines | 69899.82 |
| Singapore | 213926.13 |
| Spain | 898969.60 |
| Sweden | 158031.27 |
| Switzerland | 98617.40 |
| UK | 336388.26 |
| USA | 2736064.54 |

Name: SALES, dtype: float64

10. Create a pivot table showing the total sales per country and product line.

```
import pandas as pd
import os

folder_path = '/kaggle/input/sample-sales-data'
files = os.listdir(folder_path)

csv_file = [file for file in files if file.endswith('.csv')][0]
file_path = os.path.join(folder_path, csv_file)

data = pd.read_csv(file_path, encoding='latin1')

# Create the pivot table
pivot_table = pd.pivot_table(
    data,
    values='SALES',
    index='COUNTRY',
    columns='PRODUCTLINE',
    aggfunc='sum',
    fill_value=0 # Fill missing values with 0
)

print("Pivot Table - Total Sales per Country and Product Line:")
print(pivot_table)
```

Output :-

| Pivot Table - Total Sales per Country and Product Line: | | | | | | PRODUCTLINE | Trucks and Buses | Vintage Cars |
|---|--------------|-------------|-----------|-----------|----------|-------------|------------------|--------------|
| PRODUCTLINE | Classic Cars | Motorcycles | Planes | Ships | Trains | \ COUNTRY | | |
| COUNTRY | | | | | | Australia | 77318.50 | 189555.32 |
| Australia | 193085.54 | 89968.76 | 74853.87 | 4159.76 | 1681.35 | Austria | 20472.75 | 27197.48 |
| Austria | 101459.47 | 26047.66 | 17860.44 | 9024.73 | 0.00 | Belgium | 0.00 | 41925.60 |
| Belgium | 20136.96 | 0.00 | 5624.79 | 31708.01 | 9017.26 | Canada | 51945.98 | 40512.79 |
| Canada | 61623.22 | 4177.49 | 25510.07 | 40309.01 | 0.00 | Denmark | 9588.82 | 21105.81 |
| Denmark | 157182.48 | 0.00 | 7586.45 | 38697.26 | 11476.33 | Finland | 40479.33 | 18383.00 |
| Finland | 153552.24 | 47866.72 | 34375.13 | 29808.44 | 5117.05 | France | 116982.22 | 176609.81 |
| France | 388951.20 | 226390.31 | 108155.51 | 66486.67 | 27340.80 | Germany | 10178.00 | 20935.91 |
| Germany | 148315.00 | 7497.50 | 23001.26 | 5501.00 | 5043.42 | Ireland | 3983.05 | 2234.40 |
| Ireland | 31688.82 | 4953.20 | 11784.36 | 0.00 | 3112.60 | Italy | 5914.97 | 110450.74 |
| Italy | 128576.65 | 7567.80 | 98185.65 | 17703.54 | 6274.96 | Japan | 13349.44 | 29449.82 |
| Japan | 47271.49 | 26536.41 | 49176.96 | 18860.02 | 3523.67 | Norway | 37075.64 | 43021.00 |
| Norway | 134787.37 | 51768.63 | 29500.70 | 0.00 | 11310.36 | Philippines | 0.00 | 1935.09 |
| Philippines | 53112.09 | 18061.68 | 20906.87 | 0.00 | 0.00 | Singapore | 89027.68 | 34960.46 |
| Singapore | 132890.44 | 4175.60 | 0.00 | 14155.52 | 13278.71 | Spain | 177556.78 | 229514.51 |
| Spain | 476165.15 | 74634.82 | 89985.51 | 124459.97 | 43370.18 | Sweden | 47931.27 | 33804.46 |
| Sweden | 69088.06 | 15567.25 | 8899.60 | 30915.89 | 3807.68 | Switzerland | 0.00 | 0.00 |
| Switzerland | 117713.56 | 0.00 | 0.00 | 0.00 | 0.00 | UK | 28142.99 | 123798.74 |
| UK | 159377.70 | 40802.81 | 41163.51 | 72959.17 | 12635.54 | USA | 397842.42 | 757755.90 |
| USA | 1344638.22 | 520371.70 | 328432.89 | 209688.14 | 69253.56 | | | |

11. Create a boolean mask to filter out all transactions with sales greater than \$5000

```
import pandas as pd
import numpy as np
import os

folder_path = '/kaggle/input/sample-sales-data'

print(os.listdir(folder_path))

file_path = os.path.join(folder_path, 'sales_data_sample.csv')

df = pd.read_csv(file_path, encoding='latin1')

mask = df['SALES'].values > 5000

high_sales_df = df[mask]

print(high_sales_df.head())
```

Output :-

| | | | | | | |
|---------------------------|------------------------|-----------------|-----------------|------------------|------------|-----|
| ['sales_data_sample.csv'] | | | | | | |
| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | \ |
| 4 | 10159 | 49 | 100.0 | 14 | 5205.27 | |
| 7 | 10188 | 48 | 100.0 | 1 | 5512.32 | |
| 20 | 10341 | 41 | 100.0 | 9 | 7737.93 | |
| 25 | 10417 | 66 | 100.0 | 2 | 7516.08 | |
| 26 | 10103 | 26 | 100.0 | 11 | 5404.62 | |
| | | | | | | |
| | ORDERDATE | STATUS | QTR_ID | MONTH_ID | YEAR_ID | ... |
| 4 | 10/10/2003 0:00 | Shipped | 4 | 10 | 2003 | ... |
| 7 | 11/18/2003 0:00 | Shipped | 4 | 11 | 2003 | ... |
| 20 | 11/24/2004 0:00 | Shipped | 4 | 11 | 2004 | ... |
| 25 | 5/13/2005 0:00 | Disputed | 2 | 5 | 2005 | ... |
| 26 | 1/29/2003 0:00 | Shipped | 1 | 1 | 2003 | ... |
| | | | | | | |
| | ADDRESSLINE1 | ADDRESSLINE2 | CITY | STATE | POSTALCODE | \ |
| 4 | 7734 Strong St. | NaN | San Francisco | CA | NaN | |
| 7 | Drammen 121, PR | 744 Sentrum | Bergen | NaN | N 5804 | |
| 20 | Geislweg 14 | NaN | Salzburg | NaN | 5020 | |
| 25 | C/ Moralzarzal, 86 | NaN | Madrid | NaN | 28034 | |
| 26 | Erling Skakkes gate 78 | NaN | Stavern | NaN | 4110 | |
| | | | | | | |
| | COUNTRY | TERRITORY | CONTACTLASTNAME | CONTACTFIRSTNAME | DEALSIZE | |
| 4 | USA | NaN | Brown | Julie | Medium | |
| 7 | Norway | EMEA | Oeztan | Veysel | Medium | |
| 20 | Austria | EMEA | Pipps | Georg | Large | |
| 25 | Spain | EMEA | Freyre | Diego | Large | |
| 26 | Norway | EMEA | Bergulfsen | Jonas | Medium | |

[5 rows x 25 columns]

12. Normalize the SALES column (Min-Max scaling between 0 and 1).

```
import pandas as pd
import numpy as np
import os

folder_path = '/kaggle/input/sample-sales-data'

file_path = os.path.join(folder_path, 'sales_data_sample.csv')

df = pd.read_csv(file_path, encoding='latin1')

sales_values = df['SALES'].values
sales_min = np.min(sales_values)
sales_max = np.max(sales_values)
df['normalized_sales'] = (sales_values - sales_min) / (sales_max - sales_min)

print(df[['SALES', 'normalized_sales']].head())
```

```
↩→
```

| | SALES | normalized_sales |
|---|---------|------------------|
| 0 | 2871.00 | 0.175644 |
| 1 | 2765.90 | 0.167916 |
| 2 | 3884.34 | 0.250150 |
| 3 | 3746.70 | 0.240030 |
| 4 | 5205.27 | 0.347273 |

13. Calculate the sum of all SALES values using only NumPy

```
import numpy as np
import os

folder_path = '/kaggle/input/sample-sales-data'
|
file_path = os.path.join(folder_path, 'sales_data_sample.csv')

# Load the data using NumPy's loadtxt (skip the header row)
data = np.loadtxt(file_path, delimiter=',', skiprows=1, usecols=11, dtype=float, encoding='latin1')

# Calculate the sum
total_sales = np.sum(data)

# Print the total sales
print(f"Total sales: {total_sales}")
```

→ Total sales: 284320.0

14. List the top 5 cities (from CITY) by total sales volume.

Output :-

```
import pandas as pd
import numpy as np
import os

folder_path = '/kaggle/input/sample-sales-data'
file_path = os.path.join(folder_path, 'sales_data_sample.csv')

df = pd.read_csv(file_path, encoding='latin1')

unique_cities = df['CITY'].unique()
sales_by_city = {}
for city in unique_cities:
    sales_by_city[city] = df[df['CITY'] == city]['SALES'].sum()

# Sort cities by total sales in descending order
sorted_cities = sorted(sales_by_city.items(), key=lambda item: item[1], reverse=True)

top_5_cities = sorted_cities[:5]

print("Top 5 Cities by Total Sales Volume:")
for city, sales in top_5_cities:
    print(f"{city}: ${sales:,.2f}")
```

```
Top 5 Cities by Total Sales Volume:
Madrid: $1,082,551.44
San Rafael: $654,858.06
NYC: $560,787.77
Singapore: $288,488.41
Paris: $268,944.68
```

15. Calculate the standard deviation of QUANTITYORDERED for Vintage Cars.

```
import pandas as pd
import numpy as np
import os

folder_path = '/kaggle/input/sample-sales-data'

file_path = os.path.join(folder_path, 'sales_data_sample.csv')

df = pd.read_csv(file_path, encoding='latin1')

vintage_cars_data = df[df['PRODUCTLINE'] == 'Vintage Cars']

std_dev = np.std(vintage_cars_data['QUANTITYORDERED'])

# Print the result
print(f"Standard deviation of QUANTITYORDERED for Vintage Cars: {std_dev:.2f}")
```

Standard deviation of QUANTITYORDERED for Vintage Cars: 9.79

16. Count how many sales are between 3000 and 7000

```
import pandas as pd
import numpy as np
import os

folder_path = '/kaggle/input/sample-sales-data'

file_path = os.path.join(folder_path, 'sales_data_sample.csv')

df = pd.read_csv(file_path, encoding='latin1')

sales_values = df['SALES'].values

mask = np.logical_and(sales_values >= 3000, sales_values <= 7000)

count = np.sum(mask)

print(f"Number of sales between $3000 and $7000: {count}")
```

```
➞ Number of sales between $3000 and $7000: 1384
```

17. Find the index positions where the MSRP is above \$100.

```
import pandas as pd
import numpy as np
import os

folder_path =  '/kaggle/input/sample-sales-data' 

file_path = os.path.join(folder_path, 'sales_data_sample.csv')

df = pd.read_csv(file_path, encoding='latin1')

msrp_values = df['MSRP'].values

indices = np.where(msrp_values > 100)[0]

print("Index positions where MSRP is above $100:")
print(indices)
```

```
→ Index positions where MSRP is above $100:
[ 26  27  28 ... 2663 2664 2665]
```


18. Generate a boolean array that identifies orders placed in year 2004.

```
import pandas as pd
import numpy as np
import os

folder_path = '/kaggle/input/sample-sales-data'

file_path = os.path.join(folder_path, 'sales_data_sample.csv')

df = pd.read_csv(file_path, encoding='latin1')

# Convert 'ORDERDATE' to datetime objects
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])

# Create the boolean array
orders_in_2004 = df['ORDERDATE'].dt.year == 2004

print("Boolean array for orders placed in 2004:")
print(orders_in_2004)
```

Output :-

```
→ Boolean array for orders placed in 2004:
0      False
1      False
2      False
3      False
4      False
...
2818    True
2819    False
2820    False
2821    False
2822    False
Name: ORDERDATE, Length: 2823, dtype: bool
```

19. Create a new column where you classify sales into "High" (>5000), "Medium" (1000-5000), and "Low" (<1000) sales.



```
import pandas as pd
import numpy as np
import os

folder_path = '/kaggle/input/sample-sales-data'

file_path = os.path.join(folder_path, 'sales_data_sample.csv')

df = pd.read_csv(file_path, encoding='latin1')

conditions = [
    df['SALES'] > 5000,
    (df['SALES'] >= 1000) & (df['SALES'] <= 5000),
    df['SALES'] < 1000
]

values = ['High', 'Medium', 'Low']

df['sales_category'] = np.select(conditions, values, default=None).astype(object)

print(df[['SALES', 'sales_category']].head())
```

Output :-



| | SALES | sales_category |
|---|---------|----------------|
| 0 | 2871.00 | Medium |
| 1 | 2765.90 | Medium |
| 2 | 3884.34 | Medium |
| 3 | 3746.70 | Medium |
| 4 | 5205.27 | High |

20. Create a mask to filter out all orders with DEALSIZE equal to 'Small' using NumPy arrays.

```
import pandas as pd
import numpy as np
import os

dealsize_values = df['DEALSIZE'].values

mask = np.not_equal(dealsize_values, 'Small')

filtered_df = df[mask]
print(filtered_df[['DEALSIZE', 'SALES']].head())
```

```
DEALSIZE  SALES
2   Medium 3884.34
3   Medium 3746.70
4   Medium 5205.27
5   Medium 3479.76
7   Medium 5512.32
```

Thank You!!