Practical No:01

Aim: Convert a Nested List into a NumPy Array & Perform Operations

Task:

- **Convert a nested list [[1, 2, 3], [4, 5, 6], [7, 8, 9]] into a 2D NumPy array.**
- **Compute the sum of each column using np.sum().**
- **Compute the product of each row using np.prod().**

Code:

```
import numpy as np
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
arr_2d = np.array(nested_list)
column_sums = np.sum(arr_2d, axis=0)
row_products = np.prod(arr_2d, axis=1)
print("2D Array:\n", arr_2d)
print("Sum of Each Column:", column_sums)
print("Product of Each Row:", row_products)
```

Output:

```
2D Array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Sum of Each Column: [12 15 18]
Product of Each Row: [  6 120 504]
```

Practical No:02

Aim: Convert a Tuple to an Array & Compute Logarithms

Task:

- **Convert a tuple (1, 10, 100, 1000, 10000) into a NumPy array.**
- **Compute the natural logarithm (np.log()).**
- **Compute the base-10 logarithm (np.log10()).**

Code:

```
tuple_data = (1, 10, 100, 1000, 10000)
arr = np.array(tuple_data)
log_natural = np.log(arr)
log_base10 = np.log10(arr)
print("Original Array:", arr)
print("Natural Log:", log_natural)
print("Log Base 10:", log_base10)
```

Output:

```
Original Array: [    1    10   100  1000 10000]
Natural Log: [0.         2.30258509 4.60517019 6.90775528 9.21034037]
Log Base 10: [0. 1. 2. 3. 4.]
```

Practical No:03

Aim: Compare Memory Allocation Between List & NumPy Array

Task:

- **Create a list and a NumPy array with 1 million random integers.**
- **Compare their memory usage using sys.getsizeof().**

Code:

```
import sys
import numpy as np
list_data = list(range(1_000_000))
array_data = np.arange(1_000_000)
list_memory = sys.getsizeof(list_data) + sum(sys.getsizeof(x) for x in
list_data)
array_memory = array_data.nbytes
print("Memory Usage (List):", list_memory, "bytes")
print("Memory Usage (NumPy Array):", array_memory, "bytes")
```

Output:

```
Memory Usage (List): 36000056 bytes
Memory Usage (NumPy Array): 8000000 bytes
```

Practical No:04

Aim: Create a 3D NumPy Array & Extract a Subarray

Task:

● Create a 3D array of shape (3, 3, 3) filled with random integers from 1 to 50. ● Extract the first two matrices and all columns of index 1 and 2.

Code:

```
arr_3d = np.random.randint(1, 51, size=(3, 3, 3))
subarray = arr_3d[:2, :, 1:3]
print("Original 3D Array:\n", arr_3d)
print("Extracted Subarray:\n", subarray)
```

Output:

```
Original 3D Array:
 [[[25 26 16]
  [ 8 11 35]
  [43 40  6]]

 [[ 8 43 25]
  [41 49  2]
  [11  7 31]]

 [[23  1 21]
  [28 47 43]
  [31 40 24]]]
Extracted Subarray:
 [[[26 16]
  [11 35]
  [40  6]]

 [[43 25]
  [49  2]
  [ 7 31]]]
```

Practical No:05

Aim: Generate a Structured 2D Array with Specific Data Types

Task:

- **Create a structured array with names, ages, and weights.**
- **Assign dtype=[('name', 'U10'), ('age', 'i4'), ('weight', 'f4')].**
- **Add and print three records.**

Code:

```
dtype = [('name', 'U10'), ('age', 'i4'), ('weight', 'f4')]
structured_array = np.array([('Alice', 25, 55.5), ('Bob', 30, 72.0),
('Charlie', 35, 68.2)], dtype=dtype)
print("Structured Array:\n", structured_array)
```

Output:

```
Structured Array:
 [('Alice', 25, 55.5) ('Bob', 30, 72. ) ('Charlie', 35, 68.2)]
```

Practical No:06

Aim: Pandas Series Operations

Task:

- **Create a Pandas Series of 1000 random numbers from -500 to 500.**
- **Replace all negative values with their absolute values.**
- **Compute mean, median, standard deviation, and skewness.**
- **Find the top 10 highest values and their indices.**
- **Convert the series into a cumulative sum series.**

Code:

```
import pandas as pd
import numpy as np
series = pd.Series(np.random.randint(-500, 500, 1000))
series = series.abs()
mean_val = series.mean()
median_val = series.median()
std_dev = series.std()
skewness = series.skew()
top_10 = series.nlargest(10)
cumulative_series = series.cumsum()
print(f"Mean: {mean_val}, Median: {median_val}, Std Dev: {std_dev},
Skewness: {skewness}")
print("Top 10 values:\n", top_10)
```

Output:

```
Mean: 258.182, Median: 264.0, Std Dev: 145.5383877136829, Skewness: -0.06644257881644754
Top 10 values:
 811   500
870   500
580   499
952   499
977   499
334   498
741   498
108   497
123   497
342   496
dtype: int64
```

Practical No:07

Aim: Pandas DataFrame Operations

Task:

- **Generate a DataFrame (1000 rows × 5 columns) with random numbers (1–1000).**
- **Rename columns as 'A', 'B', 'C', 'D', 'E'.**
- **Add a new column 'F' as the product of 'A' and 'B'.**
- **Compute row-wise sum, mean, and variance.**
- **Select only rows where 'C' is greater than 700 and 'D' is even.**

Code:

```
df = pd.DataFrame(np.random.randint(1, 1001, (1000, 5)), columns=['A',
'B', 'C', 'D', 'E'])
df['F'] = df['A'] * df['B']
df['Row_Sum'] = df.sum(axis=1)
df['Row_Mean'] = df.mean(axis=1)
df['Row_Variance'] = df.var(axis=1)
filtered_df = df[(df['C'] > 700) & (df['D'] % 2 == 0)]
print(filtered_df.head())
```

Output:

```
     A    B    C    D    E       F  Row_Sum       Row_Mean  Row_Variance
5   554  610  788   62  968  337940   340922   97406.285714
2.343109e+10
6   822  124  778  776  288  101928   104716   29918.857143
2.155825e+09
15  334  619  881  610  854  206746   210044   60012.571429
8.807743e+09
26  164  560  913  278  731   91840    94486   26996.000000
1.751784e+09
48  743  629  793  746  850  467347   471108  134602.285714
4.479063e+10
```

Practical No:08

Aim: Handling Large CSV Files

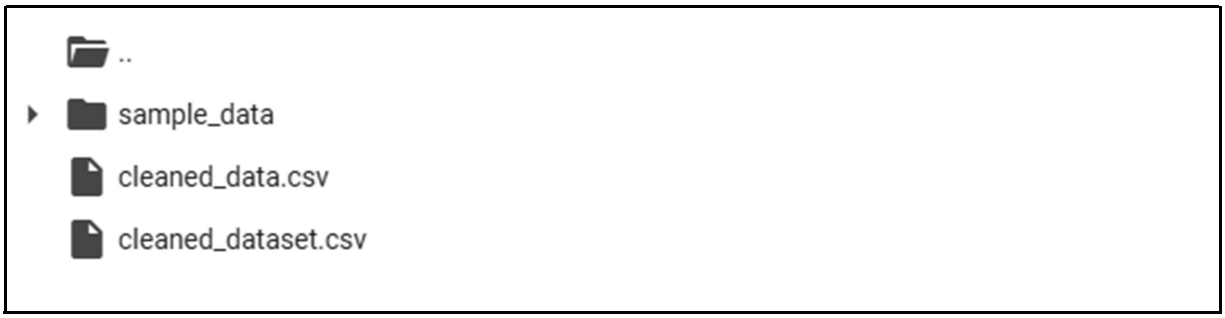
Task:

- **Read a CSV file with 1M+ rows in chunks of 50,000 rows.**
- **Drop columns with >30% missing values.**
- **Convert date columns to datetime format.**
- **Normalize numerical columns using Min-Max scaling.**
- **Save the cleaned DataFrame as a new CSV file.**

Code:

```
import pandas as pd
chunk_size = 50000
chunks = pd.read_csv("cleaned_data.csv", chunksize=chunk_size)
processed_chunks = []
for chunk in chunks:
    chunk = chunk.dropna(thresh=0.7 * len(chunk), axis=1)
    if 'date' in chunk.columns:
        chunk['date'] = pd.to_datetime(chunk['date'], errors='coerce')
    num_cols = chunk.select_dtypes(include=['number']).columns
    chunk[num_cols] = (chunk[num_cols] - chunk[num_cols].min()) /
(chunk[num_cols].max() - chunk[num_cols].min())
    processed_chunks.append(chunk)
final_df = pd.concat(processed_chunks, ignore_index=True)
final_df.to_csv("cleaned_dataset.csv", index=False)
```

Output:



Practical No:09

Aim: Handling JSON Files

Task:

- **Load a nested JSON file into a DataFrame.**
- **Flatten nested columns into separate columns.**
- **Convert Unix timestamps into readable datetime format.**
- **Extract specific fields into a subset DataFrame.**

Code:

```
import json
import pandas as pd
with open('sample_data.json', 'r') as f:
    data = json.load(f)
df = pd.json_normalize(data['records'])
if 'timestamp' in df.columns:
    df['timestamp'] = pd.to_datetime(df['timestamp'], unit='s')
subset_df = df[['id', 'user.name', 'timestamp', 'location.city']]
subset_df.rename(columns={'user.name': 'name', 'location.city': 'city'},
inplace=True)
print(subset_df.head())
```

Output:

sample_data.json

```
  id     name           timestamp      city
0   1    Alice 2024-03-01 12:00:00  New York
1   2      Bob 2024-03-02 14:30:00    London
2   3  Charlie 2024-03-03 16:45:00    Sydney
<ipython-input-20-cb6dfe1b6037>:9: 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
  subset_df.rename(columns={'user.name': 'name', 'location.city':
'city'}, inplace=True)
```

Practical No:10

Aim: Data Cleaning: Missing & Wrong Data

Task:

- **Handle missing values, wrong formats, and duplicates.**
- **Replace missing numeric values with the median.**
- **Replace missing categorical values with the mode.**
- **Standardize date formats.**
- **Remove duplicates.**

Code:

```
import pandas as pd
df = pd.read_csv("missing_values_dataset.csv")
num_cols = df.select_dtypes(include=['number']).columns
df[num_cols] = df[num_cols].fillna(df[num_cols].median())
cat_cols = df.select_dtypes(include=['object']).columns
for col in cat_cols:
    df[col] = df[col].fillna(df[col].mode()[0] if not
df[col].mode().empty else "Unknown")
df['timestamp'] = pd.to_datetime(df['timestamp'], errors='coerce')
df = df.drop_duplicates()
print(df.info())
```

Output:

missing_values_dataset.csv

```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, 0 to 4
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   id         5 non-null      int64
 1   name       5 non-null      object
 2   age        5 non-null      float64
 3   timestamp  1 non-null      datetime64[ns]
 4   category   5 non-null      object
dtypes: datetime64[ns](1), float64(1), int64(1), object(2)
memory usage: 240.0+ bytes
None
```

Practical No:11

Aim: Histogram for Customer Age Distribution

Dataset: customer_data.csv

Columns: (Customer_ID, Age, Purchase_Amount, Gender)

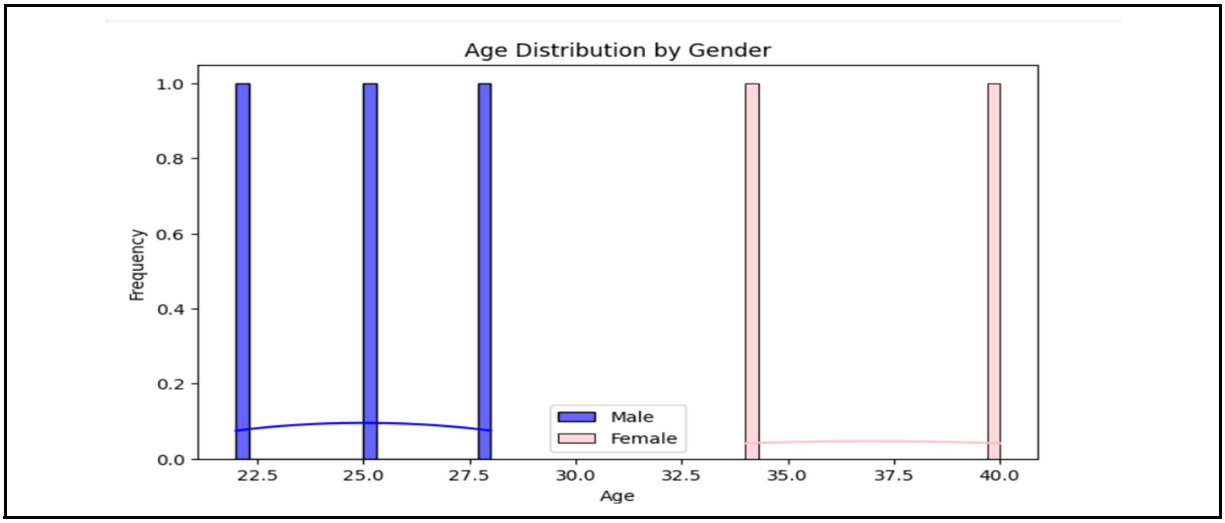
Task:

- **Plot a histogram of age distribution.**
- **Overlay histograms for male and female customers.**
- **Add a Kernel Density Estimation (KDE) curve.**

Code:

```
import seaborn as sns
df = pd.read_csv("customer_data.csv")
plt.figure(figsize=(8, 5))
sns.histplot(df[df["Gender"] == "Male"]["Age"], bins=20, kde=True,
color="blue", label="Male", alpha=0.6)
sns.histplot(df[df["Gender"] == "Female"]["Age"], bins=20, kde=True,
color="pink", label="Female", alpha=0.6)
plt.title("Age Distribution by Gender")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

Output:



Practical No:12

Aim: Scatter Plot for House Prices vs. Size

Dataset: real_estate.csv

Columns: (Property_ID, Size_sqft, Price, Bedrooms, Location)

Task:

- **Create a scatter plot of Size vs. Price.**
- **Use different colors for different cities.**
- **Fit a trendline (linear regression).**

Code:

```
from sklearn.linear_model import LinearRegression
df = pd.read_csv("real_estate.csv")
# Fit linear regression
model = LinearRegression()
X = df["Size_sqft"].values.reshape(-1, 1)
y = df["Price"]
model.fit(X, y)
trendline = model.predict(X)
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df["Size_sqft"], y=df["Price"], hue=df["Location"],
alpha=0.7)
plt.plot(df["Size_sqft"], trendline, color="red", label="Trendline")
plt.title("House Prices vs. Size")
plt.xlabel("Size (sqft)")
plt.ylabel("Price")
plt.legend()
plt.show()
```

Output:

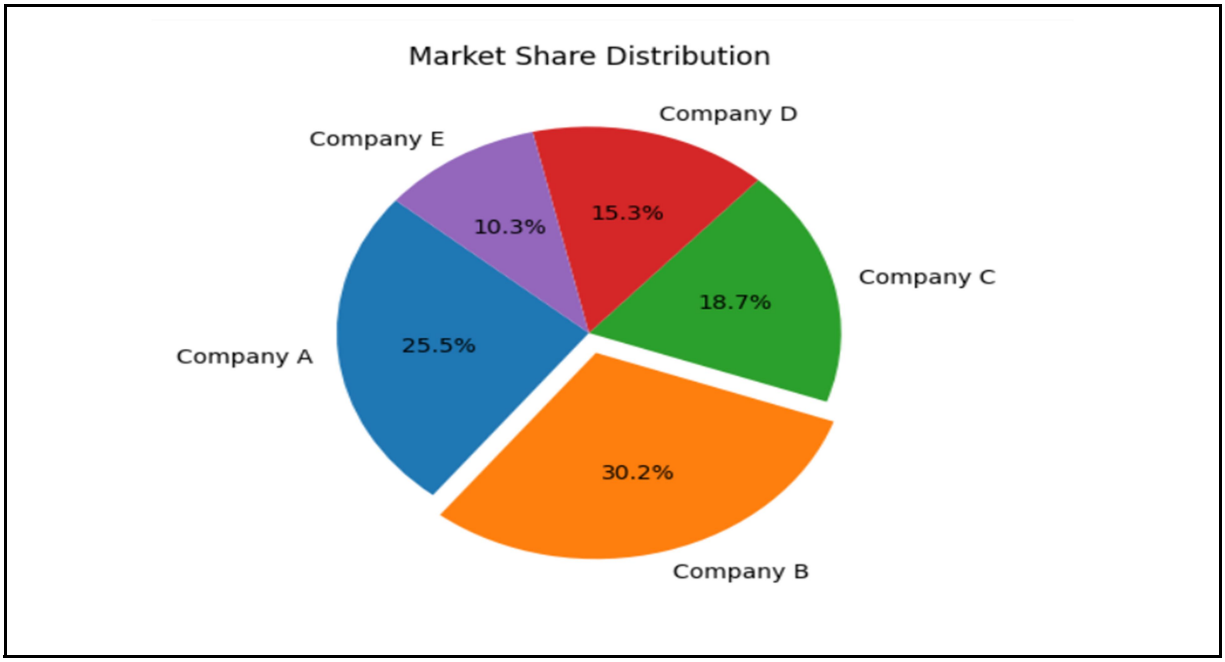
Practical No 13

Aim: Pie Chart for Market Share
Dataset: market_share.csv
Columns: (Company, Market_Share)
Task:
- **Create a pie chart of market share.**
- **Explode the largest company slice.**

Code:

```
df = pd.read_csv("market_share.csv")
explode = [0.1 if share == df["Market_Share"].max() else 0 for share
in df["Market_Share"]]
plt.pie(df["Market_Share"], labels=df["Company"], autopct='%1.1f%%',
explode=explode, startangle=140)
plt.title("Market Share Distribution")
plt.show()
```

Output:



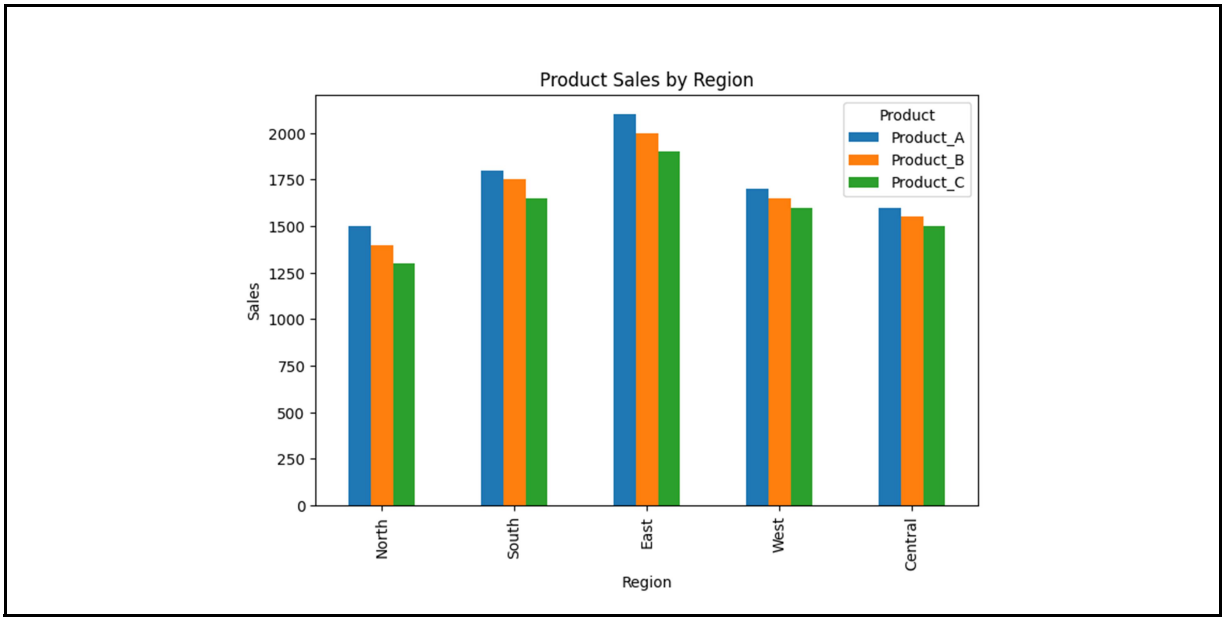
Practical No:14

Aim: Bar Plot for Product Sales in Different Regions
Dataset: sales_by_region.csv
Columns: (Region, Product_A, Product_B, Product_C)
Task:
- **Plot bar charts for product sales in each region.**
- **Use different colors for each product.**

Code:

```
df = pd.read_csv("sales_by_region.csv")
df.set_index("Region").plot(kind="bar", figsize=(8, 5))
plt.title("Product Sales by Region")
plt.xlabel("Region")
plt.ylabel("Sales")
plt.legend(title="Product")
plt.show()
```

Output:



Practical No:15

Aim: Histogram for Age Distribution of Customers

Dataset: customer_data.csv

Columns: (Customer_ID, Age, Gender)

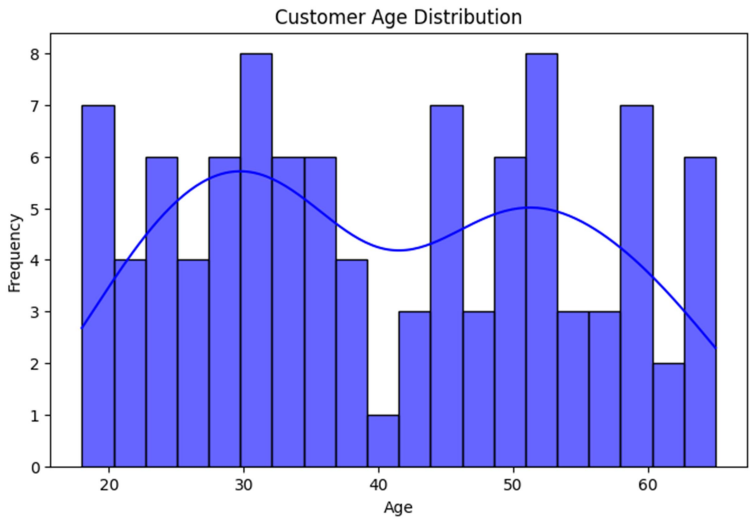
Task:

- **Plot a histogram of age distribution.**
- **Add bins and color customization.**

Code:

```
import seaborn as sns
df = pd.read_csv("customer_data.csv")
plt.figure(figsize=(8, 5))
sns.histplot(df["Age"], bins=20, kde=True, color="blue", alpha=0.6)
plt.title("Customer Age Distribution")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

Output:

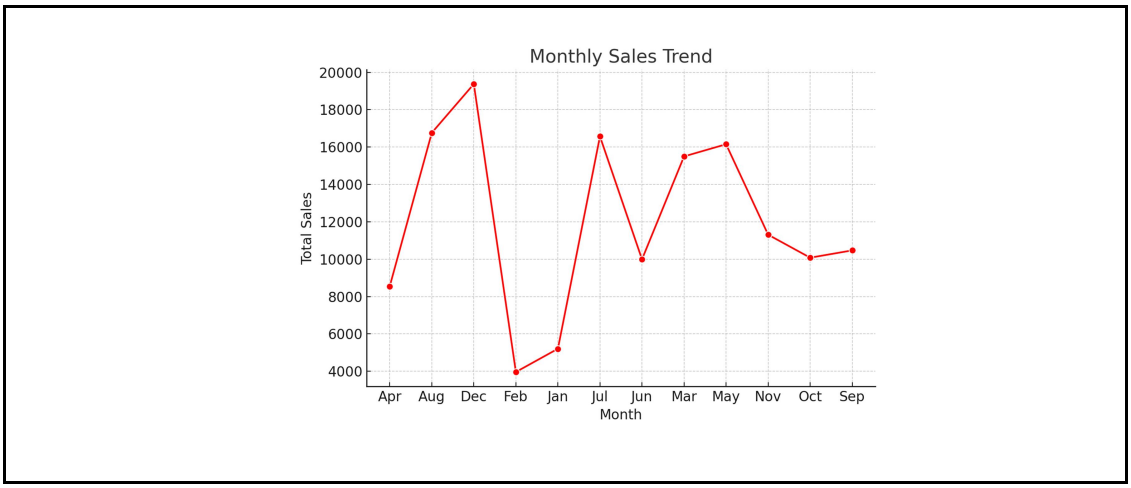


Practical No:16

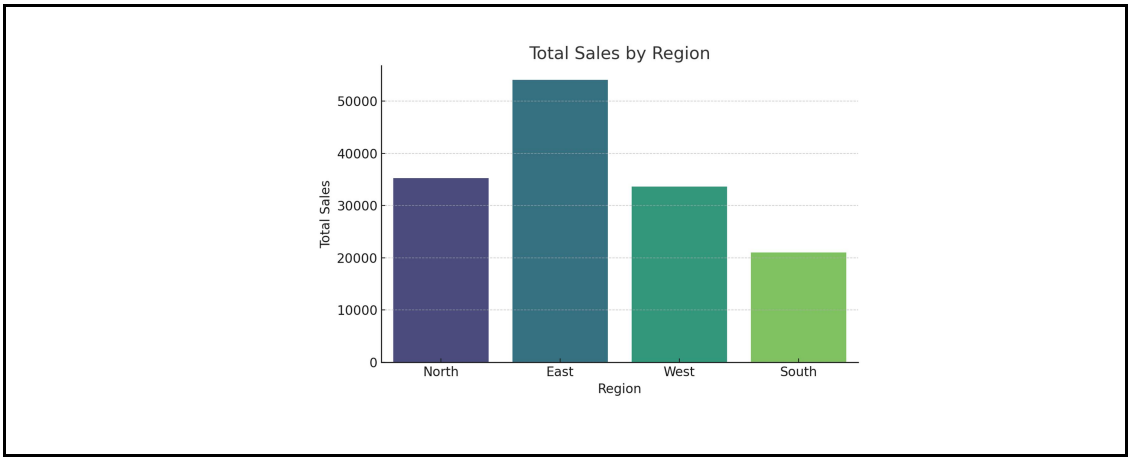
Sales Dataset

Dataset: A table with columns like Date, Product, Region, Sales, Profit, Quantity.

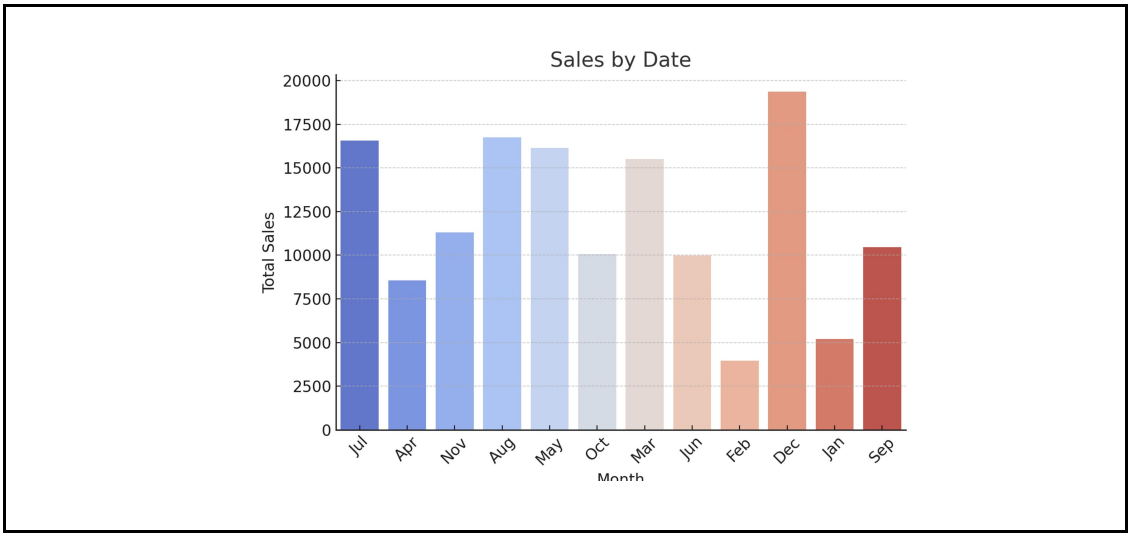
1. **Load the dataset into Power BI.**
2. **Create a relationship between the Sales table and a separate Product table.**
3. **Use DAX to calculate total sales.**



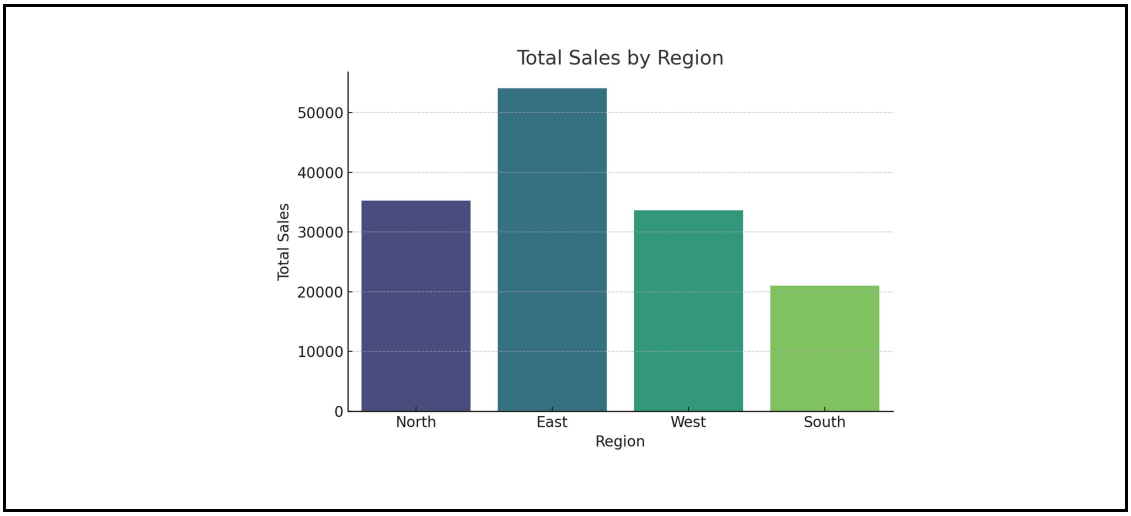
4. **Create a bar chart showing sales by region.**



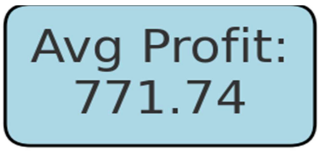
5. Add a slicer for the Date field.



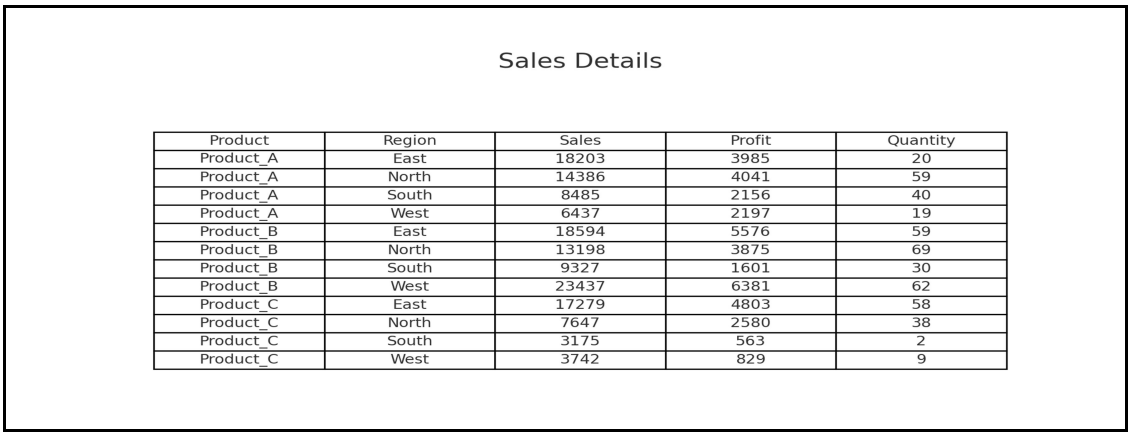
6. Use a matrix visual to display sales by product and region.



7. Create a measure to calculate average profit.



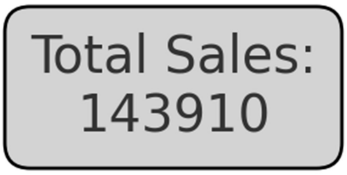
8. Add a tooltip to a visual showing additional details.



Sales Details

Product	Region	Sales	Profit	Quantity
Product_A	East	18203	3985	20
Product_A	North	14386	4041	59
Product_A	South	8485	2156	40
Product_A	West	6437	2197	19
Product_B	East	18594	5576	59
Product_B	North	13198	3875	69
Product_B	South	9327	1601	30
Product_B	West	23437	6381	62
Product_C	East	17279	4803	58
Product_C	North	7647	2580	38
Product_C	South	3175	563	2
Product_C	West	3742	829	9

9. Use a card visual to display total sales.

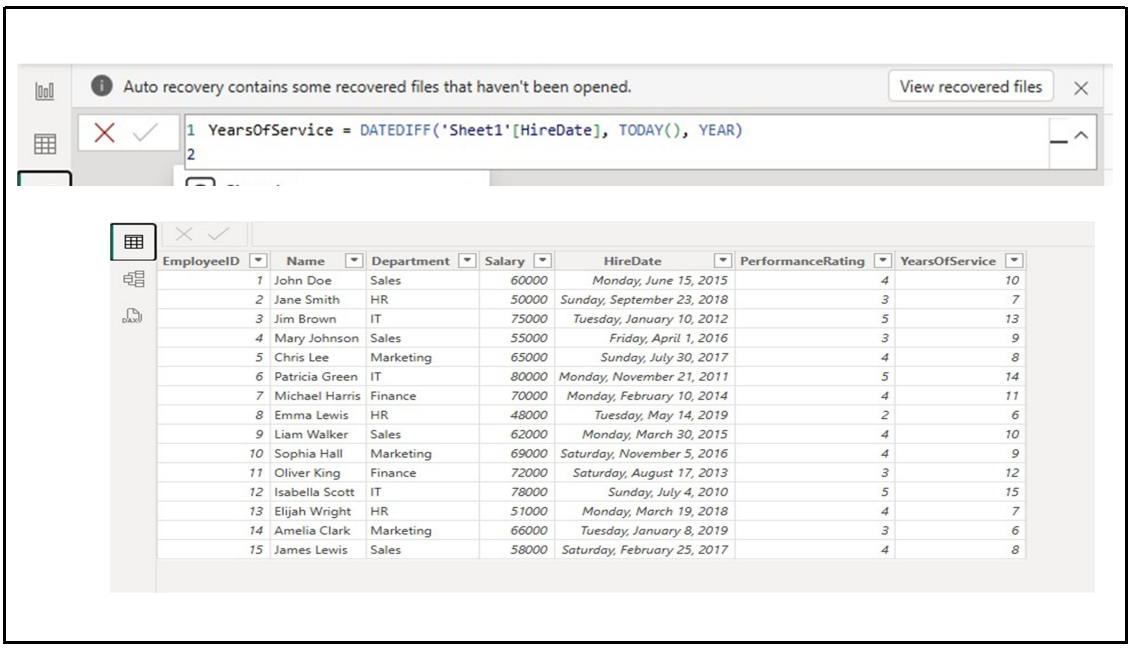


Practical No:17

HR Dataset

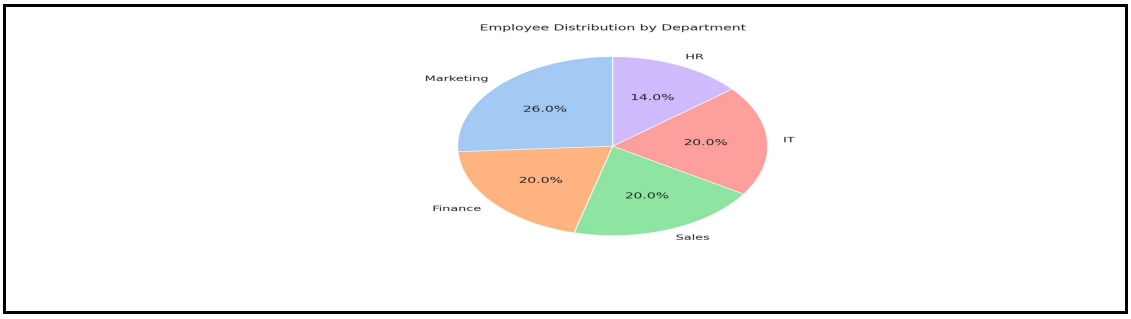
Dataset: A table with columns like EmployeeID, Name, Department, Salary, HireDate, PerformanceRating.

1. **Load the dataset into Power BI.**
2. **Create a calculated column for YearsOfService using DAX.**

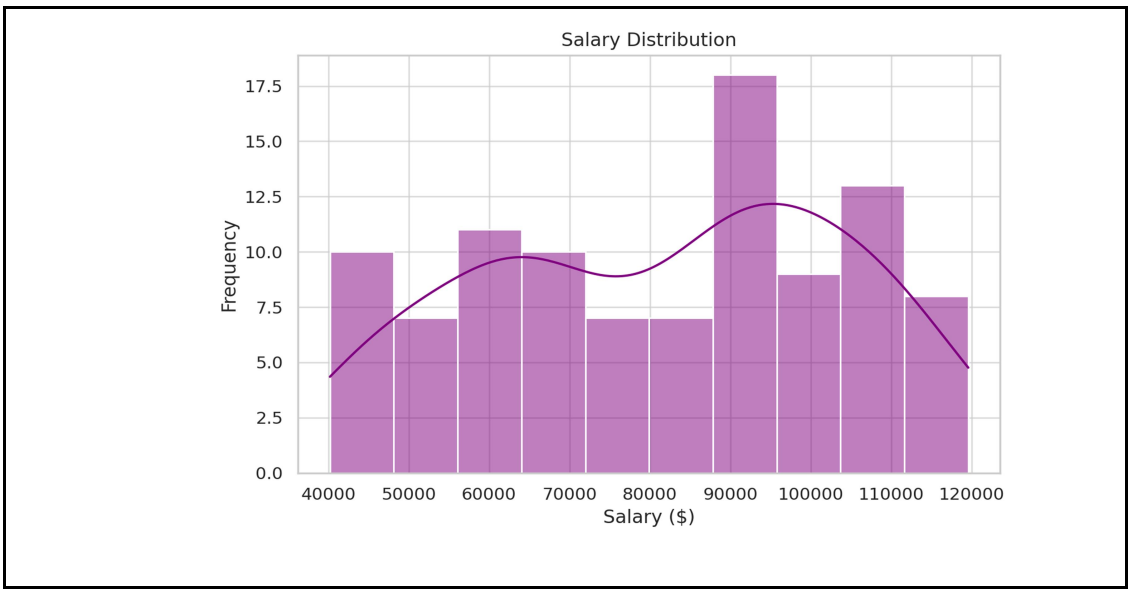


EmployeeID	Name	Department	Salary	HireDate	PerformanceRating	YearsOfService
1	John Doe	Sales	60000	Monday, June 15, 2015	4	10
2	Jane Smith	HR	50000	Sunday, September 23, 2018	3	7
3	Jim Brown	IT	75000	Tuesday, January 10, 2012	5	13
4	Mary Johnson	Sales	55000	Friday, April 1, 2016	3	9
5	Chris Lee	Marketing	65000	Sunday, July 30, 2017	4	8
6	Patricia Green	IT	80000	Monday, November 21, 2011	5	14
7	Michael Harris	Finance	70000	Monday, February 10, 2014	4	11
8	Emma Lewis	HR	48000	Tuesday, May 14, 2019	2	6
9	Liam Walker	Sales	62000	Monday, March 30, 2015	4	10
10	Sophia Hall	Marketing	69000	Saturday, November 5, 2016	4	9
11	Oliver King	Finance	72000	Saturday, August 17, 2013	3	12
12	Isabella Scott	IT	78000	Sunday, July 4, 2010	5	15
13	Elijah Wright	HR	51000	Monday, March 19, 2018	4	7
14	Amelia Clark	Marketing	66000	Tuesday, January 8, 2019	3	6
15	James Lewis	Sales	58000	Saturday, February 25, 2017	4	8

3. **Use a pie chart to show the distribution of employees by department.**

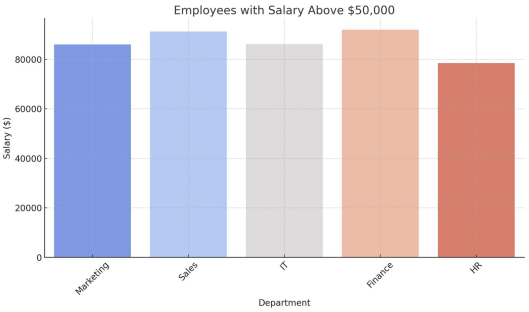


4. Create a measure to calculate the average salary.

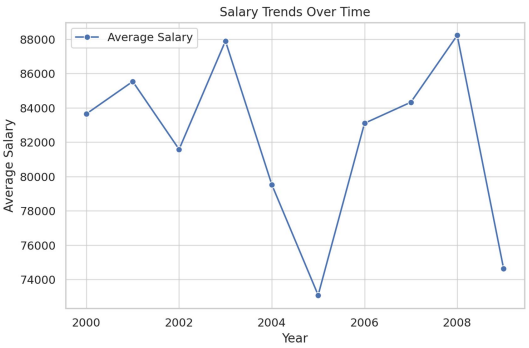


5. Add a slicer for PerformanceRating.

6. Use a table visual to display employees with salaries above $50,000.



7. Add a trend line to a visual showing salary trends over time.



8. Use a card visual to display the total number of employees.

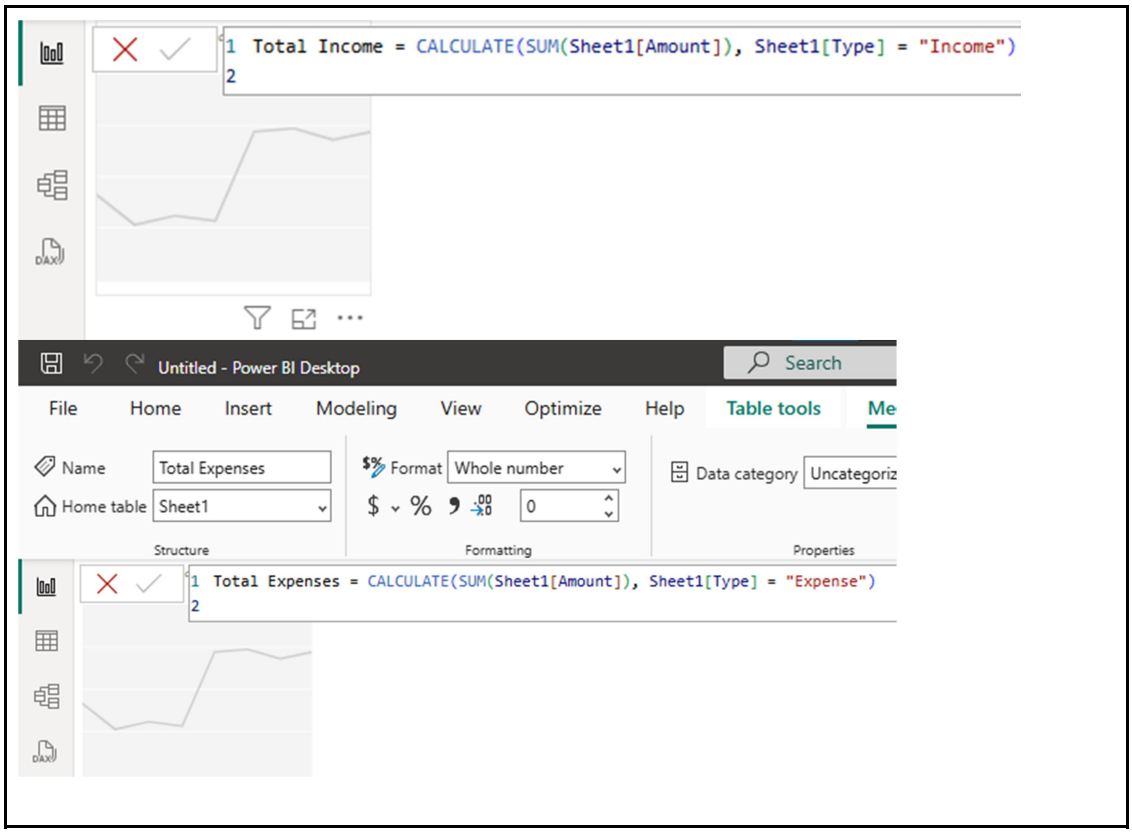
Total Employees: 100

Practical No:18

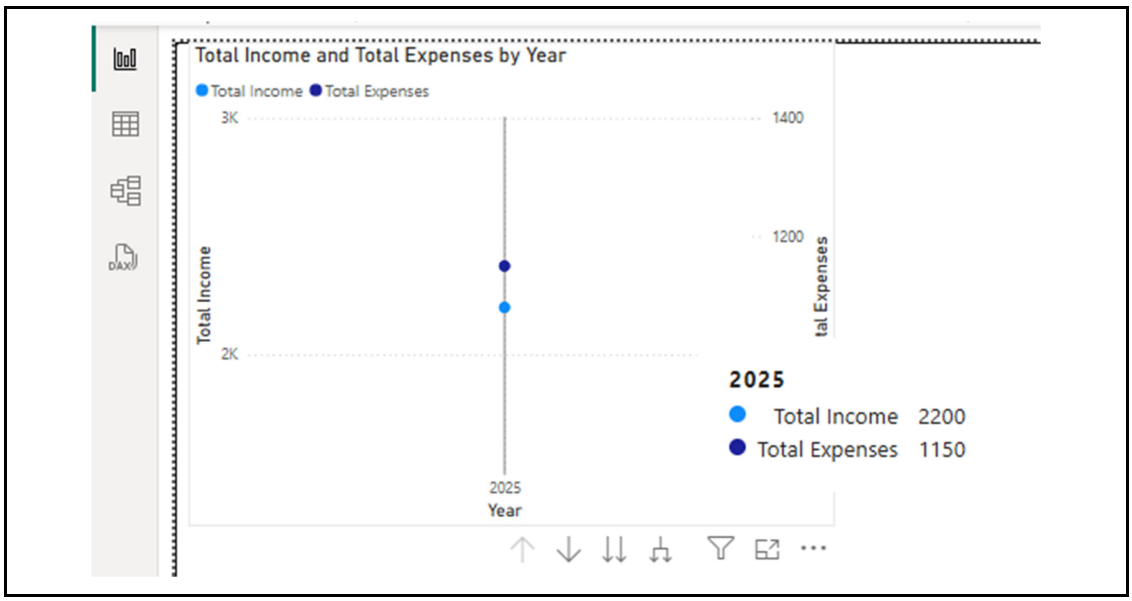
Financial Dataset

Dataset: A table with columns like AccountID, TransactionDate, Amount, Category, Type (Income/Expense).

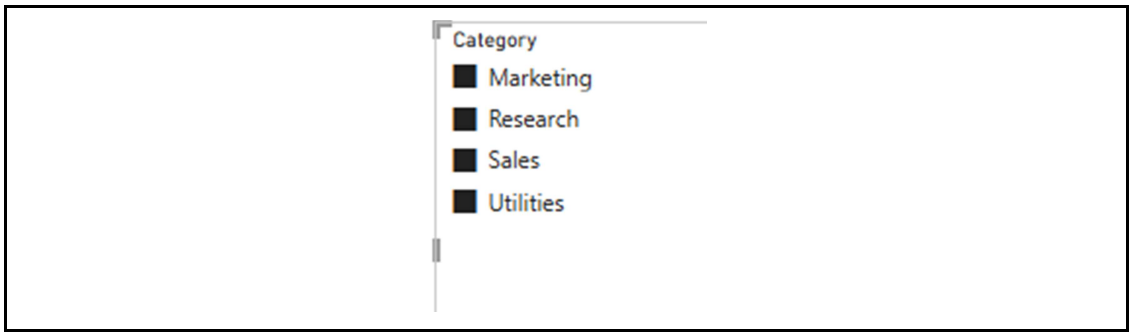
1. **Load the dataset into Power BI.**
2. **Create a measure to calculate total income & total expense.**



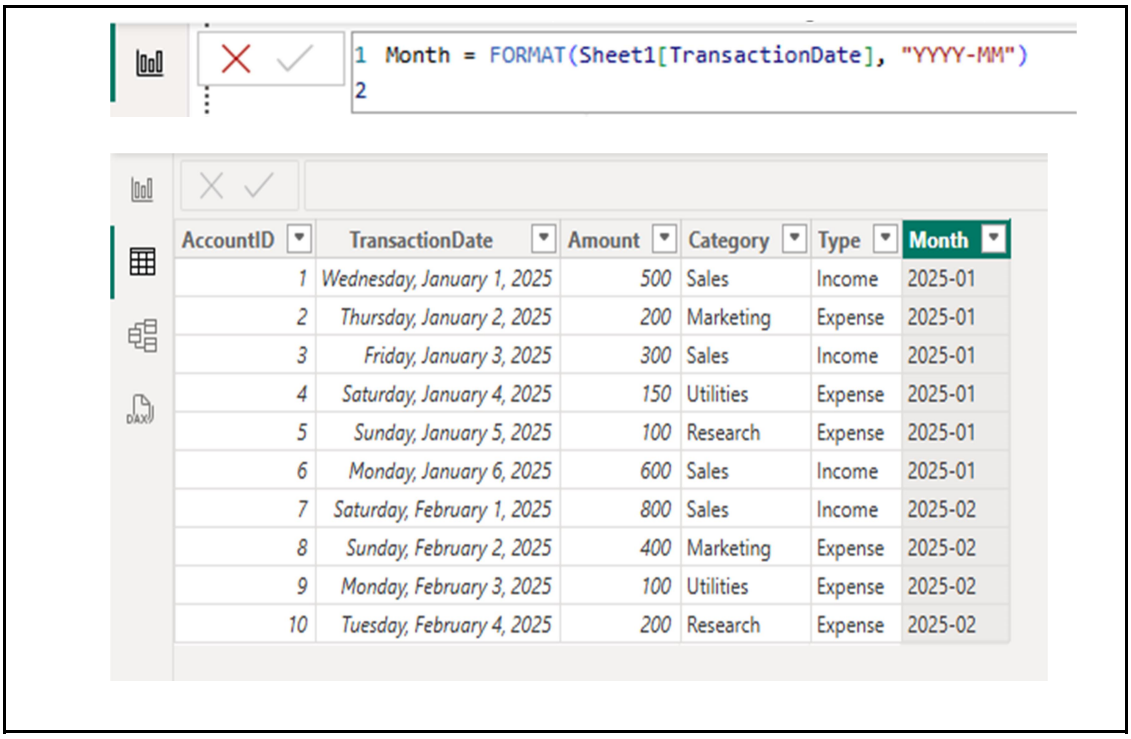
3. Use a line chart to show income vs. expenses over time.



4. Add a slicer for Category.



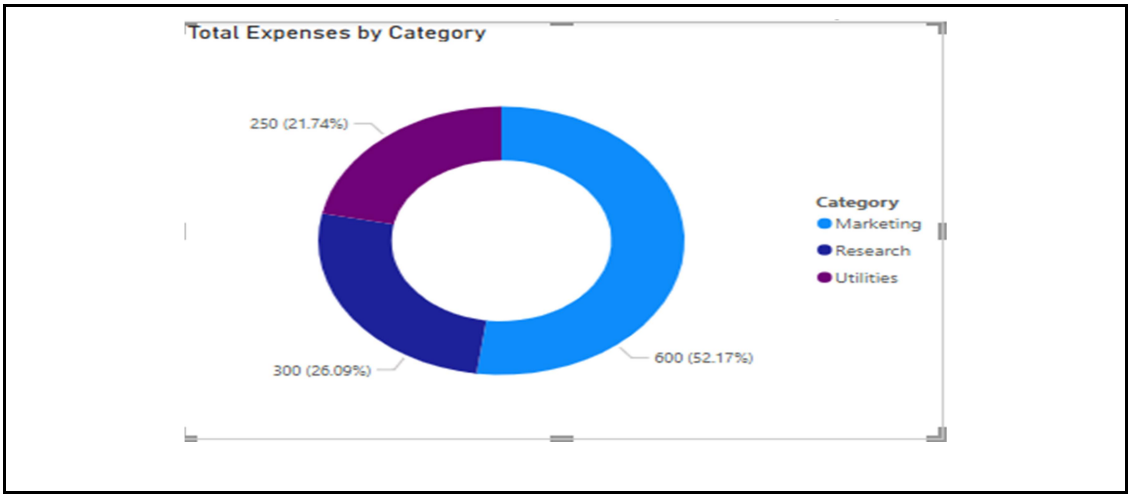
5. Create a calculated column for Month using DAX



```
1 Month = FORMAT(Sheet1[TransactionDate], "YYYY-MM")
2
```

AccountID	TransactionDate	Amount	Category	Type	Month
1	Wednesday, January 1, 2025	500	Sales	Income	2025-01
2	Thursday, January 2, 2025	200	Marketing	Expense	2025-01
3	Friday, January 3, 2025	300	Sales	Income	2025-01
4	Saturday, January 4, 2025	150	Utilities	Expense	2025-01
5	Sunday, January 5, 2025	100	Research	Expense	2025-01
6	Monday, January 6, 2025	600	Sales	Income	2025-01
7	Saturday, February 1, 2025	800	Sales	Income	2025-02
8	Sunday, February 2, 2025	400	Marketing	Expense	2025-02
9	Monday, February 3, 2025	100	Utilities	Expense	2025-02
10	Tuesday, February 4, 2025	200	Research	Expense	2025-02

6. Use a donut chart to show expenses by category.



Practical No:19

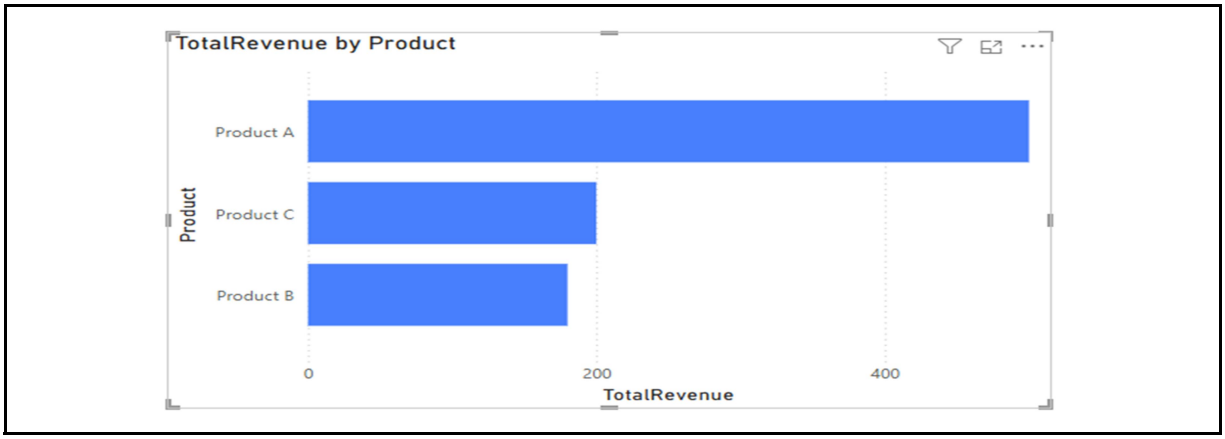
Retail Dataset

Dataset: A table with columns like OrderID, CustomerID, OrderDate, Product, Quantity, Price.

1. **Load the dataset into Power BI.**
2. **Create a measure to calculate total revenue.**

```
Total Revenue = SUMX(YourTable, YourTable[Quantity] * YourTable[Price])
```

3. **Use a bar chart to show revenue by product.**

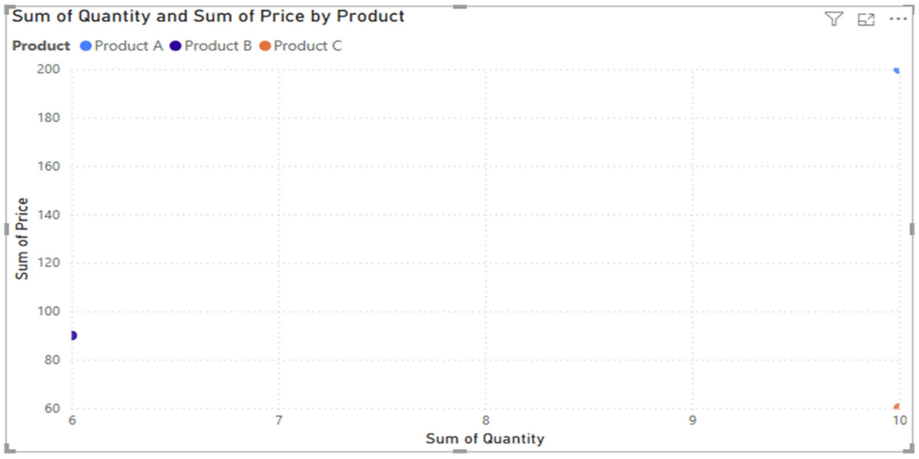


4. **Add a slicer for CustomerID.**

5. **Create a calculated column for TotalCost (Quantity * Price).**

```
TotalCost = 'Dataset'[Quantity] * 'Dataset'[Price]
```

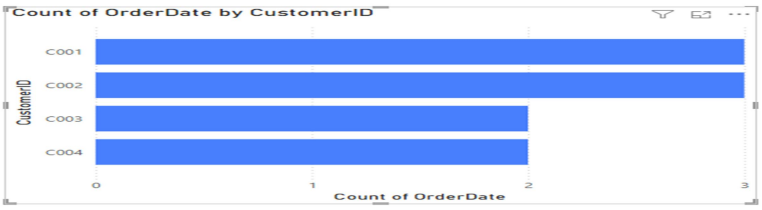
6. **Use a scatter plot to show the relationship between quantity and price.**



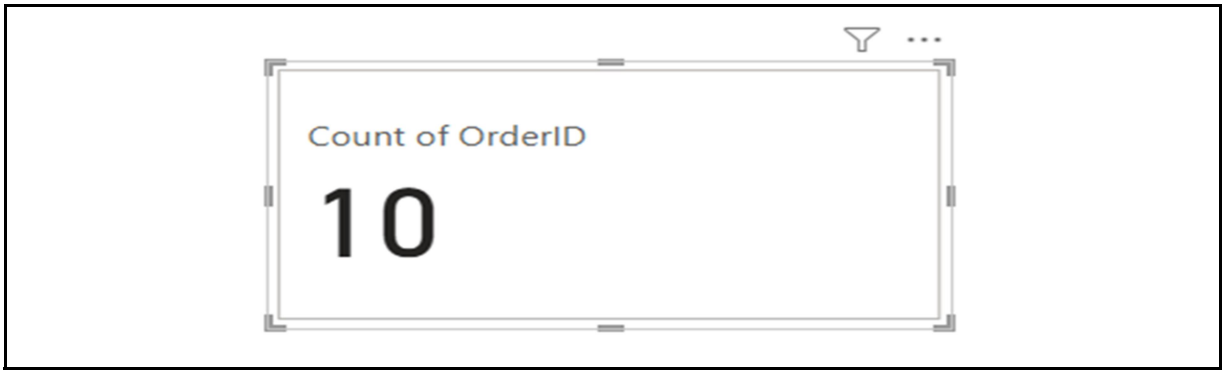
7. Create a measure to calculate the average order value.

```
AvgOrderValue = DIVIDE([TotalRevenue], DISTINCTCOUNT('Dataset'[OrderID]))
```

8. Add a tooltip to show customer details.



9. Use a card visual to display the total number of orders.



Practical No:20

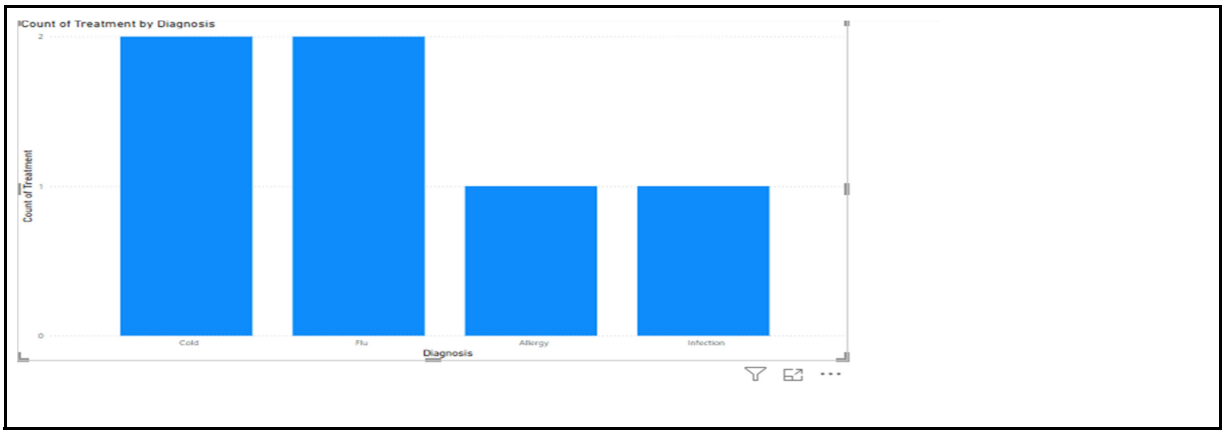
Healthcare Dataset

Dataset: A table with columns like PatientID, VisitDate, Diagnosis, Treatment, Cost, Doctor.

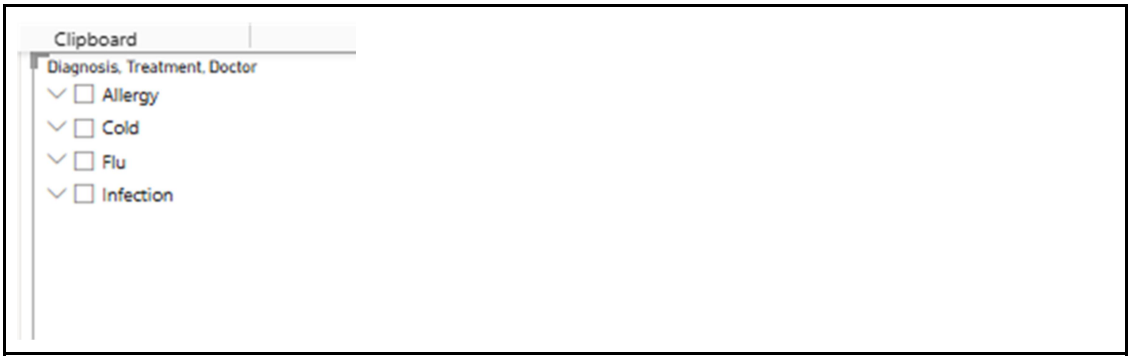
1. **Load the dataset into Power BI.**
2. **Create a measure to calculate total treatment costs.**

```
TotalTreatmentCost = SUM(HealthcareDataset[Cost])
```

3. **Use a bar chart to show costs by diagnosis.**



4. **Add a slicer for Doctor.**



5. **Create a calculated column for VisitMonth using DAX.**

```
VisitMonth = FORMAT(HealthcareDataset[VisitDate], "YYYY-MM")
```

6. Use a pie chart to show the distribution of treatments.