In [5]:

```
import pandas as pd

#1.
# Question 1:A retail store wants to identify customers who make frequent purchases. Given the dataset below, write a Python program to:
# Group customers by their IDs.


# Calculate the total purchase amount per customer.


# Identify the top 3 customers with the highest purchase amounts.


# Dataset:
data = {'Customer_ID': [101, 102, 103, 101, 104, 102, 101, 105, 102, 103],
        'Purchase_Amount': [200, 150, 180, 220, 300, 200, 100, 400, 250, 300]}

df = pd.DataFrame(data)

# Grouping by Customer_ID and calculating total purchases
total_purchases = df.groupby('Customer_ID')['Purchase_Amount'].sum().reset_index()

# Finding the top 3 frequent customers
top_customers = total_purchases.sort_values(by='Purchase_Amount', ascending=False).head(3)

print("Total Purchases per Customer:")
print(total_purchases)
print("\nTop 3 Frequent Customers:")
print(top_customers)

# Expected Output:
# Total Purchases per Customer:
#    Customer_ID  Purchase_Amount
# 0          101              520
# 1          102              600
# 2          103              480
# 3          104              300
# 4          105              400

# Top 3 Frequent Customers:
#    Customer_ID  Purchase_Amount
# 1          102              600
# 0          101              520
# 2          103              480
```

```
Total Purchases per Customer:
   Customer_ID  Purchase_Amount
0          101              520
1          102              600
2          103              480
3          104              300
4          105              400

Top 3 Frequent Customers:
   Customer_ID  Purchase_Amount
1          102              600
0          101              520
2          103              480
```

In [6]:

```
#2. Predicting House Prices with Linear Regression
# A real estate company wants to predict house prices based on square footage. Write a Python program to:
# Train a Linear Regression model.


# Predict house prices for given test data.
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Creating a DataFrame
data = {'Square_Feet': [1500, 2000, 2500, 3000, 3500],
        'Price': [300000, 400000, 500000, 600000, 700000]}

df = pd.DataFrame(data)

# Splitting into features and target
X = df[['Square_Feet']]
y = df['Price']

# Training a Linear Regression Model
model = LinearRegression()
model.fit(X, y)

# Predicting for test data
test_data = [[1800], [2800]]
predictions = model.predict(test_data)

print("Predicted Prices:")
print(predictions)

# Test Data: [[1800], [2800]]
# Expected Output:
# Predicted Prices:
# [360000. 560000.]
```

```
Predicted Prices:
[360000. 560000.]
```

```
/home/deehub/anaconda3/envs/AI/lib/python3.13/site-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [7]:

```
#3:
# Identifying Frequent Labels in a Dataset
# A company wants to identify the top 3 most common categories in a dataset. Given the dataset below, write a Python program to:
# Group the data by Category.


# Count the total occurrences of each category.


# Identify the top 3 most frequent categories.


import pandas as pd

# Creating a DataFrame
data = {'Category': ['A', 'B', 'C', 'A', 'D', 'B', 'A', 'E', 'B', 'C', 'C', 'A'],
        'Value': [10, 15, 20, 30, 25, 18, 22, 40, 35, 50, 45, 15]}

df = pd.DataFrame(data)

# Counting occurrences per category
category_counts = df.groupby('Category').size().reset_index(name='Count')

# Finding the top 3 most frequent categories
top_categories = category_counts.sort_values(by='Count', ascending=False).head(3)

# Printing results
print("Total Occurrences per Category:")
print(category_counts)
print("\nTop 3 Frequent Categories:")
print(top_categories)

# Expected Output:
```

```
# Total Occurrences per Category:
#   Category  Count
# 0        A      4
# 1        B      3
# 2        C      3
# 3        D      1
# 4        E      1

# Top 3 Frequent Categories:
#   Category  Count
# 0        A      4
# 1        B      3
# 2        C      3
```

```
Total Occurrences per Category:
  Category  Count
0        A      4
1        B      3
2        C      3
3        D      1
4        E      1

Top 3 Frequent Categories:
  Category  Count
0        A      4
1        B      3
2        C      3
```

In [8]:

```
# #4.
# Predicting Missing Values Using Mean Imputation
# A dataset contains missing values in the Age column. Write a Python program to:
# Replace missing values with the mean of the column.


# Display the updated DataFrame.


import pandas as pd

# Creating a DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'Age': [25, 30, None, 35, None]}

df = pd.DataFrame(data)

# Display original data
print("Original Data:")
print(df)

# Replacing missing values with mean
df['Age'].fillna(df['Age'].mean(), inplace=True)

# Display updated data
print("\nData after Imputation:")
print(df)

# Expected Output:

# Original Data:
#       Name   Age
# 0  Alice  25.0
# 1    Bob  30.0
# 2 Charlie   NaN
# 3  David  35.0
# 4    Eve   NaN

# Data after Imputation:
```

```
#      Name   Age
# 0  Alice  25.0
# 1    Bob  30.0
# 2 Charlie  30.0
# 3  David  35.0
# 4    Eve  30.0
```

```
Original Data:
      Name   Age
0    Alice  25.0
1      Bob  30.0
2  Charlie   NaN
3    David  35.0
4      Eve   NaN

Data after Imputation:
      Name   Age
0    Alice  25.0
1      Bob  30.0
2  Charlie  30.0
3    David  35.0
4      Eve  30.0
```

```
/tmp/ipykernel_1167947/3813205300.py:23: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Age'].fillna(df['Age'].mean(), inplace=True)
```

In [10]:

```
# # 5:
# Implementing a Simple Linear Regression Model
# You are given a dataset with Experience (years) and Salary ($). Write a Python program to:
# Train a Linear Regression model.


# Predict the salary for an individual with 6 years of experience.


# Dataset:
# import pandas as pd

from sklearn.linear_model import LinearRegression

# Creating a DataFrame
data = {'Experience': [1, 2, 3, 4, 5],
        'Salary': [30000, 35000, 40000, 45000, 50000]}
df = pd.DataFrame(data)

# Splitting into X and y
X = df[['Experience']]
y = df['Salary']

# Training a Linear Regression Model
model = LinearRegression()
model.fit(X, y)

# Predicting salary for 6 years of experience
predicted_salary = model.predict([[6]])
print(f"Predicted Salary for 6 years of experience: ${predicted_salary[0]:.2f}")


# data = {'word_count': [100, 150, 200, 120, 180, 220],
#         'is_spam': ['ham', 'spam', 'spam', 'ham', 'spam', 'spam']}
# df = pd.DataFrame(data)
# Expected Output (Example):
# Prediction for email with 200 words: Spam
```

```
Predicted Salary for 6 years of experience: $55000.00
```

```
/home/deehub/anaconda3/envs/AI/lib/python3.13/site-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [ ]: