#1.Identify Customers with High Purchase Frequency import pandas as pd 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) Task: Group customers by Customer_ID and count the number of purchases per customer. Identify the top 3 customers with the highest number of purchases. Expected Output: Customer Purchase Frequency: Customer_ID Purchase_Count 0 101 3 1 102 3 2 103 2 3 104 1 4 105 1 Top 3 Frequent Customers: Customer_ID Purchase_Count 0 101 3 1 102 3 2 103 2

In [1]:

```
import pandas as pd

# Data
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)

# Group customers by ID and count purchases
purchase_freq = df.groupby('Customer_ID').size().reset_index(name='Purchase_Count')

print("Customer Purchase Frequency:")
print(purchase_freq)

# Sort by count and get top 3
top_customers = purchase_freq.sort_values('Purchase_Count', ascending=False).head(3)

print("\nTop 3 Frequent Customers:")
print(top_customers)
```

```
Customer Purchase Frequency:
   Customer_ID  Purchase_Count
0          101               3
1          102               3
2          103               2
3          104               1
4          105               1

Top 3 Frequent Customers:
   Customer_ID  Purchase_Count
0          101               3
1          102               3
2          103               2
```

#2. Find Students with the Highest Average Exam Scores import pandas as pd data = {'Student_ID': [201, 202, 203, 201, 204, 202, 201, 205, 202, 203], 'Exam_Score': [85, 90, 78, 88, 92, 87, 80, 95, 89, 84]} df = pd.DataFrame(data) Task: Calculate the average exam score per student. Display the top 3 students with the highest average scores. Expected Output: Average Exam Scores per Student: Student_ID Avg_Score 0 201 84.33 1 202 88.67 2 203 81.00 3 204 92.00 4 205 95.00 Top 3 Students: Student_ID Avg_Score 0 205 95.00 1 204 92.00 2 202 88.67

In [2]:

```
import pandas as pd

# Step 1: Create the data
data = {'Student_ID': [201, 202, 203, 201, 204, 202, 201, 205, 202, 203],
        'Exam_Score': [85, 90, 78, 88, 92, 87, 80, 95, 89, 84]}

df = pd.DataFrame(data)

# Step 2: Calculate average score for each student
avg_scores = df.groupby('Student_ID')['Exam_Score'].mean().reset_index(name='Avg_Score')

print("Average Exam Scores per Student:")
print(avg_scores)
```

```
# Step 3: Get top 3 students by average score
top_students = avg_scores.sort_values('Avg_Score', ascending=False).head(3)

print("\nTop 3 Students:")
print(top_students)
```

```
Average Exam Scores per Student:
   Student_ID  Avg_Score
0         201  84.333333
1         202  88.666667
2         203  81.000000
3         204  92.000000
4         205  95.000000

Top 3 Students:
   Student_ID  Avg_Score
4         205  95.000000
3         204  92.000000
1         202  88.666667
```

3. Predict House Prices Using Linear Regression Dataset: import pandas as pd data = {'Size_sqft': [1500, 1800, 2400, 3000, 3500, 4000], 'Price': [300000, 350000, 450000, 550000, 650000, 700000]} df = pd.DataFrame(data) Task: Train a Linear Regression model to predict house prices based on Size_sqft. Predict the price of a house of size 2800 sqft. Expected Output: Predicted Price for 2800 sqft: $516491

In [3]:

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

# Step 1: Create dataset
data = {'Size_sqft': [1500, 1800, 2400, 3000, 3500, 4000],
        'Price': [300000, 350000, 450000, 550000, 650000, 700000]}

df = pd.DataFrame(data)

# Step 2: Separate features (X) and target (y)
X = df[['Size_sqft']]   # Features must be in 2D format for sklearn
y = df['Price']         # Target variable

# Step 3: Create and train Linear Regression model
model = LinearRegression()
model.fit(X, y)

# Step 4: Predict price for a house of size 2800 sqft
predicted_price = model.predict([[2800]])[0]

# Step 5: Print the result
print(f"Predicted Price for 2800 sqft: ${predicted_price:.0f}")
```

```
Predicted Price for 2800 sqft: $516492
C:\Anaconda3\envs\dineshML\Lib\site-packages\sklearn\base.py:465: UserWarning: X does no
t have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

4. Identify the Most Commonly Purchased Products dataset: import pandas as pd data = {'Product_ID': ['P101', 'P102', 'P103', 'P101', 'P104', 'P102', 'P101', 'P105', 'P102', 'P103'], 'Purchase_Count': [5, 3, 4, 2, 1, 6, 7, 3, 2, 5]} df = pd.DataFrame(data) Task: Group products by Product_ID and sum their Purchase_Count. Identify the top 3 most purchased products. Expected Output: Product Purchase Counts: Product_ID Total_Purchases 0 P101 14 1 P102 11 2 P103 9 3 P104 1 4 P105 3 Top 3 Purchased Products: Product_ID Total_Purchases 0 P101 14 1 P102 11 2 P103 9

In [4]:

```python
import pandas as pd

# Step 1: Create dataset
data = {'Product_ID': ['P101', 'P102', 'P103', 'P101', 'P104', 'P102', 'P101', 'P105', '
        'Purchase_Count': [5, 3, 4, 2, 1, 6, 7, 3, 2, 5]}

df = pd.DataFrame(data)

# Step 2: Group by Product_ID and sum Purchase_Count
product_purchases = df.groupby('Product_ID')['Purchase_Count'].sum().reset_index(name='T

print("Product Purchase Counts:")
print(product_purchases)

# Step 3: Sort by total purchases and get top 3
top_products = product_purchases.sort_values('Total_Purchases', ascending=False).head(3)

print("\nTop 3 Purchased Products:")
print(top_products)
```

```
Product Purchase Counts:
  Product_ID  Total_Purchases
0       P101               14
1       P102               11
2       P103                9
3       P104                1
4       P105                3

Top 3 Purchased Products:
  Product_ID  Total_Purchases
0       P101               14
1       P102               11
2       P103                9
```

5. Cluster Customers Based on Their Purchase Amounts Dataset: import pandas as pd data = {'Customer_ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'Total_Spend': [500, 1500, 2000, 2500, 3000, 3500, 4000, 1000, 1200, 2700]} df = pd.DataFrame(data) Task: Apply K-Means Clustering to segment customers into 3 clusters. Print the cluster labels for each customer. Expected Output: Customer Clusters: Customer_ID Total_Spend Cluster_Label 0 1 500 1 1 2 1500 1 2 3 2000 2 3 4 2500 2 4 5 3000 2 5 6 3500 0 6 7 4000 0 7 8 1000 1 8 9 1200 1 9 10 2700 2

In [5]:

```python
import pandas as pd
from sklearn.cluster import KMeans

# Step 1: Create dataset
data = {'Customer_ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'Total_Spend': [500, 1500, 2000, 2500, 3000, 3500, 4000, 1000, 1200, 2700]}

df = pd.DataFrame(data)

# Step 2: Prepare the data for clustering
X = df[['Total_Spend']]   # Only use Total_Spend for clustering

# Step 3: Apply K-Means clustering with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster_Label'] = kmeans.fit_predict(X)

# Step 4: Print results
```

```python
print("Customer Clusters:")
print(df)
```

```
C:\Anaconda3\envs\dineshML\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
f `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
Customer Clusters:
   Customer_ID  Total_Spend  Cluster_Label
0            1          500              0
1            2         1500              0
2            3         2000              2
3            4         2500              2
4            5         3000              2
5            6         3500              1
6            7         4000              1
7            8         1000              0
8            9         1200              0
9           10         2700              2
```

In [7]:

```python
#6.sln:
import pandas as pd
from sklearn.cluster import KMeans

# Creating the DataFrame
data = {'Customer_ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'Total_Spend': [500, 1500, 2000, 2500, 3000, 3500, 4000, 1000, 1200, 2700]}

df = pd.DataFrame(data)

# Applying K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
df['Cluster_Label'] = kmeans.fit_predict(df[['Total_Spend']])

print("Customer Clusters:")
print(df)
```

```
Customer Clusters:
   Customer_ID  Total_Spend  Cluster_Label
0            1          500              0
1            2         1500              0
2            3         2000              2
3            4         2500              2
4            5         3000              2
5            6         3500              1
6            7         4000              1
7            8         1000              0
8            9         1200              0
9           10         2700              2
```

In [ ]:

In [ ]:

In [ ]: