Week-6 Case Assessment

1. **Marketing Analytics Tasks**

Excel:

1. PivotTable to segment customers by **Product Category** and **age group**

Count of Customer ID	Column Labels						
Row Labels	18-25	26-35	36-45	46-60	60+	#N/A	Grand Total
Books	11419	14060	14207	21000	14207	19	74912
Clothing	11614	14132	14173	21036	14073	24	75052
Electronics	7734	9283	9420	14192	9545	11	50185
Home	7582	9431	9386	14062	9377	13	49851
Grand Total	**38349**	**46906**	**47186**	**70290**	**47202**	**67**	**250000**

2. . Calculate the **CTR, Conversion Rate, CPA, and ROI**

Metric	Value		CTR	Conversion Rate	CPA	ROI
Impressions	50,000		4	7.5	66.66666667	120%
Clicks	2,000					
Conversions	150					
Cost (₹)	10,000					
Revenue (₹)	22,000					

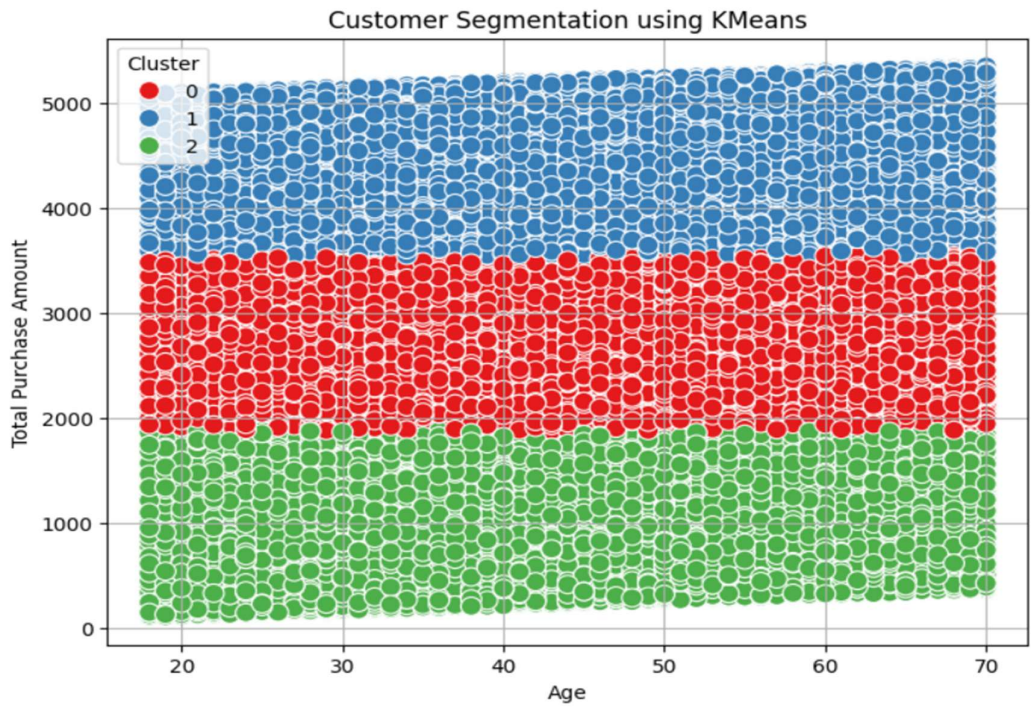
Python:

1. KMeans clustering to segment customers based on Age and Annual Spend.

```
X = df[['Age', 'Total Purchase Amount']]
```

```
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X)
```

```
plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x='Age', y='Total Purchase Amount', hue='Cluster', palette='Set1', s=100)
plt.title('Customer Segmentation using KMeans')
plt.xlabel('Age')
plt.ylabel('Total Purchase Amount')
plt.grid(True)
plt.show()
```



2. Use mlxtend to perform Market Basket Analysis on transaction data.

```
frequent_itemsets = apriori(df_encoded, min_support=0.1, use_colnames=True)
```

```
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.6)
```

```
frequent_itemsets['itemsets'] = frequent_itemsets['itemsets'].apply(lambda x: ', '.join(list(x)))
rules['antecedents'] = rules['antecedents'].apply(lambda x: ', '.join(list(x)))
rules['consequents'] = rules['consequents'].apply(lambda x: ', '.join(list(x)))
```

```
print("🔹 Frequent Itemsets:")
print(frequent_itemsets[['itemsets', 'support']])
```

```
🔹 Frequent Itemsets:
                              itemsets   support
0                                Books  0.780424
1                             Clothing  0.781431
2                          Electronics  0.639522
3                                 Home  0.636322
4                       Clothing, Books  0.605540
5                    Electronics, Books  0.497252
6                          Home, Books  0.493789
7                 Electronics, Clothing  0.497453
8                       Home, Clothing  0.493628
9                     Electronics, Home  0.404284
10         Electronics, Clothing, Books  0.386528
11                Home, Clothing, Books  0.382965
12             Electronics, Home, Books  0.314960
13          Electronics, Home, Clothing  0.314779
14  Electronics, Home, Clothing, Books  0.244982
```

```
print("\n Association Rules:")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
 Association Rules:
                  antecedents      consequents   support  confidence  \
0                    Clothing            Books  0.605540    0.774912
1                       Books         Clothing  0.605540    0.775912
2                 Electronics            Books  0.497252    0.777536
3                       Books      Electronics  0.497252    0.637156
4                        Home            Books  0.493789    0.776006
5                       Books             Home  0.493789    0.632719
6                 Electronics         Clothing  0.497453    0.777851
7                    Clothing      Electronics  0.497453    0.636593
8                        Home         Clothing  0.493628    0.775753
9                    Clothing             Home  0.493628    0.631698
10                Electronics             Home  0.404284    0.632165
11                       Home      Electronics  0.404284    0.635345
12       Electronics, Clothing            Books  0.386528    0.777013
13          Electronics, Books         Clothing  0.386528    0.777328
14             Clothing, Books      Electronics  0.386528    0.638319
15                Electronics  Clothing, Books  0.386528    0.604401
16              Home, Clothing            Books  0.382965    0.775816
17                 Home, Books         Clothing  0.382965    0.775563
18             Clothing, Books             Home  0.382965    0.632435
19                       Home  Clothing, Books  0.382965    0.601841
20           Electronics, Home            Books  0.314960    0.779056
21           Electronics, Books             Home  0.314960    0.633401
...
27  0.995380
28  0.996041
29  1.000277
30  1.000702
```

SQL

1. Number of customers based on Payment Method

Result Grid | Filter Rows:

PaymentMethod	NumberOfCustomers
Cash	31391
Credit Card	43298
Crypto	19567
PayPal	38811

2. Conversion rates by campaign ID.

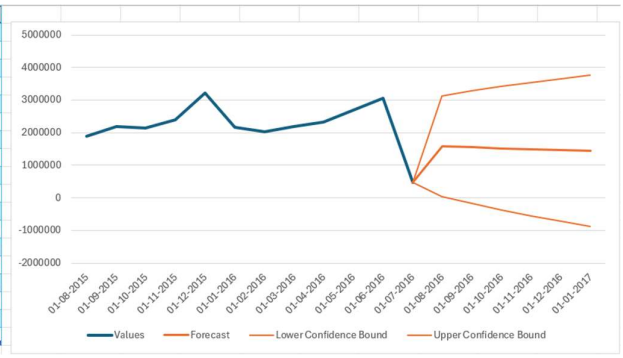
CampaignID	Total_Interactions	Conversions	ConversionRate
1	83527	79065	0.9466
2	83205	78634	0.9451
3	83268	78851	0.9470

Finance Analytics Tasks

Excel:

1. Forecast Sheet to predict monthly revenue based on the last 12 months' data.

Timeline	Values	Forecast	Lower Confidence Bound	Upper Confidence Bound
01-08-2015	1887693			
01-09-2015	2188334			
01-10-2015	2132020			
01-11-2015	2401136			
01-12-2015	3210409			
01-01-2016	2159886			
01-02-2016	2015459			
01-03-2016	2193450			
01-04-2016	2327352			
01-05-2016	2698638			
01-06-2016	3066664			
01-07-2016	471069	471069	471069.00	471069.00
01-08-2016		1574067.6	30403.30	3117731.92
01-09-2016		1548588.1	-177971.97	3275148.17
01-10-2016		1523108.6	-369379.16	3415596.33
01-11-2016		1497629.1	-547952.73	3543210.87
01-12-2016		1472149.6	-716388.66	3660687.77
01-01-2017		1446670	-876560.14	3769900.22

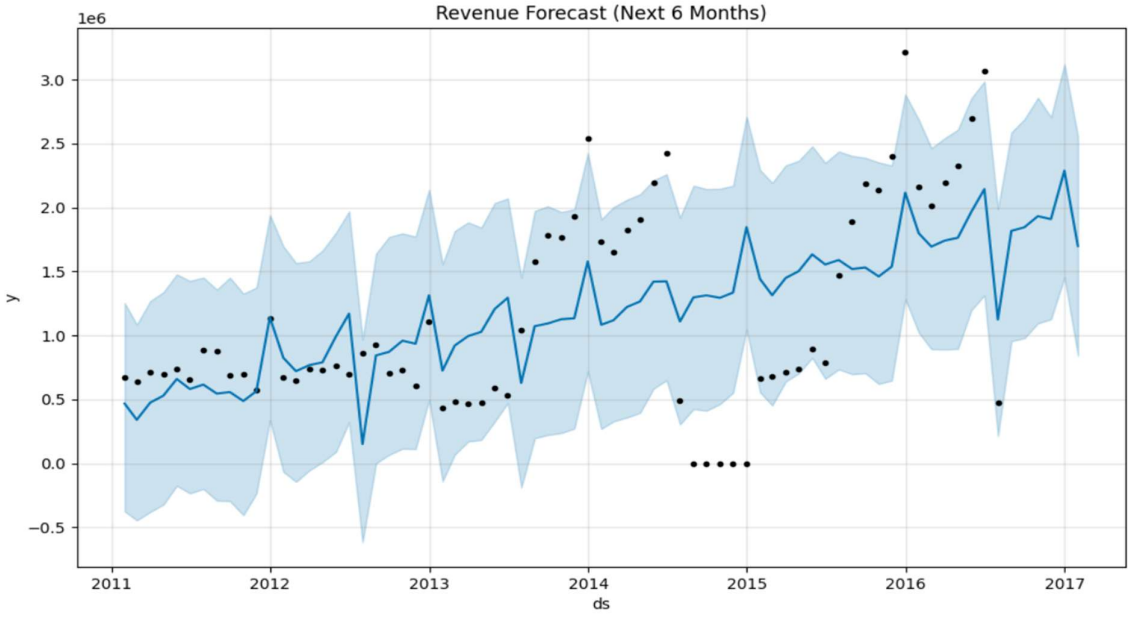


2. NPV and IRR for a project given a series of cash flows.

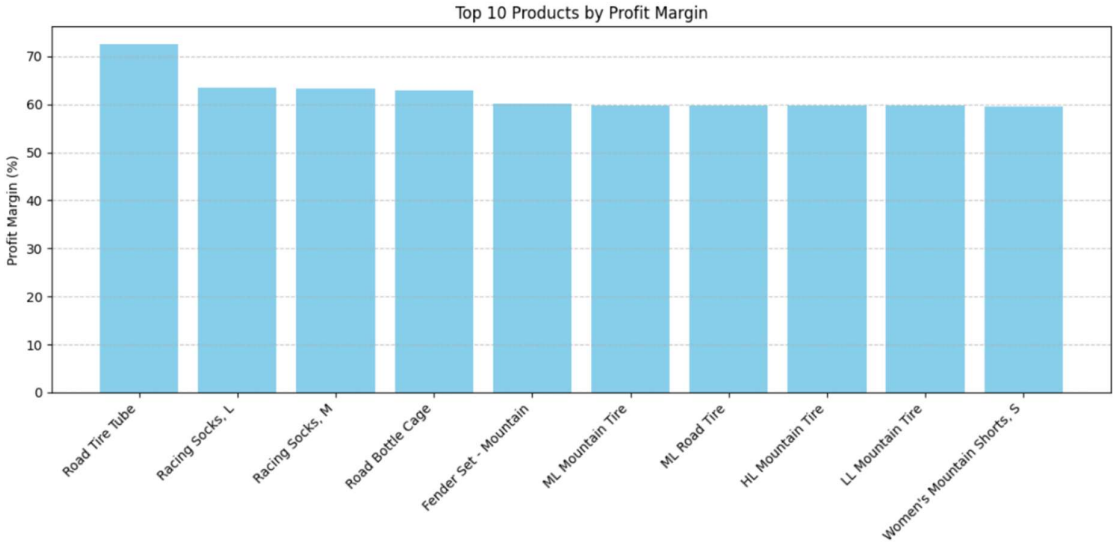
Year	Cash Flow (₹)	NPV	₹ 29,078.68
0	-100000	IRR	20%
1	25000		
2	30000		
3	35000		
4	40000		
5	45000		

Python

1. Prophet to forecast revenue for the next 6 months



2. Profit Margin bar plot of products using pandas and matplotlib



SQL

1. Total sales per month for the past year

Month	Total_Sales
2015-07	57903
2015-08	1887693
2015-09	2188334
2015-10	2132020
2015-11	2401136
2015-12	3210409
2016-01	2159886
2016-02	2015459
2016-03	2193450
2016-04	2327352
2016-05	2698638
2016-06	3066664
2016-07	471069

2. Calculate Return on Assets (ROA) by year from a financials table

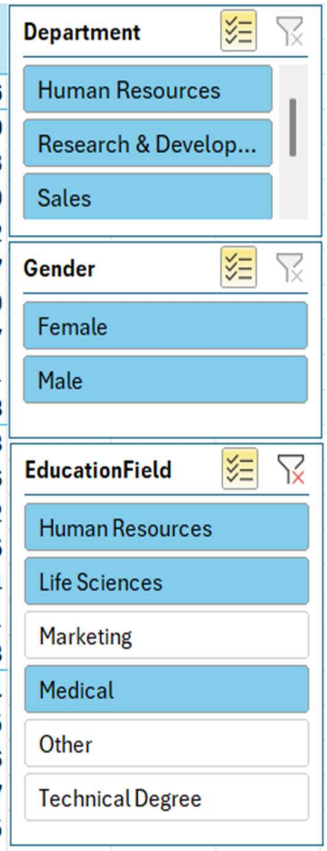
Year	ROA
2011	0.0483
2012	0.0494
2013	0.0117
2014	0.0094
2015	0.0145
2016	0.011

HR Analytics Tasks

Excel:

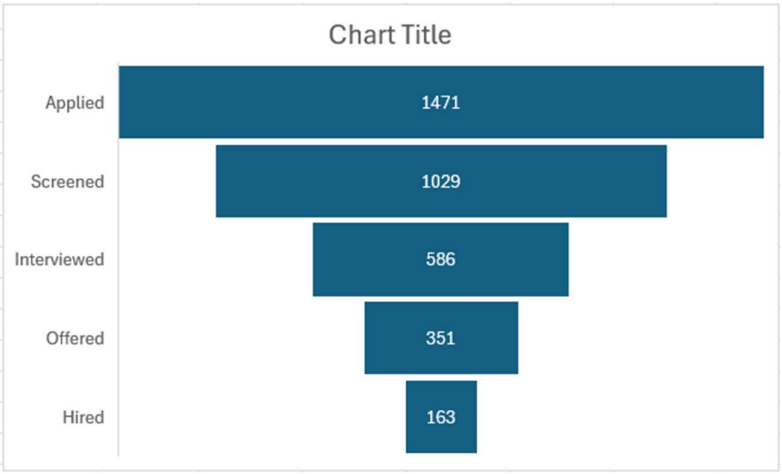
1. Attrition dashboard using PivotTables and Slicers.

Count of EmployeeID	Column Labels		
Row Labels	No	Yes	Grand Total
Human Resources	**46**	**10**	**56**
Female	**14**	**5**	**19**
Human Resources	5	3	8
Life Sciences	8	1	9
Medical	1	1	2
Male	**32**	**5**	**37**
Human Resources	15	4	19
Life Sciences	7		7
Medical	10	1	11
Research & Development	**697**	**106**	**803**
Female	**282**	**36**	**318**
Life Sciences	142	24	166
Medical	140	12	152
Male	**415**	**70**	**485**
Life Sciences	239	35	274
Medical	176	35	211
Sales	**195**	**43**	**238**
Female	**82**	**19**	**101**
Life Sciences	52	13	65
Medical	30	6	36
Male	**113**	**24**	**137**
Life Sciences	69	16	85
Medical	44	8	52
Grand Total	**938**	**159**	**1097**



2. Recruitment funnel tracker using Conditional Formatting to highlight drop-offs

Stage	Count
Applied	1471
Screened	1029
Interviewed	586
Offered	351
Hired	163



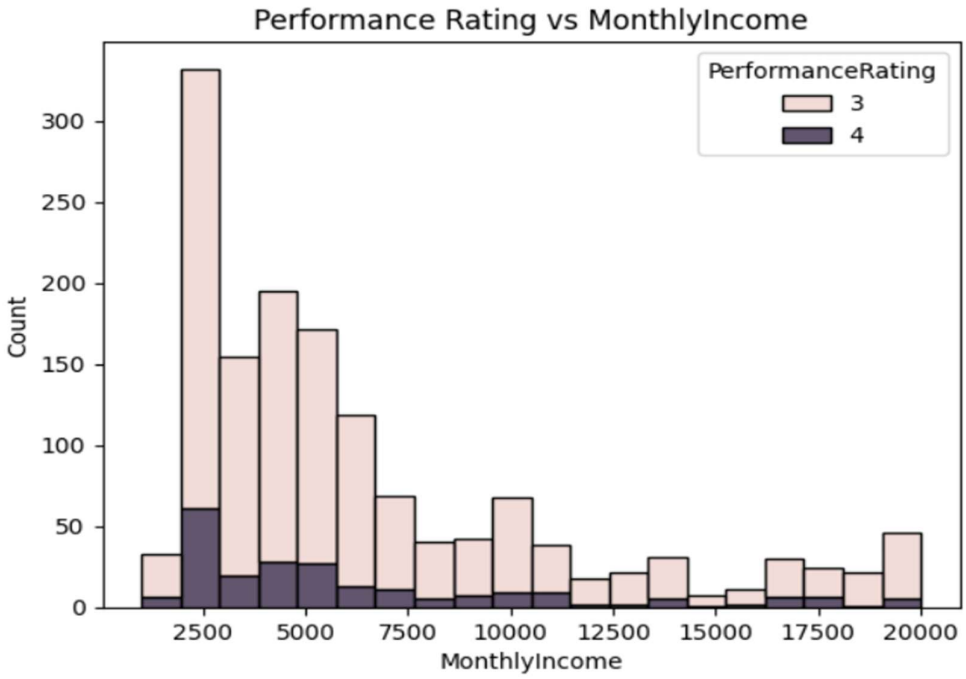
Python

1. Random forest to predict attrition from sample employee data.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Features and Target
X = df.drop('Attrition', axis=1)
y = df['Attrition']

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       255
           1       0.57      0.10      0.17        39

    accuracy                           0.87       294
   macro avg       0.72      0.55      0.55       294
weighted avg       0.84      0.87      0.83       294
```

2. Performance ratings vs. Monthly Income



SQL

1. Count attrition by department

Department	Total_Employees	Attrition_Count
Sales	446	92
Research & Development	961	133
Human Resources	63	12

2. Calculate average tenure of employees

	Avg_Tenure
▶	7.0082

Operations Analytics Tasks

Excel:

1. Use formulas to calculate EOQ and ROP. Highlight reorder alert when stock goes below threshold.
2. Use Goal Seek to find the optimal order quantity to minimize cost.

Product	Annual_Demand	Unit_Cost	Unit_Price	Ordering_Cost	Holding_Cost	Lead_Time (days)	Daily_Usage	Current_Stock	EOQ	ROP	total_cost	Goal seek
Pen	10000	5	8	500	2	4	40	150	2236.067977	160	4472.136	-750297673
Notebook	5000	15	22	700	3	6	20	70	1527.525232	120	4582.576	-512550888
Marker	3000	8	12	400	1.5	5	12	50	1264.911064	60	1897.367	-424432458

EOQ	SQRT((2 * D * S) / H)
ROP	Lead_Time * Daily_Usage
Total cost	(D/Q)*S + (Q/2)*H

conditional formatting
current stock < ROP

D = Annual demand
S = Ordering cost per order
H = Holding cost per unit per year

Python:

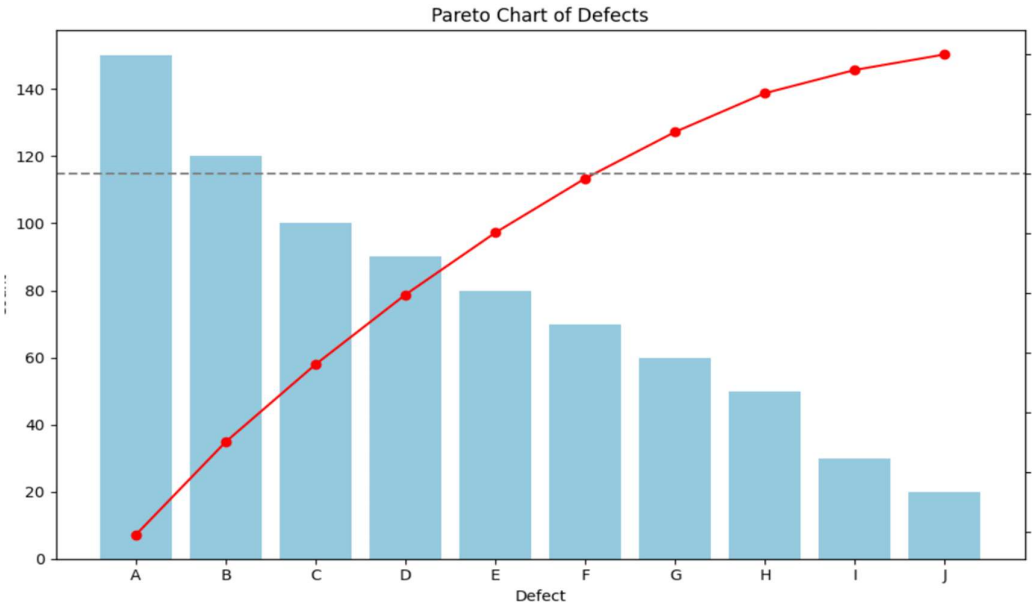
1. Use PuLP to create a simple linear optimization model to minimize inventory cost

```
import numpy as np

# Parameters
D = 10000  # Annual demand
S = 500    # Ordering cost per order
H = 2      # Holding cost per unit

# Try different order quantities
q_range = np.arange(1, 1000)  # Test from 1 to 1000
costs = [(q, (D / q) * S + (q / 2) * H) for q in q_range]

# Find the EOQ with minimum cost
optimal_q, min_cost = min(costs, key=lambda x: x[1])

print(f"Optimal EOQ: {optimal_q}")
print(f"Minimum Total Cost: ₹{min_cost:.2f}")
```

```
Optimal EOQ: 999
Minimum Total Cost: ₹6004.01
```

2. Pareto chart for top 10 defect types

```
ax2.plot(df['Defect'], df['Cumulative'], color='red', marker='o')
ax2.axhline(0.8, color='gray', linestyle='--')
ax.set_title("Pareto Chart of Defects")
ax.set_ylabel("Count")
ax2.set_ylabel("Cumulative %")
plt.tight_layout()
plt.show()
```



SQL

1. Stock movement summary by product

```
30 •  SELECT
31        Product,
32        SUM(Units_Received) AS Total_Received,
33        SUM(Units_Sold) AS Total_Sold,
34        SUM(Units_Received - Units_Sold) AS Net_Stock_Change
35    FROM inventory_log
36    GROUP BY Product;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Product	Total_Received	Total_Sold	Net_Stock_Change
Pen	1000	850	150
Notebook	600	500	100
Marker	400	300	100

2. Late deliveries by supplier

```
38 •  SELECT
39        Supplier,
40        COUNT(*) AS Late_Deliveries
41    FROM deliveries
42    WHERE Actual_Delivery_Date > Expected_Delivery_Date
43    GROUP BY Supplier
44    ORDER BY Late_Deliveries DESC;
```

Supplier	Late_Deliveries
XYZ Supplies	2
ABC Corp	1