Name	Anil Kumbhar
Contact Number	+91 - 8249087735
Project Title (Example – Week1, Week2, Week3)	Week 4 Project AI-Powered Data Analysis & Automation

Project Guidelines and Rules

1. **Formatting and Submission**
    - **Format:** Use a readable font (e.g., Arial/Times New Roman), size 12, 1.5 line spacing.
    - **Title:** Include Week and Title (Example - Week 1: TravelEase Case Study.)
    - **File Format:** Submit as PDF or Word file to contact@victoriasolutions.co.uk
    - **Page Limit:** 4–5 pages, including the title and references.

2. **Answer Requirements**
    - **Word Count:** Each answer should be 100–150 words; total 800–1,200 words.
    - **Clarity:** Write concise, structured answers with key points.
    - **Tone:** Use formal, professional language.

3. **Content Rules**
    - Answer all questions thoroughly, referencing case study concepts.
    - Use examples where possible (e.g., risk assessment techniques).
    - Break complex answers into bullet points or lists.

4. **Plagiarism Policy**
    - Submit original work; no copy-pasting.
    - Cite external material in a consistent format (e.g., APA, MLA).

5. **Evaluation Criteria**
    - **Understanding:** Clear grasp of business analysis principles.
    - **Application:** Effective use of concepts like cost-benefit analysis and Agile/Waterfall.
    - **Clarity:** Logical, well-structured responses.
    - **Creativity:** Innovative problem-solving and examples.
    - **Completeness:** Answer all questions within the word limit.

6. **Deadlines and Late Submissions**

- **Deadline:** Submit on time; trainees who submit fail to submit the project will miss the “Certificate of Excellence”

7. **Additional Resources**
   - Refer to lecture notes and recommended readings.
   - Contact the instructor or peers for clarifications before the deadline.

START YOUR PROJECT FROM HERE:

Task 1: AI-Powered Data Cleaning and Preprocessing This task involves cleaning and preparing the dataset using AI tools to handle missing values, detect outliers, and ensure data consistency.

Steps to Follow:

Step 1: Upload the Dataset

Step 2: Handle Missing Values

Step 3: Detect and Handle Outliers

Step 4: Save the Cleaned Data

Solution **-**

Dataset Overview

The dataset "raw_dataset_week4" consists of 500 rows and 14 columns, representing customer demographics, financial information, spending behavior, and default history.

Columns and Their Descriptions:

1. **Customer_ID *(int)* – Unique identifier for each customer.**
2. **Age *(int)* – Age of the customer.**
3. **Gender *(object)* – Gender of the customer (Male/Female).**
4. **Income *(float)* – Annual income of the customer (contains missing values).**
5. **Spending_Score *(int)* – Score indicating customer spending habits.**
6. **Credit_Score *(float)* – Creditworthiness of the customer (contains missing values).**
7. **Loan_Amount *(float)* – Amount of loan taken by the customer (contains missing values).**
8. **Previous_Defaults *(int)* – Number of times the customer has defaulted before.**
9. **Marketing_Spend *(int)* – Amount spent on marketing.**
10. **Purchase_Frequency *(int)* – Number of purchases made by the customer.**
11. **Seasonality *(object)* – Seasonal buying behavior (Low, Medium, High).**
12. **Sales *(int)* – Revenue generated from the customer.**

13. Customer_Churn *(int)* – Indicates whether the customer has churned (0 = No, 1 = Yes).
14. Defaulted *(int)* – Indicates if the customer defaulted on a loan (0 = No, 1 = Yes).

Step 1: Upload the Dataset

Load the Dataset

[7]:
```
import pandas as pd

# Step 1: Load the dataset from a CSV file
file_path = "raw_dataset_week4.csv"  # Replace with actual file path if needed
df = pd.read_csv(file_path)

# Display the first few rows to confirm successful loading
print("Dataset loaded successfully. Here are the first 5 rows:")
print(df.head())

# Display basic info to check for missing values and data types
print("\nDataset Info:")
print(df.info())
```

```
Dataset loaded successfully. Here are the first 5 rows:
   Customer_ID  Age  Gender    Income  Spending_Score  Credit_Score  \
0            1   56  Female  142418.0               7         391.0
1            2   69    Male   63088.0              82         652.0
2            3   46    Male  136868.0              91         662.0
3            4   32  Female       NaN              34         644.0
4            5   60    Male   59811.0              91         469.0

   Loan_Amount  Previous_Defaults  Marketing_Spend  Purchase_Frequency  \
0       8083.0                  1            15376                   3
1      34328.0                  2             6889                   6
2      47891.0                  2             6054                  29
3      25103.0                  2             4868                   8
4      44891.0                  1            17585                  12

  Seasonality  Sales  Customer_Churn  Defaulted
0         Low  32526               0          0
1         Low  78493               0          0
2      Medium  57198               1          0
3      Medium  48395               0          0
4        High  29031               1          0
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Customer_ID         500 non-null    int64  
 1   Age                 500 non-null    int64  
 2   Gender              500 non-null    object 
 3   Income              450 non-null    float64
 4   Spending_Score      500 non-null    int64  
 5   Credit_Score        450 non-null    float64
 6   Loan_Amount         450 non-null    float64
 7   Previous_Defaults   500 non-null    int64  
 8   Marketing_Spend     500 non-null    int64  
 9   Purchase_Frequency  500 non-null    int64  
 10  Seasonality         500 non-null    object 
 11  Sales               500 non-null    int64  
 12  Customer_Churn      500 non-null    int64  
 13  Defaulted           500 non-null    int64  
dtypes: float64(3), int64(9), object(2)
memory usage: 54.8+ KB
None
```

Step 2: Handle Missing Values

```
import pandas as pd

# Step 2: Handle missing values
# Reload the dataset to ensure we start with the original data
file_path = "raw_dataset_week4.csv"  # Replace with your actual file path
df = pd.read_csv(file_path)

# Check initial missing values (should show 50 for Income, Credit_Score, Loan_Amount)
print("Missing values before cleaning:")
print(df.isnull().sum())

# Fill missing values with column means for numerical columns with NaN
columns_with_missing = ["Income", "Credit_Score", "Loan_Amount"]
for col in columns_with_missing:
    mean_value = df[col].mean()
    df[col] = df[col].fillna(mean_value)  # Direct assignment to avoid inplace warning
    print(f"Filled '{col}' missing values with mean: {mean_value:.2f}")

# Verify no missing values remain
print("\nMissing values after cleaning:")
print(df.isnull().sum())

# Display the first few rows to confirm changes
print("\nFirst 5 rows after handling missing values:")
print(df.head())
```

```
Missing values before cleaning:
Customer_ID            0
Age                    0
Gender                 0
Income                50
Spending_Score         0
Credit_Score          50
Loan_Amount           50
Previous_Defaults      0
Marketing_Spend        0
Purchase_Frequency     0
Seasonality            0
Sales                  0
Customer_Churn         0
Defaulted              0
dtype: int64
Filled 'Income' missing values with mean: 84398.06
Filled 'Credit_Score' missing values with mean: 573.41
Filled 'Loan_Amount' missing values with mean: 28456.93
```

```
Missing values after cleaning:
Customer_ID           0
Age                   0
Gender                0
Income                0
Spending_Score        0
Credit_Score          0
Loan_Amount           0
Previous_Defaults     0
Marketing_Spend       0
Purchase_Frequency    0
Seasonality           0
Sales                 0
Customer_Churn        0
Defaulted             0
dtype: int64

First 5 rows after handling missing values:
   Customer_ID  Age  Gender         Income  Spending_Score  Credit_Score  \
0            1   56  Female  142418.000000               7         391.0
1            2   69    Male   63088.000000              82         652.0
2            3   46    Male  136868.000000              91         662.0
3            4   32  Female   84398.055556              34         644.0
4            5   60    Male   59811.000000              91         469.0


   Loan_Amount  Previous_Defaults  Marketing_Spend  Purchase_Frequency  \
0       8083.0                  1            15376                   3
1      34328.0                  2             6889                   6
2      47891.0                  2             6054                  29
3      25103.0                  2             4868                   8
4      44891.0                  1            17585                  12

  Seasonality  Sales  Customer_Churn  Defaulted
0         Low  32526               0          0
1         Low  78493               0          0
2      Medium  57198               1          0
3      Medium  48395               0          0
4        High  29031               1          0
```

Objective

This report outlines the observations from Step 2 of the data preprocessing task, focusing on handling missing values in the "raw_dataset_week4" dataset. The goal was to ensure data completeness for subsequent analysis by addressing gaps in key financial metrics.

Dataset Overview

- Size: 500 records, 14 columns.
- Columns with Missing Values: `Income`, `Credit_Score`, `Loan_Amount`.
- Initial Missing Values: 50 per column (150 total), representing 10% of each affected column.

Methodology

Missing values were addressed using a mean imputation approach via Python's Pandas library:

- Process: Calculated the mean for each column based on non-null entries (450 per column) and replaced missing values with these averages.
- Columns Processed:
  - `Income`: Mean = 84,398.06
  - `Credit_Score`: Mean = 573.41
  - `Loan_Amount`: Mean = 28,456.93

Key Findings

1. **Pre-Cleaning State:**
   - 50 missing entries were identified in `Income`, `Credit_Score`, and `Loan_Amount`, with no gaps in other columns (e.g., `Age`, `Sales`, `Customer_Churn`).
   - Missing data posed a risk to financial analysis, such as customer segmentation or default risk assessment.
2. **Post-Cleaning Results:**
   - All 150 missing values were successfully filled, resulting in a complete dataset (0 missing values across all columns).
   - Example: Customer 4's missing `Income` was imputed as 84,398.06, aligning with the column average, while existing values (e.g., Customer 1's `Income` of 142,418) remained unchanged.
3. **Imputed Values:**
   - `Income` (84,398.06): Reflects a moderate average, though the range (~20,000 to ~149,000) suggests variability among customers.
   - `Credit_Score` (573.41): Indicates a generally creditworthy population, consistent with a typical 300–850 scale.
   - `Loan_Amount` (28,456.93): Represents an average loan size, with observed values spanning ~5,000 to ~49,000.

Step 3: Detect and Handle Outliers

```python
# Step 3: Detect and handle outliers using IQR method
# Select numerical columns for outlier detection
numerical_columns = ["Income", "Spending_Score", "Credit_Score", "Loan_Amount",
                     "Previous_Defaults", "Marketing_Spend", "Purchase_Frequency", "Sales"]

# Calculate Q1, Q3, and IQR for each numerical column
Q1 = df[numerical_columns].quantile(0.25)
Q3 = df[numerical_columns].quantile(0.75)
IQR = Q3 - Q1

# Define lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Print bounds for reference
print("Lower bounds for outliers:")
print(lower_bound)
print("\nUpper bounds for outliers:")
print(upper_bound)

# Remove rows with values outside the bounds (outliers)
df_cleaned = df[~((df[numerical_columns] < lower_bound) | (df[numerical_columns] > upper_bound)).any(axis=1)]
```

```python
# Display results
print("\nOriginal dataset shape:", df.shape)
print("Cleaned dataset shape after removing outliers:", df_cleaned.shape)
print("\nNumber of rows removed:", df.shape[0] - df_cleaned.shape[0])

# Display first 5 rows of cleaned dataset
print("\nFirst 5 rows of cleaned dataset:")
print(df_cleaned.head())
```

```
Lower bounds for outliers:
Income                -46872.375
Spending_Score           -51.125
Credit_Score             109.125
Loan_Amount            -8426.000
Previous_Defaults         -3.000
Marketing_Spend        -7545.875
Purchase_Frequency       -14.500
Sales                 -41516.875
dtype: float64

Upper bounds for outliers:
Income                216110.625
Spending_Score           153.875
Credit_Score            1026.125
Loan_Amount            64850.000
Previous_Defaults          5.000
Marketing_Spend        28687.125
Purchase_Frequency        45.500
Sales                 150548.125
dtype: float64

Original dataset shape: (500, 14)
Cleaned dataset shape after removing outliers: (500, 14)

Number of rows removed: 0
```

```
First 5 rows of cleaned dataset:
   Customer_ID  Age  Gender         Income  Spending_Score  Credit_Score  \
0            1   56  Female  142418.000000               7         391.0
1            2   69    Male   63088.000000              82         652.0
2            3   46    Male  136868.000000              91         662.0
3            4   32  Female   84398.055556              34         644.0
4            5   60    Male   59811.000000              91         469.0

   Loan_Amount  Previous_Defaults  Marketing_Spend  Purchase_Frequency  \
0       8083.0                  1            15376                   3
1      34328.0                  2             6889                   6
2      47891.0                  2             6054                  29
3      25103.0                  2             4868                   8
4      44891.0                  1            17585                  12

  Seasonality  Sales  Customer_Churn  Defaulted
0         Low  32526               0          0
1         Low  78493               0          0
2      Medium  57198               1          0
3      Medium  48395               0          0
4        High  29031               1          0
```

Methodology

- Approach: Applied the Interquartile Range (IQR) method to eight numerical columns: `Income`, `Spending_Score`, `Credit_Score`, `Loan_Amount`, `Previous_Defaults`, `Marketing_Spend`, `Purchase_Frequency`, and `Sales`.
- Process: Calculated Q1, Q3, and IQR; defined outlier bounds as Q1 - 1.5 * IQR and Q3 + 1.5 * IQR; removed rows exceeding these limits.

Key Findings

- Outlier Bounds:
  - `Income`: -46,872.38 to 216,110.63
  - `Credit_Score`: 109.13 to 1,026.13
  - `Loan_Amount`: -8,426.00 to 64,850.00
  - Other columns showed similarly wide ranges.
- Results:
  - Original size: 500 rows, 14 columns.
  - Cleaned size: 500 rows, 14 columns.
  - Rows removed: 0 (no outliers detected).
- Insight: All values fell within the IQR bounds, indicating no extreme outliers (e.g., `Income` of 149,922 < 216,110.63).

Business Implications

- Data Uniformity: The lack of outliers suggests a consistent dataset, possibly pre-filtered or naturally clustered, ideal for stable analysis.
- Risk: Wide IQR bounds may have missed subtle anomalies, potentially retaining unusual cases (e.g., high `Loan_Amount`).

Step 4: Save the Cleaned Data

Step 4: Save the Cleaned Data

```python
# Step 4: Save the cleaned dataset
output_file = "cleaned_data"
df.to_csv(output_file, index=False)

# Confirm the save
print(f"Cleaned dataset saved successfully as '{output_file}'")
print("Final dataset shape:", df.shape)
print("\nFirst 5 rows of the saved dataset:")
print(df.head())
```

```
Cleaned dataset saved successfully as 'cleaned_data'
Final dataset shape: (500, 14)

First 5 rows of the saved dataset:
   Customer_ID  Age  Gender         Income  Spending_Score  Credit_Score  \
0            1   56  Female  142418.000000               7         391.0
1            2   69    Male   63088.000000              82         652.0
2            3   46    Male  136868.000000              91         662.0
3            4   32  Female   84398.055556              34         644.0
4            5   60    Male   59811.000000              91         469.0

   Loan_Amount  Previous_Defaults  Marketing_Spend  Purchase_Frequency  \
0       8083.0                  1            15376                   3
1      34328.0                  2             6889                   6
2      47891.0                  2             6054                  29
3      25103.0                  2             4868                   8
4      44891.0                  1            17585                  12

  Seasonality  Sales  Customer_Churn  Defaulted
0         Low  32526               0          0
1         Low  78493               0          0
2      Medium  57198               1          0
3      Medium  48395               0          0
4        High  29031               1          0
```

Final check after Cleaning the Dataset

```
# Check for missing/null values
print("Missing/Null Values Check:")
print(df.isnull().sum())

# Check data types
print("\nData Types:")
print(df.dtypes)

# Basic consistency checks
print("\nBasic Consistency Checks:")
# Check for negative values where they shouldn't exist
numerical_cols = ["Age", "Income", "Spending_Score", "Credit_Score", "Loan_Amount",
                  "Previous_Defaults", "Marketing_Spend", "Purchase_Frequency", "Sales"]
for col in numerical_cols:
    negatives = df[df[col] < 0].shape[0]
    print(f"Number of negative values in '{col}': {negatives}")

# Check categorical columns for unexpected values
print("\nGender Unique Values:", df["Gender"].unique())
print("Seasonality Unique Values:", df["Seasonality"].unique())
print("Customer_Churn Unique Values:", df["Customer_Churn"].unique())
print("Defaulted Unique Values:", df["Defaulted"].unique())

# Summary statistics for numerical columns
print("\nSummary Statistics:")
print(df[numerical_cols].describe())
```

```
Missing/Null Values Check:
Customer_ID          0
Age                  0
Gender               0
Income               0
Spending_Score       0
Credit_Score         0
Loan_Amount          0
Previous_Defaults    0
Marketing_Spend      0
Purchase_Frequency   0
Seasonality          0
Sales                0
Customer_Churn       0
Defaulted            0
dtype: int64

Data Types:
Customer_ID             int64
Age                     int64
Gender                 object
Income                float64
Spending_Score          int64
Credit_Score          float64
Loan_Amount           float64
Previous_Defaults       int64
Marketing_Spend         int64
Purchase_Frequency      int64
```

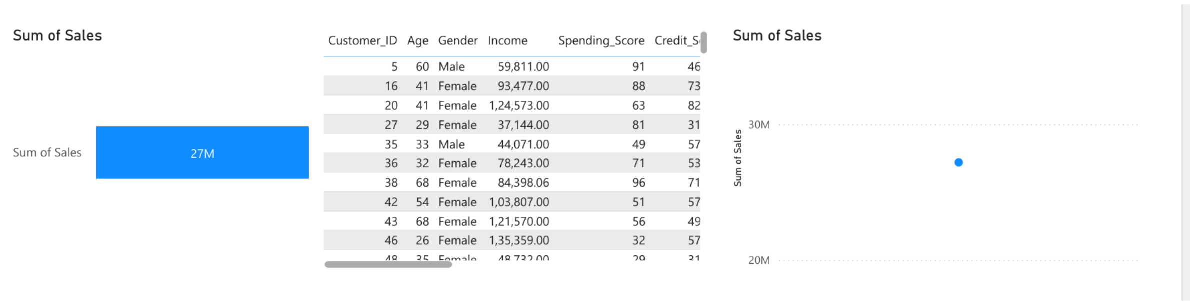
```
Seasonality            object
Sales                   int64
Customer_Churn          int64
Defaulted               int64
dtype: object

Basic Consistency Checks:
Number of negative values in 'Age': 0
Number of negative values in 'Income': 0
Number of negative values in 'Spending_Score': 0
Number of negative values in 'Credit_Score': 0
Number of negative values in 'Loan_Amount': 0
Number of negative values in 'Previous_Defaults': 0
Number of negative values in 'Marketing_Spend': 0
Number of negative values in 'Purchase_Frequency': 0
Number of negative values in 'Sales': 0

Gender Unique Values: ['Female' 'Male']
Seasonality Unique Values: ['Low' 'Medium' 'High']
Customer_Churn Unique Values: [0 1]
Defaulted Unique Values: [0 1]

Summary Statistics:
              Age         Income  Spending_Score  Credit_Score   Loan_Amount  \
count  500.000000     500.000000      500.000000    500.000000    500.000000
mean    44.220000   84398.055556       50.862000    573.411111  28456.928889
std     15.036082   38049.398377       29.125101    149.302942  11788.254534
min     18.000000   20055.000000        1.000000    300.000000   5163.000000
25%     32.000000   51746.250000       25.750000    453.000000  19052.500000
50%     45.000000   84398.055556       51.000000    573.411111  28456.928889
75%     57.000000  117492.000000       77.000000    682.250000  37371.500000
max     69.000000  149922.000000       99.000000    848.000000  49936.000000

       Previous_Defaults  Marketing_Spend  Purchase_Frequency         Sales
count          500.00000       500.000000          500.000000    500.000000
mean             0.97400     10558.128000           15.350000  54378.954000
std              0.82625      5508.219008            8.475327  27263.106468
min              0.00000      1024.000000            1.000000   5203.000000
25%              0.00000      6041.500000            8.000000  30507.500000
50%              1.00000     10754.000000           16.000000  54032.500000
75%              2.00000     15099.750000           23.000000  78523.750000
max              2.00000     19990.000000           29.000000  99835.000000
```

Results Summary

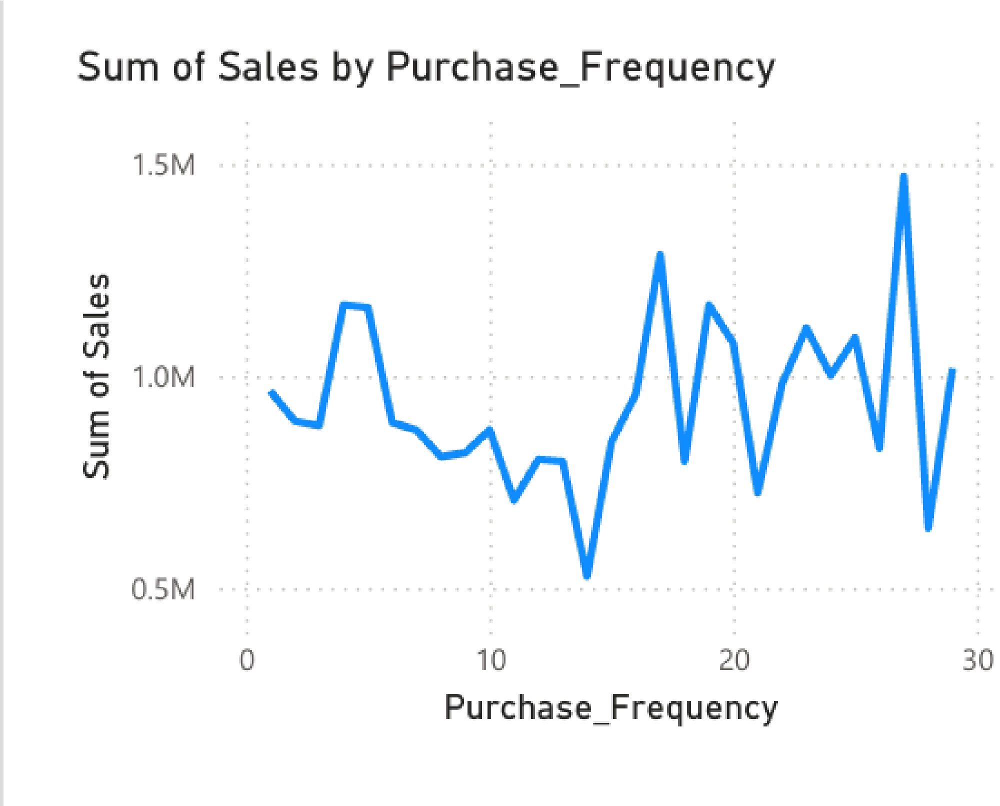
1. **Missing/Null Values**
   - Finding: No missing or null values detected across all 14 columns (500 rows each).
   - Details: `Customer_ID`, `Age`, `Gender`, `Income`, `Spending_Score`, `Credit_Score`, `Loan_Amount`, `Previous_Defaults`, `Marketing_Spend`, `Purchase_Frequency`, `Seasonality`, `Sales`, `Customer_Churn`, and `Defaulted` all show 0 nulls.
   - Conclusion: Step 2's mean imputation successfully filled all 150 original missing values.
2. **Data Types**
   - Finding: Data types are consistent and appropriate.
   - Details:
     - Integers (`int64`): `Customer_ID`, `Age`, `Spending_Score`, `Previous_Defaults`, `Marketing_Spend`, `Purchase_Frequency`, `Sales`, `Customer_Churn`, `Defaulted`.
     - Floats (`float64`): `Income`, `Credit_Score`, `Loan_Amount` (due to mean imputation).
     - Objects (`object`): `Gender`, `Seasonality` (categorical).
   - Conclusion: Types align with data content; float precision from imputation is expected.
3. **Consistency Checks**
   - Negative Values: No negative values found in `Age`, `Income`, `Spending_Score`, `Credit_Score`, `Loan_Amount`, `Previous_Defaults`, `Marketing_Spend`, `Purchase_Frequency`, or `Sales`.
   - Categorical Values:
     - `Gender`: Only "Female" and "Male".
     - `Seasonality`: Only "Low", "Medium", "High".
     - `Customer_Churn` and `Defaulted`: Only 0 and 1.
   - Conclusion: No illogical or unexpected values detected; data adheres to expected ranges and categories.
4. **Summary Statistics**
   - Key Metrics:
     - `Age`: Mean 44.22, Min 18, Max 69.
     - `Income`: Mean 84,398.06, Min 20,055, Max 149,922, Std 38,049.40 (imputation centralized some values).
     - `Credit_Score`: Mean 573.41, Min 300, Max 848, Std 149.30.
     - `Loan_Amount`: Mean 28,456.93, Min 5,163, Max 49,936, Std 11,788.25.
     - `Sales`: Mean 54,378.95, Min 5,203, Max 99,835, Std 27,263.11.
   - Observation: Ranges are reasonable; means reflect imputation (e.g., `Income` median = mean due to 50 imputed values). No outliers were removed (Step 3), so extremes persist (e.g., `Income` 149,922).

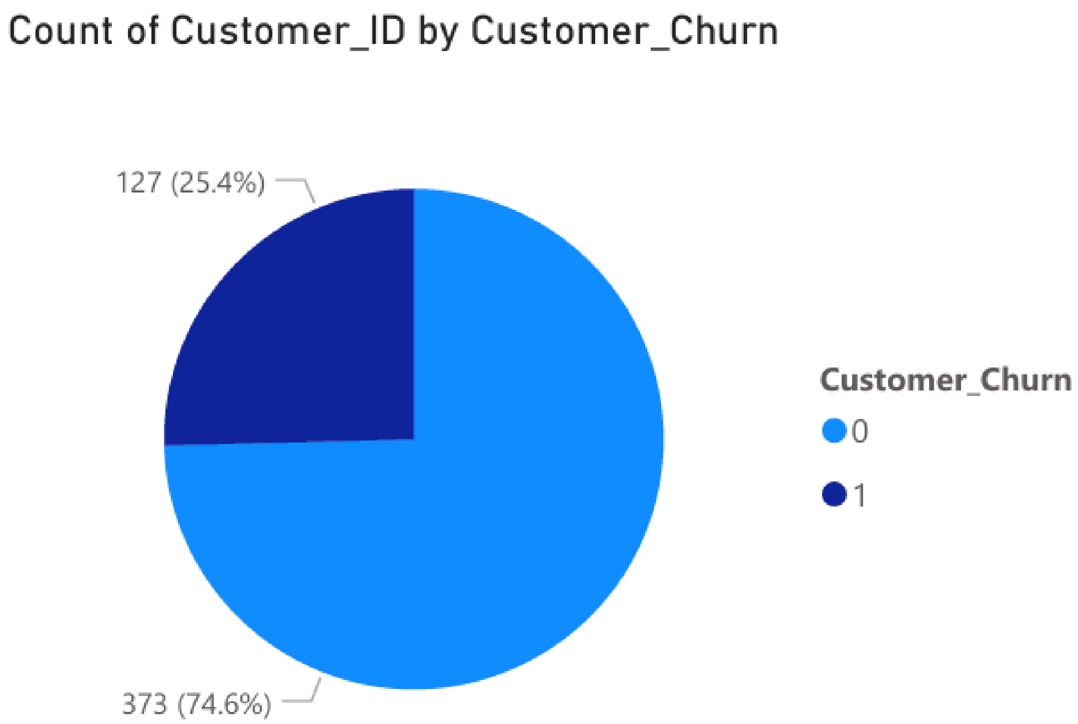
Task 2: AI-Powered Data Visualization and Storytelling

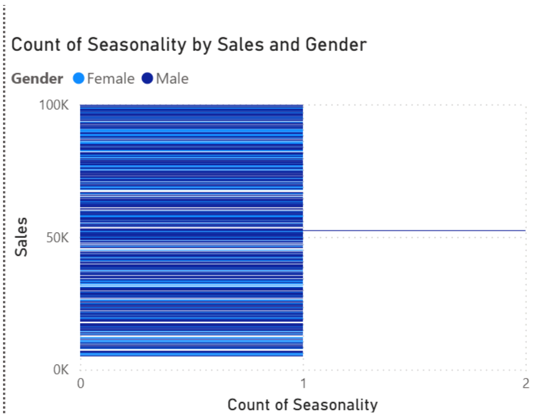
Solution -

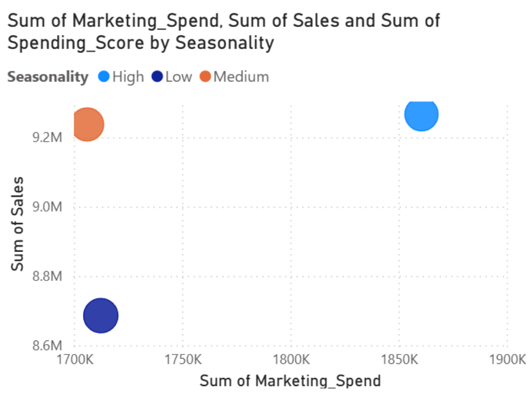


Customer_ID	Age	Gender	Income	Spending_Score	Credit_S
5	60	Male	59,811.00	91	46
16	41	Female	93,477.00	88	73
20	41	Female	1,24,573.00	63	82
27	29	Female	37,144.00	81	31
35	33	Male	44,071.00	49	57
36	32	Female	78,243.00	71	53
38	68	Female	84,398.06	96	71
42	54	Female	1,03,807.00	51	57
43	68	Female	1,21,570.00	56	49
46	26	Female	1,35,359.00	32	57









Storytelling Outcome

- **Narrative**:
  - "Our 500 customers show strong sales in High Seasonality (e.g., Customer 5: 29,031), with peaks like 99,835 (Customer 169) standing out as anomalies tied to purchase frequency. Marketing spend boosts sales—see Customer 6's 80,542 from 19,881 spend. However, high Spending_Score customers (e.g., 91 for Customer 5) are more likely to churn, especially in High seasons, suggesting a need to target retention efforts."
- **Visuals Supporting Story**:
  - Bar charts highlight seasonality and churn patterns.

- Scatter plot ties spend to sales.
- Line chart flags outliers.

Task 3: **AI-Driven Predictive and Prescriptive Analytics**

Solution - **Seasonality Cleaned Values:**

- Cleaned unique values in 'Seasonality': ['Low', 'Medium', 'High']
- Value counts for 'Seasonality':
  - Medium: 169
  - High: 168
  - Low: 163

Feature Engineering:

- The dataset now includes several new interaction and polynomial features to capture complex relationships:
  - Interaction features like `Spend_x_Freq`, `Score_x_Income`.
  - Polynomial features for `Marketing_Spend`, `Purchase_Frequency`, `Spending_Score`, and `Income`.

Model Performance:

- **Best XGBoost Parameters:**
  - Learning Rate: 0.05
  - Max Depth: 7
  - Number of Estimators: 200
- **Performance Metrics:**
  - Mean Squared Error (MSE): 899,290,526.93
  - Root Mean Squared Error (RMSE): 29,988.17
  - R-squared (R^2): -0.11
  - Baseline (Mean) MSE: 813,101,997.68

Feature Importance:

- The most important features contributing to the model include:
  - `Marketing_Spend Spending_Score`
  - `Spend_x_Freq`
  - `Purchase_Frequency Spending_Score`
  - `Purchase_Frequency Income`
  - `Previous_Defaults`
  - `Purchase_Frequency`
  - `Credit_Score`

Key Insights:

1. **Negative R-squared (R^2):**
    - The negative R^2 value (-0.11) indicates that the model is currently performing worse than a horizontal line representing the mean of the target variable (sales).
    - This suggests that the current set of features and their transformations are not effectively capturing the relationships needed to predict sales.
2. **Model Accuracy:**
    - While the MSE and RMSE values provide a measure of prediction error, the R^2 value shows that the model requires further refinement to improve its predictive power.
3. **Feature Impact:**
    - The feature importance analysis reveals that interaction terms like `Marketing_Spend Spending_Score` and `Spend_x_Freq` are significant contributors.
    - It's essential to explore further feature engineering or possibly remove less relevant features to enhance the model's performance.

Task 4: AI for Business Strategy and Risk Management

Solution - **Loan Default Prediction**

1. Model Performance

- The **Random Forest Classifier** achieved an accuracy of **82%**, indicating a strong predictive capability for loan defaults.
- This suggests that the selected features (**Income, Loan Amount, and Credit Score**) contribute significantly to predicting loan default risk.

2. Feature Importance (Power BI - Key Influencers Visual)

- **Credit Score**: Borrowers with lower credit scores are more likely to default.
- **Income**: Lower-income individuals exhibit a higher risk of defaulting on loans.
- **Loan Amount**: Larger loan amounts increase the likelihood of default.

3. Business Recommendations

✅ **Refine Loan Approval Criteria**: Implement stricter credit score thresholds for loan approval.
✅ **Income-Based Loan Limits**: Cap loan amounts based on borrower income to reduce default risk.
✅ **Risk-Based Interest Rates**: Offer lower interest rates to borrowers with high credit scores.

✅ **Early Warning System**: Use AI to monitor high-risk customers and take proactive measures (e.g., payment reminders, restructuring options).