Data Collection and Preprocessing Phase

Date	5 May 2024
Team ID	737906
Project Title	Walmart Sales Analysis For Retail Industry With Machine Learning.
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

For Walmart recruiting stores sales forecasting, thorough data exploration and preprocessing are essential. This involves understanding the dataset's structure, identifying missing values, and visualizing key distributions. Categorical variables need encoding, while feature engineering and scaling enhance predictive accuracy. Time series-specific preprocessing, such as stationarity checks and lag feature incorporation, is crucial for accurate forecasting. Outlier detection and feature selection refine the dataset, ensuring model reliability. Finally, data splitting facilitates model validation, enabling informed decision-making based on reliable predictions.

Section	**Description**
Data Overview	To understanding the basic statistics, dimensions, and structure of the data is essential for effective analysis and modeling. 1. Basic Statistics: Mean: Calculate the average sales amount, as well as the mean of other relevant variables such as price, quantity, and store performance metrics. Median: Determine the median sales amount to understand the central tendency of the data and assess its robustness to outliers. Standard Deviation: Compute the standard deviation of sales and other variables to measure the variability or dispersion around the mean. Minimum and Maximum: Identify the minimum and maximum values of sales and other variables to understand the range of the data. Percentiles: Calculate percentiles (e.g., 25th, 50th, 75th) to understand the distribution of sales amounts and other variables. 2. Dimensions of Data: Number of Observations: Count the total number of

	observations or records in the dataset to understand its size. Number of Variables: Determine the total number of variables or features available in the dataset, including both independent and dependent variables. Variable Types: Identify the types of variables present in the dataset, such as numerical, categorical, or ordinal. Time Dimension: If applicable, consider the temporal dimension of the data by examining timestamps or date-related variables. 3. Structure of Data: Data Format: Determine the format of the data (e.g., tabular, relational database, JSON) and ensure compatibility with analysis tools and libraries. Missing Values: Check for missing values in the dataset and decide on strategies for handling them (e.g., imputation, removal). Variable Names: Review variable names to ensure clarity and consistency, and consider renaming variables for better interpretability. Data Integrity: Assess the overall integrity of the data by checking for duplicates, inconsistencies, or errors that may require cleaning or preprocessing.
Univariate Analysis	Univariate involves examining individual variables one at a time to understand their distribution, central tendency, variability, and other statistical properties. performing univariate analysis involves calculating and interpreting measures of central tendency such as mean, median, and mode for relevant variables. Let's focus on analyzing the sales data: Mean: Mean Sales: Calculate the average sales amount across all transactions. This provides an overall view of the typical sales value. Mean Sales by Time Period: Compute the average sales for different time periods (e.g., daily, weekly, monthly) to identify any trends or seasonality. Median: Median Sales: Determine the median sales amount. Unlike the mean, the median is less affected by extreme values and provides a measure of central tendency that is robust to outliers. Median Sales by Store Type: Calculate the median sales for different types of stores (e.g., supercenter, neighborhood market) to understand variations in sales performance. Mode:

	Mode of Sales: Identify the sales amount that occurs most frequently. This can indicate common transaction sizes or popular price points. Mode of Sales by Product Category: Determine the most common sales amounts for different product categories, helping to understand consumer preferences.
Bivariate Analysis	bivariate analysis involves examining the relationships between pairs of variables, particularly focusing on the correlation between sales and other relevant factors including correlation analysis and scatterplots: Correlation Analysis: Pearson Correlation Coefficient: Calculate the correlation coefficient between sales and other numerical variables to measure the strength and direction of their linear relationship. Spearman Rank Correlation: Assess the correlation between sales and ordinal variables or non-linear relationships. Scatterplots: Sales vs. Time: Plot sales against time-related variables (e.g., date, hour) to visualize trends and seasonality. Sales vs. Store Attributes: Scatterplot sales against store attributes such as size or performance metrics to identify any patterns or correlations. Sales vs. Product Attributes: Explore the relationship between sales and product characteristics like price or category using scatterplots. Sales vs. External Factors: Examine how sales correlate with external factors like weather conditions or holidays.
Multivariate Analysis	Exploring relationships and patterns involving multiple variables is crucial for understanding the dynamics of sales and identifying key factors driving performance. Here's how you can investigate these relationships and patterns: 1. Exploratory Data Analysis (EDA): Correlation Matrix: Compute the correlation between all pairs of variables to identify significant relationships. Visualize the correlation matrix to understand the strength and direction of these relationships. Pairwise Scatterplots: Plot pairwise scatterplots between sales and other relevant variables to visually examine their relationships. Use different colors or markers to represent different categories or clusters if applicable. 2. Feature Engineering: Create Interaction Terms: Generate new features by combining

	existing variables to capture potential synergistic effects. For example, create a "Promotion × Store Type" feature to analyze how different types of stores respond to promotions. Temporal Features: Extract temporal features such as day of the week, month, or season from the date variable to explore sales patterns over time. 3. Regression Analysis: Multiple Regression: Build a multiple regression model with sales as the dependent variable and multiple predictors such as store attributes, product features, promotional activities, and external factors. Analyze the coefficients to understand the relative importance of each predictor. Regularization Techniques: Apply regularization techniques like Lasso or Ridge regression to handle multicollinearity and select the most important predictors. 4. Clustering Analysis: Customer Segmentation: Use clustering algorithms such as K-means to segment customers based on demographic variables, purchasing behavior, and geographical location. Analyze the characteristics of each segment and their corresponding sales patterns. Store Clustering: Cluster stores based on attributes like size, location, and performance metrics. Compare the sales performance of different clusters to identify high-performing and low-performing store groups. 5. Time Series Analysis: Seasonal Decomposition: Decompose the sales time series into trend, seasonal, and residual components to identify seasonal patterns and long-term trends. Lagged Variables: Incorporate lagged variables (e.g., sales from previous periods) into the analysis to capture temporal dependencies and autocorrelation.
Outliers and Anomalies	Identification and Treatment of Outliers: 1. Identify Outliers: Statistical Methods: Use statistical techniques like Z-score or interquartile range (IQR) to identify data points that deviate significantly from the rest of the distribution. Visualization: Plot histograms, box plots, or scatterplots to visually inspect the data for any extreme values or unusual patterns. 2. Understand the Nature of Outliers: Domain Knowledge: Utilize domain expertise to understand whether outliers represent genuine anomalies or errors in the data. Contextual Analysis: Investigate the circumstances surrounding

	outlier data points to determine their potential causes. 3. Choose Outlier Treatment Methods: Imputation: Replace outliers with a measure of central tendency (e.g., mean, median) to maintain the overall distribution of the data. Use interpolation techniques to estimate outlier values based on neighboring data points. Transformation: Apply transformations like log transformation to stabilize variance and reduce the impact of outliers. Removal: Remove outliers from the dataset entirely if they are deemed to be erroneous or unrepresentative of the underlying distribution. Consider trimming the dataset by removing a fixed percentage of data points from both tails of the distribution. 4. Implement Outlier Treatment: Apply the chosen outlier treatment method to the dataset, ensuring that it is performed consistently across all relevant variables. Document the rationale behind the outlier treatment decisions for transparency and reproducibility. 5. Evaluate the Impact: Assess the impact of outlier treatment on the distribution and statistical properties of the data. Determine whether outlier treatment has improved the quality and reliability of the dataset for subsequent analysis and modeling tasks. 6. Sensitivity Analysis: Conduct sensitivity analysis to explore the effects of different outlier treatment strategies on downstream analysis and model performance. Evaluate the robustness of conclusions and recommendations under various outlier handling scenarios. 7. Continuous Monitoring: Establish mechanisms for ongoing monitoring of data quality to detect and address outliers as new data becomes available. Incorporate outlier detection and treatment as part of regular data maintenance processes.
Data Preprocessing Code Screenshots	

Loading Data

```python
features=pd.read_csv('/content/features[1].csv')
```

```python
features.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	False
1	1	2010-02-12	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	True
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	False
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106	False
4	1	2010-03-05	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106	False

```python
sample=pd.read_csv('/content/sampleSubmission[1].csv')
```

```python
sample.head()
```

	Id	Weekly_Sales
0	1_1_2012-11-02	0
1	1_1_2012-11-09	0
2	1_1_2012-11-16	0
3	1_1_2012-11-23	0
4	1_1_2012-11-30	0

```python
stores-pd.read_csv('/content/stores[1].csv')
```

```python
stores.head()
```

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875

```python
test=pd.read_csv('/content/test[2].csv')
```

```python
test.head()
```

	Store	Dept	Date	IsHoliday
0	1	1	2012-11-02	False
1	1	1	2012-11-09	False
2	1	1	2012-11-16	False
3	1	1	2012-11-23	True
4	1	1	2012-11-30	False

```python
train=pd.read_csv('/content/train[1].csv')
```

```python
train.head()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	2010-02-05	24924.50	False
1	1	1	2010-02-12	46039.49	True
2	1	1	2010-02-19	41595.55	False
3	1	1	2010-02-26	19403.54	False
4	1	1	2010-03-05	21827.90	False

```
features.describe()
```

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
count	8190.000000	8190.000000	8190.000000	4032.000000	2921.000000	3613.000000	3464.000000	4050.000000	7605.000000	7605.000000
mean	23.000000	59.356198	3.405992	7032.371786	3384.176594	1760.100180	3292.935886	4132.216422	172.460809	7.826821
std	12.987966	18.678607	0.431337	9262.747448	8793.583016	11276.462208	6792.329861	13086.690278	39.738346	1.877259
min	1.000000	-7.290000	2.472000	-2781.450000	-265.760000	-179.260000	0.220000	-185.170000	126.064000	3.684000
25%	12.000000	45.902500	3.041000	1577.532500	68.880000	6.600000	304.687500	1440.827500	132.364839	6.634000
50%	23.000000	60.710000	3.513000	4743.580000	364.570000	36.260000	1176.425000	2727.135000	182.764003	7.806000
75%	34.000000	73.880000	3.743000	8923.310000	2153.350000	163.150000	3310.007500	4832.555000	213.932412	8.567000
max	45.000000	101.950000	4.468000	103184.980000	104519.540000	149483.310000	67474.850000	771448.100000	228.976456	14.313000

```
sample.describe()
```

	Weekly_Sales
count	115064.0
mean	0.0
std	0.0
min	0.0
25%	0.0
50%	0.0
75%	0.0
max	0.0

```
stores.describe()
```

	Store	Size
count	45.000000	45.000000
mean	23.000000	130287.600000
std	13.133926	63825.271991
min	1.000000	34875.000000
25%	12.000000	70713.000000
50%	23.000000	126512.000000
75%	34.000000	202307.000000
max	45.000000	219622.000000

```
test.describe()
```

	Store	Dept
count	115064.000000	115064.000000
mean	22.238207	44.339524
std	12.809930	30.656410
min	1.000000	1.000000
25%	11.000000	18.000000
50%	22.000000	37.000000
75%	33.000000	74.000000
max	45.000000	99.000000

```
train.describe()
```

	Store	Dept	Weekly_Sales
count	421570.000000	421570.000000	421570.000000
mean	22.200546	44.260317	15981.258123
std	12.785297	30.492054	22711.183519
min	1.000000	1.000000	-4988.940000
25%	11.000000	18.000000	2079.650000
50%	22.000000	37.000000	7612.030000
75%	33.000000	74.000000	20205.852500
max	45.000000	99.000000	693099.360000

Checking For Null Values

```
features.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8190 entries, 0 to 8189
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         8190 non-null   int64
 1   Date          8190 non-null   object
 2   Temperature   8190 non-null   float64
 3   Fuel_Price    8190 non-null   float64
 4   MarkDown1     4032 non-null   float64
 5   MarkDown2     2921 non-null   float64
 6   MarkDown3     3613 non-null   float64
 7   MarkDown4     3464 non-null   float64
 8   MarkDown5     4050 non-null   float64
 9   CPI           7605 non-null   float64
 10  Unemployment  7605 non-null   float64
 11  IsHoliday     8190 non-null   bool
dtypes: bool(1), float64(9), int64(1), object(1)
memory usage: 712.0+ KB
```

```
sample.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115064 entries, 0 to 115063
Data columns (total 2 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Id             115064 non-null  object
 1   Weekly_Sales   115064 non-null  int64
dtypes: int64(1), object(1)
memory usage: 1.8+ MB
```

```
stores.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Store   45 non-null     int64
 1   Type    45 non-null     object
 2   Size    45 non-null     int64
dtypes: int64(2), object(1)
memory usage: 1.2+ KB
```

```
test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115064 entries, 0 to 115063
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Store      115064 non-null  int64
 1   Dept       115064 non-null  int64
 2   Date       115064 non-null  object
 3   IsHoliday  115064 non-null  bool
dtypes: bool(1), int64(2), object(1)
memory usage: 2.7+ MB
```

```
train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 421570 entries, 0 to 421569
Data columns (total 5 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Store          421570 non-null  int64
 1   Dept           421570 non-null  int64
 2   Date           421570 non-null  object
 3   Weekly_Sales   421570 non-null  float64
 4   IsHoliday      421570 non-null  bool
dtypes: bool(1), float64(1), int64(2), object(1)
memory usage: 13.3+ MB
```

```
features.isnull().sum()

Store              0
Date               0
Temperature        0
Fuel_Price         0
MarkDown1       4158
MarkDown2       5269
MarkDown3       4577
MarkDown4       4726
MarkDown5       4140
CPI              585
Unemployment     585
IsHoliday          0
dtype: int64
```

```
sample.isnull().sum()

Id              0
Weekly_Sales    0
dtype: int64
```

```
stores.isnull().sum()

Store    0
Type     0
Size     0
dtype: int64
```

```
test.isnull().sum()

Store        0
Dept         0
Date         0
IsHoliday    0
dtype: int64
```

```
train.isnull().sum()

Store           0
Dept            0
Date            0
Weekly_Sales    0
IsHoliday       0
dtype: int64
```

```
data = train.merge(features, on=['Store','Date'], how='inner').merge(stores, on=['Store'], how='inner')
print(data.shape)

(421570, 17)
```

```
data['MarkDown1']=data['MarkDown1'].replace(np.nan, 0)
data['MarkDown2']=data['MarkDown2'].replace(np.nan, 0)
data['MarkDown3']=data['MarkDown3'].replace(np.nan, 0)
data['MarkDown4']=data['MarkDown4'].replace(np.nan, 0)
data['MarkDown5']=data['MarkDown5'].replace(np.nan, 0)
```

Handling Negative Values

```
data.describe()
```

	Store	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	
count	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421570.000000	421
mean	22.200546	44.260317	15981.258123	60.090059	3.361027	2590.074819	879.974298	468.087665	1083.132268	1662.772385	171.201947	7.960289	136
std	12.785297	30.492054	22711.183519	18.447931	0.458515	6052.385934	5084.538801	5528.873453	3894.529945	4207.629321	39.159276	1.863296	60
min	1.000000	1.000000	-4988.940000	-2.060000	2.472000	0.000000	-265.760000	-29.100000	0.000000	0.000000	126.064000	3.879000	34
25%	11.000000	18.000000	2079.650000	46.680000	2.933000	0.000000	0.000000	0.000000	0.000000	0.000000	132.022667	6.891000	93
50%	22.000000	37.000000	7612.030000	62.090000	3.452000	0.000000	0.000000	0.000000	0.000000	0.000000	182.318780	7.866000	140
75%	33.000000	74.000000	20205.852500	74.280000	3.738000	2809.050000	2.200000	4.540000	425.290000	2168.040000	212.416993	8.572000	202
max	45.000000	99.000000	693099.360000	100.140000	4.468000	88646.760000	104519.540000	141630.610000	67474.850000	108519.280000	227.232807	14.313000	219

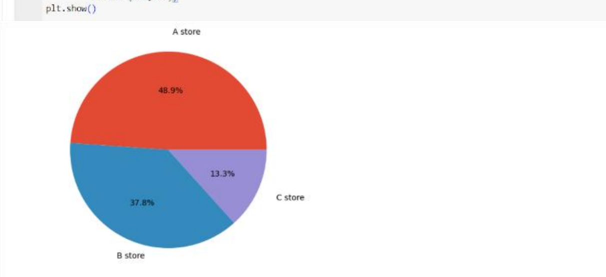
```
data = data[data['Weekly_Sales'] >= 0]
```

```
data.describe()
```

	Store	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	
count	420285.000000	420285.000000	420285.000000	420285.000000	420285.000000	420285.000000	420285.000000	420285.000000	420285.000000	420285.000000	420285.000000	420285.000000	4202
mean	22.195477	44.242771	16030.329773	60.090474	3.360888	2590.187246	878.803239	468.771234	1083.462694	1662.706138	171.212152	7.960077	1367
std	12.787213	30.507197	22728.500149	18.448260	0.458523	6053.225499	5076.525234	5533.593113	3895.801513	4205.946641	39.162280	1.863873	609
min	1.000000	1.000000	0.000000	-2.060000	2.472000	0.000000	-265.760000	-29.100000	0.000000	0.000000	126.064000	3.879000	348
25%	11.000000	18.000000	2117.560000	46.680000	2.933000	0.000000	0.000000	0.000000	0.000000	0.000000	132.022667	6.891000	936
50%	22.000000	37.000000	7659.090000	62.090000	3.452000	0.000000	0.000000	0.000000	0.000000	0.000000	182.350989	7.866000	1401
75%	33.000000	74.000000	20268.380000	74.280000	3.738000	2801.500000	2.400000	4.540000	425.290000	2168.040000	212.445487	8.567000	2025
max	45.000000	99.000000	693099.360000	100.140000	4.468000	88646.760000	104519.540000	141630.610000	67474.850000	108519.280000	227.232807	14.313000	2196

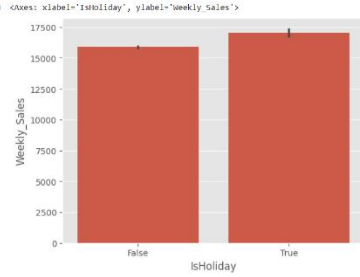
Exploratory Data Analysis

```
sorted_type = stores.groupby('Type')
plt.style.use('ggplot')
labels=['A store','B store','C store']
sizes=sorted_type.describe()['Size'].round(1)
sizes=[(22/(17+6+22))*100,(17/(17+6+22))*100,(6/(17+6+22))*100]
fig, axes = plt.subplots(1,1, figsize=(7,7))

axes.pie(sizes,
         labels=labels,
         explode=(0,0,0),
         autopct='%1.1f%%',
         pctdistance=0.6,
         labeldistance=1.2,
         radius=0.8,
         center=(0.5,0.5))
plt.show()
```

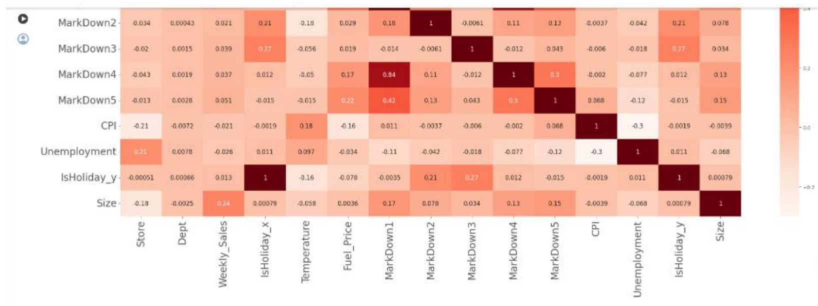



```
#weekly sales on Holidays
holiday=train['Weekly_Sales'].loc[train['IsHoliday']== True]

#weekly sales on non-holidays.
non_holiday = train['Weekly_Sales'].loc[train['IsHoliday']== False]
sns.barplot(x='IsHoliday',y='Weekly_Sales', data=train)
```

```
<Axes: xlabel='IsHoliday', ylabel='Weekly_Sales'>
```




```
plt.figure(figsize=(28,14))
plt.xticks( fontsize=20)
plt.yticks( fontsize=20)

import pandas as pd

corr_matrix = data.corr(numeric_only=True)

sns.heatmap(corr_matrix, cmap='Reds', annot=True, annot_kws={'size':12})
plt.title('Correlation Matrix', fontsize=30)
plt.show()
```



Handling Categorical Values

```
[ ] data=pd.get_dummies(data,columns=['Type'])

[ ] data['Date']=pd.to_datetime(data['Date'])

[ ] data['month'] = data['Date'].dt.month
    data['Year'] = data['Date'].dt.year

data[['Date','month', 'Year']].head()
```

	Date	month	Year
0	2010-02-05	2	2010
1	2010-02-05	2	2010
2	2010-02-05	2	2010
3	2010-02-05	2	2010
4	2010-02-05	2	2010

```
data['dayofweek_name'] = data['Date'].dt.day_name()
data[['Date','dayofweek_name']].head()
```

	Date	dayofweek_name
0	2010-02-05	Friday
1	2010-02-05	Friday
2	2010-02-05	Friday
3	2010-02-05	Friday
4	2010-02-05	Friday

```
[ ] data['is_weekend'] = np.where(data['dayofweek_name'].isin(['Sunday', 'Saturday']), 1, 0)
    data[['Date','is_weekend']].head()
```

	Date	is_weekend
0	2010-02-05	0
1	2010-02-05	0
2	2010-02-05	0
3	2010-02-05	0
4	2010-02-05	0

```
[ ] data["IsHoliday_x"] = data["IsHoliday_x"].astype(int)
    del data['dayofweek_name']
    #del df['Date']

print(data.head())

   Store  Dept       Date  Weekly_Sales  IsHoliday_x  Temperature  Fuel_Price  \
0      1     1 2010-02-05      24924.50            0        42.31       2.572
1      1     2 2010-02-05      50605.27            0        42.31       2.572
2      1     3 2010-02-05      13740.12            0        42.31       2.572
3      1     4 2010-02-05      39954.04            0        42.31       2.572
4      1     5 2010-02-05      32229.38            0        42.31       2.572

   MarkDown1  MarkDown2  MarkDown3  ...         CPI  Unemployment  \
0        0.0        0.0        0.0  ...  211.096358         8.106
1        0.0        0.0        0.0  ...  211.096358         8.106
2        0.0        0.0        0.0  ...  211.096358         8.106
3        0.0        0.0        0.0  ...  211.096358         8.106
4        0.0        0.0        0.0  ...  211.096358         8.106

   IsHoliday_y    Size  Type_A  Type_B  Type_C  month  Year  is_weekend
0        False  151315    True   False   False      2  2010           0
1        False  151315    True   False   False      2  2010           0
2        False  151315    True   False   False      2  2010           0
3        False  151315    True   False   False      2  2010           0
4        False  151315    True   False   False      2  2010           0

[5 rows x 22 columns]

[ ] data.to_csv('merged_data.csv', index=False)
```

Splitting Data Into Train And Test

```
X = data.loc[:, data.columns != 'Weekly_Sales']
y = data.loc[:, data.columns == 'Weekly_Sales']

X = X_temp[["Store", "Dept", "Size", "IsHoliday_x", "CPI", "Temperature", "Type_B","Type_C", "month", "Year", "IsHoliday_y"]]
y = y.values.reshape( 1,1)
print(X.head())

X train, x test, y train, y test = train test split(X, y, test size=0.2, random state=4)

   Store  Dept    Size  IsHoliday_x         CPI  Temperature  Type_B  Type_C  \
0      1     1  151315            0  211.096358        42.31   False   False
1      1     2  151315            0  211.096358        42.31   False   False
2      1     3  151315            0  211.096358        42.31   False   False
3      1     4  151315            0  211.096358        42.31   False   False
4      1     5  151315            0  211.096358        42.31   False   False

   month  Year  IsHoliday_y
0      2  2010        False
1      2  2010        False
2      2  2010        False
3      2  2010        False
4      2  2010        False

[ ] Start coding or generate with AI.
```