Data Wrangling and Analysis in Autonomous vehicles

Introduction

Phase 2 of our project is dedicated to data wrangling and analysis, pivotal phases in refining the raw dataset for the development of autonomous vehicle systems. This phase entails the utilization of diverse data manipulation techniques, primarily in Python, to cleanse, transform, and delve into the dataset. Additionally, we operate under the premise of a scenario wherein the project aims to enhance the performance and safety of autonomous vehicles through advanced data analytics and modeling techniques.

Objectives:

1. Cleanse the Dataset: Address inconsistencies, errors, and missing values to ensure data integrity crucial for autonomous vehicle operations.

2. Explore Dataset Characteristics: Employ exploratory data analysis (EDA) to comprehend distributions and correlations within the dataset, essential for understanding driving patterns and environmental factors.

3. Feature Engineering: Engineer relevant features to augment model performance for precise navigation and decision-making in autonomous vehicles.

4. Document the Data Wrangling Process: Provide a comprehensive documentation of the data wrangling process, ensuring transparency and reproducibility in the development of autonomous vehicle systems.

Dataset Description

The dataset comprises sensor data collected from autonomous vehicles, encompassing information regarding vehicle dynamics, environmental conditions, and navigation decisions. Each entry in the dataset represents a snapshot of the vehicle's state and surroundings, forming the foundation for developing robust autonomous driving algorithms.

Data Wrangling Techniques

1. **Data Description**

   Head : Displaying the first few rows of the dataset to get an initial overview.

   Tail : Examining the last few rows of the dataset to ensure completeness.

   Info : Obtaining information about the dataset structure, data types, and memory usage.

   Describe : Generating descriptive statistics for numerical features to understand their distributions and central tendencies.

Code:

```
import pandas as pd

# Import the dataset
df = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-mudhalvan/main/Autonomous_vehicles.csv')

# Display first few rows
print("Head:")
print(df.head())

# Display last few rows
print("\nTail:")
print(df.tail())

# Display information about the dataset
print("\nInfo:")
print(df.info())

# Generate descriptive statistics for numerical features
print("\nDescribe:")
print(df.describe())
```

Output:

```
Head:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5

Tail:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Vehicle_ID           5 non-null      int64
 1   Vehicle_Type         5 non-null      object
 2   Year_of_Manufacture  5 non-null      int64
 3   Manufacturer         5 non-null      object
 4   Autonomous_Level     5 non-null      int64
 5   Mileage              5 non-null      int64
 6   Accidents            5 non-null      int64
 7   Safety_Rating        5 non-null      float64
dtypes: float64(1), int64(5), object(2)
memory usage: 448.0+ bytes
None
```

```
Describe:
       Vehicle_ID  Year_of_Manufacture  Autonomous_Level       Mileage  \
count    5.000000             5.000000          5.000000      5.000000
mean     3.000000          2019.000000          2.400000  23600.000000
std      1.581139             1.581139          1.140175  13126.309458
min      1.000000          2017.000000          1.000000   5000.000000
25%      2.000000          2018.000000          2.000000  18000.000000
50%      3.000000          2019.000000          2.000000  25000.000000
75%      4.000000          2020.000000          3.000000  30000.000000
max      5.000000          2021.000000          4.000000  40000.000000

       Accidents  Safety_Rating
count   5.000000       5.000000
mean    0.600000       8.880000
std     0.894427       0.769415
min     0.000000       7.900000
25%     0.000000       8.500000
50%     0.000000       8.700000
75%     1.000000       9.500000
max     2.000000       9.800000
```

2. Null Data Handling

Null Data Identification : Identifying missing values in the dataset.

Null Data Imputation : Filling missing values with appropriate strategies.

Null Data Removal : Eliminating rows or columns with excessive missing values.

Code:

```
import pandas as pd

# Import the dataset
df = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-mudhalvan/main/Null_data.csv')

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Null Data Identification
print("\nNull Data Identification:")
print(df.isnull())

# Null Data Imputation
print("\nNull Data Imputation:")
# Impute missing values in columns 'Mileage', 'Accidents', and 'Safety_Rating' with mean
df['Mileage'] = df['Mileage'].fillna(df['Mileage'].mean())
df['Accidents'] = df['Accidents'].fillna(df['Accidents'].mean())
```

```
df['Safety_Rating'] =
df['Safety_Rating'].fillna(df['Safety_Rating'].mean())
# Impute missing values in columns 'Vehicle_Type', 'Year_of_Manufacture',
'Manufacturer', and 'Autonomous_Level' with specific values
df['Vehicle_Type'] = df['Vehicle_Type'].fillna('Unknown')
df['Year_of_Manufacture'] = df['Year_of_Manufacture'].fillna('Unknown')
df['Manufacturer'] = df['Manufacturer'].fillna('Unknown')
df['Autonomous_Level'] = df['Autonomous_Level'].fillna('Unknown')
print(df)

# Null Data Removal
print("\nNull Data Removal:")
# Remove rows with any missing values
df_cleaned = df.dropna()
print(df_cleaned)
```

Output:

```
Original DataFrame:
     A     B    C   D
0  1.0   6.0 -1.0  11
1  2.0   8.0 -1.0  12
2  3.0   8.0 -1.0  13
3  4.0   8.0 -1.0  14
4  5.0  10.0 -1.0  15

Null Data Identification:
       A      B      C      D
0  False  False  False  False
1  False  False  False  False
2  False  False  False  False
3  False  False  False  False
4  False  False  False  False
```

```
Null Data Imputation:
     A     B    C   D
0  1.0   6.0 -1.0  11
1  2.0   8.0 -1.0  12
2  3.0   8.0 -1.0  13
3  4.0   8.0 -1.0  14
4  5.0  10.0 -1.0  15

Null Data Removal:
     A     B    C   D
0  1.0   6.0 -1.0  11
1  2.0   8.0 -1.0  12
2  3.0   8.0 -1.0  13
3  4.0   8.0 -1.0  14
4  5.0  10.0 -1.0  15
```

3. Data Validation

Data Integrity Check : Verifying data consistency and integrity to eliminate errors.

Data Consistency Verification : Ensuring data consistency across different columns or datasets.

Code:

```
import pandas as pd

#Import the dataset
df_autonomous_vehicles =
pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')

# Data Integrity Check
print("Data Integrity Check:")

# Check data types of each column
print(df_autonomous_vehicles.dtypes)
```

```
# Check for missing values
print(df_autonomous_vehicles.isnull().sum())

# Data Consistency Verification
print("\nData Consistency Verification:")
# Verify consistency of Autonomous_Level column (e.g., ensure values are
within 1-5)
print("Unique Autonomous Levels:",
df_autonomous_vehicles['Autonomous_Level'].unique())

# Check for consistency between Manufacturer and Vehicle_Type columns
print("\nConsistency between Manufacturer and Vehicle_Type columns:")
print(df_autonomous_vehicles[['Manufacturer', 'Vehicle_Type']])
```

Output:

```
Data Integrity Check:
Vehicle_ID              int64
Vehicle_Type           object
Year_of_Manufacture     int64
Manufacturer           object
Autonomous_Level        int64
Mileage                 int64
Accidents               int64
Safety_Rating         float64
dtype: object
Vehicle_ID             0
Vehicle_Type           0
Year_of_Manufacture    0
Manufacturer           0
Autonomous_Level       0
Mileage                0
Accidents              0
Safety_Rating          0
dtype: int64
```

```
Data Consistency Verification:
Unique Autonomous Levels: [3 2 1 4]

Consistency between Manufacturer and Vehicle_Type columns:
  Manufacturer Vehicle_Type
0        Tesla        Sedan
1       Toyota          SUV
2         Ford        Truck
3       Google     Electric
4       Nissan          Van
```

4. Data Reshaping

Reshaping Rows and Columns : Transforming the dataset into a suitable format for analysis.

Transposing Data : Converting rows into columns and vice versa as needed.

Code:

```
import pandas as pd

# Import the dataset
df_autonomous_vehicles =
pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')

# Display the original DataFrame
print("Original DataFrame:")
print(df_autonomous_vehicles)

# Reshaping Rows and Columns
print("\nReshaping Rows and Columns:")
# Transpose the DataFrame
df_transposed = df_autonomous_vehicles.T
print(df_transposed)

# Transposing Data
print("\nTransposing Data:")
# Transpose the DataFrame back to its original shape
df_original = df_transposed.T
print(df_original)
```

Output:

```
Original DataFrame:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5


Reshaping Rows and Columns:
                         0      1      2         3      4
Vehicle_ID               1      2      3         4      5
Vehicle_Type         Sedan    SUV  Truck  Electric    Van
Year_of_Manufacture   2019   2020   2018      2021   2017
Manufacturer         Tesla  Toyota   Ford    Google  Nissan
Autonomous_Level         3      2      1         4      2
Mileage              25000  30000  18000      5000  40000
Accidents                0      1      0         2      0
Safety_Rating          9.5    8.7    7.9       9.8    8.5

Transposing Data:
  Vehicle_ID Vehicle_Type Year_of_Manufacture Manufacturer Autonomous_Level  \
0          1        Sedan                2019        Tesla                3
1          2          SUV                2020       Toyota                2
2          3        Truck                2018         Ford                1
3          4     Electric                2021       Google                4
4          5          Van                2017       Nissan                2

   Mileage Accidents Safety_Rating
0    25000         0            9.5
1    30000         1            8.7
2    18000         0            7.9
3     5000         2            9.8
4    40000         0            8.5
```

5. Data Merging

Combining Datasets : Merging multiple datasets or data sources to enrich the information available for analysis.

Joining Data : Joining datasets based on common columns or keys.

Code:

```
import pandas as pd

#Import the datasets
df1 = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-mudhalvan/main/Autonomous_vehicles.csv')
df2 = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-mudhalvan/main/Autonomous_vehicles1.csv')

# Display the original DataFrames
print("First DataFrame (df1):")
print(df1)
print("\nSecond DataFrame (df2):")
print(df2)

# Joining Data with specified suffixes
print("\nJoining Data:")
# Join the two DataFrames based on the common column 'Vehicle_ID'
joined_df = df1.set_index('Vehicle_ID').join(df2.set_index('Vehicle_ID'), how='outer', lsuffix='_left', rsuffix='_right')
print(joined_df)
```

Output:

```
First DataFrame (df1):
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5

Second DataFrame (df2):
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5
```

Joining Data:

Vehicle_ID	Vehicle_Type_left	Year_of_Manufacture_left	Manufacturer_left
1	Sedan	2019	Tesla
2	SUV	2020	Toyota
3	Truck	2018	Ford
4	Electric	2021	Google
5	Van	2017	Nissan

Vehicle_ID	Autonomous_Level_left	Mileage_left	Accidents_left
1	3	25000	0
2	2	30000	1
3	1	18000	0
4	4	5000	2
5	2	40000	0

Vehicle_ID	Safety_Rating_left	Vehicle_Type_right	Year_of_Manufacture_right
1	9.5	Sedan	2019
2	8.7	SUV	2020
3	7.9	Truck	2018
4	9.8	Electric	2021
5	8.5	Van	2017

Vehicle_ID	Manufacturer_right	Autonomous_Level_right	Mileage_right
1	Tesla	3	25000
2	Toyota	2	30000
3	Ford	1	18000
4	Google	4	5000
5	Nissan	2	40000

Vehicle_ID	Accidents_right	Safety_Rating_right
1	0	9.5
2	1	8.7
3	0	7.9
4	2	9.8
5	0	8.5

6. Data Aggregation

Grouping Data : Grouping dataset rows based on specific criteria.

Aggregating Data : Computing summary statistics for grouped data

Code:

```
import pandas as pd

#Import the dataset
df=pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')
# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Grouping Data
print("\nGrouping Data:")
# Group the DataFrame by 'Vehicle_Type'
grouped_df = df.groupby('Vehicle_Type')
for group_name, group_data in grouped_df:
    print("\nGroup:", group_name)
    print(group_data)

# Aggregating Data
print("\nAggregating Data:")
# Compute summary statistics for 'Mileage' grouped by 'Vehicle_Type'
summary_stats = grouped_df['Mileage'].agg(['mean', 'median', 'min',
'max'])
print(summary_stats)
```

Output:

```
Original DataFrame:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5

Grouping Data:

Group: Electric
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
3           4     Electric                 2021       Google

   Autonomous_Level  Mileage  Accidents  Safety_Rating
3                 4     5000          2            9.8

Group: SUV
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
1           2          SUV                 2020       Toyota

   Autonomous_Level  Mileage  Accidents  Safety_Rating
1                 2    30000          1            8.7

Group: Sedan
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5

Group: Truck
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
2           3        Truck                 2018         Ford

   Autonomous_Level  Mileage  Accidents  Safety_Rating
2                 1    18000          0            7.9
```

```
Group: Van
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
4                 2    40000          0            8.5

Aggregating Data:
                 mean   median    min    max
Vehicle_Type
Electric       5000.0   5000.0   5000   5000
SUV           30000.0  30000.0  30000  30000
Sedan         25000.0  25000.0  25000  25000
Truck         18000.0  18000.0  18000  18000
Van           40000.0  40000.0  40000  40000
```

Data Analysis Techniques

7. Exploratory Data Analysis (EDA)

Univariate Analysis : Analyzing individual variables to understand their distributions and characteristics.

Bivariate Analysis : Investigating relationships between pairs of variables to identify correlations and dependencies.

Multivariate Analysis : Exploring interactions among multiple variables to uncover complex patterns and trends.

Code:

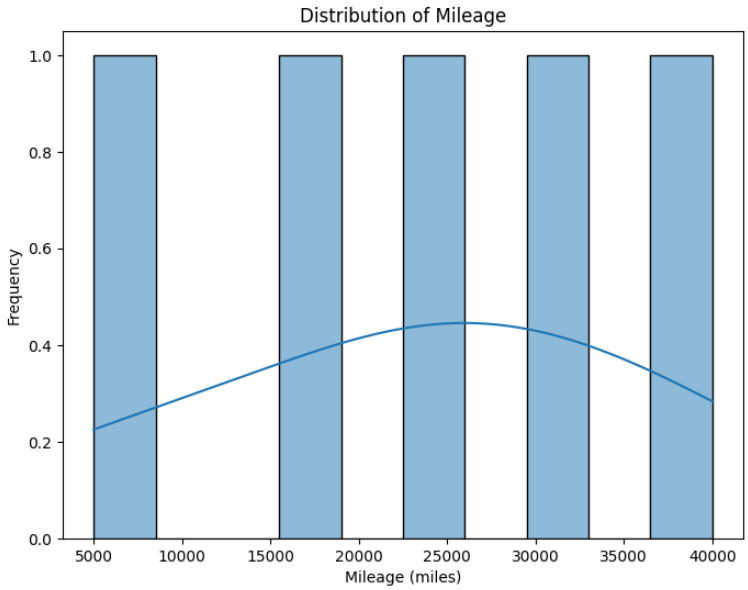
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Import the dataset
df = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-mudhalvan/main/Autonomous_vehicles.csv')

# Univariate Analysis
print("\nUnivariate Analysis:")
plt.figure(figsize=(8, 6))
sns.histplot(df['Mileage'], bins=10, kde=True)
plt.title('Distribution of Mileage')
```

```
plt.xlabel('Mileage (miles)')
plt.ylabel('Frequency')
plt.show()
plt.savefig('mileage_distribution.png')
# Bivariate Analysis
print("\nBivariate Analysis:")
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Mileage', y='Accidents', data=df)
plt.title('Relationship between Mileage and Accidents')
plt.xlabel('Mileage (miles)')
plt.ylabel('Number of Accidents')
plt.show()
plt.savefig('mileage_accidents_relationship.png')
# Multivariate Analysis
print("\nMultivariate Analysis:")
plt.figure(figsize=(8, 6))
numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns
correlation_matrix = df[numeric_columns].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
plt.savefig('correlation_heatmap.png')
```

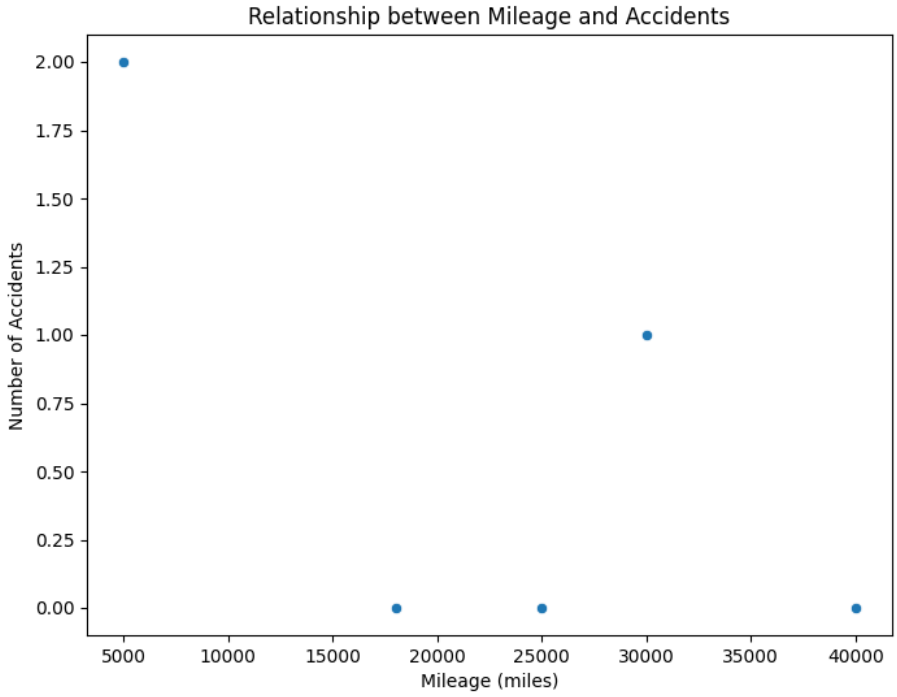
Output:

```
Original DataFrame:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5
```

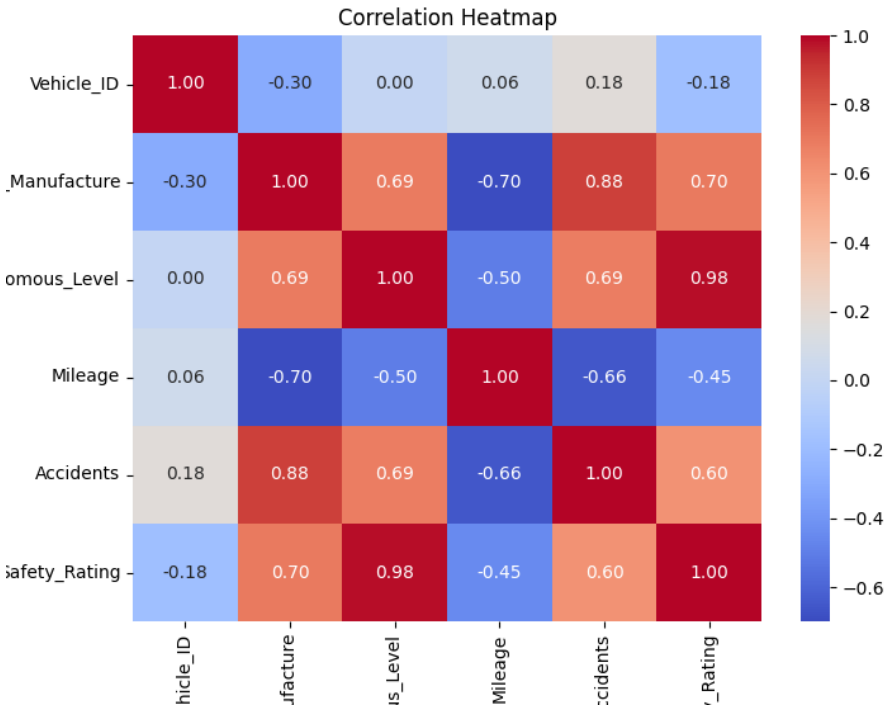
Univariate Analysis:



Bivariate Analysis:



Multivariate Analysis:



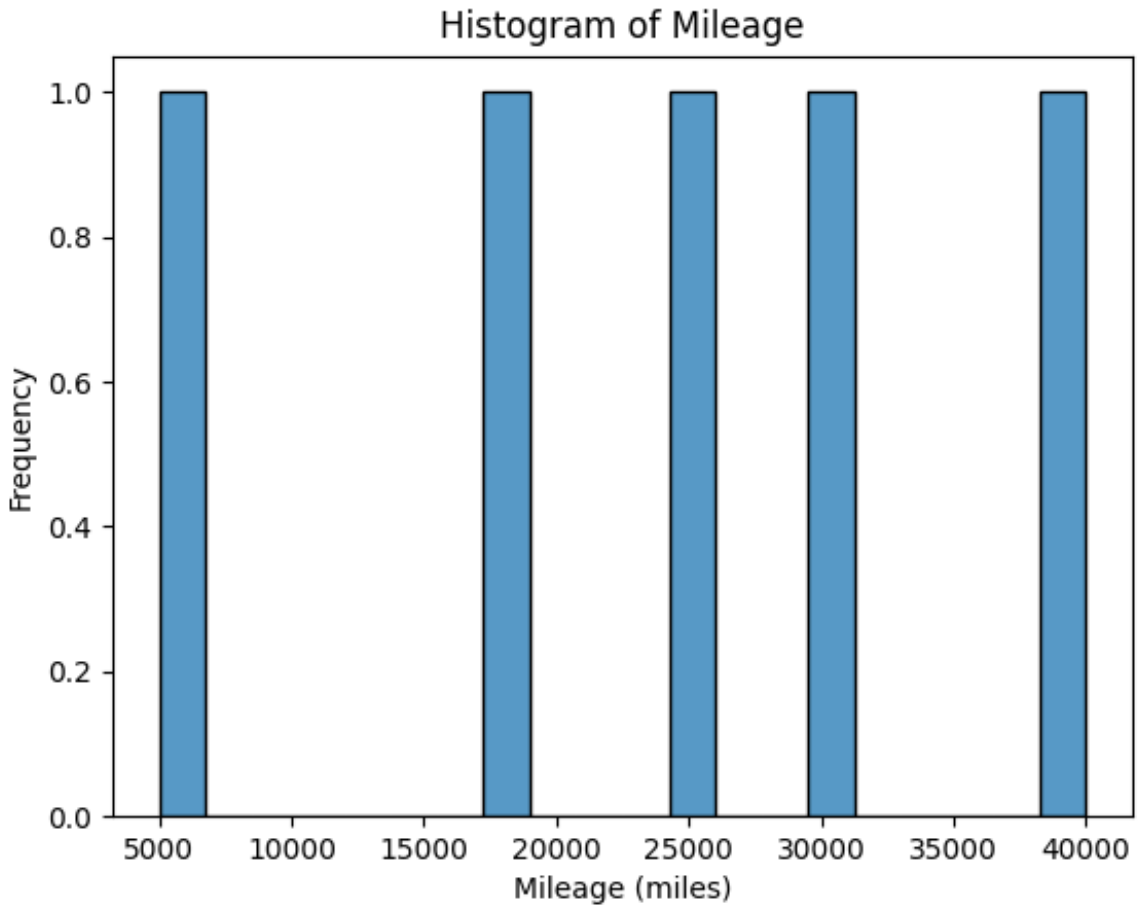
Univariate analysis – Histogram:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the autonomous vehicles dataset
autonomous_vehicles_data =
pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')

# Univariate Analysis - Histogram
print('\nUnivariate Analysis - Histogram')
sns.histplot(autonomous_vehicles_data['Mileage'], bins=20)
plt.title('Histogram of Mileage')
plt.xlabel('Mileage (miles)')
plt.ylabel('Frequency')
plt.show()
plt.savefig('Histogram of Mileage.png')

# Bivariate Analysis - Scatter plot
print('\nBivariate Analysis - Scatter plot')
```

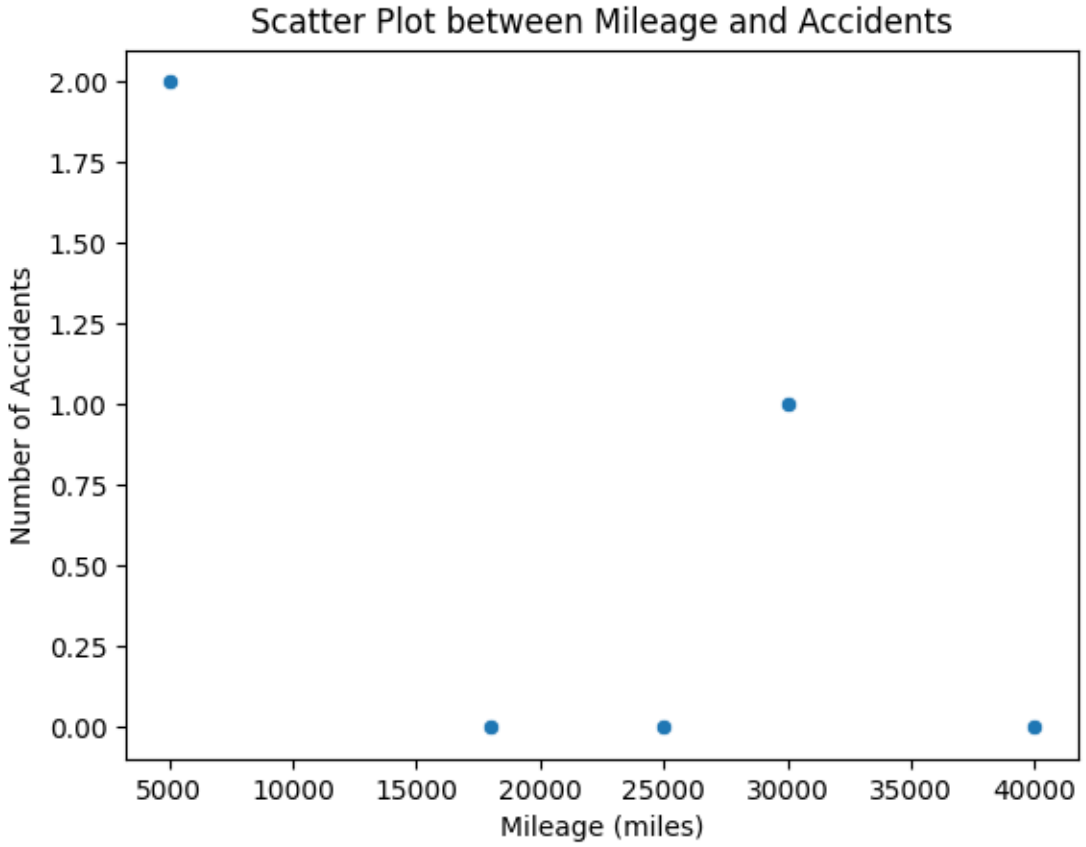
```
sns.scatterplot(x='Mileage', y='Accidents', data=autonomous_vehicles_data)
plt.title('Scatter Plot between Mileage and Accidents')
plt.xlabel('Mileage (miles)')
plt.ylabel('Number of Accidents')
plt.show()
plt.savefig('Scatter Plot between Mileage and Accidents.png')

# Multivariate Analysis - Pair plot
print('\nMultivariate Analysis - Pair plot')
sns.pairplot(autonomous_vehicles_data)
plt.title('Pair Plot of Autonomous Vehicles Data')
plt.show()
plt.savefig('Pair Plot of Autonomous Vehicles Data.png')
```

Output:

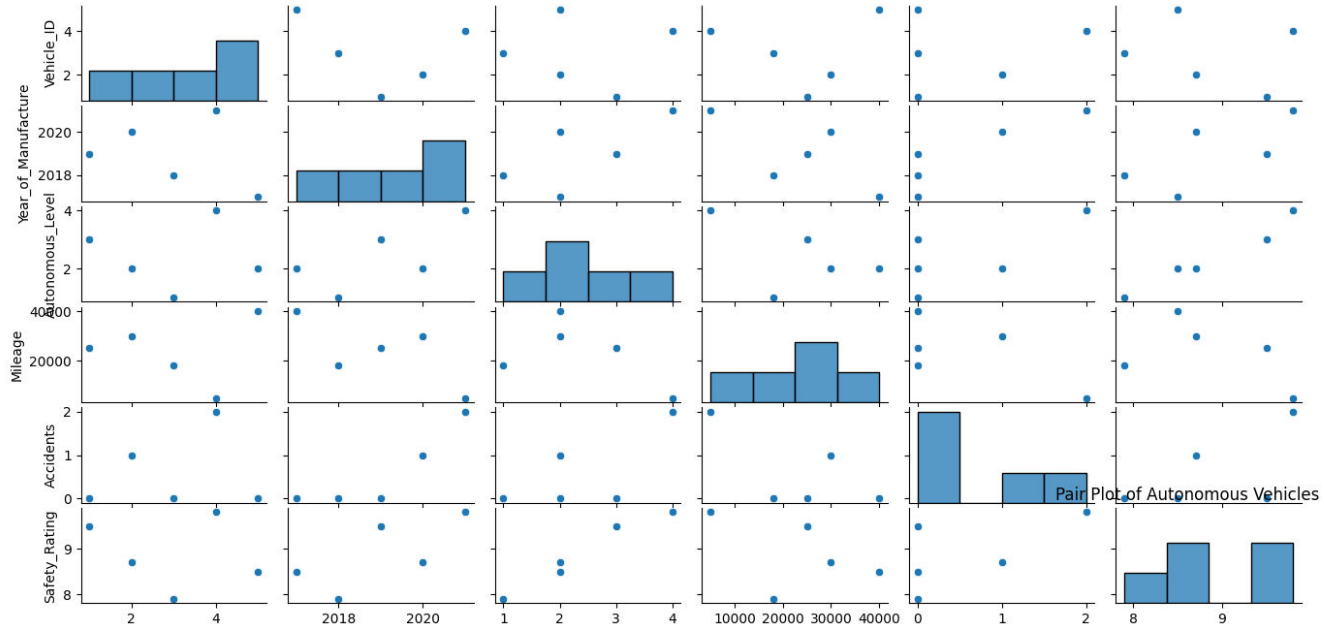
```
Univariate Analysis - Histogram
```



Bivariate Analysis - Scatter plot



Multivariate Analysis - Pair plot



8.Feature Engineering

Creating User Profiles: Aggregating user interaction data to construct comprehensive user profiles capturing preferences and behaviours.

Temporal Analysis: Incorporating temporal features such as time of day or day of week to capture temporal trends in user behaviour.

Content Embeddings: Generating embeddings for content items to represent their characteristics and relationships.

```
import pandas as pd
from datetime import datetime
from gensim.models import Word2Vec

# Load the dataset
autonomous_vehicles_data =
pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')

# Creating User Profiles
user_profiles =
autonomous_vehicles_data.groupby('Vehicle_ID')['Accidents'].sum().reset_in
dex()
user_profiles.rename(columns={'Accidents': 'Total_Accidents'},
inplace=True)
print("User Profiles:")
print(user_profiles)

# Temporal Analysis
current_year = datetime.now().year
autonomous_vehicles_data['Year_Manufacture_Age'] = current_year -
autonomous_vehicles_data['Year_of_Manufacture']
print("\nTemporal Analysis:")
print(autonomous_vehicles_data.head())

# Content Embeddings (Not applicable for this dataset, but let's
demonstrate with a dummy example)
content_data = autonomous_vehicles_data[['Vehicle_Type', 'Manufacturer',
'Autonomous_Level']].copy()

# Generate embeddings for each feature using Word2Vec
embeddings_model = Word2Vec(sentences=content_data.values.tolist(),
vector_size=10, window=5, min_count=1, workers=4)
print("\nContent Embeddings:")
print(embeddings_model)
```

Output:

```
User Profiles:
   Vehicle_ID  Total_Accidents
0           1                0
1           2                1
2           3                0
3           4                2
4           5                0

Temporal Analysis:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating  Year_Manufacture_Age
0                 3    25000          0            9.5                     5
1                 2    30000          1            8.7                     4
2                 1    18000          0            7.9                     6
3                 4     5000          2            9.8                     3
4                 2    40000          0            8.5                     7

Content Embeddings:
Word2Vec<vocab=14, vector_size=10, alpha=0.025>
```

Assumed Scenario:

The project aims to enhance the safety and efficiency of autonomous vehicles by leveraging historical data and advanced analytics techniques.

Objective:

Improve the performance and reliability of autonomous vehicles by utilizing historical interaction data and predictive analytics to anticipate and prevent potential issues.

Target Audience:

Developers, engineers, and stakeholders involved in the development and deployment of autonomous vehicle technologies.

Conclusion:

Phase 2 of the project focuses on data wrangling and analysis to harness the power of historical interaction data in optimizing the performance and safety of autonomous vehicles. By leveraging Python-based data manipulation techniques and predictive analytics, we aim to derive actionable insights that drive advancements in autonomous vehicle technology, ultimately leading to safer and more efficient transportation solutions.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from gensim.models import Word2Vec
#1
df = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-mudhalvan/main/Autonomous_vehicles.csv')
# Display first few rows
print("Head:")
print(df.head())

# Display last few rows
print("\nTail:")
print(df.tail())

# Display information about the dataset
print("\nInfo:")
print(df.info())

# Generate descriptive statistics for numerical features
print("\nDescribe:")
print(df.describe())

#2
df = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-mudhalvan/main/Null_data.csv')
# Display the original DataFrame
print("Original DataFrame:")
print(df)
```

```
# Null Data Identification
print("\nNull Data Identification:")
print(df.isnull())

# Null Data Imputation
print("\nNull Data Imputation:")
# Impute missing values in columns 'Mileage', 'Accidents', and
'Safety_Rating' with mean
df['Mileage'] = df['Mileage'].fillna(df['Mileage'].mean())
df['Accidents'] = df['Accidents'].fillna(df['Accidents'].mean())
df['Safety_Rating'] =
df['Safety_Rating'].fillna(df['Safety_Rating'].mean())
# Impute missing values in columns 'Vehicle_Type', 'Year_of_Manufacture',
'Manufacturer', and 'Autonomous_Level' with specific values
df['Vehicle_Type'] = df['Vehicle_Type'].fillna('Unknown')
df['Year_of_Manufacture'] = df['Year_of_Manufacture'].fillna('Unknown')
df['Manufacturer'] = df['Manufacturer'].fillna('Unknown')
df['Autonomous_Level'] = df['Autonomous_Level'].fillna('Unknown')
print(df)

# Null Data Removal
print("\nNull Data Removal:")
# Remove rows with any missing values
df_cleaned = df.dropna()
print(df_cleaned)

#3
#Import the dataset
df_autonomous_vehicles =
pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')

# Data Integrity Check
print("Data Integrity Check:")

# Check data types of each column
print(df_autonomous_vehicles.dtypes)

# Check for missing values
print(df_autonomous_vehicles.isnull().sum())

# Data Consistency Verification
print("\nData Consistency Verification:")
# Verify consistency of Autonomous_Level column (e.g., ensure values are
within 1-5)
```

```
print("Unique Autonomous Levels:",
df_autonomous_vehicles['Autonomous_Level'].unique())

# Check for consistency between Manufacturer and Vehicle_Type columns
print("\nConsistency between Manufacturer and Vehicle_Type columns:")
print(df_autonomous_vehicles[['Manufacturer', 'Vehicle_Type']])

#4
# Import the dataset
df_autonomous_vehicles =
pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')

# Display the original DataFrame
print("Original DataFrame:")
print(df_autonomous_vehicles)

# Reshaping Rows and Columns
print("\nReshaping Rows and Columns:")
# Transpose the DataFrame
df_transposed = df_autonomous_vehicles.T
print(df_transposed)

# Transposing Data
print("\nTransposing Data:")
# Transpose the DataFrame back to its original shape
df_original = df_transposed.T
print(df_original)

#5
#Import the dataset
df1 = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')
df2 = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles1.csv')

# Display the original DataFrames
print("First DataFrame (df1):")
print(df1)
print("\nSecond DataFrame (df2):")
print(df2)

# Joining Data with specified suffixes
print("\nJoining Data:")
# Join the two DataFrames based on the common column 'Vehicle_ID'
```

```
joined_df = df1.set_index('Vehicle_ID').join(df2.set_index('Vehicle_ID'),
how='outer', lsuffix='_left', rsuffix='_right')
print(joined_df)

#6
df=pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')
# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Grouping Data
print("\nGrouping Data:")
# Group the DataFrame by 'Vehicle_Type'
grouped_df = df.groupby('Vehicle_Type')
for group_name, group_data in grouped_df:
    print("\nGroup:", group_name)
    print(group_data)

# Aggregating Data
print("\nAggregating Data:")
# Compute summary statistics for 'Mileage' grouped by 'Vehicle_Type'
summary_stats = grouped_df['Mileage'].agg(['mean', 'median', 'min',
'max'])
print(summary_stats)

#7
# Import the dataset
df = pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')

# Univariate Analysis
print("\nUnivariate Analysis:")
plt.figure(figsize=(8, 6))
sns.histplot(df['Mileage'], bins=10, kde=True)
plt.title('Distribution of Mileage')
plt.xlabel('Mileage (miles)')
plt.ylabel('Frequency')
plt.show()
plt.savefig('mileage_distribution.png')
# Bivariate Analysis
print("\nBivariate Analysis:")
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Mileage', y='Accidents', data=df)
plt.title('Relationship between Mileage and Accidents')
```

```
plt.xlabel('Mileage (miles)')
plt.ylabel('Number of Accidents')
plt.show()
plt.savefig('mileage_accidents_relationship.png')
# Multivariate Analysis
print("\nMultivariate Analysis:")
plt.figure(figsize=(8, 6))
numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns
correlation_matrix = df[numeric_columns].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
plt.savefig('correlation_heatmap.png')

#8
autonomous_vehicles_data =
pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')

# Univariate Analysis - Histogram
print('\nUnivariate Analysis - Histogram')
sns.histplot(autonomous_vehicles_data['Mileage'], bins=20)
plt.title('Histogram of Mileage')
plt.xlabel('Mileage (miles)')
plt.ylabel('Frequency')
plt.show()
plt.savefig('Histogram of Mileage.png')

# Bivariate Analysis - Scatter plot
print('\nBivariate Analysis - Scatter plot')
sns.scatterplot(x='Mileage', y='Accidents', data=autonomous_vehicles_data)
plt.title('Scatter Plot between Mileage and Accidents')
plt.xlabel('Mileage (miles)')
plt.ylabel('Number of Accidents')
plt.show()
plt.savefig('Scatter Plot between Mileage and Accidents.png')

# Multivariate Analysis - Pair plot
print('\nMultivariate Analysis - Pair plot')
sns.pairplot(autonomous_vehicles_data)
plt.title('Pair Plot of Autonomous Vehicles Data')
plt.show()
plt.savefig('Pair Plot of Autonomous Vehicles Data.png')

#9
```

```
# Load the dataset
autonomous_vehicles_data =
pd.read_csv('https://raw.githubusercontent.com/Tech-master1234/Naan-
mudhalvan/main/Autonomous_vehicles.csv')

# Creating User Profiles
user_profiles =
autonomous_vehicles_data.groupby('Vehicle_ID')['Accidents'].sum().reset_in
dex()
user_profiles.rename(columns={'Accidents': 'Total_Accidents'},
inplace=True)
print("User Profiles:")
print(user_profiles)

# Temporal Analysis
current_year = datetime.now().year
autonomous_vehicles_data['Year_Manufacture_Age'] = current_year -
autonomous_vehicles_data['Year_of_Manufacture']
print("\nTemporal Analysis:")
print(autonomous_vehicles_data.head())

# Content Embeddings (Not applicable for this dataset, but let's
demonstrate with a dummy example)
content_data = autonomous_vehicles_data[['Vehicle_Type', 'Manufacturer',
'Autonomous_Level']].copy()

# Generate embeddings for each feature using Word2Vec
embeddings_model = Word2Vec(sentences=content_data.values.tolist(),
vector_size=10, window=5, min_count=1, workers=4)
print("\nContent Embeddings:")
print(embeddings_model)
```

Output:

```
Head:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5

Tail:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5

Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Vehicle_ID           5 non-null      int64
 1   Vehicle_Type         5 non-null      object
 2   Year_of_Manufacture  5 non-null      int64
 3   Manufacturer         5 non-null      object
 4   Autonomous_Level     5 non-null      int64
 5   Mileage              5 non-null      int64
 6   Accidents            5 non-null      int64
 7   Safety_Rating        5 non-null      float64
```

```
dtypes: float64(1), int64(5), object(2)
memory usage: 448.0+ bytes
None

Describe:
       Vehicle_ID  Year_of_Manufacture  Autonomous_Level       Mileage  \
count    5.000000             5.000000          5.000000      5.000000
mean     3.000000          2019.000000          2.400000  23600.000000
std      1.581139             1.581139          1.140175  13126.309458
min      1.000000          2017.000000          1.000000   5000.000000
25%      2.000000          2018.000000          2.000000  18000.000000
50%      3.000000          2019.000000          2.000000  25000.000000
75%      4.000000          2020.000000          3.000000  30000.000000
max      5.000000          2021.000000          4.000000  40000.000000

       Accidents  Safety_Rating
count   5.000000       5.000000
mean    0.600000       8.880000
std     0.894427       0.769415
min     0.000000       7.900000
25%     0.000000       8.500000
50%     0.000000       8.700000
75%     1.000000       9.500000
max     2.000000       9.800000
Original DataFrame:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1          SUV               2018.0       Toyota
1           2          NaN                  NaN          NaN
2           3        Truck               2020.0         Ford
3           4          NaN                  NaN          NaN
4           5     Electric               2019.0        Tesla

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0               3.0  25000.0        0.0            9.5
1               NaN      NaN        1.0            NaN
2               NaN      NaN        0.0            7.9
3               NaN  18000.0        NaN            NaN
4               4.0  40000.0        0.0            8.5
```

Null Data Identification:

	Vehicle_ID	Vehicle_Type	Year_of_Manufacture	Manufacturer
0	False	False	False	False
1	False	True	True	True
2	False	False	False	False
3	False	True	True	True
4	False	False	False	False

	Autonomous_Level	Mileage	Accidents	Safety_Rating
0	False	False	False	False
1	True	True	False	True
2	True	True	False	False
3	True	False	True	True
4	False	False	False	False

Null Data Imputation:

	Vehicle_ID	Vehicle_Type	Year_of_Manufacture	Manufacturer	Autonomous_Level
0	1	SUV	2018.0	Toyota	3.0
1	2	Unknown	Unknown	Unknown	Unknown
2	3	Truck	2020.0	Ford	Unknown
3	4	Unknown	Unknown	Unknown	Unknown
4	5	Electric	2019.0	Tesla	4.0

	Mileage	Accidents	Safety_Rating
0	25000.000000	0.00	9.500000
1	27666.666667	1.00	8.633333
2	27666.666667	0.00	7.900000
3	18000.000000	0.25	8.633333
4	40000.000000	0.00	8.500000

Null Data Removal:

	Vehicle_ID	Vehicle_Type	Year_of_Manufacture	Manufacturer	Autonomous_Level
0	1	SUV	2018.0	Toyota	3.0
1	2	Unknown	Unknown	Unknown	Unknown
2	3	Truck	2020.0	Ford	Unknown
3	4	Unknown	Unknown	Unknown	Unknown
4	5	Electric	2019.0	Tesla	4.0

	Mileage	Accidents	Safety_Rating
0	25000.000000	0.00	9.500000
1	27666.666667	1.00	8.633333
2	27666.666667	0.00	7.900000
3	18000.000000	0.25	8.633333
4	40000.000000	0.00	8.500000

```
Data Integrity Check:
Vehicle_ID               int64
Vehicle_Type            object
Year_of_Manufacture      int64
Manufacturer            object
Autonomous_Level         int64
Mileage                  int64
Accidents                int64
Safety_Rating          float64
dtype: object
Vehicle_ID             0
Vehicle_Type           0
Year_of_Manufacture    0
Manufacturer           0
Autonomous_Level       0
Mileage                0
Accidents              0
Safety_Rating          0
dtype: int64

Data Consistency Verification:
Unique Autonomous Levels: [3 2 1 4]

Consistency between Manufacturer and Vehicle_Type columns:
  Manufacturer Vehicle_Type
0        Tesla        Sedan
1       Toyota          SUV
2         Ford        Truck
3       Google     Electric
4       Nissan          Van
Original DataFrame:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan
```

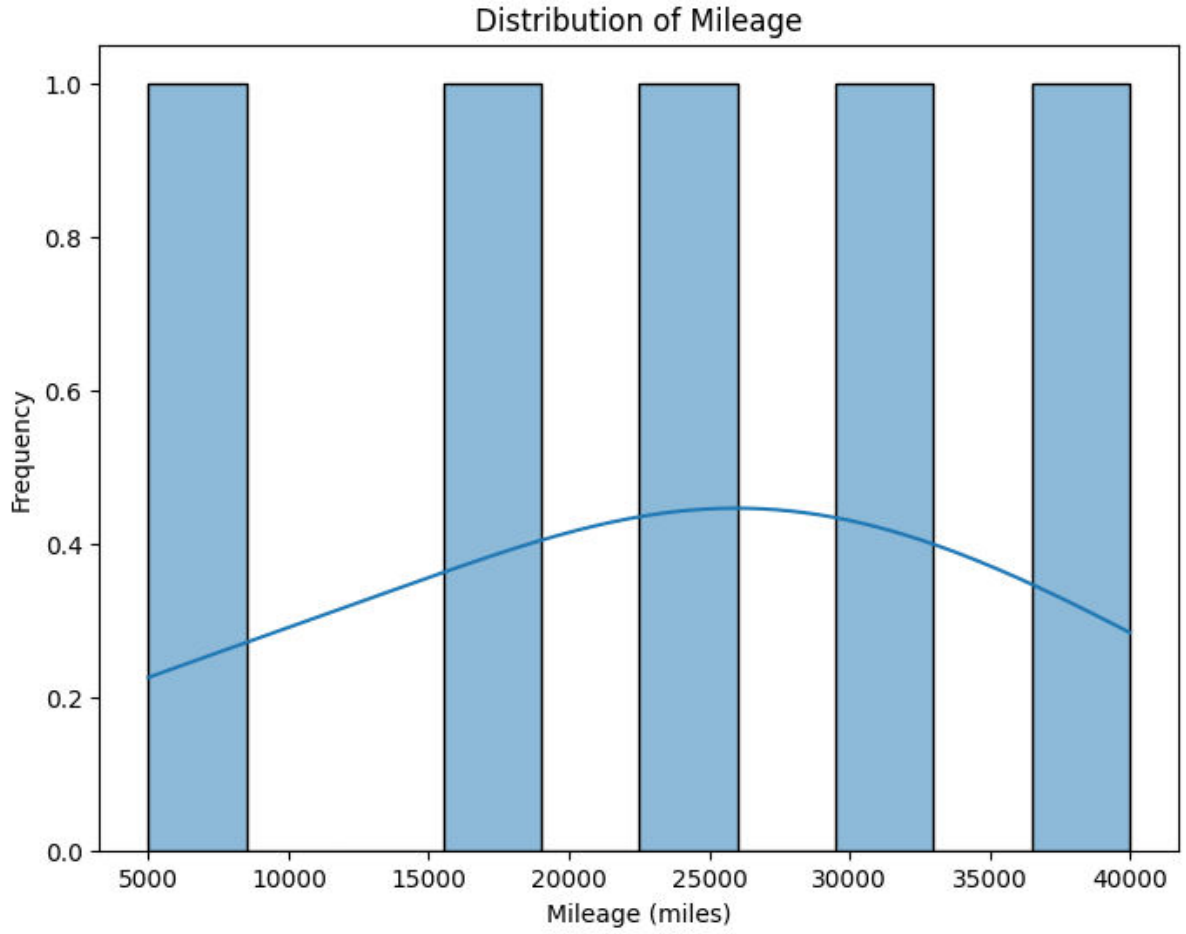
```
   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5

Reshaping Rows and Columns:
                         0       1      2         3       4
Vehicle_ID               1       2      3         4       5
Vehicle_Type         Sedan     SUV  Truck  Electric     Van
Year_of_Manufacture   2019    2020   2018      2021    2017
Manufacturer         Tesla  Toyota   Ford    Google  Nissan
Autonomous_Level         3       2      1         4       2
Mileage              25000   30000  18000      5000   40000
Accidents                0       1      0         2       0
Safety_Rating          9.5     8.7    7.9       9.8     8.5

Transposing Data:
  Vehicle_ID Vehicle_Type Year_of_Manufacture Manufacturer Autonomous_Level  \
0          1        Sedan                2019        Tesla                3
1          2          SUV                2020       Toyota                2
2          3        Truck                2018         Ford                1
3          4     Electric                2021       Google                4
4          5          Van                2017       Nissan                2

  Mileage Accidents Safety_Rating
0   25000         0           9.5
1   30000         1           8.7
2   18000         0           7.9
3    5000         2           9.8
4   40000         0           8.5
First DataFrame (df1):
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan
```

```
   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5

Second DataFrame (df2):
   Vehicle_ID  Service_Cost Warranty_Expiry
0           1           200      2023-05-15
1           2           150      2024-01-10
2           3           300      2022-11-20
3           6           250      2023-09-30
4           7           180      2023-06-25

Joining Data:
           Vehicle_Type  Year_of_Manufacture Manufacturer  Autonomous_Level  \
Vehicle_ID
1                 Sedan               2019.0        Tesla               3.0
2                   SUV               2020.0       Toyota               2.0
3                 Truck               2018.0         Ford               1.0
4              Electric               2021.0       Google               4.0
5                   Van               2017.0       Nissan               2.0
6                   NaN                  NaN          NaN               NaN
7                   NaN                  NaN          NaN               NaN

            Mileage  Accidents  Safety_Rating  Service_Cost Warranty_Expiry
Vehicle_ID
1           25000.0        0.0            9.5         200.0      2023-05-15
2           30000.0        1.0            8.7         150.0      2024-01-10
3           18000.0        0.0            7.9         300.0      2022-11-20
4            5000.0        2.0            9.8           NaN             NaN
5           40000.0        0.0            8.5           NaN             NaN
6               NaN        NaN            NaN         250.0      2023-09-30
7               NaN        NaN            NaN         180.0      2023-06-25
Original DataFrame:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan
```

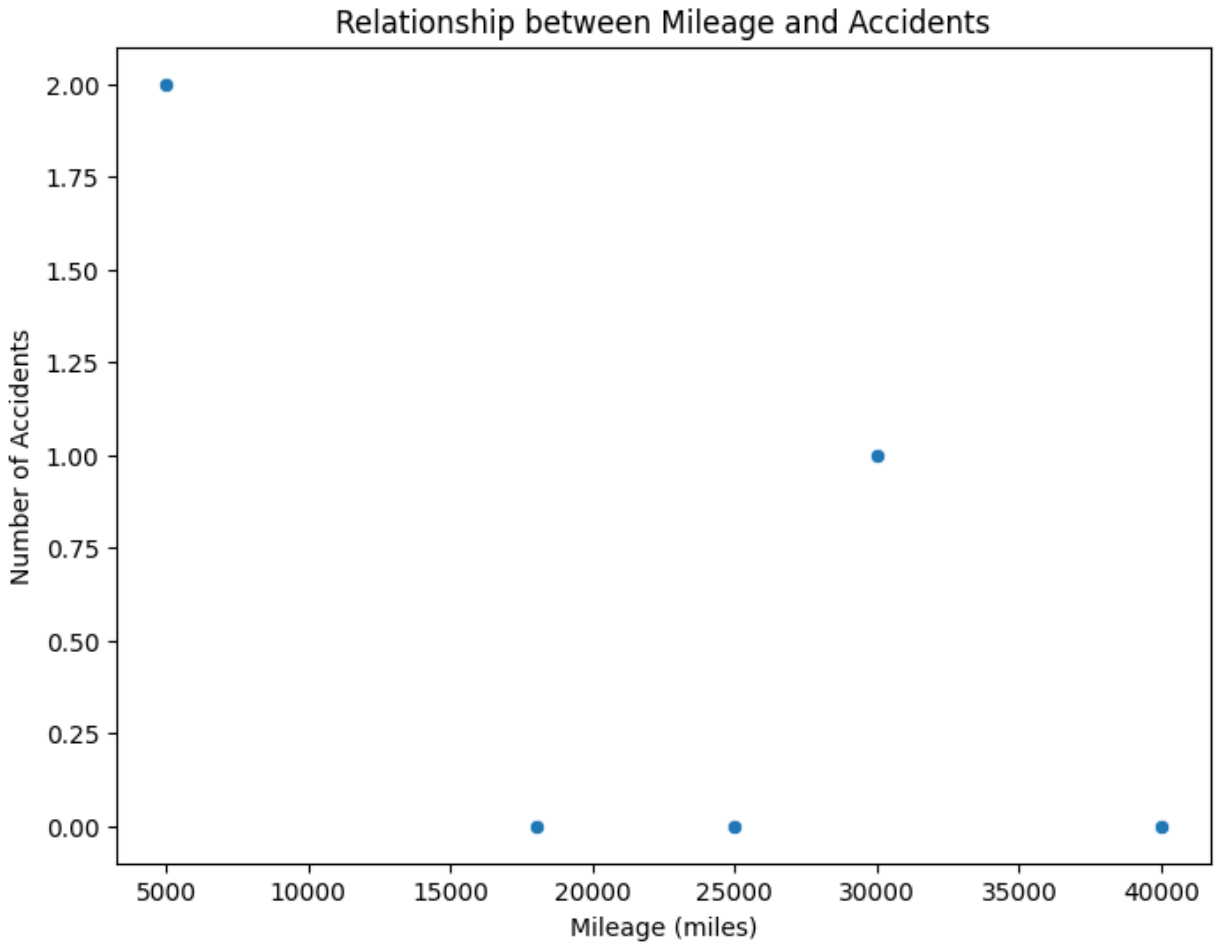
```
   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5
1                 2    30000          1            8.7
2                 1    18000          0            7.9
3                 4     5000          2            9.8
4                 2    40000          0            8.5

Grouping Data:

Group: Electric
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
3           4     Electric                 2021       Google

   Autonomous_Level  Mileage  Accidents  Safety_Rating
3                 4     5000          2            9.8

Group: SUV
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
1           2          SUV                 2020       Toyota

   Autonomous_Level  Mileage  Accidents  Safety_Rating
1                 2    30000          1            8.7

Group: Sedan
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla

   Autonomous_Level  Mileage  Accidents  Safety_Rating
0                 3    25000          0            9.5

Group: Truck
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
2           3        Truck                 2018         Ford

   Autonomous_Level  Mileage  Accidents  Safety_Rating
2                 1    18000          0            7.9
```

```
Group: Van
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating
4                 2    40000          0            8.5

Aggregating Data:
                mean   median    min    max
Vehicle_Type
Electric       5000.0   5000.0   5000   5000
SUV           30000.0  30000.0  30000  30000
Sedan         25000.0  25000.0  25000  25000
Truck         18000.0  18000.0  18000  18000
Van           40000.0  40000.0  40000  40000
```

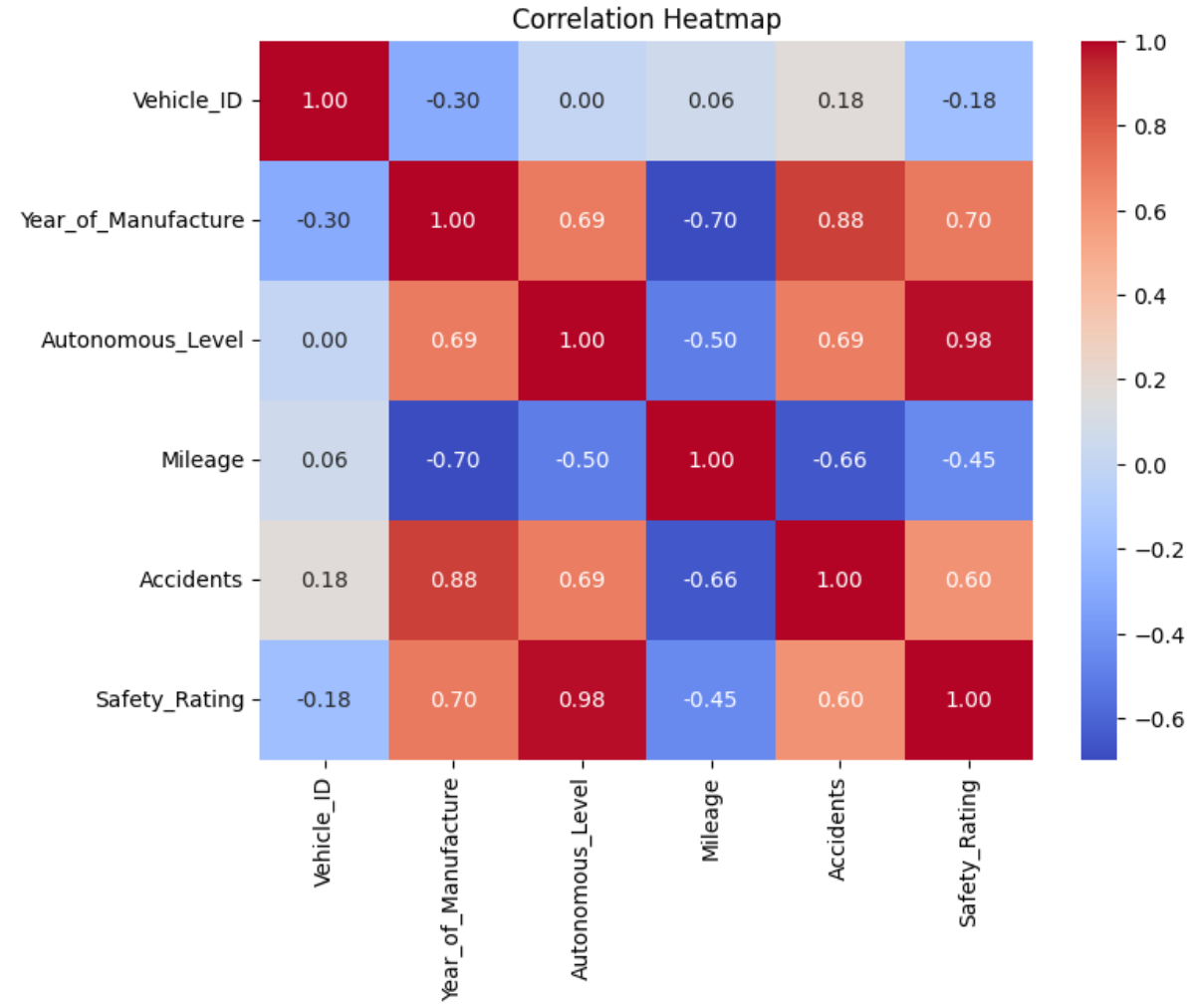
Univariate Analysis:



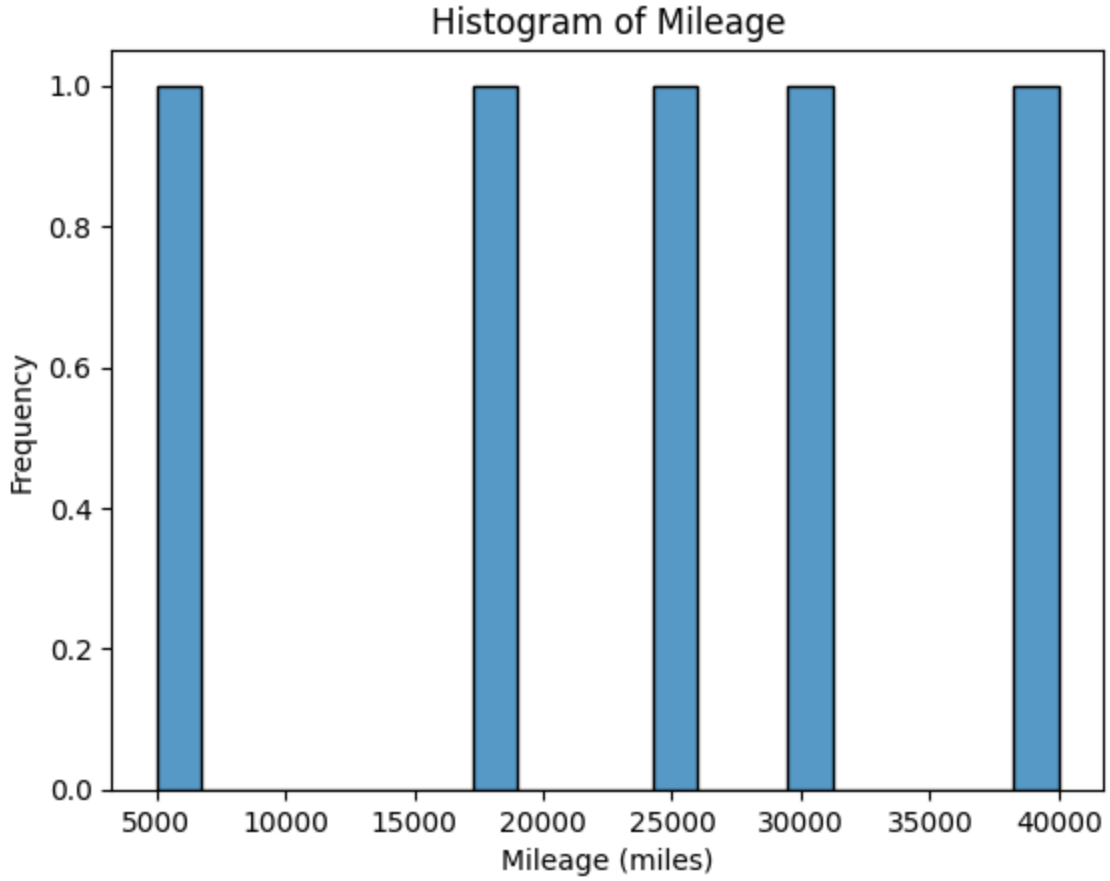
```
Bivariate Analysis:
<Figure size 640x480 with 0 Axes>
```



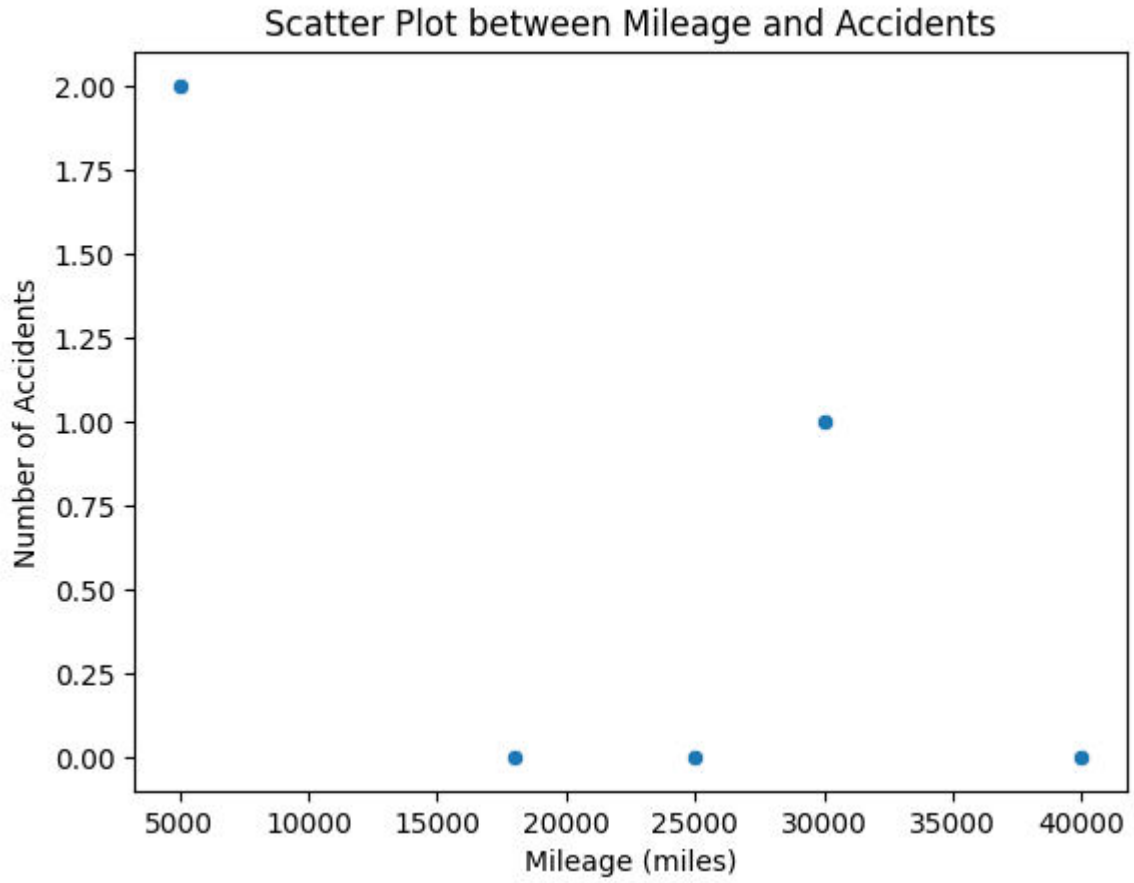
```
Multivariate Analysis:
<Figure size 640x480 with 0 Axes>
```



Univariate Analysis - Histogram

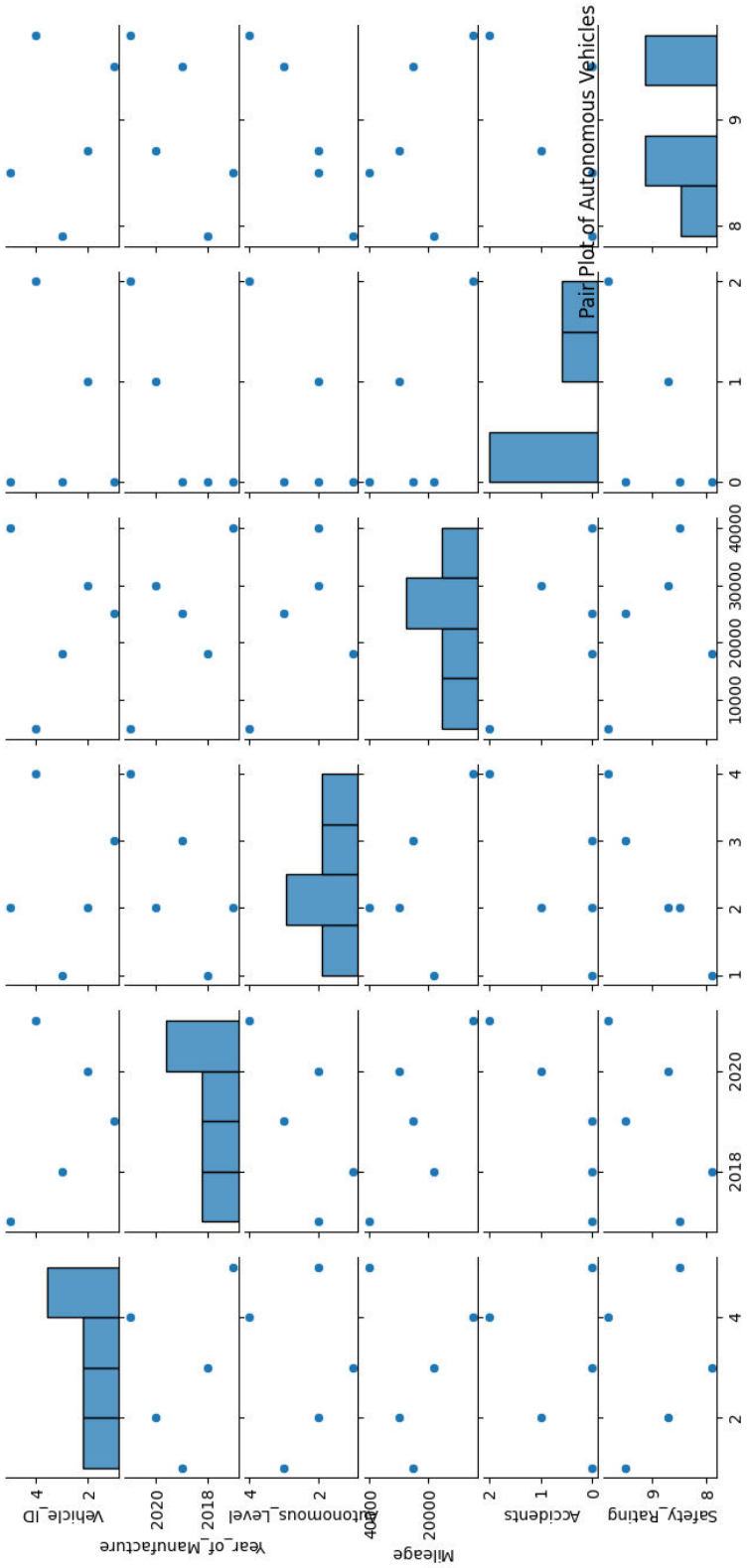


```
Bivariate Analysis - Scatter plot
```



Multivariate Analysis - Pair plot

<Figure size 640x480 with 0 Axes>



```
User Profiles:
   Vehicle_ID  Total_Accidents
0           1                0
1           2                1
2           3                0
3           4                2
4           5                0

Temporal Analysis:
   Vehicle_ID Vehicle_Type  Year_of_Manufacture Manufacturer  \
0           1        Sedan                 2019        Tesla
1           2          SUV                 2020       Toyota
2           3        Truck                 2018         Ford
3           4     Electric                 2021       Google
4           5          Van                 2017       Nissan

   Autonomous_Level  Mileage  Accidents  Safety_Rating  Year_Manufacture_Age
0                 3    25000          0            9.5                     5
1                 2    30000          1            8.7                     4
2                 1    18000          0            7.9                     6
3                 4     5000          2            9.8                     3
4                 2    40000          0            8.5                     7

Content Embeddings:
Word2Vec<vocab=14, vector_size=10, alpha=0.025>
<Figure size 640x480 with 0 Axes>
```