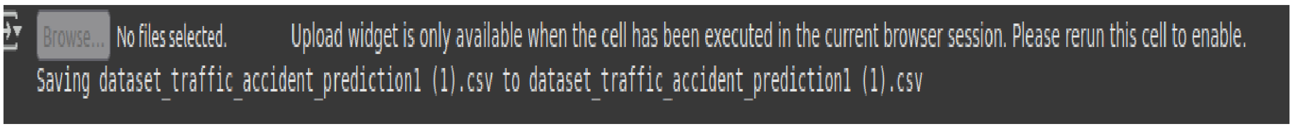
1.)UPLOADING FILES

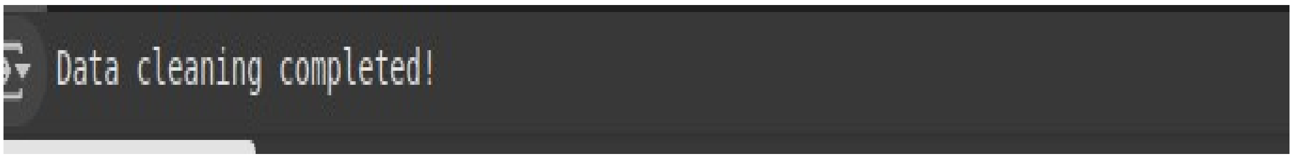
```
from google.colab import files
uploaded = files.upload()
```



```
Browse... No files selected.    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving dataset_traffic_accident_prediction1 (1).csv to dataset_traffic_accident_prediction1 (1).csv
```

2.)DATA CLEANING

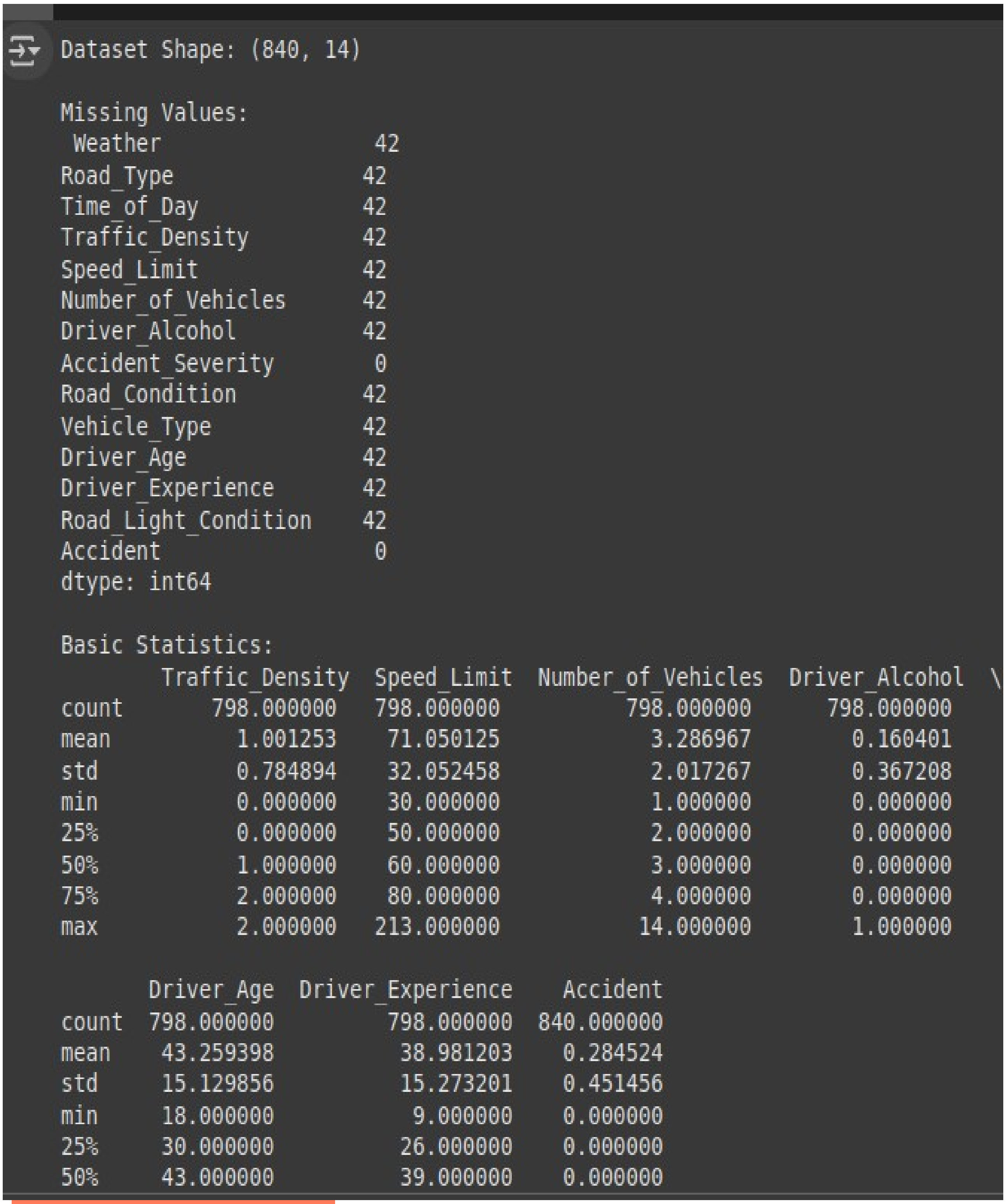
```
import pandas as pd
df = pd.read_csv('dataset_traffic_accident_prediction1 (1).csv')
# Handle missing values
df['Accident'] = df['Accident'].fillna(0)
df['Accident_Severity'] = df['Accident_Severity'].fillna('Unknown')
# Convert to proper data types
numeric_cols = ['Traffic_Density', 'Speed_Limit', 'Number_of_Vehicles',
'Driver_Age', 'Driver_Experience']
df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric, errors='coerce')
df.to_csv('cleaned_dataset.csv', index=False)
print("Data cleaning completed!")
```



```
Data cleaning completed!
```

3.)MISSING DATAVALUES

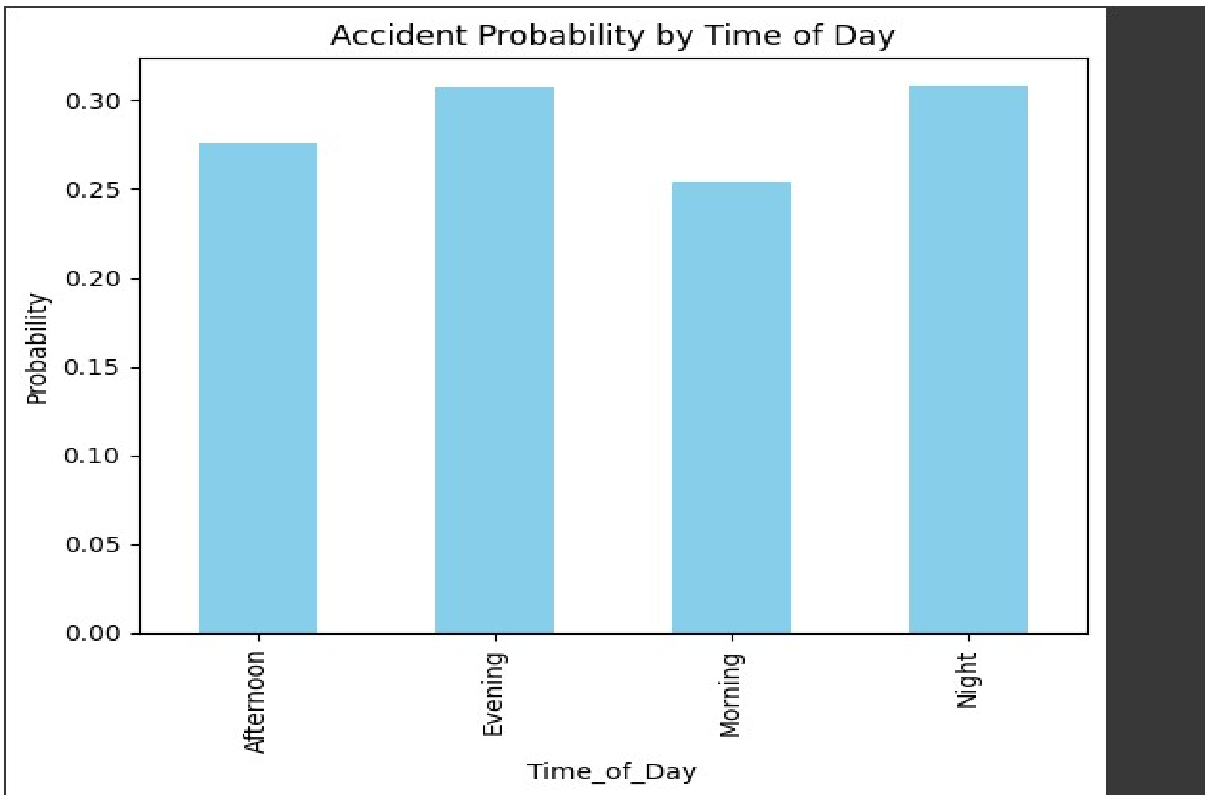
```
import pandas as pd
import seaborn as sns
df = pd.read_csv('cleaned_dataset.csv')
print("Dataset Shape:", df.shape)
print("\nMissing Values:\n", df.isnull().sum())
print("\nBasic Statistics:\n", df.describe())
# Accident distribution
sns.countplot(x='Accident', data=df)
plt.title('Accident Distribution')
plt.show()
```



```
Dataset Shape: (840, 14)

Missing Values:
 Weather              42
Road_Type            42
Time_of_Day          42
Traffic_Density      42
Speed_Limit          42
Number_of_Vehicles   42
Driver_Alcohol       42
Accident_Severity     0
Road_Condition       42
Vehicle_Type         42
Driver_Age           42
Driver_Experience    42
Road_Light_Condition 42
Accident              0
dtype: int64

Basic Statistics:
        Traffic_Density  Speed_Limit  Number_of_Vehicles  Driver_Alcohol  \
count        798.000000   798.000000          798.000000      798.000000
mean           1.001253    71.050125            3.286967        0.160401
std            0.784894    32.052458            2.017267        0.367208
min            0.000000    30.000000            1.000000        0.000000
25%            0.000000    50.000000            2.000000        0.000000
50%            1.000000    60.000000            3.000000        0.000000
75%            2.000000    80.000000            4.000000        0.000000
max            2.000000   213.000000           14.000000        1.000000

       Driver_Age  Driver_Experience    Accident
count  798.000000         798.000000  840.000000
mean    43.259398          38.981203    0.284524
std     15.129856          15.273201    0.451456
min     18.000000           9.000000    0.000000
25%     30.000000          26.000000    0.000000
50%     43.000000          39.000000    0.000000
```

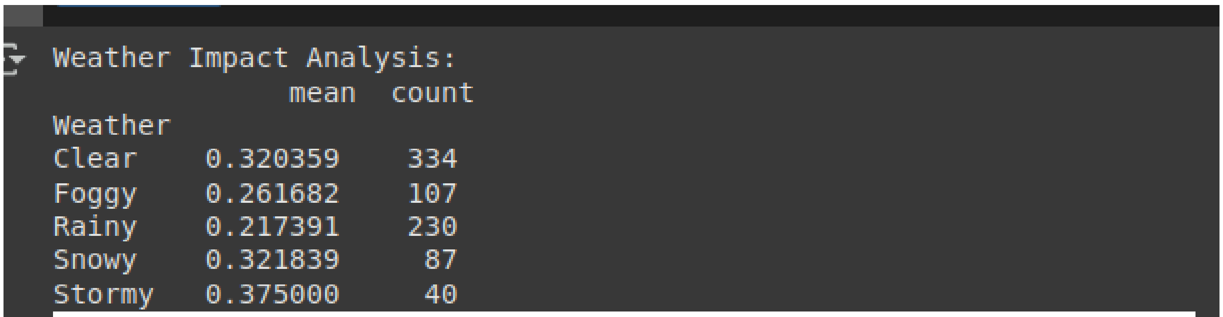
4.)ACCIDENT PROBABILITY

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('cleaned_dataset.csv')
time_accidents = df.groupby('Time_of_Day')['Accident'].mean()
time_accidents.plot(kind='bar', color='skyblue')
plt.title('Accident Probability by Time of Day')
plt.ylabel('Probability')
plt.show()
```

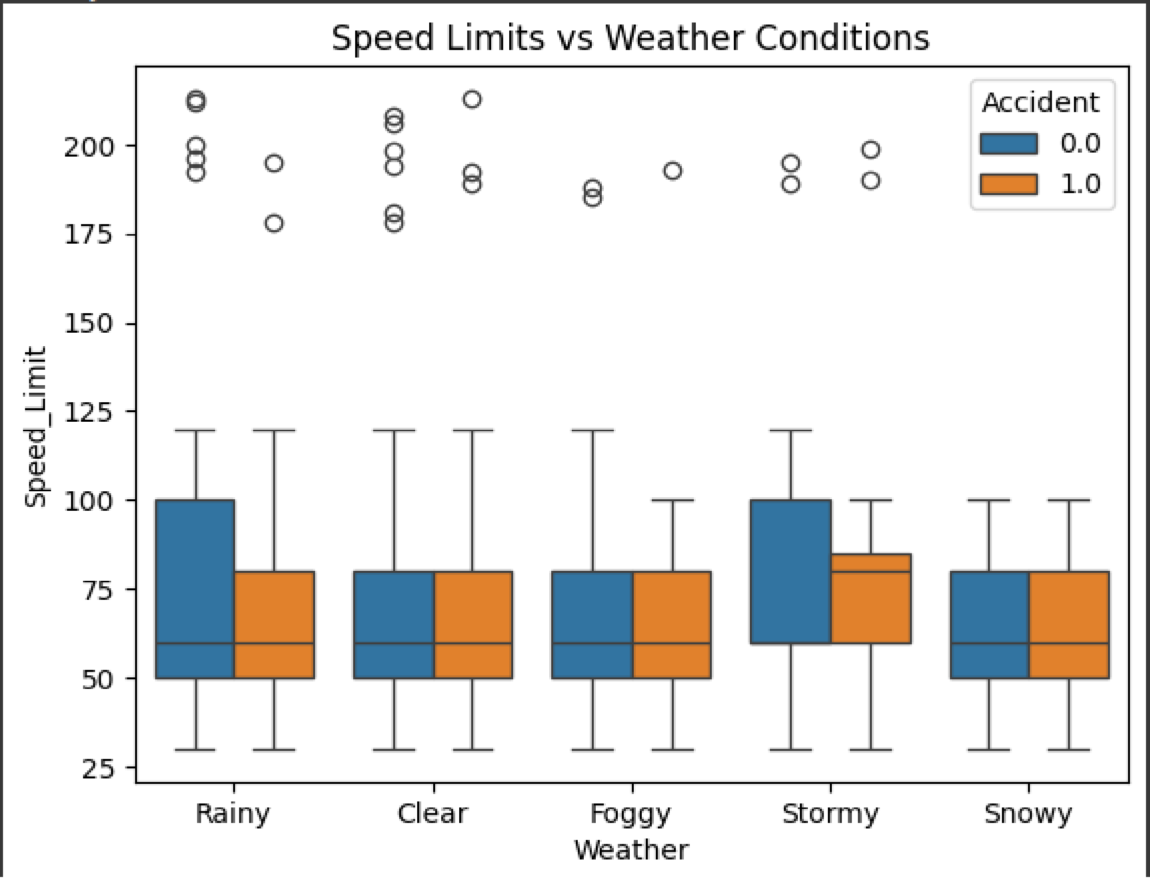


5.)WEATHER IMPORT ANALYSIS

```
import pandas as pd
import seaborn as sns
df = pd.read_csv('cleaned_dataset.csv')
weather_impact = df.groupby('Weather')['Accident'].agg(['mean', 'count'])
print("Weather Impact Analysis:\n", weather_impact)
sns.boxplot(x='Weather', y='Speed_Limit', hue='Accident', data=df)
plt.title('Speed Limits vs Weather Conditions')
plt.show()
```

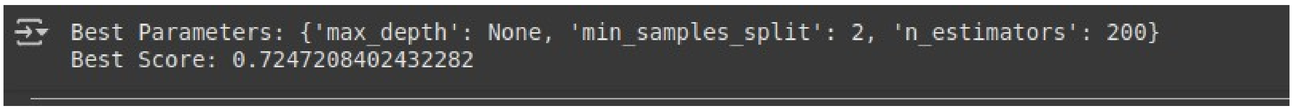


```
Weather Impact Analysis:
                mean  count
Weather
Clear       0.320359    334
Foggy       0.261682    107
Rainy       0.217391    230
Snowy       0.321839     87
Stormy      0.375000     40
```



6.)RANDOM FOREST CLASSIFIER

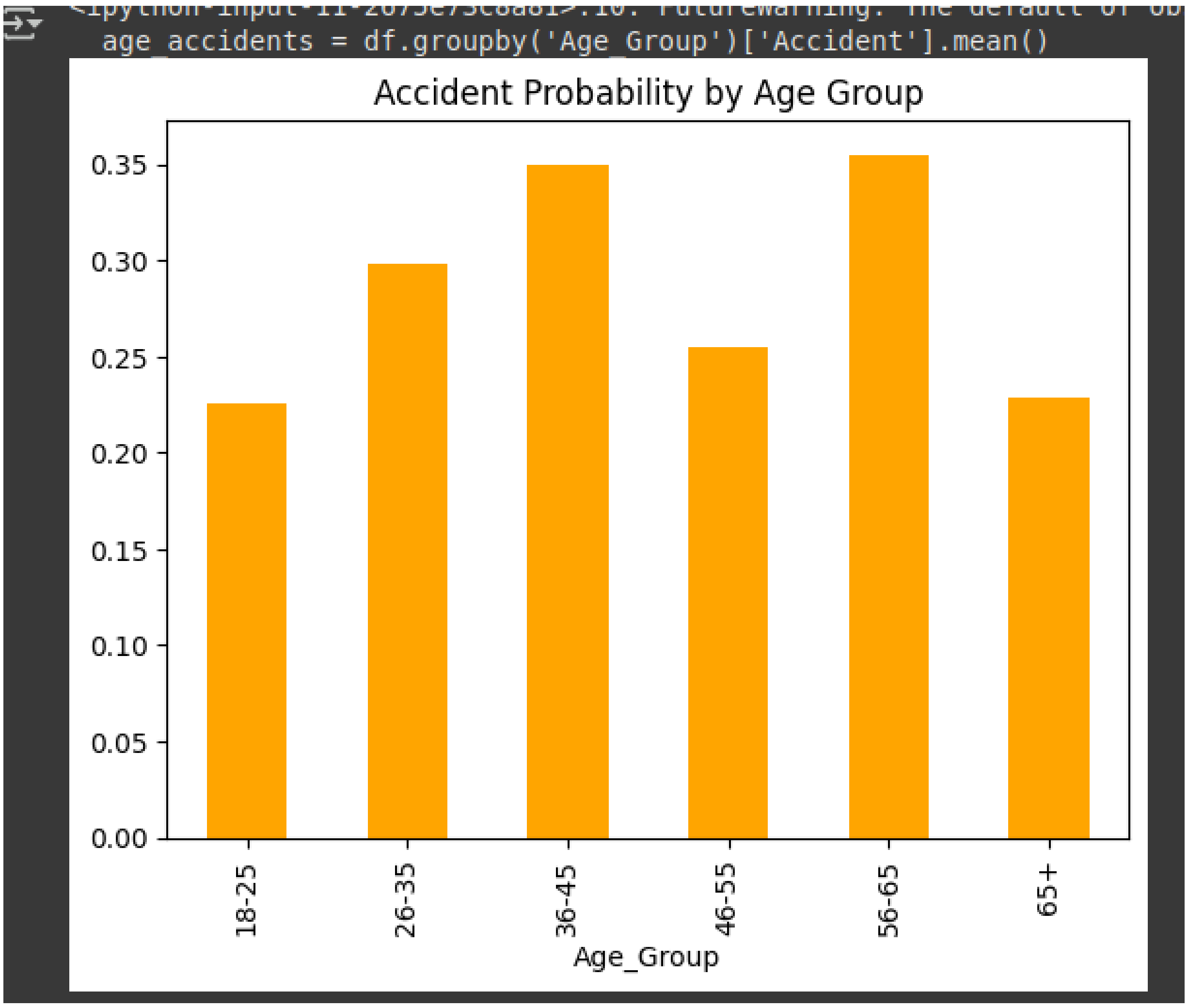
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
param_grid = {
'n_estimators': [100, 200],
'max_depth': [None, 10, 20],
'min_samples_split': [2, 5]
}
model = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
model.fit(X_train, y_train)
print("Best Parameters:", model.best_params_)
print("Best Score:", model.best_score_)
```



```
Best Parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 200}
Best Score: 0.7247208402432282
```

7.)ACCIDENT PROBABILITY BY AGE GROUP

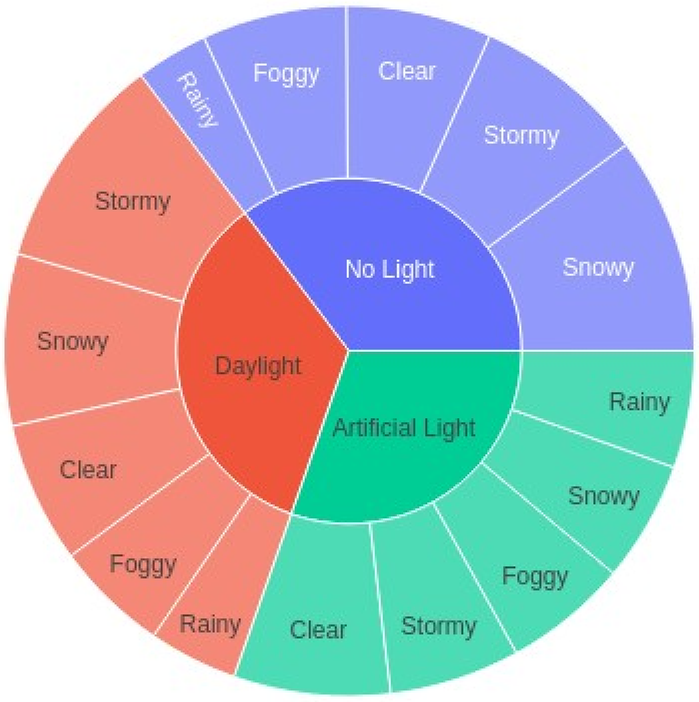
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('cleaned_dataset.csv')
bins = [18, 25, 35, 45, 55, 65, 100]
labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '65+']
df['Age_Group'] = pd.cut(df['Driver_Age'], bins=bins, labels=labels)
age_accidents = df.groupby('Age_Group')['Accident'].mean()
age_accidents.plot(kind='bar', color='orange')
plt.title('Accident Probability by Age Group')
plt.show()
```



```
<ipython-input-11-2075e75c8a81>:10: FutureWarning: The default of ob
  age_accidents = df.groupby('Age_Group')['Accident'].mean()
```

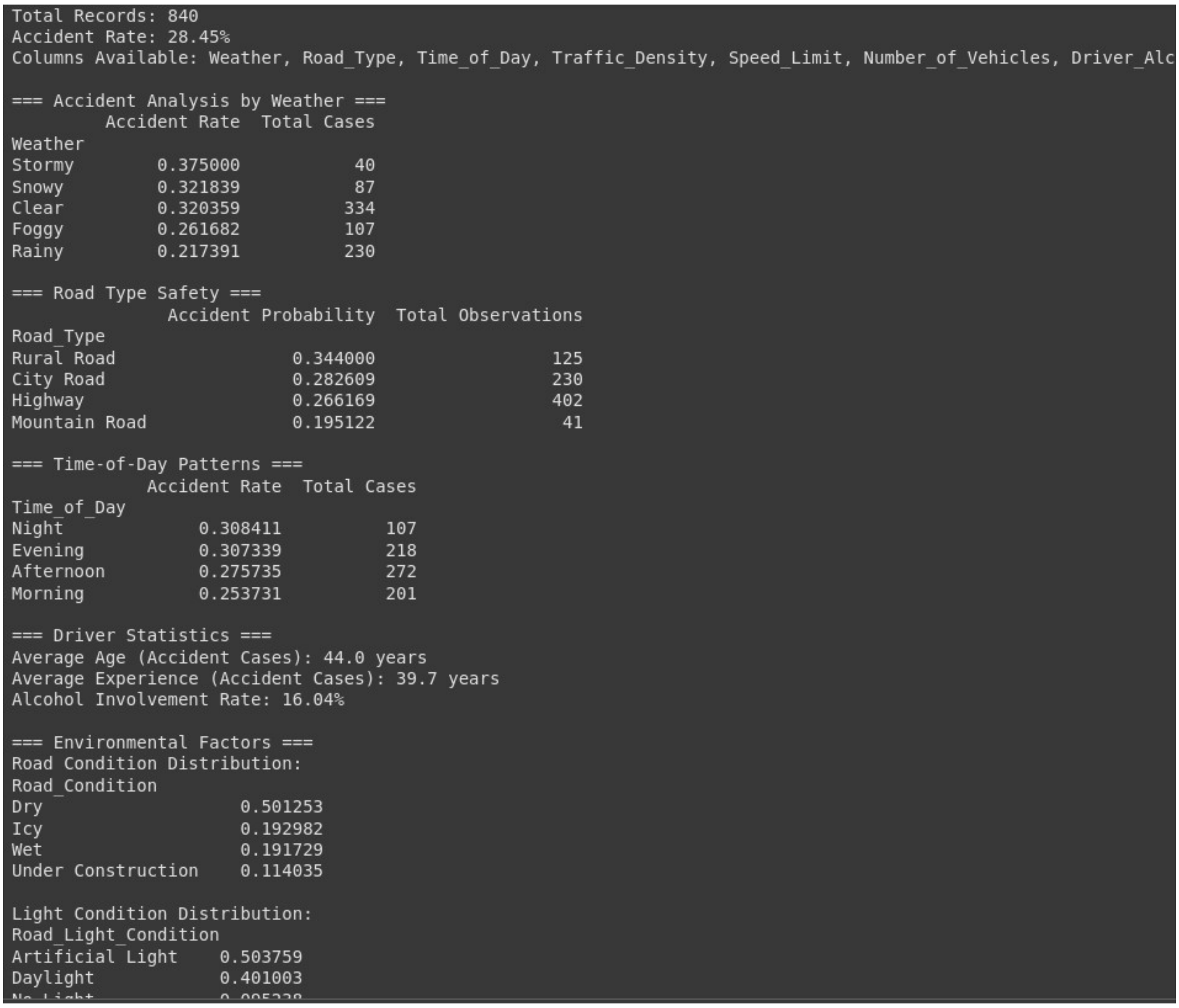
8.) ROAD LIGHT CONDITION

```
import pandas as pd
import plotly.express as px

# Preprocessing
light_analysis = df.groupby(['Road_Light_Condition', 'Weather'])
['Accident'].mean().reset_index()

# Interactive visualization
fig = px.sunburst(light_analysis,
path=['Road_Light_Condition', 'Weather'],
values='Accident',
title='Accident Probability by Lighting and Weather Conditions')
fig.show()
```



9.)TEXT BASED ANALYSIS

```
import pandas as pd
def text_based_analysis():
# Load dataset
df = pd.read_csv('dataset_traffic_accident_prediction1 (1).csv')

# Clean data
df['Accident'] = df['Accident'].fillna(0).astype(int)
df['Accident_Severity'] = df['Accident_Severity'].fillna('Unknown')

# Basic statistics
print("=== Dataset Overview ===")
print(f"Total Records: {len(df)}")
print(f"Accident Rate: {df['Accident'].mean():.2%}")
print(f"Columns Available: {', '.join(df.columns)}")

# Accident analysis by weather
print("\n=== Accident Analysis by Weather ===")
weather_stats = df.groupby('Weather')['Accident'].agg(['mean', 'count'])
weather_stats.columns = ['Accident Rate', 'Total Cases']
print(weather_stats.sort_values('Accident Rate',
ascending=False).to_string())

# Road type analysis
print("\n=== Road Type Safety ===")
road_stats = df.groupby('Road_Type')['Accident'].agg(['mean', 'count'])
road_stats.columns = ['Accident Probability', 'Total Observations']
print(road_stats.sort_values('Accident Probability',
ascending=False).to_string())

# Time of day patterns
print("\n=== Time-of-Day Patterns ===")
time_stats = df.groupby('Time_of_Day')['Accident'].agg(['mean', 'count'])
time_stats.columns = ['Accident Rate', 'Total Cases']
print(time_stats.sort_values('Accident Rate', ascending=False).to_string())

# Driver statistics
print("\n=== Driver Statistics ===")
print(f"Average Age (Accident Cases): {df[df['Accident'] == 1]
['Driver_Age'].mean():.1f} years")
print(f"Average Experience (Accident Cases): {df[df['Accident'] == 1]
['Driver_Experience'].mean():.1f} years")
print(f"Alcohol Involvement Rate: {df['Driver_Alcohol'].mean():.2%}")

# Environmental factors
print("\n=== Environmental Factors ===")
print("Road Condition Distribution:")
print(df['Road_Condition'].value_counts(normalize=True).to_string())
print("\nLight Condition Distribution:")
print(df['Road_Light_Condition'].value_counts(normalize=True).to_string())

# Numerical correlations
print("\n=== Numerical Feature Correlations ===")
numeric_cols = ['Traffic_Density', 'Speed_Limit', 'Number_of_Vehicles',
'Driver_Age', 'Driver_Experience']
corr_matrix = df[numeric_cols + ['Accident']].corr()
['Accident'].sort_values(ascending=False)
print("Correlation with Accident Probability:")
print(corr_matrix.to_string())

if __name__ == "__main__":
text_based_analysis()
```



```
Total Records: 840
Accident Rate: 28.45%
Columns Available: Weather, Road_Type, Time_of_Day, Traffic_Density, Speed_Limit, Number_of_Vehicles, Driver_Alc

=== Accident Analysis by Weather ===
         Accident Rate  Total Cases
Weather
Stormy        0.375000           40
Snowy         0.321839           87
Clear         0.320359          334
Foggy         0.261682          107
Rainy         0.217391          230

=== Road Type Safety ===
               Accident Probability  Total Observations
Road_Type
Rural Road                 0.344000                 125
City Road                  0.282609                 230
Highway                    0.266169                 402
Mountain Road              0.195122                  41

=== Time-of-Day Patterns ===
             Accident Rate  Total Cases
Time_of_Day
Night             0.308411          107
Evening           0.307339          218
Afternoon         0.275735          272
Morning           0.253731          201

=== Driver Statistics ===
Average Age (Accident Cases): 44.0 years
Average Experience (Accident Cases): 39.7 years
Alcohol Involvement Rate: 16.04%

=== Environmental Factors ===
Road Condition Distribution:
Road_Condition
Dry                   0.501253
Icy                   0.192982
Wet                   0.191729
Under Construction    0.114035

Light Condition Distribution:
Road_Light_Condition
Artificial Light    0.503759
Daylight            0.401003
```