Data Assessment – 1

Tasks:

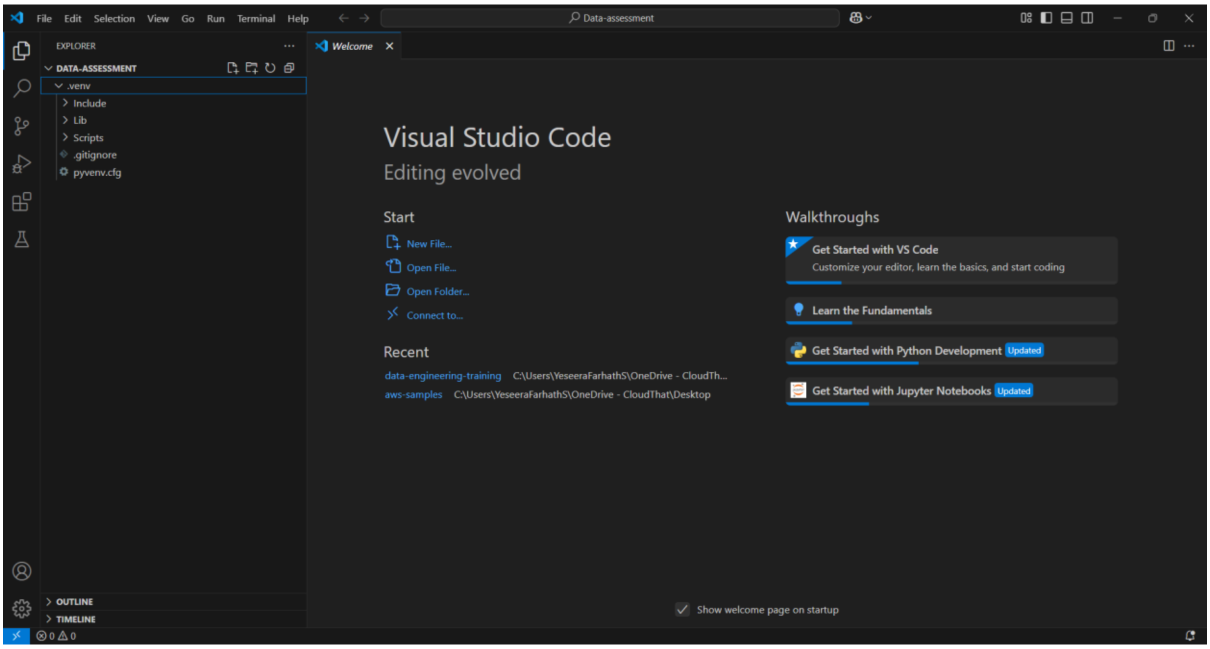
Day 1: Setup and Data Collection

1. **Team Setup**
   - Set up virtual environments for Python dependencies
   - Create a new GitHub repository for the project
   - Install required Python libraries (NumPy, Pandas, Matplotlib, scikit-learn)
2. **Dataset Selection**: Choose any from one dataset from data.gov.in
3. **Environment Setup**
   - Use VSCode Mandatory
   - Create project structure (data, notebooks, scripts, output folders)
   - Set up Jupyter Notebooks

Solution:

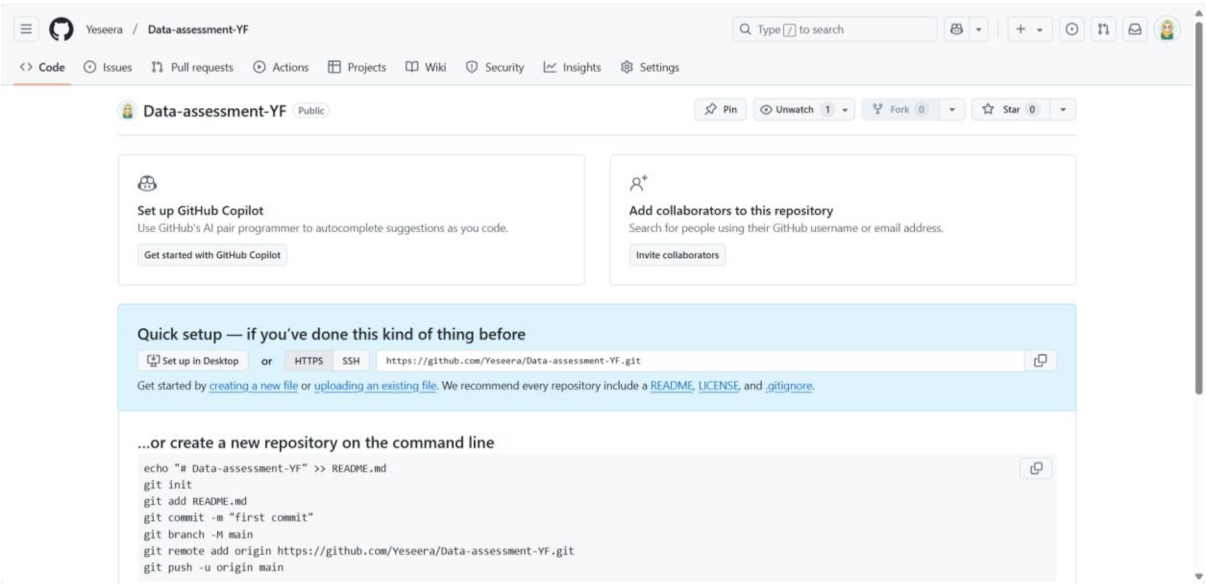
I) Team Setup:

1. **Set up virtual environments for Python dependencies**

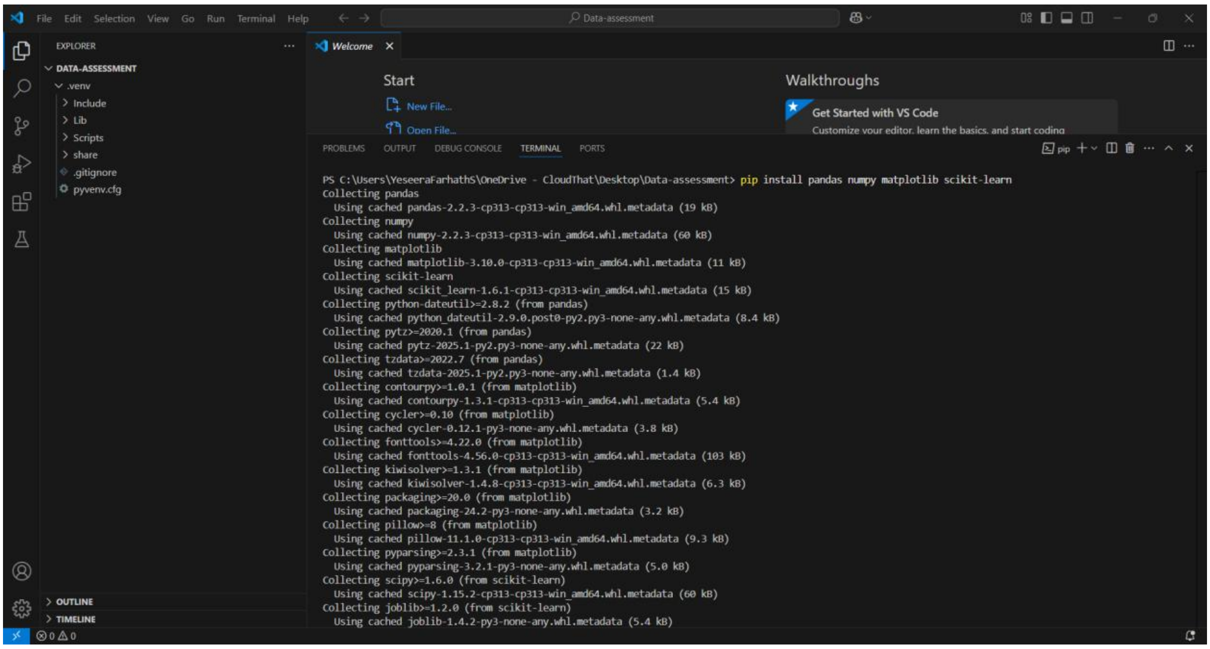


- Created a new folder
- Created a python virtual environment.

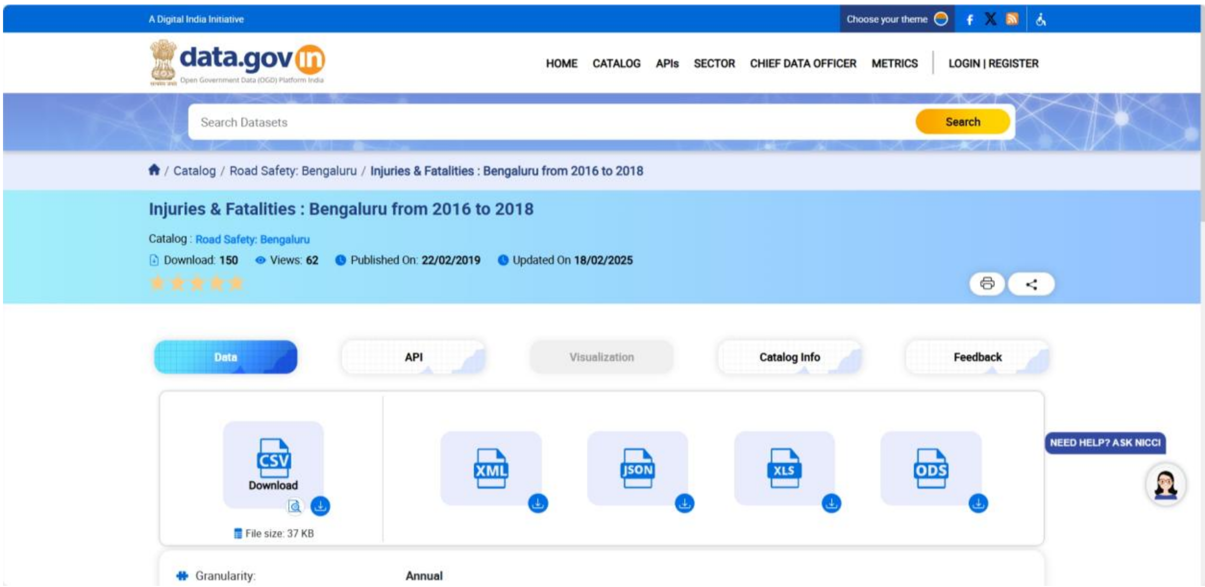
2. **Create a new GitHub repository for the project**



3. Install required Python libraries (NumPy, Pandas, Matplotlib, scikit-learn)



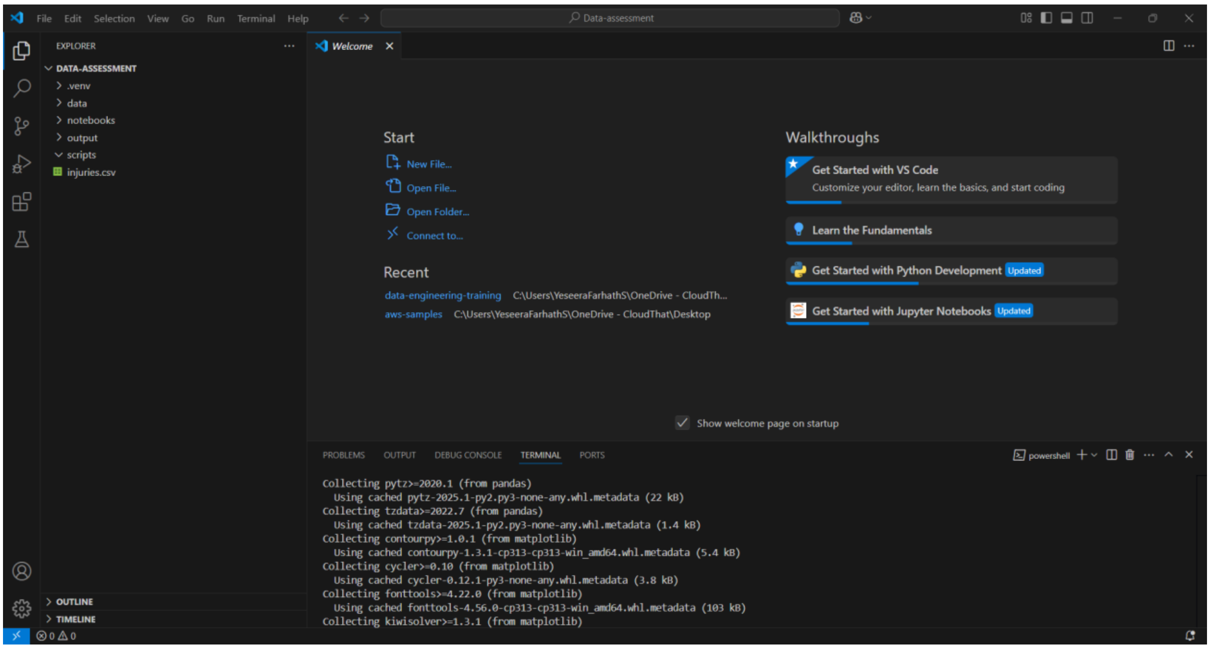
II) Dataset Selection: Choose any from one dataset from data.gov.in



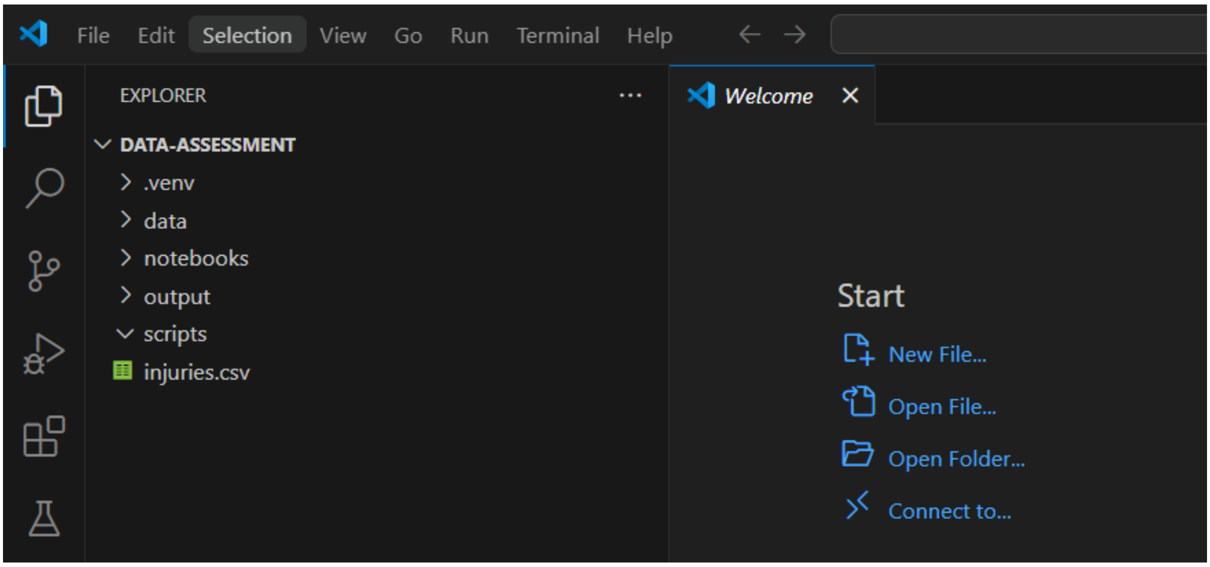
- Selected Injuries and fatalities dataset

III) Environment Setup

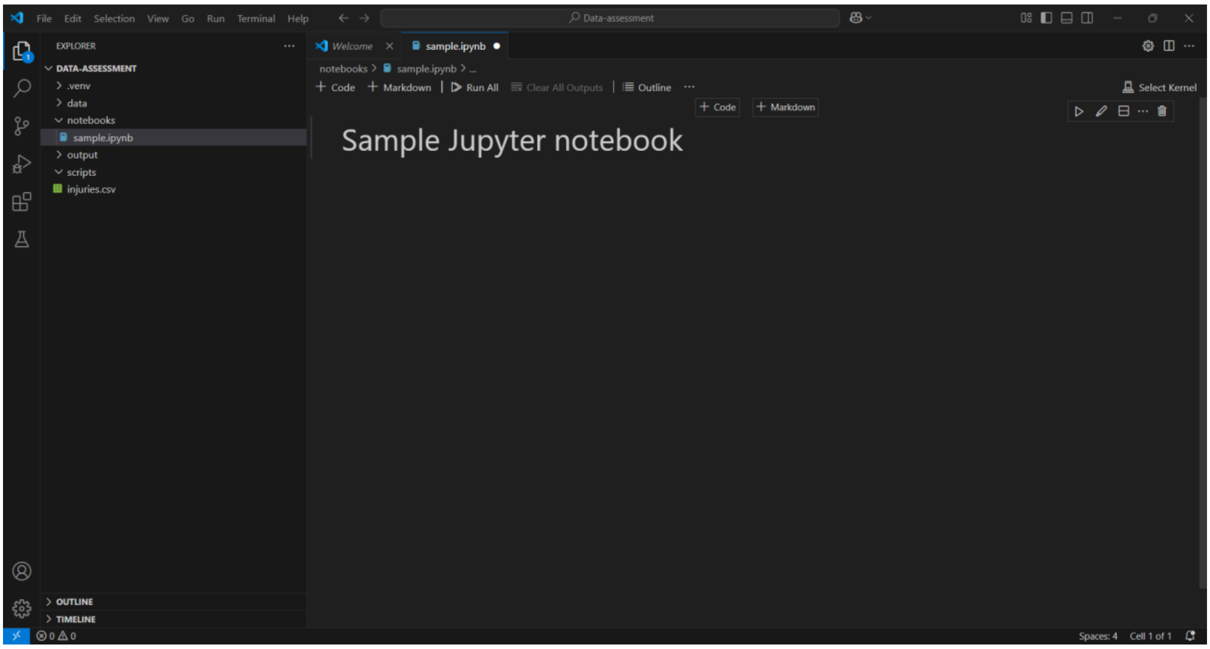
1. Use VSCode Mandatory



2. Create project structure (data, notebooks, scripts, output folders)



3. Set up Jupyter Notebooks



Day 2: Data Processing and Analysis

- 1. **Data Processing**
- o Use Pandas to:
- § Clean the dataset (handle missing values, format dates)
- § Create derived features relevant to the chosen dataset
- § Aggregate data for meaningful analysis
- 2. **Exploratory Analysis**
- o Use Jupyter Notebooks to:
- § Generate descriptive statistics
- § Create visualizations using Matplotlib
- § Document key findings and insights
- § Explore relationships between variables

Solution:

1. Import required packages:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2] ✓ 7.2s

2. Import the dataset (injuries.csv)

```
df = pd.read_csv('injuries.csv')
df.info()
```

[29] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 963 entries, 0 to 962
Data columns (total 32 columns):
 #   Column                                                                  Non-Null Count  Dtype
---  ------                                                                  --------------  -----
 0   City Name                                                               1 non-null      object
 1   2018 - Total Injuries - Pedestrian                                      1 non-null      float64
 2   2018 - Total Injuries - Bicycles                                        1 non-null      float64
 3   2018 - Total Injuries - Two-wheelers                                    1 non-null      float64
 4   2018 - Total Injuries - Other modes of road transport (auto, bus, lorry)   1 non-null      float64
 5   2018 - Total Injuries                                                   1 non-null      float64
 6   2018 - Total Fatalities - Pedestrian                                    1 non-null      float64
 7   2018 - Total Fatalities - Bicycles                                      1 non-null      float64
 8   2018 - Total Fatalities - Two-wheelers                                  1 non-null      float64
 9   2018 - Total Fatalities - Other modes of road transport (auto, bus, lorry) 1 non-null      float64
 10  2018 - Total Fatalities                                                 1 non-null      float64
 11  2017 - Total Injuries - Pedestrian                                      1 non-null      float64
 12  2017 - Total Injuries - Bicycles                                        1 non-null      float64
 13  2017 - Total Injuries - Two-wheelers                                    1 non-null      float64
 14  2017 - Total Injuries - Other modes of road transport (auto, bus, lorry)   1 non-null      float64
 15  2017 - Total Injuries                                                   1 non-null      float64
 16  2017 - Total Fatalities - Pedestrian                                    1 non-null      float64
 17  2017 - Total Fatalities - Bicycles                                      1 non-null      float64
 18  2017 - Total Fatalities - Two-wheelers                                  1 non-null      float64
 19  2017 - Total Fatalities - Other modes of road transport (auto, bus, lorry) 1 non-null      float64
...
 30  2016 - Total Fatalities.1                                               1 non-null      float64
 31  Unnamed: 31                                                             0 non-null      float64
dtypes: float64(31), object(1)
memory usage: 240.9+ KB
```

- Using Toyota.csv dataset, as injuries.csv is almost null.

```
df = pd.read_csv('Toyota.csv')
df
```

[3] ✓ 0.0s

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500	NaN	20544	Petrol	86	1.0	0	1300	3	1025
1432	1432	10845	72.0	??	Petrol	86	0.0	0	1300	3	1015
1433	1433	8500	NaN	17016	Petrol	86	0.0	0	1300	3	1015
1434	1434	7250	70.0	??	NaN	86	1.0	0	1300	3	1015
1435	1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 11 columns

3. Get information about the dataset

```
df.info()
[4] ✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1436 entries, 0 to 1435
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  1436 non-null   int64
 1   Price       1436 non-null   int64
 2   Age         1336 non-null   float64
 3   KM          1436 non-null   object
 4   FuelType    1336 non-null   object
 5   HP          1436 non-null   object
 6   MetColor    1286 non-null   float64
 7   Automatic   1436 non-null   int64
 8   CC          1436 non-null   int64
 9   Doors       1436 non-null   object
 10  Weight      1436 non-null   int64
dtypes: float64(2), int64(5), object(4)
memory usage: 123.5+ KB
```

4. Dropping the unwanted column named 'unnamed'

```
df.drop(columns=['Unnamed: 0'], inplace=True, errors='ignore')
[5] ✓ 0.0s

df.columns
[6] ✓ 0.0s

Index(['Price', 'Age', 'KM', 'FuelType', 'HP', 'MetColor', 'Automatic', 'CC',
       'Doors', 'Weight'],
      dtype='object')
```

5. Check for null values

```
df.isnull().sum()
[7] ✓ 0.0s

Price          0
Age          100
KM             0
FuelType     100
HP             0
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

6. Clean the data (missing values, wrong data types, and question mark)

```
# Replace ?? with NaN
df.replace("??", np.nan, inplace=True)

# Convert data types
df['KM'] = df['KM'].str.replace(',', '').astype(float)
df['HP'] = pd.to_numeric(df['HP'], errors='coerce')
df['Doors'] = df['Doors'].replace({'three': 3, 'four': 4, 'five': 5}).astype(float)

#  missing values
df.fillna({
    'Age': df['Age'].median(),
    'KM': df['KM'].median(),
    'HP': df['HP'].median(),
    'FuelType': df['FuelType'].mode()[0],
    'MetColor': df['MetColor'].mode()[0],
    'Doors': df['Doors'].mode()[0]
}, inplace=True)
[8] ✓ 0.0s
```

```
df.isnull().sum()
[9] ✓ 0.0s

Price        0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

7. Plot graphs of clean data

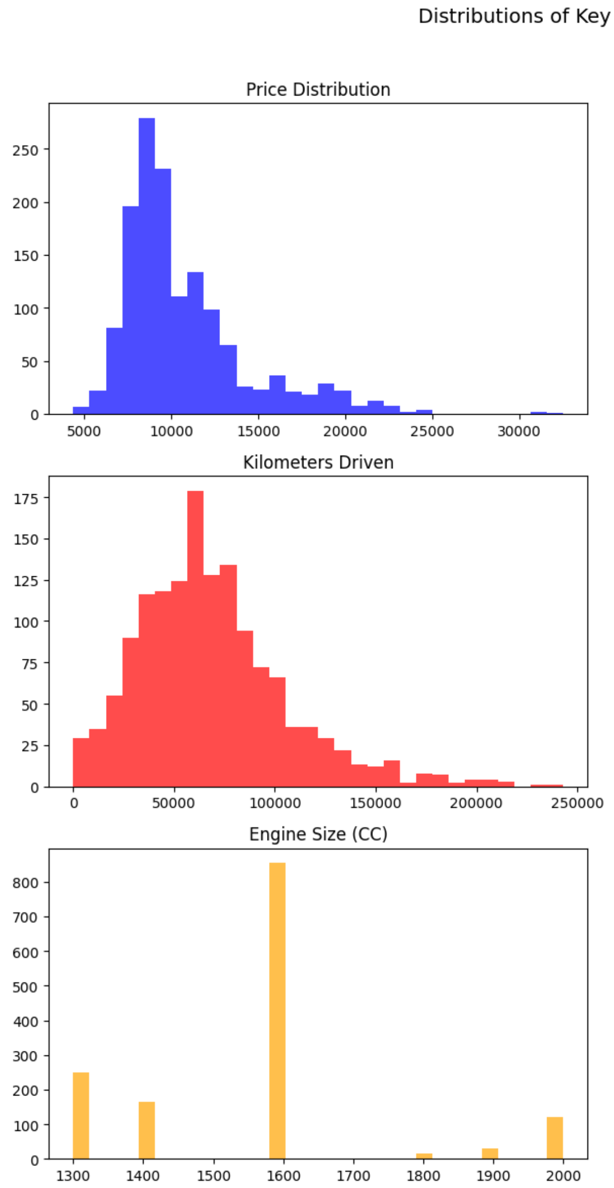
```
# Plot distributions of numeric features
fig, axes = plt.subplots(3, 2, figsize=(12, 12))
fig.suptitle("Distributions of Key Numeric Features", fontsize=14)

axes[0, 0].hist(df['Price'], bins=30, color='blue', alpha=0.7)
axes[0, 0].set_title("Price Distribution")
axes[0, 1].hist(df['Age'], bins=30, color='green', alpha=0.7)
axes[0, 1].set_title("Age Distribution")
axes[1, 0].hist(df['KM'], bins=30, color='red', alpha=0.7)
axes[1, 0].set_title("Kilometers Driven")
axes[1, 1].hist(df['HP'], bins=30, color='purple', alpha=0.7)
axes[1, 1].set_title("Horsepower Distribution")
axes[2, 0].hist(df['CC'], bins=30, color='orange', alpha=0.7)
axes[2, 0].set_title("Engine Size (CC)")
axes[2, 1].hist(df['Weight'], bins=30, color='brown', alpha=0.7)
axes[2, 1].set_title("Car Weight Distribution")

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

[11] ✓ 1.4s

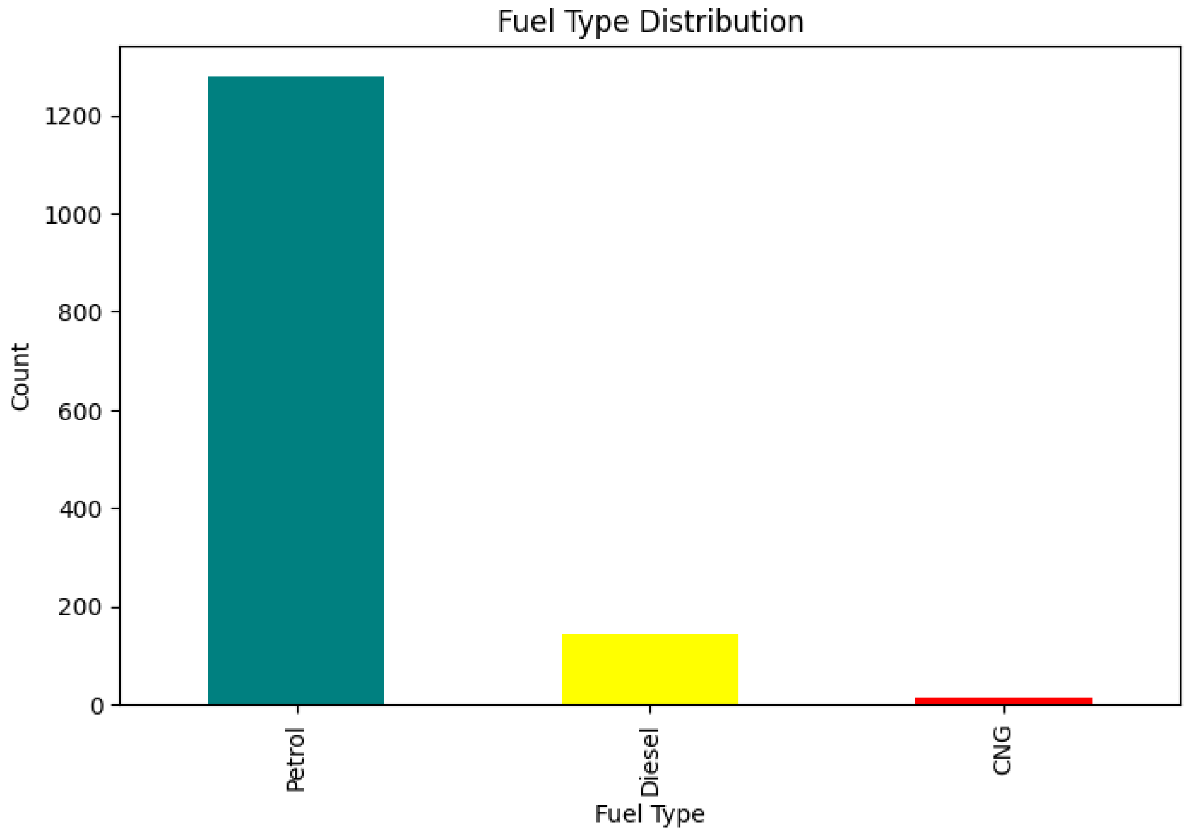
8. Plots of numeric features



9. Fuel type distribution

```
# Fuel Type Distribution
plt.figure(figsize=(8, 5))
df['FuelType'].value_counts().plot(kind='bar', color=['teal', 'yellow', 'red'])
plt.title("Fuel Type Distribution")
plt.xlabel("Fuel Type")
plt.ylabel("Count")
plt.show()
```

[15] ✓ 0.1s



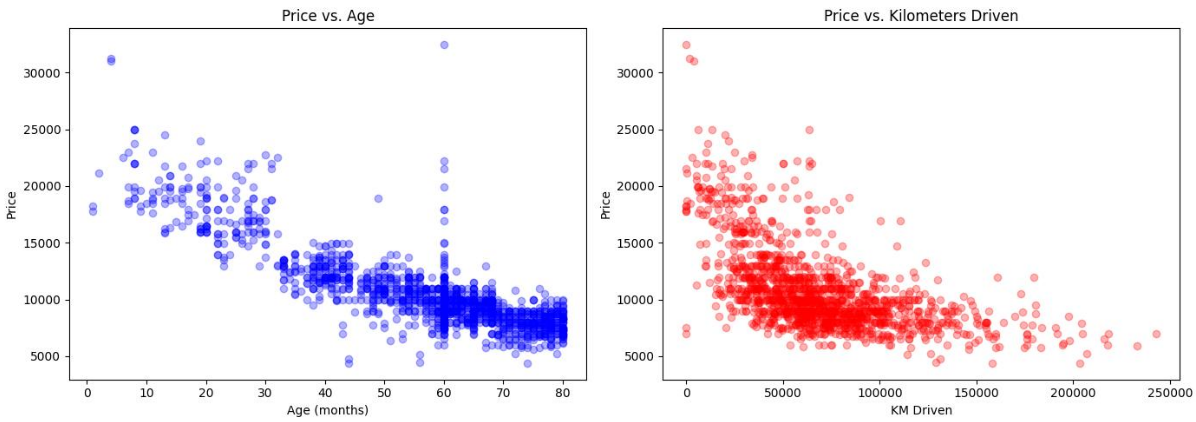
10. Plots for relationships between features

```
# Scatter plots for relationships between key variables
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

axes[0].scatter(df["Age"], df["Price"], alpha=0.3, color='blue')
axes[0].set_title("Price vs. Age")
axes[0].set_xlabel("Age (months)")
axes[0].set_ylabel("Price")

axes[1].scatter(df["KM"], df["Price"], alpha=0.3, color='red')
axes[1].set_title("Price vs. Kilometers Driven")
axes[1].set_xlabel("KM Driven")
axes[1].set_ylabel("Price")

plt.tight_layout()
plt.show()
```
[17] ✓ 0.3s



11. Insights

- **Price vs. Age**: Older cars have lower prices
- **Price vs. KM Driven**: Cars with higher kilometers driven have lower prices.
- **Fuel Type Popularity**: Petrol cars are the most common, followed by diesel, with very few LPG cars in the dataset.
- **Horsepower & Price**: Cars with higher horsepower have higher prices.

Day -3

Day 3: Modeling and Presentation

- 1. **Predictive Modeling**
- o Build an appropriate predictive model based on the dataset chosen
- o Evaluate model performance using appropriate metrics
- o Document model selection process and rationale
- 2. **Documentation and Presentation**
- o Document the entire process
- o Prepare presentation slides
- o Create a final report with actionable insights

Solution:

1. Import sklearn and define features and target

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

features = ['Age', 'KM', 'HP', 'CC', 'Weight']
target = 'Price'
```

[20] ✓ 18.0s

2. Split the dataset for training and testing

```
# Split data
X = df[features]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[22] ✓ 0.0s

3. Choose and apply model – Linear regression model is selected

```
# linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

[23] ✓ 0.0s

```
▾ LinearRegression ℹ ❓
LinearRegression()
```

4. Make predictions

```
# Make predictions
y_pred = model.predict(X_test)
```

[24] ✓ 0.0s

5. Evaluate the model using metrics

```
# Metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

[25] ✓ 0.0s

```
Mean Absolute Error: 1007.7523623460324
Mean Squared Error: 2467859.499716883
R-squared: 0.8150417031532511
```

6. Plot of actual vs predicted price

```
# Plot actual vs. predicted prices
plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, alpha=0.5, color='blue')
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs. Predicted Prices")
plt.show()
```