```
import pandas as pd
import numpy as np


# Load Salary_Data.csv
salary_df = pd.read_csv("Salary_Data.csv")

# Load Used_Car_Data.csv
car_df = pd.read_csv("used_cars_data.csv")
```

EDA of Salary Data

```
salary_df.shape
```

```
(13440, 11)
```

```
salary_df.head()
```

	empID	YearsExperience	Age	BTech?	MTech?	PhD?	Name	address	phone	branch	Salary
0	1001	1.1	30.0	1	1	1	Mr X-ds-1	Kolkata	9890078900	Ruby	39343.0
1	1002	1.3	31.0	1	1	1	Mr X-ds-2	Kolkata	9890078901	Ruby	46205.0
2	1003	1.5	32.0	1	1	1	Mr X-ds-3	Kolkata	9890078902	Ruby	37731.0
3	1004	2.0	33.0	1	1	1	Mr X-ds-4	Kolkata	9890078903	Ruby	43525.0
4	1005	2.2	34.0	1	1	1	Mr X-ds-5	Kolkata	9890078904	Ruby	39891.0

Next steps: Generate code with salary_df | View recommended plots | New interactive sheet

```
# Salary Data null check
print("Null values in Salary_Data:\n", salary_df.isnull().sum())
```

```
Null values in Salary_Data:
 empID                0
YearsExperience       0
Age                 379
BTech?                0
MTech?                0
PhD?                  0
Name                  0
address               0
phone                 0
branch                0
Salary             1920
dtype: int64
```

```
salary_df.describe()
```

	empID	YearsExperience	Age	BTech?	MTech?	PhD?	phone	
count	13440.000000	13440.000000	13061.000000	13440.000000	13440.000000	13440.000000	1.344000e+04	11520.(
mean	7720.500000	5.392381	39.937600	0.606845	0.576786	0.576786	9.890086e+09	76003.(
std	3879.938144	3.593758	6.037682	0.488469	0.494087	0.494087	3.879938e+03	26954.5
min	1001.000000	1.100000	30.000000	0.000000	0.000000	0.000000	9.890079e+09	37731.(
25%	4360.750000	2.100000	35.000000	0.000000	0.000000	0.000000	9.890082e+09	56642.(
50%	7720.500000	5.100000	40.000000	1.000000	1.000000	1.000000	9.890086e+09	65237.(
75%	11080.250000	8.100000	45.000000	1.000000	1.000000	1.000000	9.890089e+09	101302.(
max	14440.000000	14.100000	50.000000	1.000000	1.000000	1.000000	9.890092e+09	122391.(

```
# Fill numeric nulls with mean (Salary Data)
```

```
salary_df = salary_df.fillna(salary_df.median(numeric_only=True))
```

```
salary_df.describe()
```

	empID	YearsExperience	Age	BTech?	MTech?	PhD?	phone	
count	13440.000000	13440.000000	13440.000000	13440.000000	13440.000000	13440.000000	1.344000e+04	13440.(
mean	7720.500000	5.392381	39.939360	0.606845	0.576786	0.576786	9.890086e+09	74465.(
std	3879.938144	3.593758	5.951947	0.488469	0.494087	0.494087	3.879938e+03	25237.9
min	1001.000000	1.100000	30.000000	0.000000	0.000000	0.000000	9.890079e+09	37731.(
25%	4360.750000	2.100000	35.000000	0.000000	0.000000	0.000000	9.890082e+09	56957.(
50%	7720.500000	5.100000	40.000000	1.000000	1.000000	1.000000	9.890086e+09	65237.(
75%	11080.250000	8.100000	45.000000	1.000000	1.000000	1.000000	9.890089e+09	98273.(
max	14440.000000	14.100000	50.000000	1.000000	1.000000	1.000000	9.890092e+09	122391.(

```
salary_df.describe()# Salary Data null check
print("Null values in Salary_Data:\n", salary_df.isnull().sum())
```

```
Null values in Salary_Data:
 empID              0
YearsExperience    0
Age                0
BTech?             0
MTech?             0
PhD?               0
Name               0
address            0
phone              0
branch             0
Salary             0
dtype: int64
```

```
# Summary for Salary_Data
print("\nSalary Data - Mean:\n", salary_df.mean(numeric_only=True))
print("Salary Data - Median:\n", salary_df.median(numeric_only=True))
print("Salary Data - Mode:\n", salary_df.mode().iloc[0])
print("Salary Data - Std Deviation:\n", salary_df.std(numeric_only=True))
```

```
Salary Data - Mean:
 empID              7.720500e+03
YearsExperience    5.392381e+00
Age                3.993936e+01
BTech?             6.068452e-01
MTech?             5.767857e-01
PhD?               5.767857e-01
phone              9.890086e+09
Salary             7.446500e+04
dtype: float64
Salary Data - Median:
 empID              7.720500e+03
YearsExperience    5.100000e+00
Age                4.000000e+01
BTech?             1.000000e+00
MTech?             1.000000e+00
PhD?               1.000000e+00
phone              9.890086e+09
Salary             6.523700e+04
dtype: float64
Salary Data - Mode:
 empID                    1001
YearsExperience           1.1
Age                      40.0
BTech?                    1.0
MTech?                    1.0
PhD?                      1.0
Name                 Mr X-ds-1
address                Kolkata
phone               9890078900
branch                   Ruby
```

```
Salary                65237.0
Name: 0, dtype: object
Salary Data - Std Deviation:
 empID               3879.938144
YearsExperience        3.593758
Age                    5.951947
BTech?                 0.488469
MTech?                 0.494087
PhD?                   0.494087
phone               3879.938144
Salary             25237.941859
dtype: float64
```

```
print("Categorical \n",salary_df.select_dtypes(include='object'))  # for categorical
print("Numerical \n",salary_df.select_dtypes(include=['int64', 'float64']))  # for numerical
```

```
Categorical
                   Name  address branch
0          Mr X-ds-1  Kolkata   Ruby
1          Mr X-ds-2  Kolkata   Ruby
2          Mr X-ds-3  Kolkata   Ruby
3          Mr X-ds-4  Kolkata   Ruby
4          Mr X-ds-5  Kolkata   Ruby
...              ...      ...    ...
13435  Mr X-ds-13436  Kolkata   Ruby
13436  Mr X-ds-13437  Kolkata   Ruby
13437  Mr X-ds-13438  Kolkata   Ruby
13438  Mr X-ds-13439  Kolkata   Ruby
13439  Mr X-ds-13440  Kolkata   Ruby

[13440 rows x 3 columns]
Numerical
        empID  YearsExperience   Age  BTech?  MTech?  PhD?       phone  \
0       1001              1.1  30.0       1       1     1  9890078900
1       1002              1.3  31.0       1       1     1  9890078901
2       1003              1.5  32.0       1       1     1  9890078902
3       1004              2.0  33.0       1       1     1  9890078903
4       1005              2.2  34.0       1       1     1  9890078904
...      ...              ...   ...     ...     ...   ...         ...
13435  14436             10.1  35.0       1       1     1  9890092335
13436  14437             11.1  36.0       1       1     1  9890092336
13437  14438             12.1  37.0       1       1     1  9890092337
13438  14439             13.1  38.0       1       1     1  9890092338
13439  14440             14.1  39.0       1       1     1  9890092339

         Salary
0       39343.0
1       46205.0
2       37731.0
3       43525.0
4       39891.0
...         ...
13435  105582.0
13436  116969.0
13437  112635.0
13438  122391.0
13439  121872.0

[13440 rows x 8 columns]
```

EDA of Used Cars Data

```
car_df.shape
```

```
(58024, 14)
```

```
car_df.head()
```

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner Type	Mileage	Engine	Power
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp

Next steps: Generate code with car_df | View recommended plots | New interactive sheet

```
# Used Car Data null check
print("\nNull values in Used_Car_Data:\n", car_df.isnull().sum())
```

```
Null values in Used_Car_Data:
 S.No.                  0
Name                    0
Location                0
Year                    0
Kilometers_Driven       0
Fuel_Type               0
Transmission            0
Owner Type              0
Mileage                16
Engine                368
Power                 368
Seats                 424
New_Price           49976
Price                9872
dtype: int64
```

```
car_df.describe()
```

	S.No.	Year	Kilometers_Driven	Seats	Price
count	58024.000000	58024.000000	5.802400e+04	57600.000000	48152.000000
mean	29011.500000	2013.365366	5.869906e+04	5.279722	9.479468
std	16750.230347	3.254224	8.442263e+04	0.811610	11.187104
min	0.000000	1996.000000	1.710000e+02	0.000000	0.440000
25%	14505.750000	2011.000000	3.400000e+04	5.000000	3.500000
50%	29011.500000	2014.000000	5.341600e+04	5.000000	5.640000
75%	43517.250000	2016.000000	7.300000e+04	5.000000	9.950000
max	58023.000000	2019.000000	6.500000e+06	10.000000	160.000000

```
# Remove both ' kmpl' and ' km/kg'
car_df['Mileage'] = car_df['Mileage'].str.replace(' kmpl', '', regex=False)
car_df['Mileage'] = car_df['Mileage'].str.replace(' km/kg', '', regex=False)

# Now safely convert to float
car_df['Mileage'] = pd.to_numeric(car_df['Mileage'], errors='coerce')
```

```
# Clean Engine
car_df['Engine'] = car_df['Engine'].str.replace(' CC', '', regex=False)
car_df['Engine'] = pd.to_numeric(car_df['Engine'], errors='coerce')

# Clean Power
car_df['Power'] = car_df['Power'].str.replace(' bhp', '', regex=False)
car_df['Power'] = pd.to_numeric(car_df['Power'], errors='coerce')



car_df['Mileage'] = car_df['Mileage'].fillna(car_df['Mileage'].median())
car_df['Engine'] = car_df['Engine'].fillna(car_df['Engine'].median())
car_df['Power'] = car_df['Power'].fillna(car_df['Power'].median())
car_df['Seats'] = car_df['Seats'].fillna(car_df['Seats'].mode()[0])



# Used Car Data null check
print("\nNull values in Used_Car_Data:\n", car_df.isnull().sum())
```

```
Null values in Used_Car_Data:
 S.No.                    0
Name                      0
Location                  0
Year                      0
Kilometers_Driven         0
Fuel_Type                 0
Transmission              0
Owner Type                0
Mileage                   0
Engine                    0
Power                     0
Seats                     0
New_Price             49976
Price                  9872
dtype: int64
```

```
car_df.describe()
```

	S.No.	Year	Kilometers_Driven	Mileage	Engine	Power	Seats	
count	58024.000000	58024.000000	5.802400e+04	58024.000000	58024.000000	58024.000000	58024.000000	48152
mean	29011.500000	2013.365366	5.869906e+04	18.141586	1615.789742	112.312448	5.277678	9
std	16750.230347	3.254224	8.442263e+04	4.561292	593.439457	52.919388	0.808990	11
min	0.000000	1996.000000	1.710000e+02	0.000000	72.000000	34.200000	0.000000	0
25%	14505.750000	2011.000000	3.400000e+04	15.170000	1198.000000	77.000000	5.000000	3
50%	29011.500000	2014.000000	5.341600e+04	18.160000	1493.000000	94.000000	5.000000	5
75%	43517.250000	2016.000000	7.300000e+04	21.100000	1968.000000	138.030000	5.000000	9
max	58023.000000	2019.000000	6.500000e+06	33.540000	5998.000000	616.000000	10.000000	160

```
# Drop rows where Price is missing
car_df = car_df.dropna(subset=['Price'])

car_df.drop(columns=['New_Price'], inplace=True)
```

```
/tmp/ipython-input-21-2609105402.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
  car_df.drop(columns=['New_Price'], inplace=True)
```

```
# Used Car Data null check
print("\nNull values in Used_Car_Data:\n", car_df.isnull().sum())
```

```
Null values in Used_Car_Data:
 S.No.                0
Name                  0
Location              0
Year                  0
Kilometers_Driven     0
Fuel_Type             0
Transmission          0
Owner Type            0
Mileage               0
Engine                0
Power                 0
Seats                 0
Price                 0
dtype: int64
```

```
car_df.describe()
```

	S.No.	Year	Kilometers_Driven	Mileage	Engine	Power	Seats	
count	48152.000000	48152.000000	4.815200e+04	48152.000000	48152.000000	48152.000000	48152.000000	48152
mean	28394.500000	2013.358199	5.873838e+04	18.134969	1620.509221	112.795634	5.276790	9
std	16709.469641	3.269504	9.126221e+04	4.581195	599.591870	53.307852	0.806287	11
min	0.000000	1998.000000	1.710000e+02	0.000000	72.000000	34.200000	0.000000	0
25%	14197.250000	2011.000000	3.400000e+04	15.170000	1198.000000	78.000000	5.000000	3
50%	28394.500000	2014.000000	5.300000e+04	18.160000	1493.000000	94.000000	5.000000	5
75%	42591.750000	2016.000000	7.300000e+04	21.100000	1969.000000	138.030000	5.000000	9
max	56789.000000	2019.000000	6.500000e+06	33.540000	5998.000000	560.000000	10.000000	160

```
# Summary for Used_Car_Data
print("\nUsed Car Data - Mean:\n", car_df.mean(numeric_only=True))
print("Used Car Data - Median:\n", car_df.median(numeric_only=True))
print("Used Car Data - Mode:\n", car_df.mode().iloc[0])
print("Used Car Data - Std Deviation:\n", car_df.std(numeric_only=True))
```

```
Used Car Data - Mean:
 S.No.               28394.500000
Year                 2013.358199
Kilometers_Driven   58738.380296
Mileage                18.134969
Engine               1620.509221
Power                 112.795634
Seats                   5.276790
Price                   9.479468
dtype: float64
Used Car Data - Median:
 S.No.               28394.50
Year                 2014.00
Kilometers_Driven   53000.00
Mileage                18.16
Engine               1493.00
Power                  94.00
Seats                   5.00
Price                   5.64
dtype: float64
Used Car Data - Mode:
 S.No.                                  0
Name                 Mahindra XUV500 W8 2WD
Location                             Mumbai
Year                                 2014.0
Kilometers_Driven                   60000.0
Fuel_Type                           Diesel
Transmission                        Manual
Owner Type                           First
Mileage                               17.0
Engine                              1197.0
Power                                 74.0
Seats                                  5.0
```

```
Price                                4.5
Name: 0, dtype: object
Used Car Data - Std Deviation:
 S.No.               16709.469641
Year                    3.269504
Kilometers_Driven    91262.208815
Mileage                 4.581195
Engine                599.591870
Power                  53.307852
Seats                   0.806287
Price                  11.187104
dtype: float64
```

```
print("Categorical \n",car_df.select_dtypes(include='object'))  # for categorical
print("Numerical \n",car_df.select_dtypes(include=['int64', 'float64']))  # for numerical
```

```
Categorical
                                  Name    Location Fuel_Type Transmission  \
0               Maruti Wagon R LXI CNG      Mumbai       CNG       Manual
1     Hyundai Creta 1.6 CRDi SX Option        Pune    Diesel       Manual
2                         Honda Jazz V     Chennai    Petrol       Manual
3                    Maruti Ertiga VDI     Chennai    Diesel       Manual
4      Audi A4 New 2.0 TDI Multitronic  Coimbatore    Diesel    Automatic
...                                ...         ...       ...          ...
56785                  Maruti Swift VDI       Delhi    Diesel       Manual
56786          Hyundai Xcent 1.1 CRDi S      Jaipur    Diesel       Manual
56787             Mahindra Xylo D4 BSIV      Jaipur    Diesel       Manual
56788                Maruti Wagon R VXI     Kolkata    Petrol       Manual
56789             Chevrolet Beat Diesel   Hyderabad    Diesel       Manual

      Owner Type
0          First
1          First
2          First
3          First
4         Second
...          ...
56785      First
56786      First
56787     Second
56788      First
56789      First

[48152 rows x 5 columns]
Numerical
        S.No.  Year  Kilometers_Driven  Mileage  Engine   Power  Seats  Price
0          0  2010              72000    26.60   998.0   58.16    5.0   1.75
1          1  2015              41000    19.67  1582.0  126.20    5.0  12.50
2          2  2011              46000    18.20  1199.0   88.70    5.0   4.50
3          3  2012              87000    20.77  1248.0   88.76    7.0   6.00
4          4  2013              40670    15.20  1968.0  140.80    5.0  17.74
...      ...   ...                ...      ...     ...     ...    ...    ...
56785  56785  2014              27365    28.40  1248.0   74.00    5.0   4.75
56786  56786  2015             100000    24.40  1120.0   71.00    5.0   4.00
56787  56787  2012              55000    14.00  2498.0  112.00    8.0   2.90
56788  56788  2013              46000    18.90   998.0   67.10    5.0   2.65
56789  56789  2011              47000    25.44   936.0   57.60    5.0   2.50

[48152 rows x 8 columns]
```

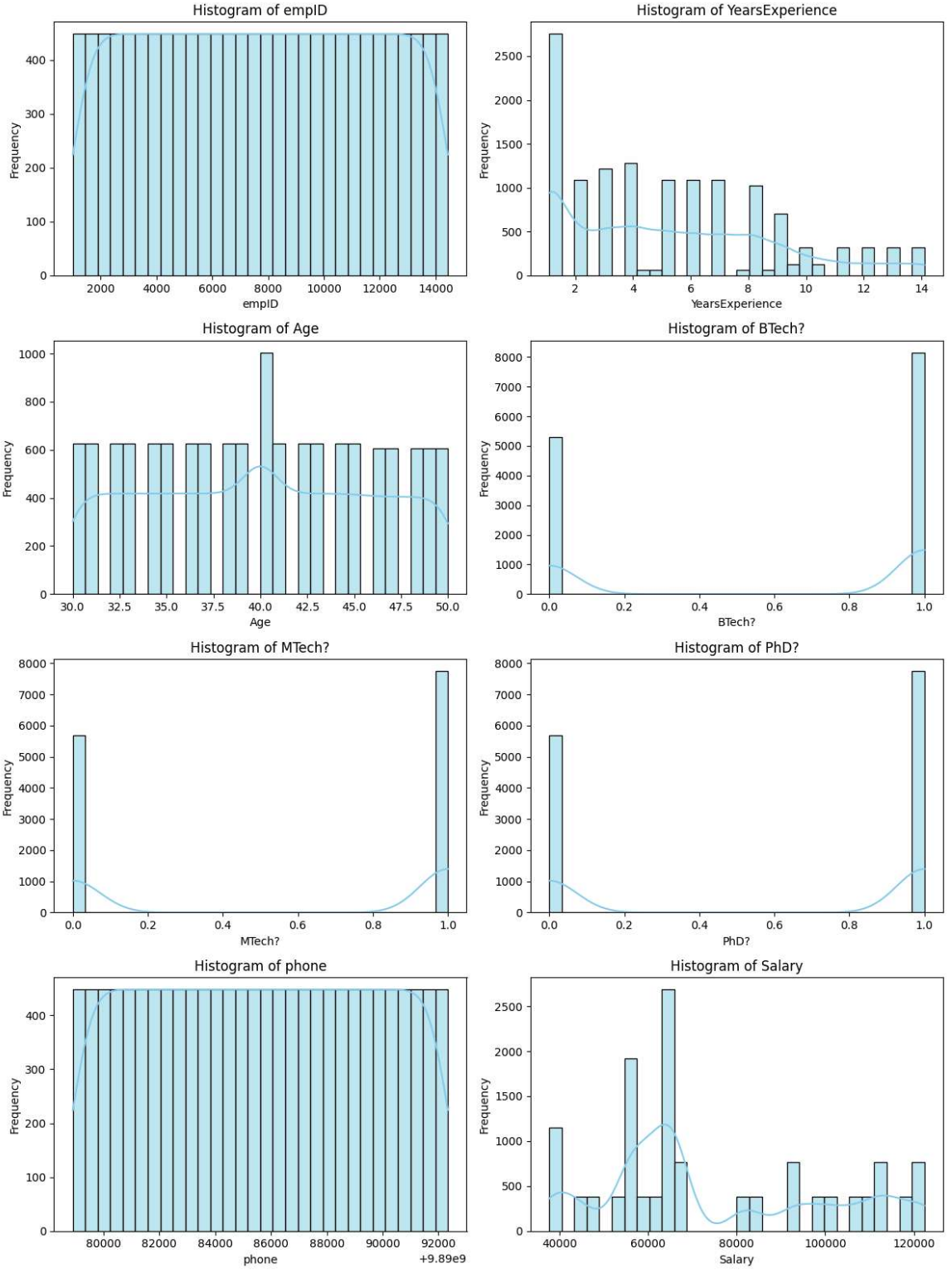
```
salary_df.to_csv("Cleaned_Salary_Data.csv", index=False)
car_df.to_csv("Cleaned_Used_Car_Data.csv", index=False)
```

Outlier Detection in Salary Data

```
# Load cleaned data
salary_df = pd.read_csv("Cleaned_Salary_Data.csv")
```

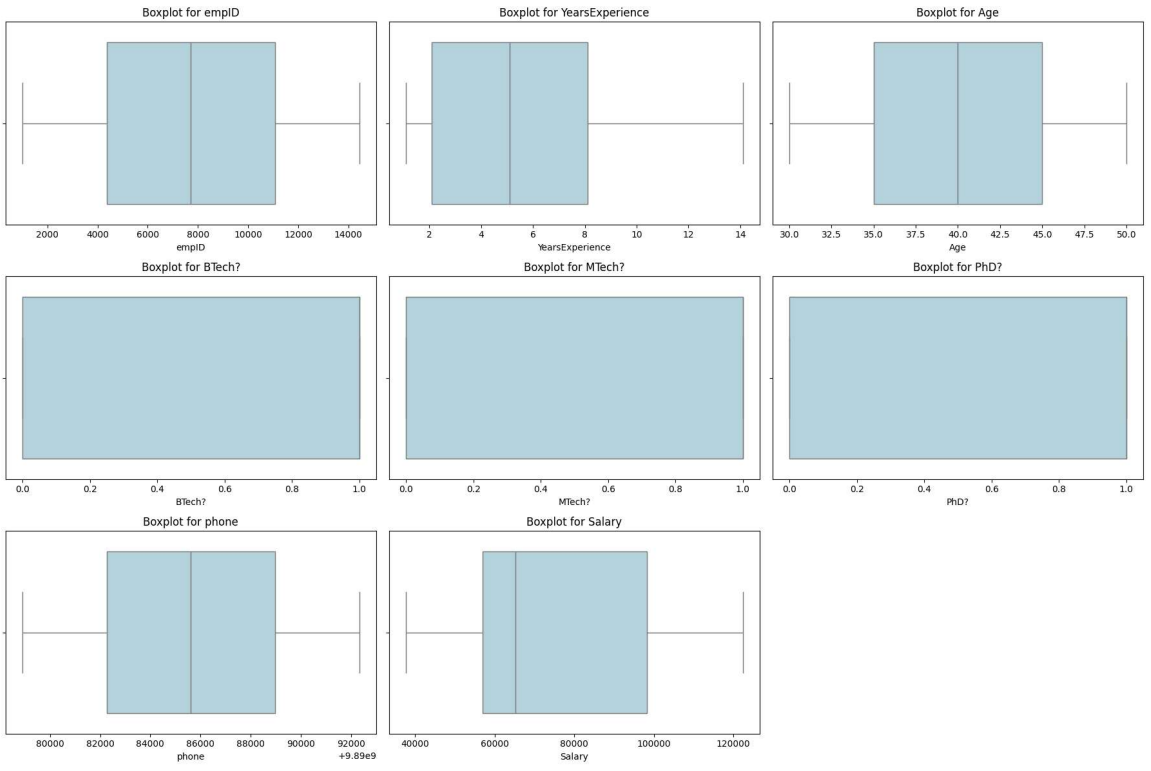
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming car_df is your cleaned DataFrame
```

```
numeric_cols = salary_df.select_dtypes(include='number').columns.tolist()

# Set number of plots per row
n_cols = 2
n_rows = int(np.ceil(len(numeric_cols) / n_cols))

plt.figure(figsize=(12, 4 * n_rows))

for i, col in enumerate(numeric_cols, 1):
    plt.subplot(n_rows, n_cols, i)
    sns.histplot(salary_df[col], kde=True, bins=30, color='skyblue')
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel("Frequency")

plt.tight_layout()
plt.show()
```


Histogram of empID
Frequency
empID
Histogram of YearsExperience
Frequency
YearsExperience
Histogram of Age
Frequency
Age
Histogram of BTech?
Frequency
BTech?
Histogram of MTech?
Frequency
MTech?
Histogram of PhD?
Frequency
PhD?
Histogram of phone
Frequency
phone
+9.89e9
Histogram of Salary
Frequency
Salary

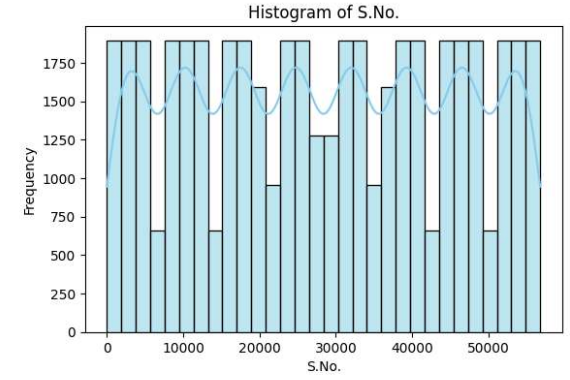
Using IQR Method

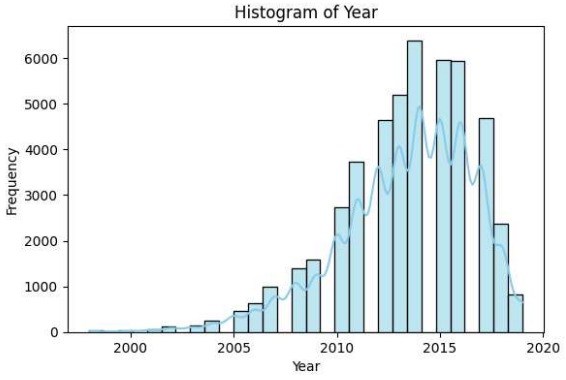
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


# Load cleaned data
salary_df = pd.read_csv("Cleaned_Salary_Data.csv")


# Select numeric columns for outlier detection
numeric_cols = salary_df.select_dtypes(include=[np.number]).columns.tolist()


# Number of plots per row
plots_per_row = 3

# Total numeric columns
num_cols = len(numeric_cols)

# Calculate number of rows needed
num_rows = int(np.ceil(num_cols / plots_per_row))

# Set figure size dynamically
plt.figure(figsize=(6 * plots_per_row, 4 * num_rows))

# Loop and plot
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(num_rows, plots_per_row, i)
    sns.boxplot(x=salary_df[col], color='lightblue')
    plt.title(f'Boxplot for {col}')
    plt.tight_layout()

plt.show()
```

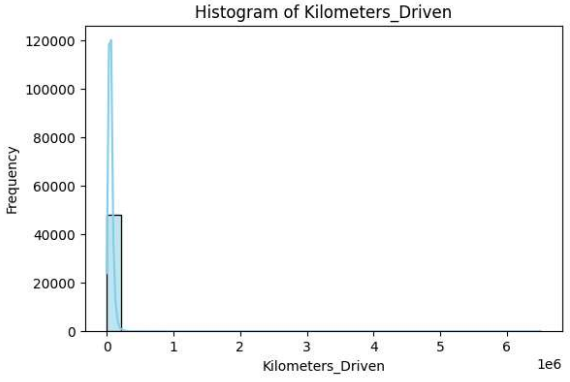


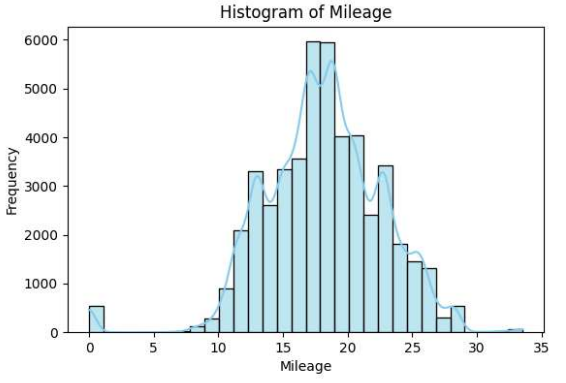
Outlier Detection in Used Cars Data

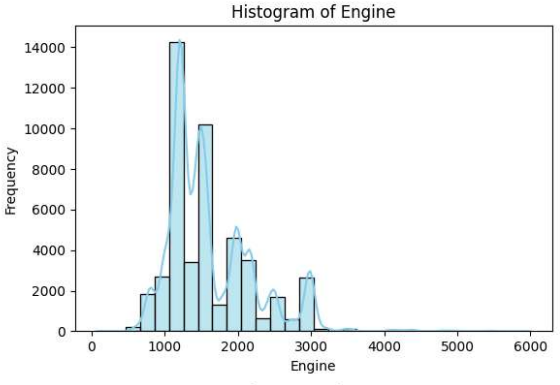
```
# Load cleaned data
car_df = pd.read_csv("Cleaned_Used_Car_Data.csv")


import matplotlib.pyplot as plt
import seaborn as sns

# Assuming car_df is your cleaned DataFrame
numeric_cols = car_df.select_dtypes(include='number').columns.tolist()

# Set number of plots per row
n_cols = 2
n_rows = int(np.ceil(len(numeric_cols) / n_cols))

plt.figure(figsize=(12, 4 * n_rows))

for i, col in enumerate(numeric_cols, 1):
    plt.subplot(n_rows, n_cols, i)
    sns.histplot(car_df[col], kde=True, bins=30, color='skyblue')
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel("Frequency")

plt.tight_layout()
plt.show()
```

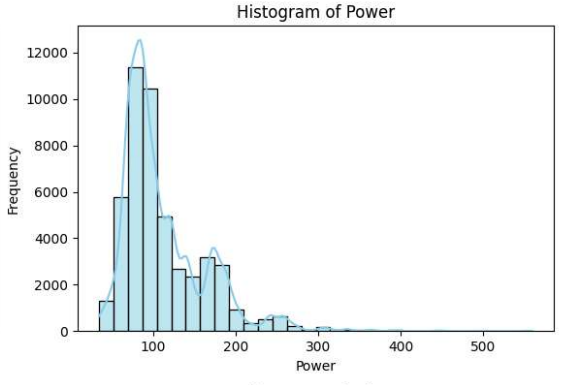

Histogram of S.No.
Frequency
S.No.

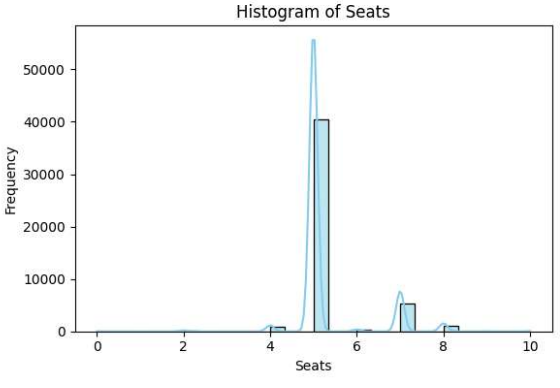

Histogram of Year
Frequency
Year

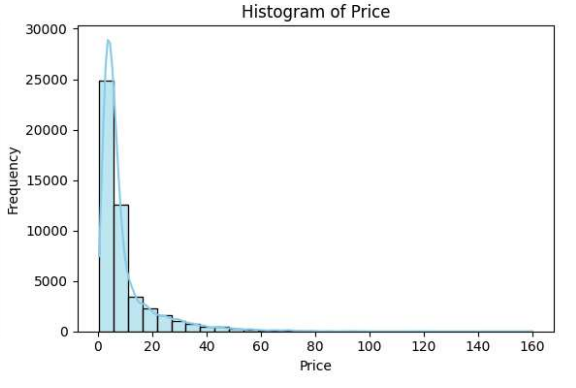

Histogram of Kilometers_Driven
Frequency
Kilometers_Driven
1e6


Histogram of Mileage
Frequency
Mileage


Histogram of Engine
Frequency
Engine

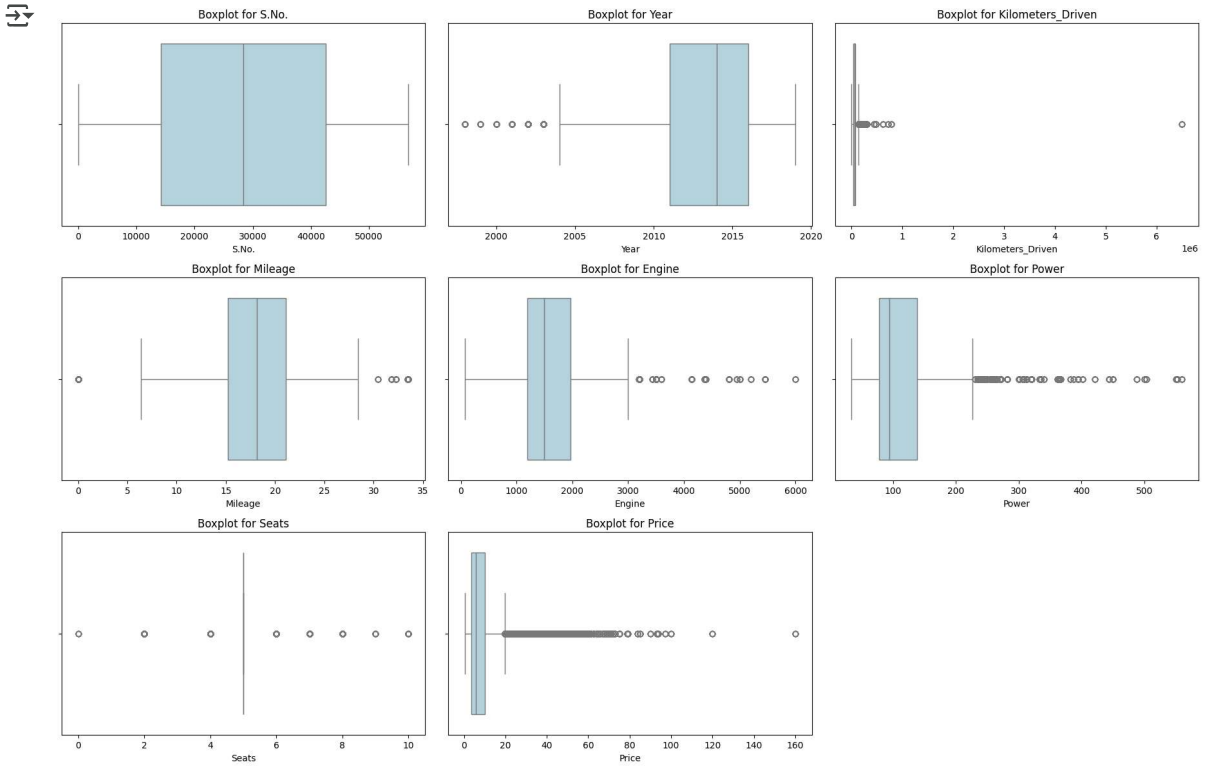

Histogram of Power
Frequency
Power


Histogram of Seats
Frequency
Seats


Histogram of Price
Frequency
Price

Using IQR Method

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


# Load cleaned data
car_df = pd.read_csv("Cleaned_Used_Car_Data.csv")


# Select numeric columns for outlier detection
numeric_cols = car_df.select_dtypes(include=[np.number]).columns.tolist()


# Number of plots per row
plots_per_row = 3

# Total numeric columns
num_cols = len(numeric_cols)

# Calculate number of rows needed
num_rows = int(np.ceil(num_cols / plots_per_row))

# Set figure size dynamically
plt.figure(figsize=(6 * plots_per_row, 4 * num_rows))

# Loop and plot
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(num_rows, plots_per_row, i)
    sns.boxplot(x=car_df[col], color='lightblue')
    plt.title(f'Boxplot for {col}')
    plt.tight_layout()

plt.show()
```

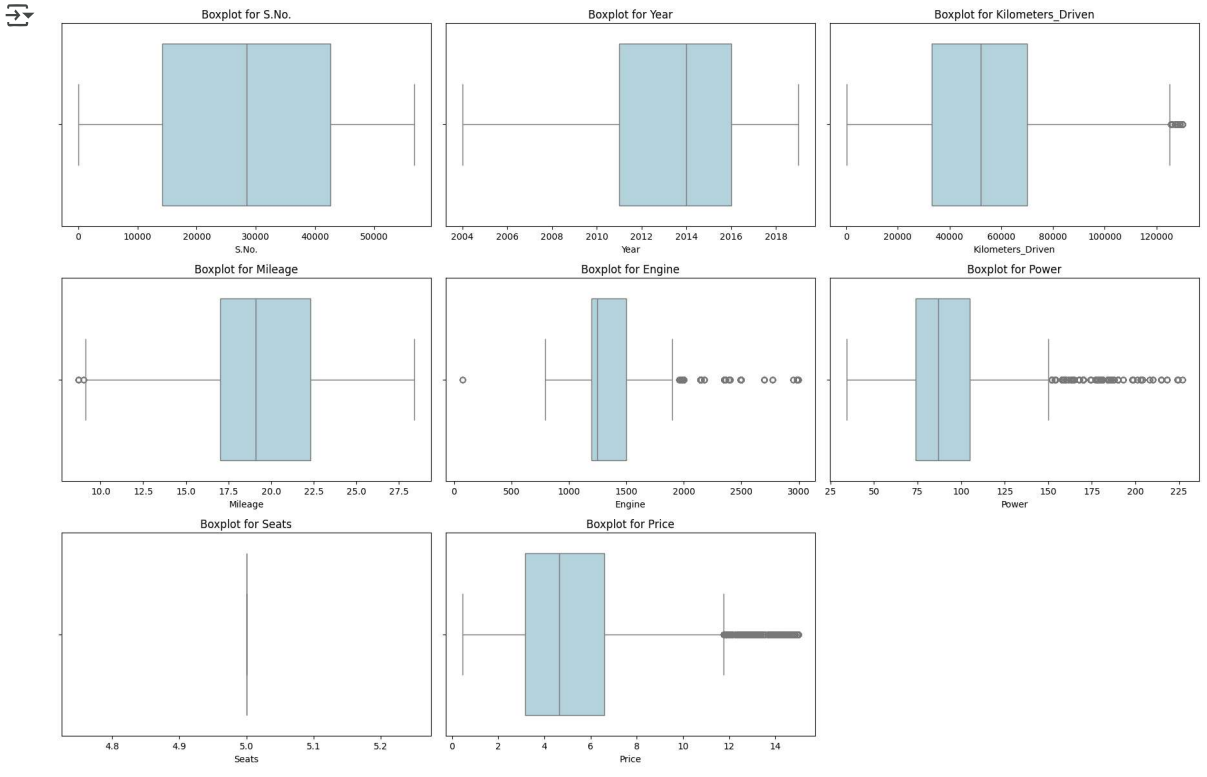



```
iqr_removed = {}
# --- IQR Method ---
def remove_outliers_iqr(df, column):
    initial_rows = df.shape[0]
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_cleaned = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    removed = initial_rows - df_cleaned.shape[0]
    return df_cleaned, removed

for col in numeric_cols:
    car_df, removed = remove_outliers_iqr(car_df, col)
    iqr_removed[col] = removed


# --- Display the Summary ---
print("🔎 Rows removed per column using IQR method:")
for col, count in iqr_removed.items():
    print(f"{col:20} : {count} rows removed")
print(f"\nFinal rows after IQR cleanup: {car_df.shape[0]}")
```

```
🔎 Rows removed per column using IQR method:
S.No.                : 0 rows removed
Year                 : 400 rows removed
Kilometers_Driven    : 1616 rows removed
Mileage              : 576 rows removed
Engine               : 456 rows removed
Power                : 1544 rows removed
Seats                : 5920 rows removed
Price                : 4440 rows removed
```

```
Final rows after IQR cleanup: 33200
```

```
# Number of plots per row
plots_per_row = 3

# Total numeric columns
num_cols = len(numeric_cols)

# Calculate number of rows needed
num_rows = int(np.ceil(num_cols / plots_per_row))

# Set figure size dynamically
plt.figure(figsize=(6 * plots_per_row, 4 * num_rows))

# Loop and plot
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(num_rows, plots_per_row, i)
    sns.boxplot(x=car_df[col], color='lightblue')
    plt.title(f'Boxplot for {col}')
    plt.tight_layout()

plt.show()
```



Using Z Score method

```
from scipy.stats import zscore
```

```
# Reload the cleaned data
car_df = pd.read_csv("Cleaned_Used_Car_Data.csv")
```

```
# Number of plots per row
plots_per_row = 3

# Total numeric columns
num_cols = len(numeric_cols)

# Calculate number of rows needed
num_rows = int(np.ceil(num_cols / plots_per_row))

# Set figure size dynamically
plt.figure(figsize=(6 * plots_per_row, 4 * num_rows))

# Loop and plot
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(num_rows, plots_per_row, i)
    sns.boxplot(x=car_df[col], color='lightblue')
    plt.title(f'Boxplot for {col}')
    plt.tight_layout()

plt.show()
```