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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_absolute_error, accuracy_score, confusion_matrix
from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
```

```
df = pd.read_csv("Food_Delivery_Times.csv")
print(df.head())
print(df.info())
```

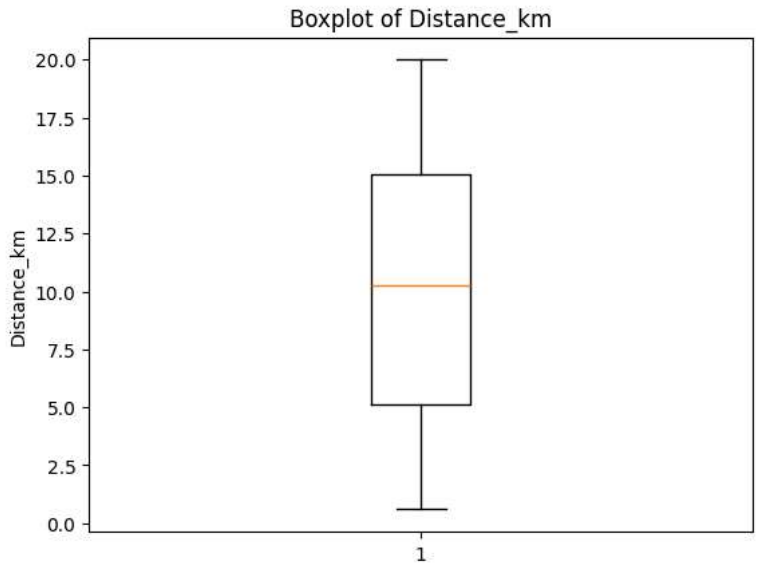
```
   Order_ID  Distance_km Weather Traffic_Level Time_of_Day Vehicle_Type  \
0       522         7.93   Windy           Low   Afternoon      Scooter
1       738        16.42   Clear        Medium     Evening         Bike
2       741         9.52   Foggy           Low       Night      Scooter
3       661         7.44   Rainy        Medium   Afternoon      Scooter
4       412        19.03   Clear           Low     Morning         Bike

   Preparation_Time_min  Courier_Experience_yrs  Delivery_Time_min
0                    12                     1.0                 43
1                    20                     2.0                 84
2                    28                     1.0                 59
3                     5                     1.0                 37
4                    16                     5.0                 68
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Order_ID                1000 non-null   int64
 1   Distance_km             1000 non-null   float64
 2   Weather                 970 non-null    object
 3   Traffic_Level           970 non-null    object
 4   Time_of_Day             970 non-null    object
 5   Vehicle_Type            1000 non-null   object
 6   Preparation_Time_min    1000 non-null   int64
 7   Courier_Experience_yrs  970 non-null    float64
 8   Delivery_Time_min       1000 non-null   int64
dtypes: float64(2), int64(3), object(4)
memory usage: 70.4+ KB
None
```

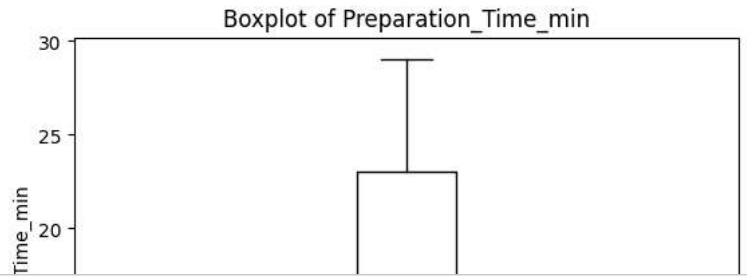
```
#--------------Data Cleaning and preprocessing----------------------------
# Dataset shape
print("\tBEFORE")
print("Initial shape:", df.shape)

# Check duplicates
print("Duplicate rows:", df.duplicated().sum())

# Check missing values
print("Missing values per column:")
print(df.isnull().sum())

#fix missing values
median_exp = df['Courier_Experience_yrs'].median()
df['Courier_Experience_yrs'] = df['Courier_Experience_yrs'].fillna(median_exp)
df.dropna(inplace=True)
print("\n\tAFTER")
print("Shape after handling missing values:", df.shape)
print("Final missing values:")
print(df.isnull().sum())
```

```
        BEFORE
Initial shape: (1000, 9)
Duplicate rows: 0
Missing values per column:
Order_ID                  0
Distance_km               0
Weather                  30
Traffic_Level            30
Time_of_Day              30
Vehicle_Type              0
```

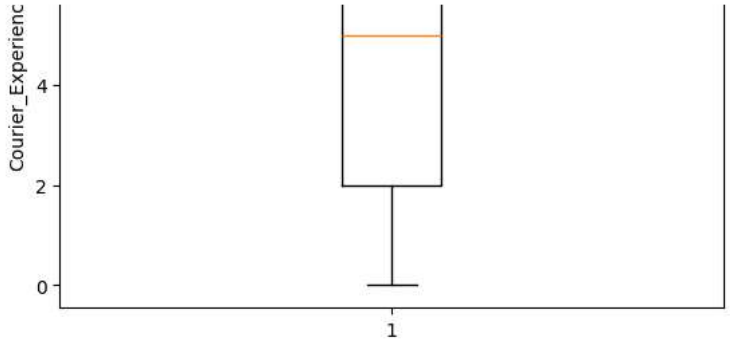
```
Preparation_Time_min       0
Courier_Experience_yrs    30
Delivery_Time_min          0
dtype: int64

        AFTER
Shape after handling missing values: (912, 9)
Final missing values:
Order_ID                  0
Distance_km               0
Weather                   0
Traffic_Level             0
Time_of_Day               0
Vehicle_Type              0
Preparation_Time_min      0
Courier_Experience_yrs    0
Delivery_Time_min         0
dtype: int64
```

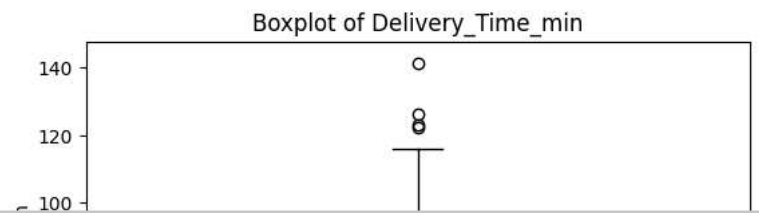
```
#Check outliers
numerical_cols = ['Distance_km', 'Preparation_Time_min','Courier_Experience_yrs', 'Delivery_Time_min']
for col in numerical_cols:
    plt.boxplot(df[col])
    plt.title(f"Boxplot of {col}")
    plt.ylabel(col)
    plt.show()
print("\nSummary statistics:")
print(df.describe())
```

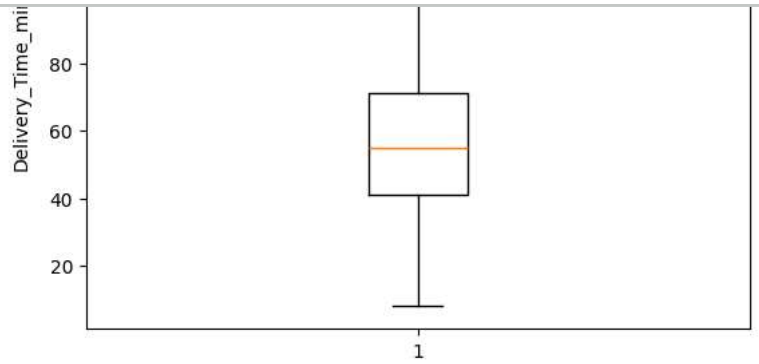





```
#Remove outliers
for col in ['Delivery_Time_min']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    df = df[(df[col] >= lower) & (df[col] <= upper)]

print("Final cleaned dataset shape:", df.shape)

for col in numerical_cols:
    plt.boxplot(df[col])
    plt.title(f"Boxplot of {col}")
    plt.ylabel(col)
    plt.show()
print("\nSummary statistics:")
print(df.describe())
```





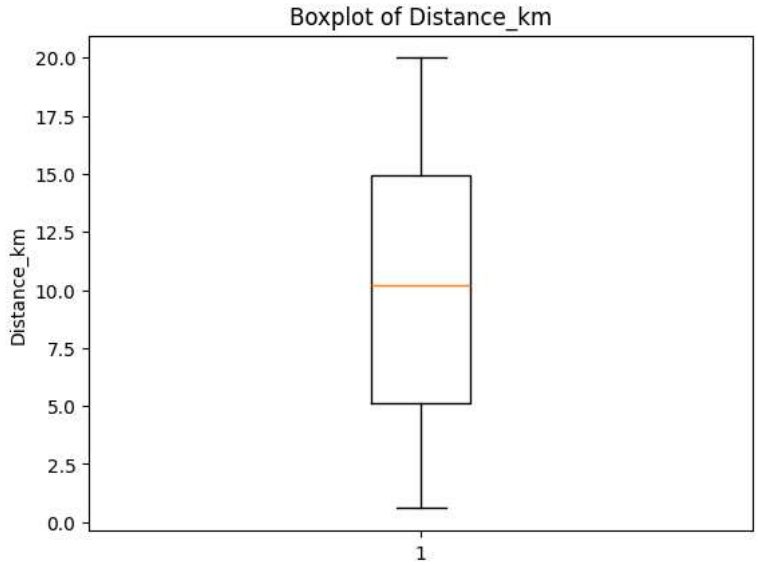


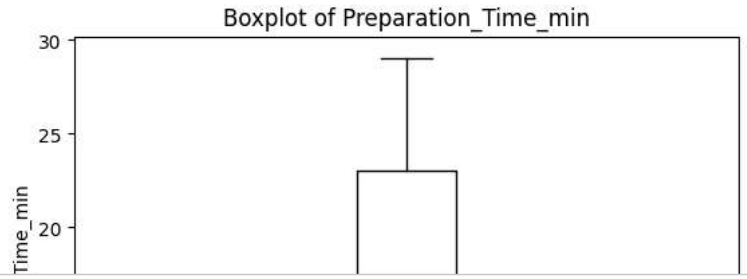
Summary statistics:

	Order_ID	Distance_km	Preparation_Time_min	Courier_Experience_yrs
count	912.000000	912.000000	912.000000	912.000000
mean	507.573465	10.069320	16.993421	4.651316
std	288.528012	5.692518	7.264636	2.875980
min	1.000000	0.590000	5.000000	0.000000
25%	259.750000	5.130000	10.000000	2.000000
50%	512.500000	10.285000	17.000000	5.000000
75%	758.250000	15.042500	23.000000	7.000000
max	1000.000000	19.990000	29.000000	9.000000

	Delivery_Time_min
count	912.000000
mean	56.450658
std	21.581281
min	8.000000
25%	41.000000
50%	55.000000
75%	71.000000
max	141.000000

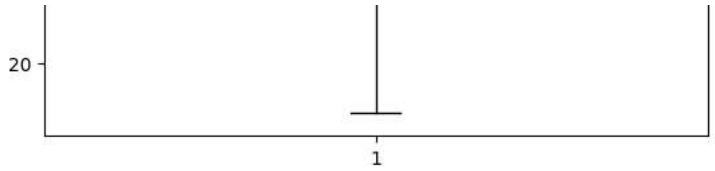
```
Final cleaned dataset shape: (908, 9)
```






```
#------------------------Feature engineering and eda--------------------
# Distance Category
df['Distance_Category'] = pd.cut(
    df['Distance_km'],
    bins=[0, 5, 10, 20, 50],
    labels=['Short', 'Medium', 'Long', 'Very Long']
)

# Peak Hour Indicator
df['Peak_Hour'] = df['Time_of_Day'].apply(
    lambda x: 1 if x in ['Evening', 'Night'] else 0
)

# Courier Experience Groups
df['Experience_Group'] = pd.cut(
    df['Courier_Experience_yrs'],
    bins=[0, 2, 5, 10, 50],
    labels=['Beginner', 'Intermediate', 'Experienced', 'Expert']
)

# Delay Status (Target for classification)
df['Delay_Status'] = np.where(
    df['Delivery_Time_min'] >
    df['Preparation_Time_min'] + 15,
    'Delayed', 'On-Time'
)

print(df.head())
# Weather Impact
df.groupby('Weather')['Delivery_Time_min'].mean().plot(kind='bar')
plt.title("Weather Impact on Delivery Time")
plt.ylabel("Average Delivery Time (min)")
plt.show()

# Traffic Impact
df.groupby('Traffic_Level')['Delivery_Time_min'].mean().plot(kind='bar')
plt.title("Traffic Level vs Delivery Time")
plt.ylabel("Average Delivery Time (min)")
plt.show()

# Distance vs Time
```

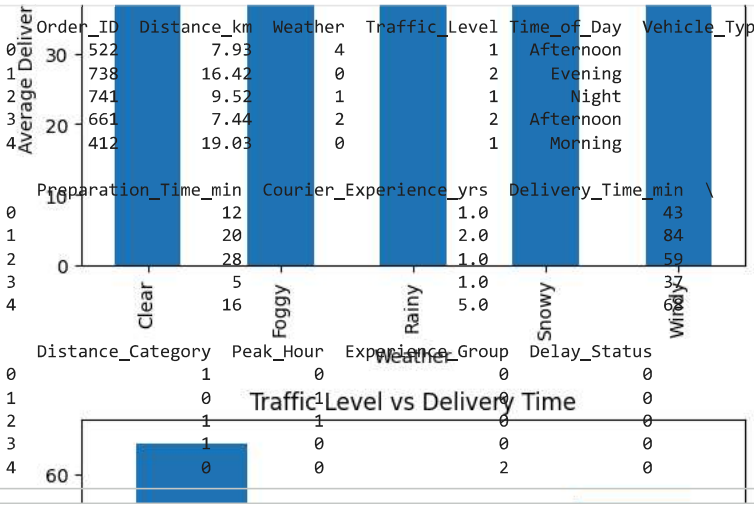
```
plt.scatter(df['Distance_km'], df['Delivery_Time_min'])
plt.xlabel("Distance (km)")
plt.ylabel("Delivery Time (min)")
plt.title("Distance vs Delivery Time")
plt.show()
```




```
Summary statistics:
          Order_ID  Distance_km  Preparation_Time_min  Courier_Experience_yrs  \
count   908.000000   908.000000            908.000000              908.000000
mean    507.378855    10.036960             16.985683                4.650881
std     288.745428     5.683134              7.270548                2.881016
min       1.000000     0.590000              5.000000                0.000000
25%     258.750000     5.105000             10.000000                2.000000
50%     513.500000    10.220000             17.000000                5.000000
75%     758.250000    14.965000             23.000000                7.000000
max    1000.000000    19.990000             29.000000                9.000000

       Delivery_Time_min
count         908.000000
mean           56.135463
std            21.091974
min             8.000000
25%            40.750000
50%            55.000000
75%            70.250000
max           116.000000
```

```
   Order_ID  Distance_km Weather Traffic_Level Time_of_Day Vehicle_Type  \
0       522         7.93   Windy           Low   Afternoon      Scooter
1       738        16.42   Clear        Medium     Evening         Bike
2       741         9.52   Foggy           Low       Night      Scooter
3       661         7.44   Rainy        Medium   Afternoon      Scooter
4       412        19.03   Clear           Low     Morning         Bike

   Preparation_Time_min  Courier_Experience_yrs  Delivery_Time_min  \
0                    12                     1.0                 43
1                    20                     2.0                 84
2                    28                     1.0                 59
3                     5                     1.0                 37
4                    16                     5.0                 68

  Distance_Category  Peak_Hour Experience_Group Delay_Status
0            Medium          0         Beginner      Delayed
1              Long          1         Beginner      Delayed
2            Medium          1         Beginner      Delayed
3            Medium          0         Beginner      Delayed
```

```
#-------------------Label Encoding for categorical columns-------------------
categorical_cols = ['Weather', 'Traffic_Level', 'Vehicle_Type','Distance_Category', 'Experience_Group', 'Delay_Status']

le_dict = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    le_dict[col] = le  # Save encoder for future use

print(df.head())
```




```
   Order_ID  Distance_km  Weather  Traffic_Level Time_of_Day  Vehicle_Type  \
0       522         7.93        4              1   Afternoon             2
1       738        16.42        0              2     Evening             0
2       741         9.52        1              1       Night             2
3       661         7.44        2              2   Afternoon             2
4       412        19.03        0              1     Morning             0

   Preparation_Time_min  Courier_Experience_yrs  Delivery_Time_min  \
0                    12                     1.0                 43
1                    20                     2.0                 84
2                    28                     1.0                 59
3                     5                     1.0                 37
4                    16                     5.0                 68

   Distance_Category  Peak_Hour  Experience_Group  Delay_Status
0                  1          0                 0             0
1                  0          1                 0             0
2                  1          1                 0             0
3                  1          0                 0             0
4                  0          0                 2             0
```



```
import seaborn as sns

# ---------------- Correlation Heatmap ----------------
plt.figure(figsize=(12, 8))

# Select only numeric columns
corr = df[['Distance_km',
           'Preparation_Time_min',
           'Courier_Experience_yrs',
           'Weather',
           'Traffic_Level',
           'Vehicle_Type',
           'Distance_Category',
           'Experience_Group',
           'Peak_Hour',
           'Delivery_Time_min',
           'Delay_Status']].corr()

sns.heatmap(
    corr,
    annot=True,
    cmap='coolwarm',
    fmt=".2f",
    linewidths=0.5
)
```