```
D:\Project\Amazon\.venv\Scripts\python.exe D:\Project
\Amazon\main.py
************Amazon Delivery Time Prediction
******************
********Step 1:- Load DataSet**************
        Order_ID  Agent_Age  ...  Delivery_Time
Category
0  ialx566343618         37  ...            120
Clothing
1  akqg208421122         34  ...            165
Electronics
2  njpu434582536         23  ...            130
       Sports
3  rjto796129700         38  ...            105
Cosmetics
4  zguw716275638         32  ...            150
         Toys

[5 rows x 16 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43739 entries, 0 to 43738
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Order_ID          43739 non-null  object
 1   Agent_Age         43739 non-null  int64
 2   Agent_Rating      43685 non-null  float64
 3   Store_Latitude    43739 non-null  float64
 4   Store_Longitude   43739 non-null  float64
 5   Drop_Latitude     43739 non-null  float64
 6   Drop_Longitude    43739 non-null  float64
 7   Order_Date        43739 non-null  object
 8   Order_Time        43739 non-null  object
 9   Pickup_Time       43739 non-null  object
 10  Weather           43648 non-null  object
 11  Traffic           43739 non-null  object
 12  Vehicle           43739 non-null  object
 13  Area              43739 non-null  object
 14  Delivery_Time     43739 non-null  int64
 15  Category          43739 non-null  object
dtypes: float64(5), int64(2), object(9)
```

```
memory usage: 5.3+ MB
None
           Agent_Age  Agent_Rating  ...  Drop_Longitude  Delivery_Time
count  43739.000000  43685.000000  ...    43739.000000   43739.000000
mean      29.567137      4.633780  ...       70.821842     124.905645
std        5.815155      0.334716  ...       21.153148      51.915451
min       15.000000      1.000000  ...        0.010000      10.000000
25%       25.000000      4.500000  ...       73.280000      90.000000
50%       30.000000      4.700000  ...       76.002574     125.000000
75%       35.000000      4.900000  ...       78.104095     160.000000
max       50.000000      6.000000  ...       88.563452     270.000000

[8 rows x 7 columns]
Original Columns ['Order_ID', 'Agent_Age', 'Agent_Rating', 'Store_Latitude', 'Store_Longitude', 'Drop_Latitude', 'Drop_Longitude', 'Order_Date', 'Order_Time', 'Pickup_Time', 'Weather', 'Traffic', 'Vehicle', 'Area', 'Delivery_Time', 'Category']
**********Step 2A :- Handling missing values*********
                 Missig Values  %Missing Percentage
Order_ID                     0             0.000000
Agent_Age                    0             0.000000
Agent_Rating                54             0.123460
Store_Latitude               0             0.000000
Store_Longitude              0             0.000000
Drop_Latitude                0             0.000000
Drop_Longitude               0             0.000000
Order_Date                   0             0.000000
Order_Time                   0             0.000000
Pickup_Time                  0             0.000000
Weather                     91             0.208052
Traffic                      0             0.000000
```

```
Vehicle                           0            0.000000
Area                              0            0.000000
Delivery_Time                     0            0.000000
Category                          0            0.000000
df2 = Case 1 Coulumn is 100% missing
        Order_ID  Agent_Age  ...  Delivery_Time     Category
0  ialx566343618         37  ...            120     Clothing
1  akqg208421122         34  ...            165  Electronics
2  njpu434582536         23  ...            130       Sports
3  rjto796129700         38  ...            105    Cosmetics
4  zguw716275638         32  ...            150         Toys

[5 rows x 16 columns]
Columns After Removing Missing Values=  ['Order_ID', 'Agent_Age', 'Agent_Rating', 'Store_Latitude', 'Store_Longitude', 'Drop_Latitude', 'Drop_Longitude', 'Order_Date', 'Order_Time', 'Pickup_Time', 'Weather', 'Traffic', 'Vehicle', 'Area', 'Delivery_Time', 'Category']
Original Columns ['Order_ID', 'Agent_Age', 'Agent_Rating', 'Store_Latitude', 'Store_Longitude', 'Drop_Latitude', 'Drop_Longitude', 'Order_Date', 'Order_Time', 'Pickup_Time', 'Weather', 'Traffic', 'Vehicle', 'Area', 'Delivery_Time', 'Category']
After comparing it is conclude that dataset is having zero column having 100% missing value
Case 2 Column Is mostly missing >50%
***********Step 2B:- Fill Missing Values************
Agent_rating(Numeric Column)==using median(safer than mean if outliers exist
Weather(Categorical Column)==using mode
Order_ID          0
Agent_Age         0
Agent_Rating      0
Store_Latitude    0
```

```
Store_Longitude    0
Drop_Latitude      0
Drop_Longitude     0
Order_Date         0
Order_Time         0
Pickup_Time        0
Weather            0
Traffic            0
Vehicle            0
Area               0
Delivery_Time      0
Category           0
dtype: int64
Missing values handled successfully!
Remaining Nulls After Filling:
 Order_ID           0
Agent_Age          0
Agent_Rating       0
Store_Latitude     0
Store_Longitude    0
Drop_Latitude      0
Drop_Longitude     0
Order_Date         0
Order_Time         0
Pickup_Time        0
Weather            0
Traffic            0
Vehicle            0
Area               0
Delivery_Time      0
Category           0
dtype: int64
***************Step 3 Handling Duplicates **********************8
Total Duplicates:  0
Empty DataFrame
Columns: [Order_ID, Agent_Age, Agent_Rating, Store_Latitude, Store_Longitude, Drop_Latitude, Drop_Longitude, Order_Date, Order_Time, Pickup_Time, Weather, Traffic, Vehicle, Area, Delivery_Time, Category]
```

```
Index: []
NO duplicates found.
**********Step 4 Feature Engineering*********
Euclidean distance is enough unless exact km/miles are needed.
   Store_Latitude  Store_Longitude  Drop_Latitude  Distance
0       22.745049        75.892471      22.765049  0.028284
1       12.913041        77.683237      13.043041  0.183848
2       12.914264        77.678400      12.924264  0.014142
3       11.003669        76.976494      11.053669  0.070711
4       12.972793        80.249982      13.012793  0.056569
===============Histogram of Eucledean distance ===============
Step 4B: Haversine gives a realistic feature instead  of a flat approximation.This can improve  model’s accuracy.
   Distance_km
0     3.025149
1    20.183530
2     1.552758
3     7.790401
4     6.210138
=================Histogram of Haversine distances (in km)==================
**********Step 5: Time-Based Feature Engineering*********
  Order_Date  Order_Weekday  Is_Weekend  Order_Hour  Pickup_Hour  Rush_Hour
0 2022-03-19              5           1        11.0           11          0
1 2022-03-25              4           0        19.0           19          1
2 2022-03-19              5           1         8.0            8          1
3 2022-04-05              1           0        18.0
```

```
            18          1
4 2022-03-26              5            1        13.0             13          0
**********Step 6: Pickup Delay Feature*********
Pickup Delay feature measures how long it takes for the agent to actually pick up the parcel after the order is placed.
       Order_DateTime     Pickup_DateTime  Pickup_Delay_Minutes
0 2022-03-19 11:30:00 2022-03-19 11:45:00                  15.0
1 2022-03-25 19:45:00 2022-03-25 19:50:00                   5.0
2 2022-03-19 08:30:00 2022-03-19 08:45:00                  15.0
3 2022-04-05 18:00:00 2022-04-05 18:10:00                  10.0
4 2022-03-26 13:30:00 2022-03-26 13:45:00                  15.0
*************Step 7: one hot encoding*************
Train Score: 0.9729247483785157
Test Score: 0.8091544277212389
Step 8 :- EDA
Boxplot to check outliers
Scatter plot vs delivery time
=======Distribution of Delivery_Time (target variable)=========
==========Numeric Features: Agent_Age, Agent_Rating, Distance_km, Pickup_Delay_Minutes
Boxplots to detect outliers
========Correlation Heatmap===========
Categorical Features Visualization
D:\Project\Amazon\main.py:211: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=col, order=df[col].value_counts().index, palette='Set2')
```

```
D:\Project\Amazon\main.py:211: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=col, order=df[col].value_counts().index, palette='Set2')
D:\Project\Amazon\main.py:211: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=col, order=df[col].value_counts().index, palette='Set2')
D:\Project\Amazon\main.py:211: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=col, order=df[col].value_counts().index, palette='Set2')
D:\Project\Amazon\main.py:211: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x=col, order=df[col].value_counts().index, palette='Set2')
Feature Relationships

Process finished with exit code 0

```