In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv(r"C:\Users\VICTUS\Desktop\Food_Delivery_Times (1).csv")
```

In [3]:
```
df.head()
```

Out[3]:

| | Order_ID | Distance_km | Weather | Traffic_Level | Time_of_Day | Vehicle_Type | Preparation_Time_min | Courier_Experience_yrs | Delivery_Time |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 522 | 7.93 | Windy | Low | Afternoon | Scooter | 12 | 1.0 | |
| **1** | 738 | 16.42 | Clear | Medium | Evening | Bike | 20 | 2.0 | |
| **2** | 741 | 9.52 | Foggy | Low | Night | Scooter | 28 | 1.0 | |
| **3** | 661 | 7.44 | Rainy | Medium | Afternoon | Scooter | 5 | 1.0 | |
| **4** | 412 | 19.03 | Clear | Low | Morning | Bike | 16 | 5.0 | |

In [4]:
```
df.info()
```

```
<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
```

In [5]:
```
df.describe()
```

Out[5]:

| | Order_ID | Distance_km | Preparation_Time_min | Courier_Experience_yrs | Delivery_Time_min |
|---|---|---|---|---|---|
| **count** | 1000.000000 | 1000.000000 | 1000.000000 | 970.000000 | 1000.000000 |
| **mean** | 500.500000 | 10.059970 | 16.982000 | 4.579381 | 56.732000 |
| **std** | 288.819436 | 5.696656 | 7.204553 | 2.914394 | 22.070915 |
| **min** | 1.000000 | 0.590000 | 5.000000 | 0.000000 | 8.000000 |
| **25%** | 250.750000 | 5.105000 | 11.000000 | 2.000000 | 41.000000 |
| **50%** | 500.500000 | 10.190000 | 17.000000 | 5.000000 | 55.500000 |
| **75%** | 750.250000 | 15.017500 | 23.000000 | 7.000000 | 71.000000 |
| **max** | 1000.000000 | 19.990000 | 29.000000 | 9.000000 | 153.000000 |

In [6]:
```
df.shape
```

Out[6]:
```
(1000, 9)
```

In [7]:
```
df.duplicated()
```

Out[7]:
```
0      False
1      False
2      False
3      False
4      False
       ...
995    False
996    False
997    False
998    False
999    False
Length: 1000, dtype: bool
```

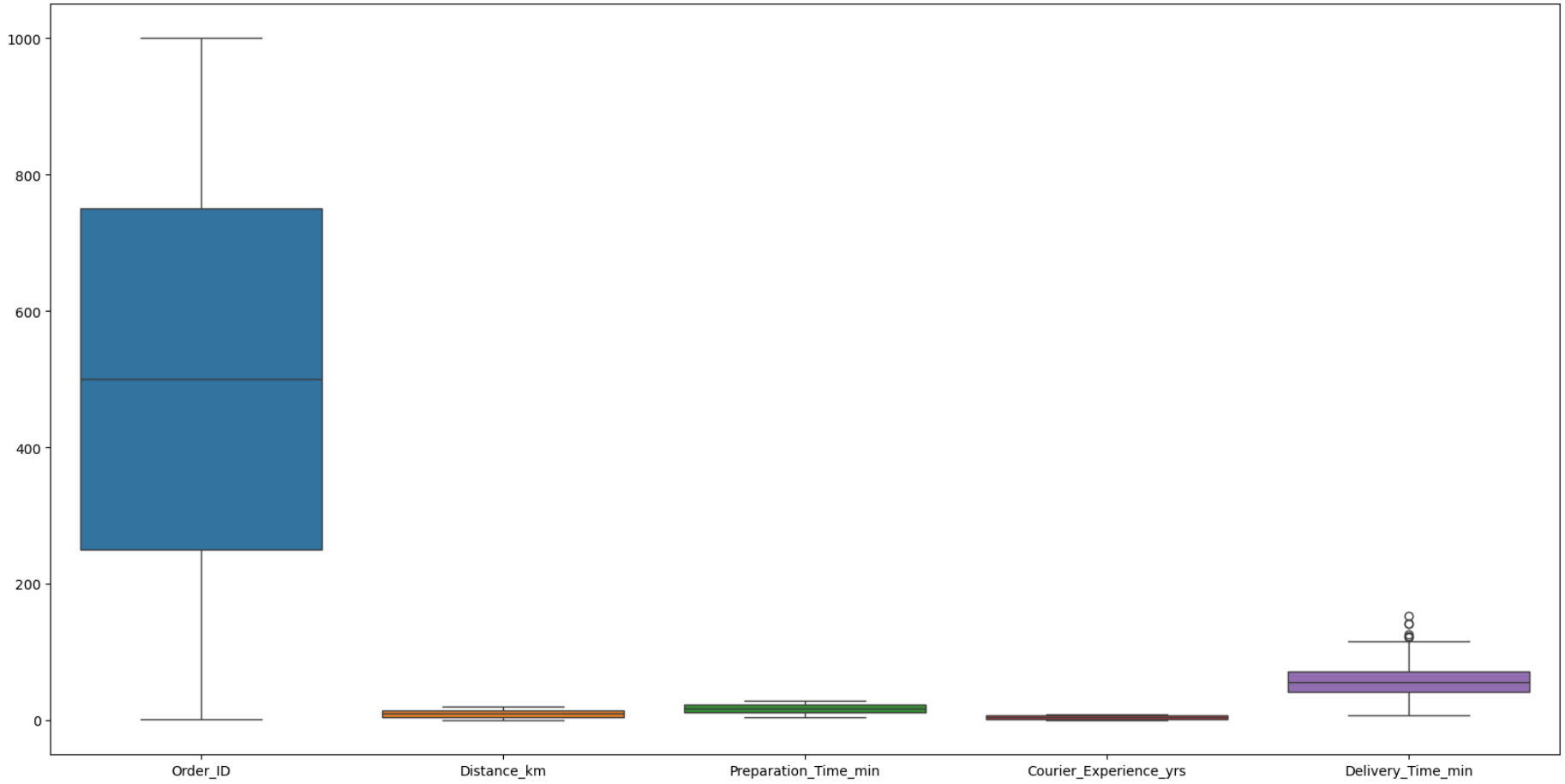
In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
def outliertreat(df,col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    UL = Q3 + IQR
    LL = Q1 - IQR
    df.loc[df[col]>UL,col] = df[col].median()
    df.loc[df[col]<LL,col] = df[col].median()
```

In [10]:
```
plt.figure(figsize=(20,10))
sns.boxplot(df)
```

```
plt.show()
```



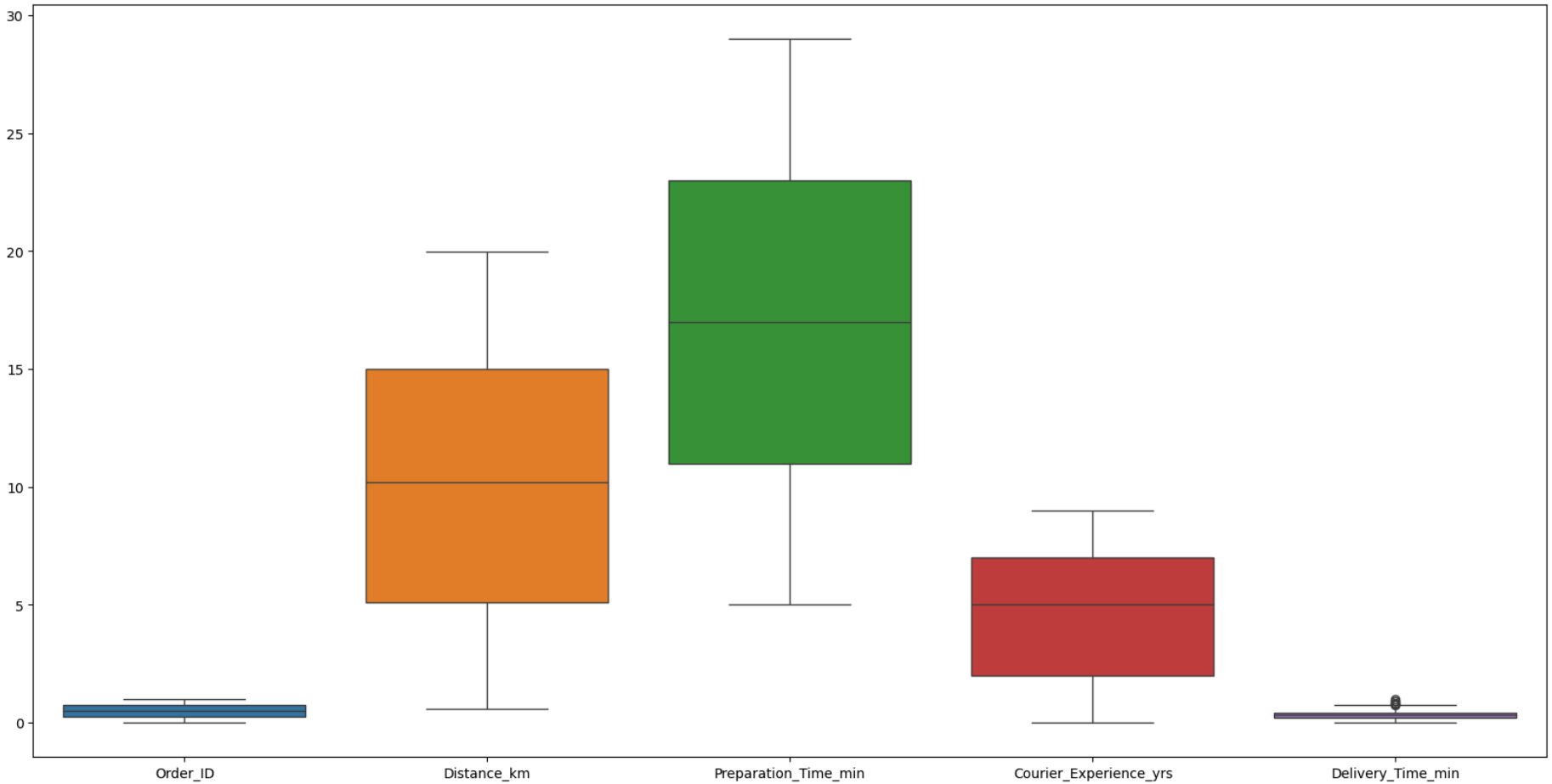

In [11]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [12]:
```
MS = MinMaxScaler()
```

In [13]:
```
df['Order_ID'] = MS.fit_transform(df[['Order_ID']])
```

In [14]:
```
df['Delivery_Time_min'] = MS.fit_transform(df[['Delivery_Time_min']])
```

In [15]:
```
plt.figure(figsize=(20,10))
sns.boxplot(df)
plt.show()
```




In [16]:
```
from sklearn.preprocessing import LabelEncoder
```

In [17]:
```
LE = LabelEncoder()
```

In [18]:
```
df['Weather'] = LE.fit_transform(df['Weather'])
```

In [19]:
```
df['Traffic_Level'] = LE.fit_transform(df['Traffic_Level'])
```

In [20]:
```
df['Time_of_Day'] = LE.fit_transform(df['Time_of_Day'])
```

In [21]:
```
df['Vehicle_Type'] = LE.fit_transform(df['Vehicle_Type'])
```

In [22]:
```
df.head()
```

Out[22]:

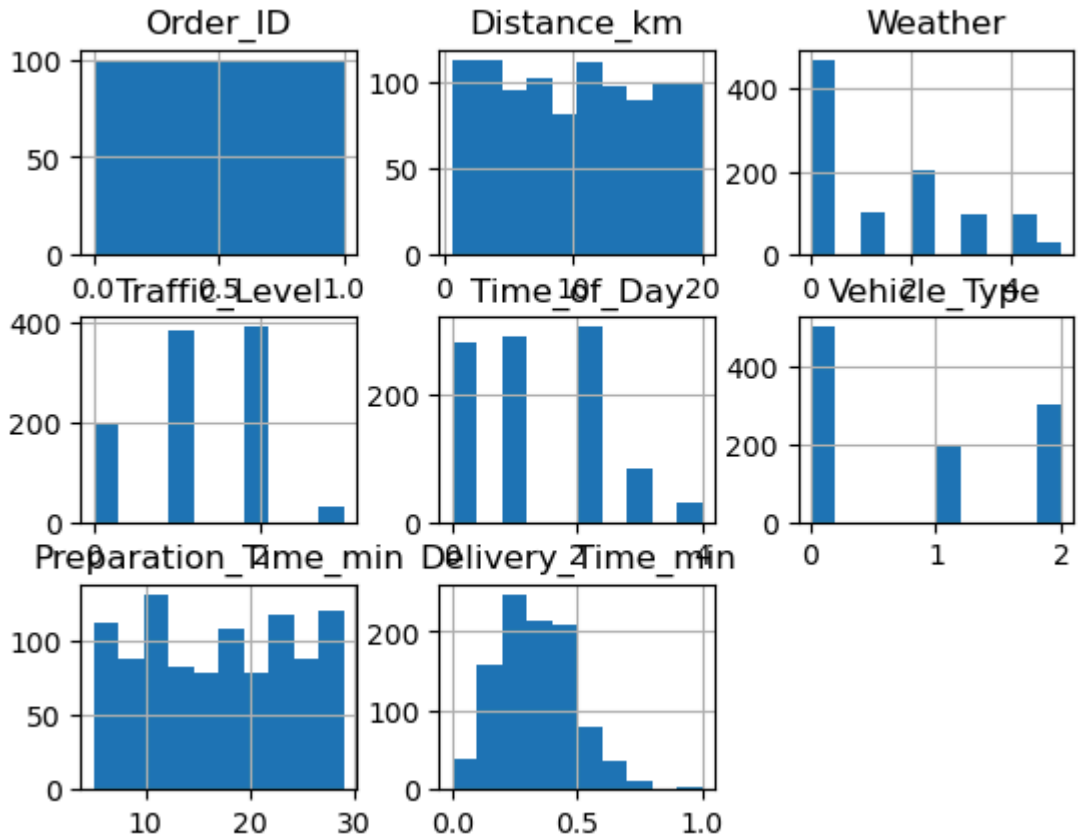
| | Order_ID | Distance_km | Weather | Traffic_Level | Time_of_Day | Vehicle_Type | Preparation_Time_min | Courier_Experience_yrs | Delivery_Time |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.521522 | 7.93 | 4 | 1 | 0 | 2 | 12 | 1.0 | 0.24 |
| **1** | 0.737738 | 16.42 | 0 | 2 | 1 | 0 | 20 | 2.0 | 0.52 |
| **2** | 0.740741 | 9.52 | 1 | 1 | 3 | 2 | 28 | 1.0 | 0.35 |
| **3** | 0.660661 | 7.44 | 2 | 2 | 0 | 2 | 5 | 1.0 | 0.20 |
| **4** | 0.411411 | 19.03 | 0 | 1 | 2 | 0 | 16 | 5.0 | 0.41 |

In [23]:
```
df.drop(columns=['Courier_Experience_yrs'], inplace=True)
```

In [24]:
```
df.isnull().sum()
```

Out[24]:
```
Order_ID                0
Distance_km             0
Weather                 0
Traffic_Level           0
Time_of_Day             0
Vehicle_Type            0
Preparation_Time_min    0
Delivery_Time_min       0
dtype: int64
```

In [25]:
```
df.hist()
plt.show()
```




In [26]:
```
df.skew(numeric_only=True)
```

Out[26]:
```
Order_ID               -1.096752e-15
Distance_km             3.884047e-02
Weather                 7.708209e-01
Traffic_Level          -1.383483e-01
Time_of_Day             4.518274e-01
Vehicle_Type            4.017149e-01
Preparation_Time_min    3.000816e-02
Delivery_Time_min       5.072512e-01
dtype: float64
```

In [27]:
```
from sklearn.preprocessing import PowerTransformer
```

In [28]:
```
PT = PowerTransformer()
```

In [29]:
```
PT.fit_transform(df[['Traffic_Level','Preparation_Time_min','Delivery_Time_min']])
```

Out[29]:
```
array([[-0.31438592, -0.64489576, -0.57624906],
       [ 0.93145906,  0.46559741,  1.21564784],
       [-0.31438592,  1.44926602,  0.19712054],
       ...,
       [-1.56398662,  1.21220682,  1.10374233],
       [-0.31438592, -1.27912648,  0.01385886],
       [-0.31438592,  0.96965152,  0.15189201]])
```

In [30]:
```
df[['Traffic_Level','Preparation_Time_min','Delivery_Time_min',]] = PT.fit_transform(df[['Traffic_Level','Preparation_Time_min
```

In [31]:
```
df.skew(numeric_only=True)
```

Out[31]:
```
Order_ID               -1.096752e-15
Distance_km             3.884047e-02
Weather                 7.708209e-01
Traffic_Level          -1.419601e-01
Time_of_Day             4.518274e-01
Vehicle_Type            4.017149e-01
Preparation_Time_min   -1.285052e-01
Delivery_Time_min       8.900913e-03
dtype: float64
```

In [32]:
```
df.corr(numeric_only=True)
```

Out[32]:

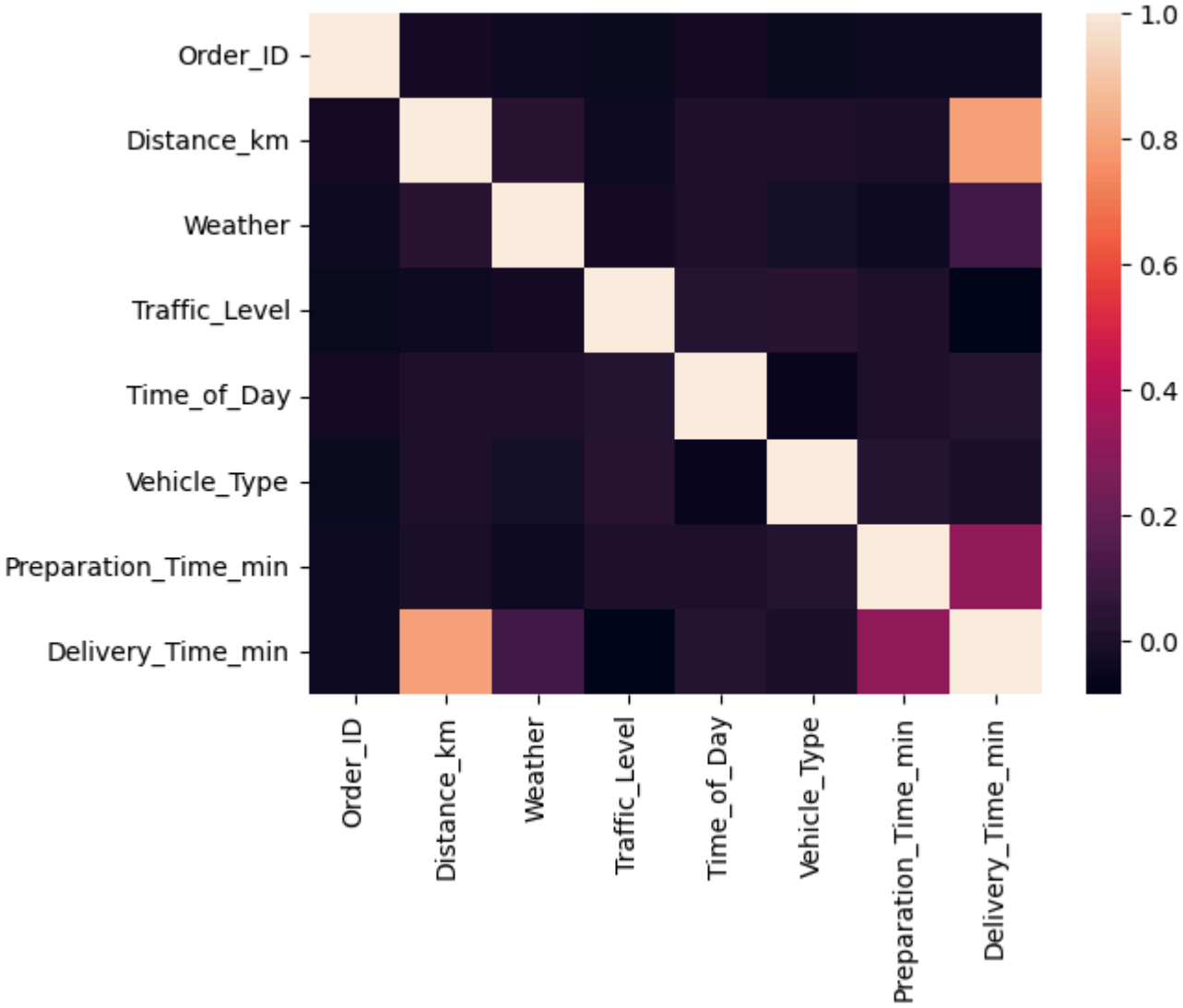
| | Order_ID | Distance_km | Weather | Traffic_Level | Time_of_Day | Vehicle_Type | Preparation_Time_min | Delivery_Time_m |
|---|---|---|---|---|---|---|---|---|
| **Order_ID** | 1.000000 | -0.024483 | -0.035785 | -0.050806 | -0.027034 | -0.045030 | -0.035859 | -0.0328 |
| **Distance_km** | -0.024483 | 1.000000 | 0.029756 | -0.036625 | 0.009034 | 0.003319 | -0.012127 | 0.7984 |
| **Weather** | -0.035785 | 0.029756 | 1.000000 | -0.031329 | 0.006595 | -0.019231 | -0.037662 | 0.1097 |
| **Traffic_Level** | -0.050806 | -0.036625 | -0.031329 | 1.000000 | 0.022548 | 0.032533 | 0.004215 | -0.0866 |
| **Time_of_Day** | -0.027034 | 0.009034 | 0.006595 | 0.022548 | 1.000000 | -0.054988 | 0.004447 | 0.0164 |
| **Vehicle_Type** | -0.045030 | 0.003319 | -0.019231 | 0.032533 | -0.054988 | 1.000000 | 0.018661 | -0.0042 |
| **Preparation_Time_min** | -0.035859 | -0.012127 | -0.037662 | 0.004215 | 0.004447 | 0.018661 | 1.000000 | 0.3190 |
| **Delivery_Time_min** | -0.032890 | 0.798499 | 0.109796 | -0.086679 | 0.016430 | -0.004227 | 0.319076 | 1.0000 |

In [33]:

```
sns.heatmap(df.corr(numeric_only=True))
```

Out[33]:

```
<Axes: >
```




In [34]:
```
from sklearn.model_selection import train_test_split
```

In [35]:
```
X = df.drop('Delivery_Time_min',axis=1)
y = df.Delivery_Time_min
```

In [36]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=42)
```

In [37]:
```
from sklearn.linear_model import LinearRegression
```

In [38]:
```
LR = LinearRegression()
```

In [39]:
```
LR.fit(X_train, y_train)
```

Out[39]:
```
▾ LinearRegression
LinearRegression()
```

In [40]:
```
LR_pred = LR.predict(X_test)
```

In [41]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [42]:
```
mean_absolute_error(y_test,LR_pred)
```

Out[42]:
```
0.3354891852033775
```

In [43]:
```
mean_squared_error(y_test,LR_pred)
```

Out[43]:
```
0.21446556795516905
```

In [44]:
```
r2_score(y_test, LR_pred)
```

Out[44]:
```
0.793097088496856
```

In [45]:
```
LR.score(X_train,y_train)*100
```

Out[45]:
```
74.54719561652814
```

In [46]:
```
LR.score(X_test,y_test)+
```

```
  Cell In[46], line 1
    LR.score(X_test,y_test)+
                           ^
SyntaxError: invalid syntax
```

In [ ]:
```
from sklearn.neighbors import KNeighborsRegressor
```

In [ ]:
```
KNR = KNeighborsRegressor(n_neighbors=7)
```

In [ ]:
```
KNR.fit(X_train,y_train)
```

In [ ]:
```
KNR_pred = KNR.predict(X_test)
```

In [ ]:
```
mean_absolute_error(y_test,KNR_pred)
```

In [ ]:
```
mean_squared_error(y_test,LR_pred)
```

In [ ]:
```
r2_score(y_test, KNR_pred)
```

In [ ]:
```
KNR.score(X_train,y_train)*100
```

In [ ]:
```
KNR.score(X_test,y_test)
```

In [ ]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [ ]:
```
DTC = DecisionTreeRegressor(random_state=42)
```

In [ ]:
```
DTC.fit(X_train , y_train)
```

In [ ]:
```
y_pred_dt = DTC.predict(X_test)
```

In [ ]:
```
mean_absolute_error(y_test, y_pred_dt)
```

In [ ]:
```
mean_squared_error(y_test, y_pred_dt)
```

In [ ]:
```
r2_score(y_test,y_pred_dt )
```

In [ ]:
```
DTC.score(X_train,y_train)*100
```

In [ ]:
```
DTC.score(X_test,y_test)
```

In [ ]:

In [ ]:
```
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
```

In [ ]:
```
RFR=RandomForestRegressor(n_estimators=100, random_state=42)
```

In [ ]:
```
RFR.fit(X_train, y_train)
```

In [ ]:
```
RFR_pred = RFR.predict(X_test)
```

In [ ]:
```
mean_absolute_error(y_test, RFR_pred)
```

In [ ]:
```
mean_squared_error(y_test, RFR_pred)
```

In [ ]:
```
r2_score(y_test, RFR_pred)
```

In [ ]:
```
RFR.score(X_train,y_train)*100
```

In [ ]:
```
RFR.score(X_test,y_test)
```

In [ ]:
```
GBL = GradientBoostingRegressor(random_state=42, n_estimators=100, learning_rate=0.1)
```

In [ ]:
```
GBL.fit(X_train, y_train)
```

In [ ]:
```
gbl_pred = GBL.predict(X_test)
```

In [ ]:
```
mean_squared_error(y_test, gbl_pred)
```

In [ ]:
```
mean_absolute_error(y_test, gbl_pred)
```

In [ ]:
```
r2_score(y_test,gbl_pred)
```

In [ ]:
```
GBL.score(X_train,y_train)*100
```

In [ ]:
```
GBL.score(X_test,y_test)
```

In [60]:
```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

In [56]:
```
param_grid = {
    'n_estimators': [50,100,200],
    'max_depth' : [None, 10,20,30],
}
```

In [58]:
```
rf_model = RandomForestRegressor(random_state=42)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[58], line 1
----> 1 rf_model = RandomForestRegressor(random_state=42)

NameError: name 'RandomForestRegressor' is not defined
```

In [ ]: