In [1]:

```
import pandas as pd
import numpy as np
import plotly.express as px
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, LSTM
```

In [2]:

```
url = 'https://raw.githubusercontent.com/ataislucky/Data-Science/main/dataset/food_
data = pd.read_csv(url)
data.sample(5)
```

Out[2]:

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurar
31784	C5E2	ALHRES08DEL01	31	4.6	
21440	D8AE	GOARES20DEL01	31	4.3	
17503	99C	INDORES16DEL01	34	5.0	
15976	55DA	JAPRES15DEL01	25	4.6	
34454	619D	RANCHIRES15DEL01	29	4.6	

In [3]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45593 entries, 0 to 45592
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ID                           45593 non-null  object 
 1   Delivery_person_ID           45593 non-null  object 
 2   Delivery_person_Age          45593 non-null  int64  
 3   Delivery_person_Ratings      45593 non-null  float64
 4   Restaurant_latitude          45593 non-null  float64
 5   Restaurant_longitude         45593 non-null  float64
 6   Delivery_location_latitude   45593 non-null  float64
 7   Delivery_location_longitude  45593 non-null  float64
 8   Type_of_order                45593 non-null  object 
 9   Type_of_vehicle              45593 non-null  object 
 10  Time_taken(min)              45593 non-null  int64  
dtypes: float64(5), int64(2), object(4)
memory usage: 3.8+ MB
```

In [4]:

```
data.isnull().sum()
```

Out[4]:

```
ID                             0
Delivery_person_ID             0
Delivery_person_Age            0
Delivery_person_Ratings        0
Restaurant_latitude            0
Restaurant_longitude           0
Delivery_location_latitude     0
Delivery_location_longitude    0
Type_of_order                  0
Type_of_vehicle                0
Time_taken(min)                0
dtype: int64
```

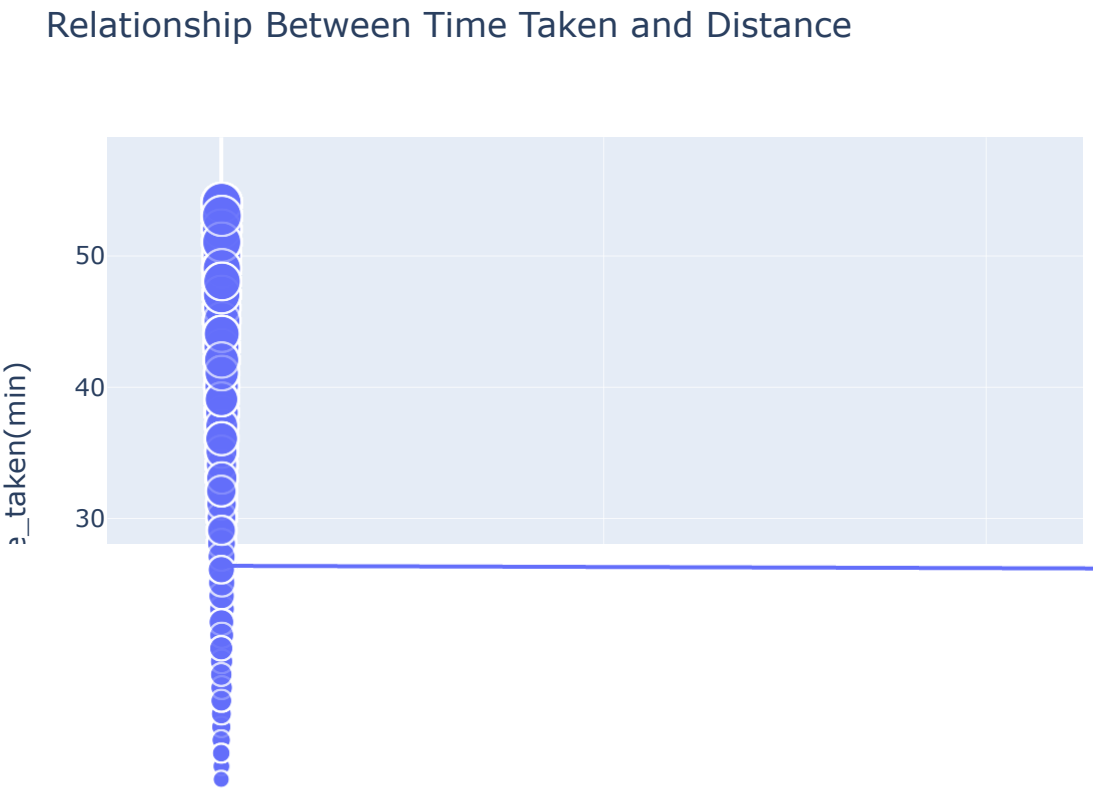
```
Haversine formula is used to find the distance between two geographical locations
```

In [6]:

```
R = 6371  ##The earth's radius (in km)

def deg_to_rad(degrees):
    return degrees * (np.pi/180)

## The haversine formula
def distcalculate(lat1, lon1, lat2, lon2):
    d_lat = deg_to_rad(lat2-lat1)
    d_lon = deg_to_rad(lon2-lon1)
    a1 = np.sin(d_lat/2)**2 + np.cos(deg_to_rad(lat1))
    a2 = np.cos(deg_to_rad(lat2)) * np.sin(d_lon/2)**2
    a = a1 * a2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))
    return R * c

# Create distance column & calculate the distance
data['distance'] = np.nan

for i in range(len(data)):
  data.loc[i, 'distance'] = distcalculate(data.loc[i, 'Restaurant_latitude'],
                                          data.loc[i, 'Restaurant_longitude'],
                                          data.loc[i, 'Delivery_location_latitude']
                                          data.loc[i, 'Delivery_location_longitude'
```

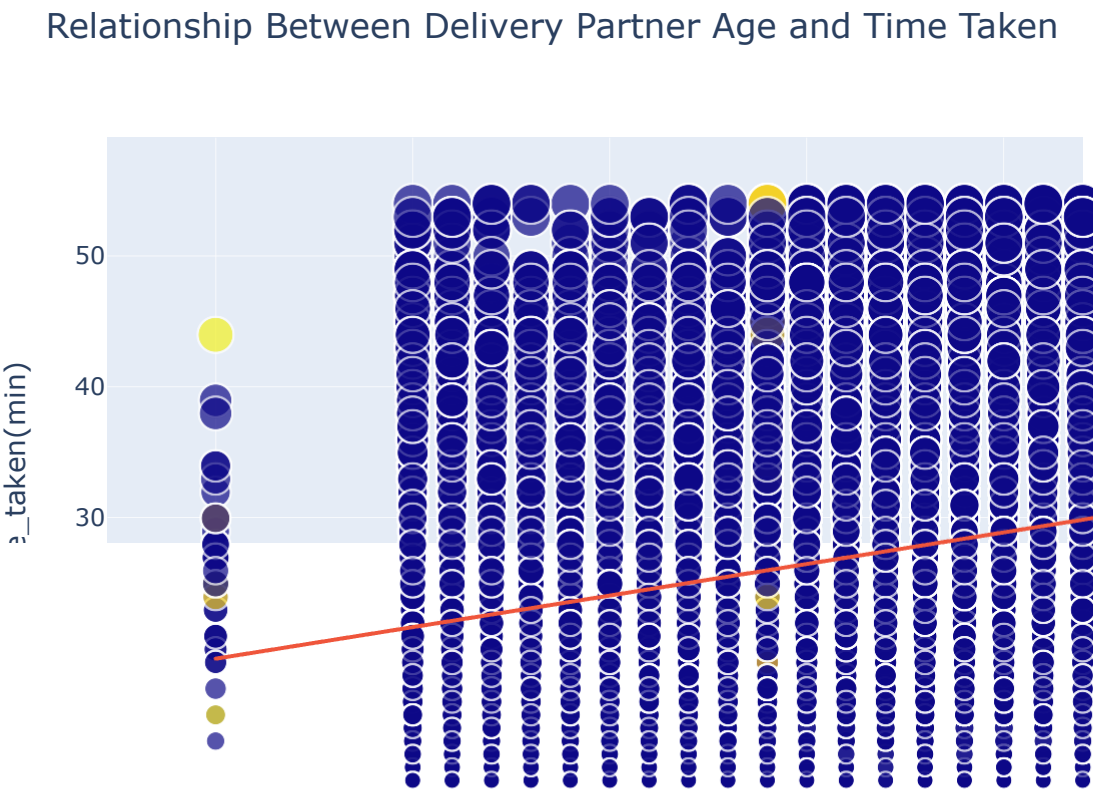
In [7]:

```
figure = px.scatter(data_frame = data,
                    x="distance",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    trendline="ols",
                    title = "Relationship Between Time Taken and Distance")
figure.show()
```



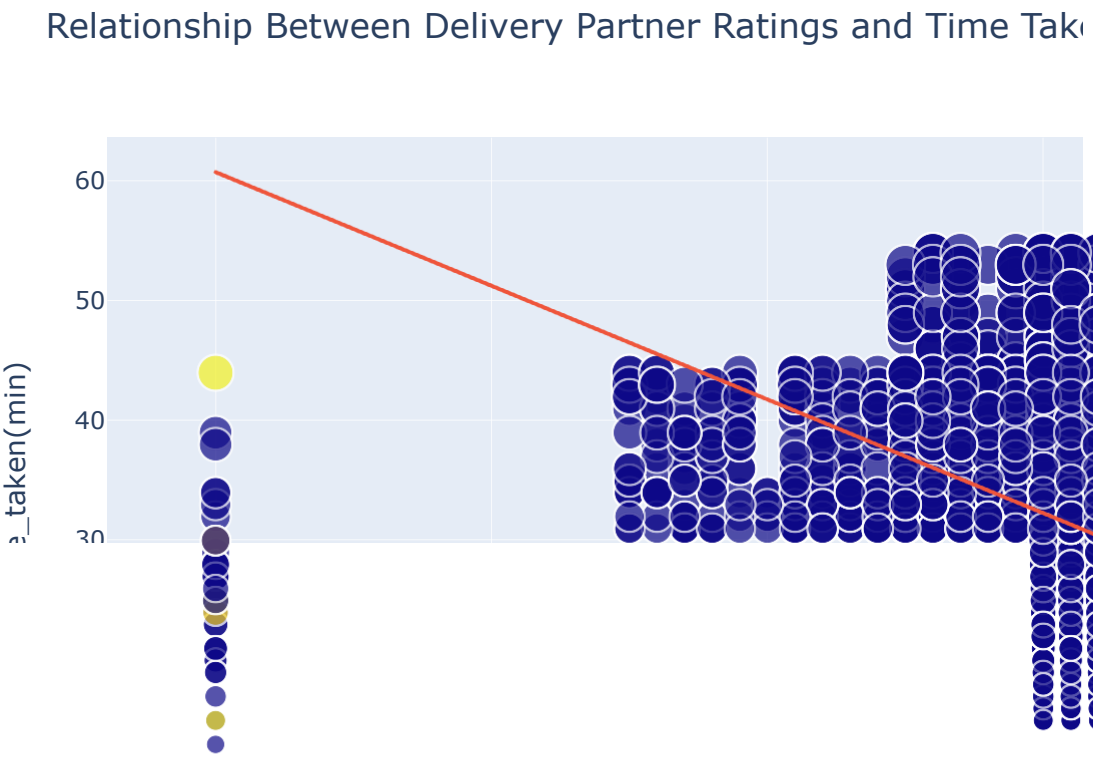
In [8]:

```
figure = px.scatter(data_frame = data,
                    x="Delivery_person_Age",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Delivery Partner Age and Time Tak
figure.show()
```



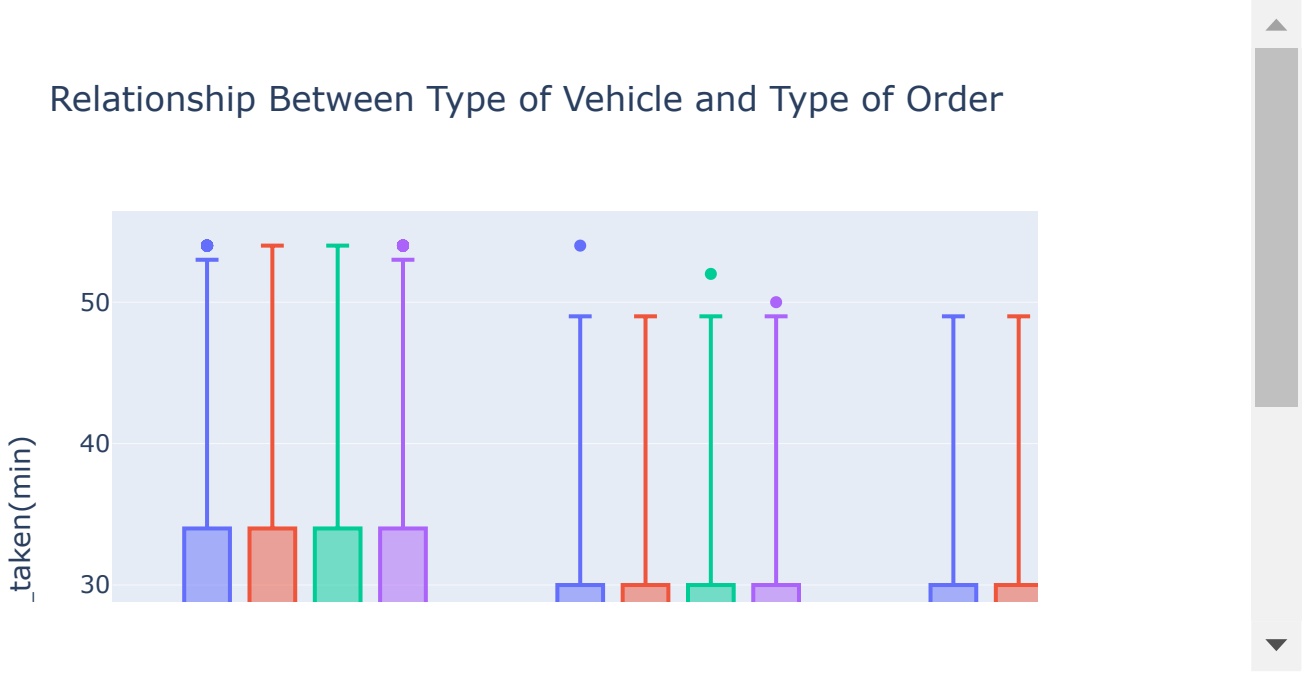
In [9]:

```
figure = px.scatter(data_frame = data,
                    x="Delivery_person_Ratings",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Delivery Partner Ratings and Time
figure.show()
```



In [10]:

```
fig = px.box(data,
             x="Type_of_vehicle",
             y="Time_taken(min)",
             color="Type_of_order",
             title = "Relationship Between Type of Vehicle and Type of Order")
fig.show()
```




```
Build an LSTM Model and Make Predictions
```

In [11]:

```
x = np.array(data[["Delivery_person_Age",
                   "Delivery_person_Ratings",
                   "distance"]])
y = np.array(data[["Time_taken(min)"]])
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.20,
                                                random_state=33)
```

In [12]:

```
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 3, 128)            66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 117,619
Trainable params: 117,619
Non-trainable params: 0
_________________________________________________________________
```

In [13]:

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=1, epochs=9)
```

```
Epoch 1/9
36474/36474 [==============================] - 217s 6ms/step - loss: 69.3
614
Epoch 2/9
36474/36474 [==============================] - 211s 6ms/step - loss: 64.7
297
Epoch 3/9
36474/36474 [==============================] - 210s 6ms/step - loss: 62.2
265
Epoch 4/9
36474/36474 [==============================] - 210s 6ms/step - loss: 60.9
183
Epoch 5/9
36474/36474 [==============================] - 212s 6ms/step - loss: 60.1
418
Epoch 6/9
36474/36474 [==============================] - 227s 6ms/step - loss: 59.7
417
Epoch 7/9
36474/36474 [==============================] - 227s 6ms/step - loss: 59.4
175
Epoch 8/9
36474/36474 [==============================] - 227s 6ms/step - loss: 59.1
459
Epoch 9/9
36474/36474 [==============================] - 227s 6ms/step - loss: 59.0
887
```

Out[13]:

```
<keras.callbacks.History at 0x1af96f9ffd0>
```

In [14]:

```
print("Food Delivery Time Prediction using LSTM")
a = int(input("Delivery Partner Age: "))
b = float(input("Previous Delivery Ratings: "))
c = int(input("Total Distance: "))

features = np.array([[a, b, c]])
print("Delivery Time Prediction in Minutes = ", model.predict(features))
```

```
Food Delivery Time Prediction using LSTM
Delivery Partner Age: 21
Previous Delivery Ratings: 4
Total Distance: 120
1/1 [==============================] - 1s 1s/step
Delivery Time Prediction in Minutes =  [[23.78782]]
```

In [ ]:

```

```