In [ ]:
```
# Problem Statement

The central objective is to develop a regression model to accurately predict food delivery times for Porter.
The model will use various factors such as order details, restaurant location,
and the availability of delivery partners to generate precise estimates for customers.
```

In [11]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
```

In [12]:
```
# load the dataset
df = pd.read_csv('Porter_data.csv')
```

In [13]:
```
# Preview first few rows
df.head()
```

Out[13]:

	market_id	created_at	actual_delivery_time	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	ma
0	1.0	2015-02-06 22:24:17	2015-02-06 23:11:17	4	1.0	4	3441	4	557	
1	2.0	2015-02-10 21:49:25	2015-02-10 22:33:25	46	2.0	1	1900	1	1400	
2	2.0	2015-02-16 00:11:35	2015-02-16 01:06:35	36	3.0	4	4771	3	820	
3	1.0	2015-02-12 03:36:46	2015-02-12 04:35:46	38	1.0	1	1525	1	1525	
4	1.0	2015-01-27 02:12:36	2015-01-27 02:58:36	38	1.0	2	3620	2	1425	

In [14]:
```
# Check shape (rows, columns)
df.shape
```

Out[14]:
```
(175777, 14)
```

In [15]:
```
# Get column info, datatypes, nulls
df.columns
```

Out[15]:
```
Index(['market_id', 'created_at', 'actual_delivery_time',
       'store_primary_category', 'order_protocol', 'total_items', 'subtotal',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_dashers', 'total_busy_dashers',
       'total_outstanding_orders',
       'estimated_store_to_consumer_driving_duration'],
      dtype='object')
```

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 14 columns):
 #   Column                                        Non-Null Count   Dtype
---  ------                                        --------------   -----
 0   market_id                                     175777 non-null  float64
 1   created_at                                    175777 non-null  object
 2   actual_delivery_time                          175777 non-null  object
 3   store_primary_category                        175777 non-null  int64
 4   order_protocol                                175777 non-null  float64
 5   total_items                                   175777 non-null  int64
 6   subtotal                                      175777 non-null  int64
 7   num_distinct_items                            175777 non-null  int64
 8   min_item_price                                175777 non-null  int64
 9   max_item_price                                175777 non-null  int64
 10  total_onshift_dashers                         175777 non-null  float64
 11  total_busy_dashers                            175777 non-null  float64
 12  total_outstanding_orders                      175777 non-null  float64
 13  estimated_store_to_consumer_driving_duration  175777 non-null  float64
dtypes: float64(6), int64(6), object(2)
memory usage: 18.8+ MB
```

In [17]:
```
# Quick stats for numerical features
df.describe()
```

Out[17]:

	market_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	to
count	175777.000000	175777.000000	175777.000000	175777.000000	175777.000000	175777.000000	175777.000000	175777.000000	
mean	2.743726	35.887949	2.911752	3.204976	2697.111147	2.675060	684.965433	1160.158616	
std	1.330963	20.728254	1.513128	2.674055	1828.554893	1.625681	519.882924	560.828571	
min	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	-86.000000	0.000000	
25%	2.000000	18.000000	1.000000	2.000000	1412.000000	1.000000	299.000000	799.000000	
50%	2.000000	38.000000	3.000000	3.000000	2224.000000	2.000000	595.000000	1095.000000	
75%	4.000000	55.000000	4.000000	4.000000	3410.000000	3.000000	942.000000	1395.000000	
max	6.000000	72.000000	7.000000	411.000000	26800.000000	20.000000	14700.000000	14700.000000	

In [ ]:
```
# Explore Columns

# Categorical variables:

market_id, store_primary_category, order_protocol

# Numerical variables:

total_items, subtotal, num_distinct_items, min_item_price, max_item_price,
total_onshift_partners, total_busy_partners, total_outstanding_orders,
estimated_store_to_consumer_driving_duration
```

In [19]:
```
# Separate numerical and categorical columns

numerical_cols = ['total_items', 'subtotal', 'num_distinct_items', 'min_item_price', 'max_item_price', 'total
categorical_cols = ['market_id', 'store_primary_category', 'order_protocol']
time_cols = ['created_at', 'actual_delivery_time']

print("Numerical variables:")
print(numerical_cols)

print("\nCategorical variables:")
print(categorical_cols)

print("\nTime variables:")
print(time_cols)
```

```
Numerical variables:
['total_items', 'subtotal', 'num_distinct_items', 'min_item_price', 'max_item_price', 'total_onshift_partne
rs', 'total_busy_partners', 'total_outstanding_orders', 'estimated_store_to_consumer_driving_duration']

Categorical variables:
['market_id', 'store_primary_category', 'order_protocol']

Time variables:
['created_at', 'actual_delivery_time']
```

In [20]:
```
# Convert timestamps to datetime

df['created_at'] = pd.to_datetime(df['created_at'], errors='coerce')
df['actual_delivery_time'] = pd.to_datetime(df['actual_delivery_time'], errors='coerce')
```

In [21]:
```
# Target variable: time taken for delivery
df['delivery_duration'] = (df['actual_delivery_time'] - df['created_at']).dt.total_seconds() / 60
```

In [22]:
```
# Feature Engineering (Time-based)

# Order hour

df['Order hour'] = df['created_at'].dt.hour

# Day of week (0 = Monday, 6 = Sunday)
df['order_dayofweek'] = df['created_at'].dt.dayofweek

# # Weekend flag
df['is_weekend'] = df['order_dayofweek'].isin([5,6]).astype(int)
```

In [23]:
```
# Null Value Handling

df.isnull().sum()
```

Out[23]:
```
market_id                                   0
created_at                                  0
actual_delivery_time                        0
store_primary_category                      0
order_protocol                              0
total_items                                 0
subtotal                                    0
num_distinct_items                          0
min_item_price                              0
max_item_price                              0
total_onshift_dashers                       0
total_busy_dashers                          0
total_outstanding_orders                    0
estimated_store_to_consumer_driving_duration    0
delivery_duration                           0
Order hour                                  0
order_dayofweek                             0
is_weekend                                  0
dtype: int64
```

In [24]:
```
# Drop rows
df = df.dropna(subset=['created_at','actual_delivery_time','delivery_duration'])
```

In [25]:
```
# Fill missing numerical values
nums_col = df.select_dtypes(['float64', 'int64']).columns

df[nums_col] = df[nums_col].fillna(df[nums_col].median())
```

In [26]:
```
# Fill missing categorical values

cat_col = df.select_dtypes(include=['object']).columns
df[cat_col] = df[cat_col].fillna('Unknown')
```

In [27]:
```
# Encode Categorical Variables

# One-hot encoding for small categories
df = pd.get_dummies(df, columns=['store_primary_category','order_protocol'], drop_first=True)
```

In [ ]:

In [28]:
```
df.columns.tolist()
```

Out[28]:
```
['market_id',
 'created_at',
 'actual_delivery_time',
 'total_items',
 'subtotal',
 'num_distinct_items',
 'min_item_price',
 'max_item_price',
 'total_onshift_dashers',
 'total_busy_dashers',
 'total_outstanding_orders',
 'estimated_store_to_consumer_driving_duration',
 'delivery_duration',
 'Order hour',
 'order_dayofweek',
 'is_weekend',
 'store_primary_category_1',
 'store_primary_category_2',
 'store_primary_category_3',
 'store_primary_category_4',
 'store_primary_category_5',
 'store_primary_category_6',
 'store_primary_category_7',
 'store_primary_category_8',
 'store_primary_category_9',
 'store_primary_category_10',
 'store_primary_category_11',
 'store_primary_category_12',
 'store_primary_category_13',
 'store_primary_category_14',
 'store_primary_category_15',
 'store_primary_category_16',
 'store_primary_category_17',
 'store_primary_category_18',
 'store_primary_category_19',
 'store_primary_category_20',
 'store_primary_category_21',
 'store_primary_category_22',
 'store_primary_category_23',
 'store_primary_category_24',
 'store_primary_category_25',
```

```
                  'store_primary_category_26',
                  'store_primary_category_27',
                  'store_primary_category_28',
                  'store_primary_category_29',
                  'store_primary_category_30',
                  'store_primary_category_31',
                  'store_primary_category_32',
                  'store_primary_category_33',
                  'store_primary_category_34',
                  'store_primary_category_35',
                  'store_primary_category_36',
                  'store_primary_category_37',
                  'store_primary_category_38',
                  'store_primary_category_39',
                  'store_primary_category_40',
                  'store_primary_category_41',
                  'store_primary_category_42',
                  'store_primary_category_43',
                  'store_primary_category_44',
                  'store_primary_category_45',
                  'store_primary_category_46',
                  'store_primary_category_47',
                  'store_primary_category_48',
                  'store_primary_category_49',
                  'store_primary_category_50',
                  'store_primary_category_51',
                  'store_primary_category_52',
                  'store_primary_category_53',
                  'store_primary_category_54',
                  'store_primary_category_55',
                  'store_primary_category_56',
                  'store_primary_category_57',
                  'store_primary_category_58',
                  'store_primary_category_59',
                  'store_primary_category_60',
                  'store_primary_category_61',
                  'store_primary_category_62',
                  'store_primary_category_63',
                  'store_primary_category_64',
                  'store_primary_category_65',
                  'store_primary_category_66',
                  'store_primary_category_67',
```

```
 'store_primary_category_68',
 'store_primary_category_69',
 'store_primary_category_70',
 'store_primary_category_71',
 'store_primary_category_72',
 'order_protocol_2.0',
 'order_protocol_3.0',
 'order_protocol_4.0',
 'order_protocol_5.0',
 'order_protocol_6.0',
 'order_protocol_7.0']
```

In [29]:
```
df.head()
```

Out[29]:

	market_id	created_at	actual_delivery_time	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_dashers	total_
0	1.0	2015-02-06 22:24:17	2015-02-06 23:11:17	4	3441	4	557	1239	33.0	
1	2.0	2015-02-10 21:49:25	2015-02-10 22:33:25	1	1900	1	1400	1400	1.0	
2	2.0	2015-02-16 00:11:35	2015-02-16 01:06:35	4	4771	3	820	1604	8.0	
3	1.0	2015-02-12 03:36:46	2015-02-12 04:35:46	1	1525	1	1525	1525	5.0	
4	1.0	2015-01-27 02:12:36	2015-01-27 02:58:36	2	3620	2	1425	2195	5.0	

5 rows × 94 columns

In [30]:
```
df.info()
```

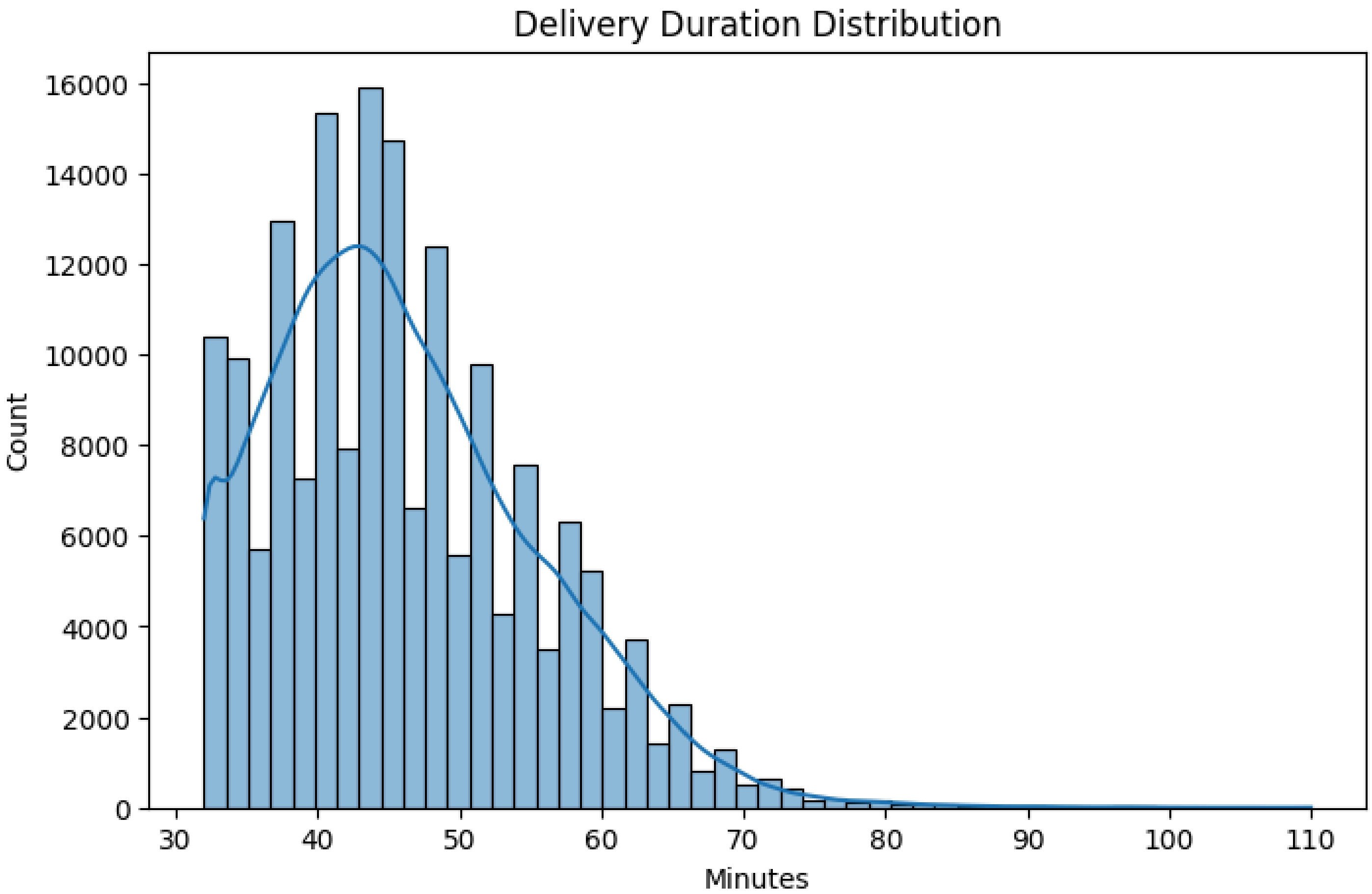
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175777 entries, 0 to 175776
Data columns (total 94 columns):
 #   Column                                    Non-Null Count   Dtype
---  ------                                    --------------   -----
 0   market_id                                 175777 non-null  float64
 1   created_at                                175777 non-null  datetime64[ns]
 2   actual_delivery_time                      175777 non-null  datetime64[ns]
 3   total_items                               175777 non-null  int64
 4   subtotal                                  175777 non-null  int64
 5   num_distinct_items                        175777 non-null  int64
 6   min_item_price                            175777 non-null  int64
 7   max_item_price                            175777 non-null  int64
 8   total_onshift_dashers                     175777 non-null  float64
 9   total_busy_dashers                        175777 non-null  float64
 10  total_outstanding_orders                  175777 non-null  float64
 11  estimated_store_to_consumer_driving_duration 175777 non-null  float64
 12  delivery_duration                         175777 non-null  float64
 13  Order hour                                175777 non-null  int32
 14  order_dayofweek                           175777 non-null  int32
 15  is_weekend                                175777 non-null  int64
 16  store_primary_category_1                  175777 non-null  bool
 17  store_primary_category_2                  175777 non-null  bool
 18  store_primary_category_3                  175777 non-null  bool
 19  store_primary_category_4                  175777 non-null  bool
 20  store_primary_category_5                  175777 non-null  bool
 21  store_primary_category_6                  175777 non-null  bool
 22  store_primary_category_7                  175777 non-null  bool
 23  store_primary_category_8                  175777 non-null  bool
 24  store_primary_category_9                  175777 non-null  bool
 25  store_primary_category_10                 175777 non-null  bool
 26  store_primary_category_11                 175777 non-null  bool
 27  store_primary_category_12                 175777 non-null  bool
 28  store_primary_category_13                 175777 non-null  bool
 29  store_primary_category_14                 175777 non-null  bool
 30  store_primary_category_15                 175777 non-null  bool
 31  store_primary_category_16                 175777 non-null  bool
 32  store_primary_category_17                 175777 non-null  bool
 33  store_primary_category_18                 175777 non-null  bool
 34  store_primary_category_19                 175777 non-null  bool
 35  store_primary_category_20                 175777 non-null  bool
```

```
 36  store_primary_category_21                     175777 non-null  bool
 37  store_primary_category_22                     175777 non-null  bool
 38  store_primary_category_23                     175777 non-null  bool
 39  store_primary_category_24                     175777 non-null  bool
 40  store_primary_category_25                     175777 non-null  bool
 41  store_primary_category_26                     175777 non-null  bool
 42  store_primary_category_27                     175777 non-null  bool
 43  store_primary_category_28                     175777 non-null  bool
 44  store_primary_category_29                     175777 non-null  bool
 45  store_primary_category_30                     175777 non-null  bool
 46  store_primary_category_31                     175777 non-null  bool
 47  store_primary_category_32                     175777 non-null  bool
 48  store_primary_category_33                     175777 non-null  bool
 49  store_primary_category_34                     175777 non-null  bool
 50  store_primary_category_35                     175777 non-null  bool
 51  store_primary_category_36                     175777 non-null  bool
 52  store_primary_category_37                     175777 non-null  bool
 53  store_primary_category_38                     175777 non-null  bool
 54  store_primary_category_39                     175777 non-null  bool
 55  store_primary_category_40                     175777 non-null  bool
 56  store_primary_category_41                     175777 non-null  bool
 57  store_primary_category_42                     175777 non-null  bool
 58  store_primary_category_43                     175777 non-null  bool
 59  store_primary_category_44                     175777 non-null  bool
 60  store_primary_category_45                     175777 non-null  bool
 61  store_primary_category_46                     175777 non-null  bool
 62  store_primary_category_47                     175777 non-null  bool
 63  store_primary_category_48                     175777 non-null  bool
 64  store_primary_category_49                     175777 non-null  bool
 65  store_primary_category_50                     175777 non-null  bool
 66  store_primary_category_51                     175777 non-null  bool
 67  store_primary_category_52                     175777 non-null  bool
 68  store_primary_category_53                     175777 non-null  bool
 69  store_primary_category_54                     175777 non-null  bool
 70  store_primary_category_55                     175777 non-null  bool
 71  store_primary_category_56                     175777 non-null  bool
 72  store_primary_category_57                     175777 non-null  bool
 73  store_primary_category_58                     175777 non-null  bool
 74  store_primary_category_59                     175777 non-null  bool
 75  store_primary_category_60                     175777 non-null  bool
 76  store_primary_category_61                     175777 non-null  bool
 77  store_primary_category_62                     175777 non-null  bool
```

```
 78  store_primary_category_63                          175777 non-null  bool
 79  store_primary_category_64                          175777 non-null  bool
 80  store_primary_category_65                          175777 non-null  bool
 81  store_primary_category_66                          175777 non-null  bool
 82  store_primary_category_67                          175777 non-null  bool
 83  store_primary_category_68                          175777 non-null  bool
 84  store_primary_category_69                          175777 non-null  bool
 85  store_primary_category_70                          175777 non-null  bool
 86  store_primary_category_71                          175777 non-null  bool
 87  store_primary_category_72                          175777 non-null  bool
 88  order_protocol_2.0                                 175777 non-null  bool
 89  order_protocol_3.0                                 175777 non-null  bool
 90  order_protocol_4.0                                 175777 non-null  bool
 91  order_protocol_5.0                                 175777 non-null  bool
 92  order_protocol_6.0                                 175777 non-null  bool
 93  order_protocol_7.0                                 175777 non-null  bool
dtypes: bool(78), datetime64[ns](2), float64(6), int32(2), int64(6)
memory usage: 33.2 MB
```

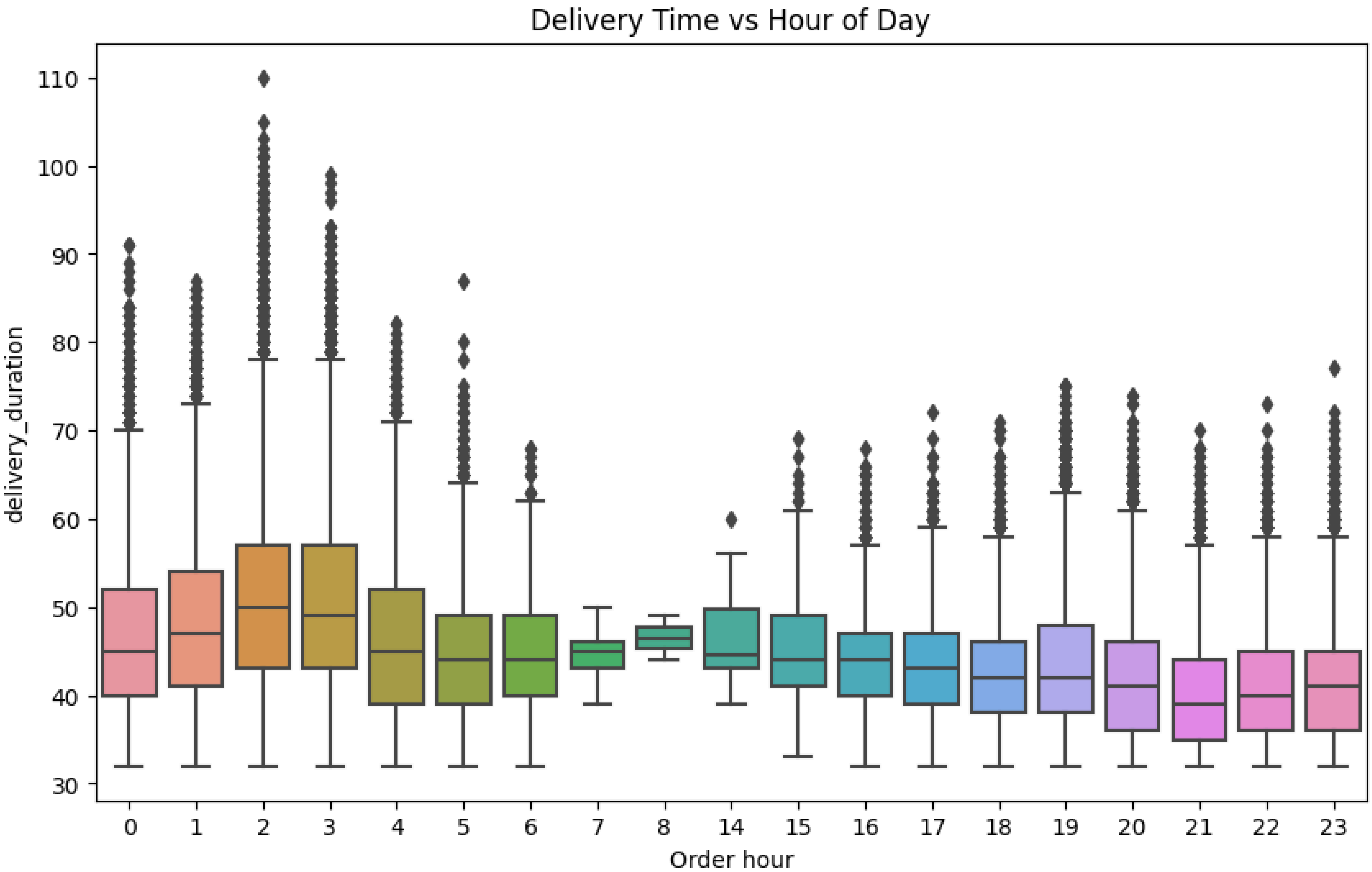
In [31]:
```
# Data Visualization

# Target variable distribution
plt.figure(figsize=(8, 5))
sns.histplot(df['delivery_duration'], bins=50, kde= True)
plt.title('Delivery Duration Distribution')
plt.xlabel('Minutes')
plt.show()
```



In [32]:
```
# Hour of order vs delivery duration

plt.figure(figsize=(10, 6))
sns.boxplot(x ='Order hour', y = 'delivery_duration', data = df)
plt.title('Delivery Time vs Hour of Day')
plt.show()
```

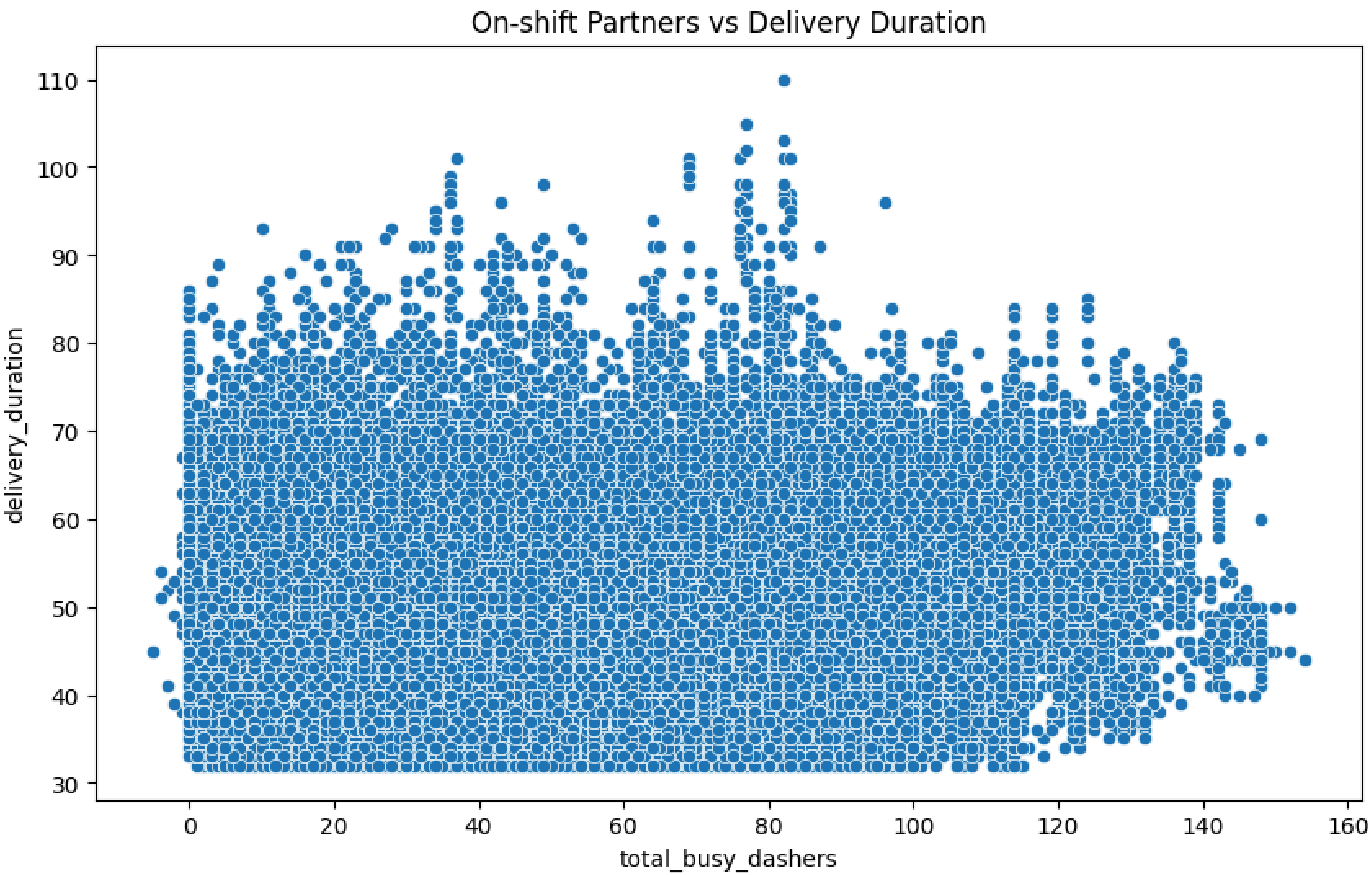


In [33]:
```
# Day of week vs delivery duration

plt.figure(figsize=(10, 6))
sns.boxplot(x = 'order_dayofweek', y ='delivery_duration', data = df)
plt.title('Delivery Time vs Day of Week')
plt.show()
```

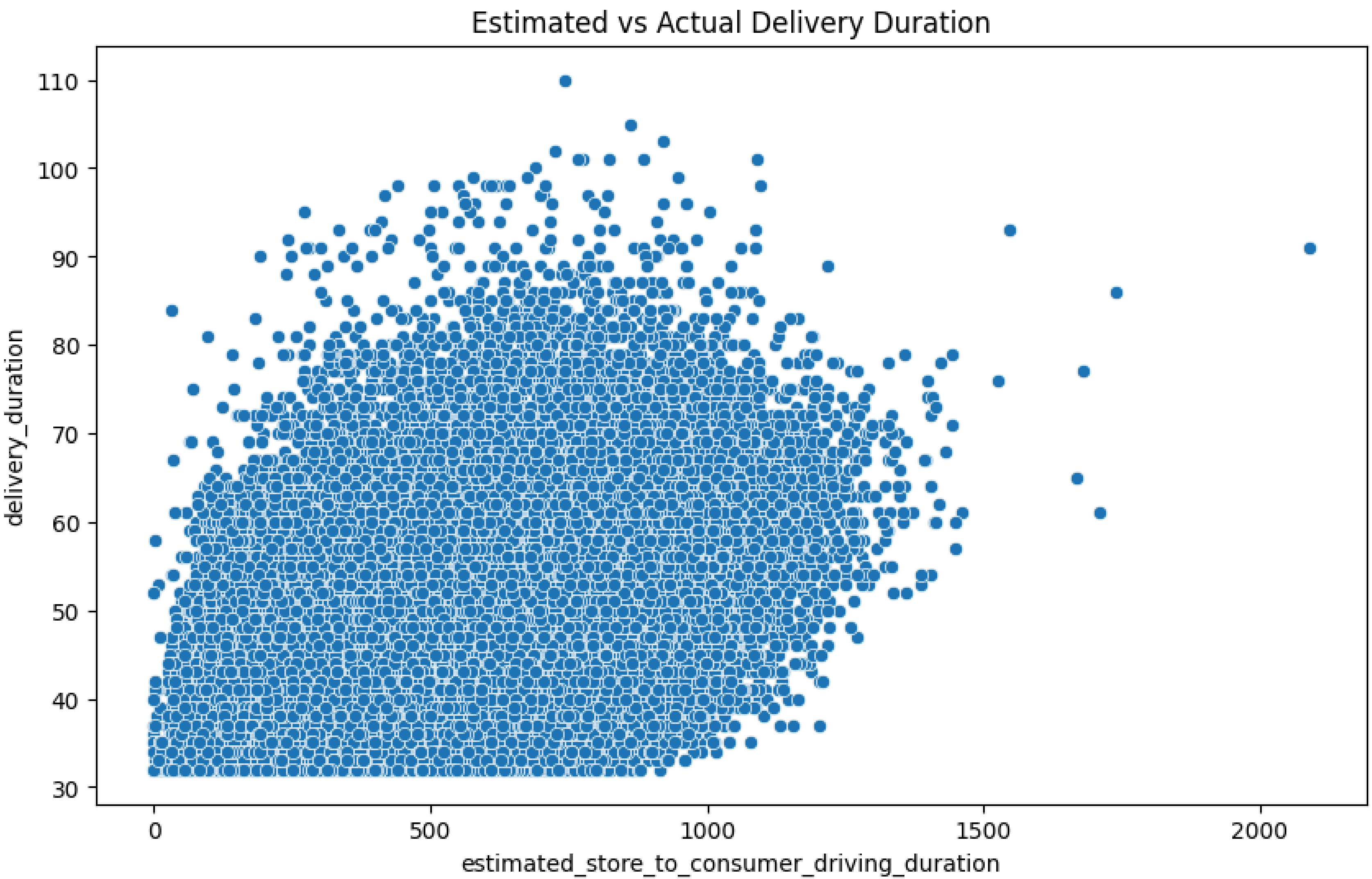
In [34]:
```
# Delivery partners vs delivery duration

plt.figure(figsize=(10, 6))
sns.scatterplot(x = 'total_busy_dashers', y ='delivery_duration', data = df)
plt.title('On-shift Partners vs Delivery Duration')
plt.show()
```



In [35]:
```
# Driving duration estimate vs actual duration

plt.figure(figsize=(10, 6))
sns.scatterplot(x = 'estimated_store_to_consumer_driving_duration', y = 'delivery_duration', data = df)
plt.title('Estimated vs Actual Delivery Duration')
plt.show()
```

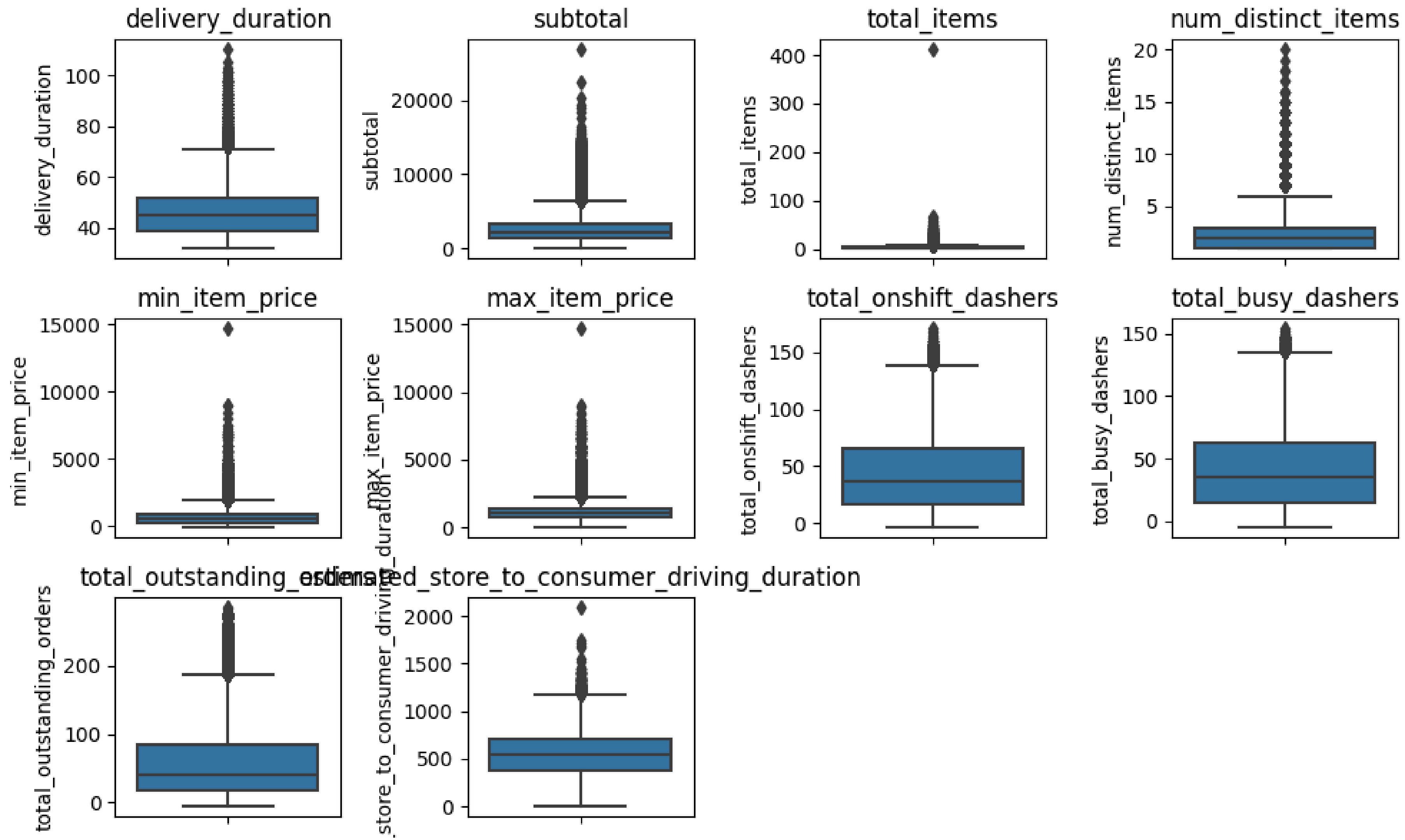


In [36]:
```
# Outlier Detection

# boxplots & IQR method for numerical columns

num_cols = ['delivery_duration','subtotal','total_items','num_distinct_items',
            'min_item_price','max_item_price',
            'total_onshift_dashers','total_busy_dashers',
            'total_outstanding_orders','estimated_store_to_consumer_driving_duration']
```

In [37]:
```
# Boxplots for numerical features
plt.figure(figsize=(10, 6))
for i, j in enumerate(num_cols, 1):
    plt.subplot(3, 4, i)
    sns.boxplot(y = df[j])
    plt.title(j)
plt.tight_layout()
plt.show()
```



In [38]:
```
# Removing Outliers (IQR method)

def remove_Outliers(data, col):
    Q1 = data[col].quantile(0.25)
    Q2 = data[col].quantile(0.75)
    IQR = Q2 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q2 + 1.5 * IQR

    return data[(data[col] >= lower) & (data[col] <= upper)]
```

In [39]:
```
df_clean = remove_Outliers(df, 'delivery_duration')
df_clean.shape
```

Out[39]:
```
(174028, 94)
```

In [40]:
```
df.shape
```

Out[40]:
```
(175777, 94)
```

In [41]:
```
num_cols = df.select_dtypes(include=['int64','float64']).columns

df_clean = df.copy()
for col in num_cols:
    df_clean = remove_Outliers(df_clean, col)

print("Before:", df.shape)
print("After removing outliers from all num cols:", df_clean.shape)
```

```
Before: (175777, 94)
After removing outliers from all num cols: (144990, 94)
```

In [42]:
```
# Data Splitting
from sklearn.model_selection import train_test_split
```

In [43]:
```
X = df_clean.drop(columns=['delivery_duration','created_at','actual_delivery_time'])
y = df_clean['delivery_duration']
```

In [44]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state= 42)
```

In [45]:
```
# Data Scaling (for Neural Networks)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [46]:
```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [47]:
```
X_train_scaled
```

Out[47]:
```
array([[-0.55842349, -0.49823908, -0.42800158, ..., -0.55829776,
        -0.06612705, -0.01098693],
       [ 0.92978203,  0.18706564,  0.04700813, ..., -0.55829776,
        -0.06612705, -0.01098693],
       [-1.30252624,  2.24297982,  2.93456719, ..., -0.55829776,
        -0.06612705, -0.01098693],
       ...,
       [ 0.92978203,  0.18706564,  0.13034317, ...,  1.7911589 ,
        -0.06612705, -0.01098693],
       [ 1.67388479,  0.87237037,  0.62201989, ..., -0.55829776,
        -0.06612705, -0.01098693],
       [ 0.92978203,  2.24297982,  2.06704945, ...,  1.7911589 ,
        -0.06612705, -0.01098693]])
```

In [48]:
```
X_test_scaled
```

Out[48]:
```
array([[ 0.92978203,  0.18706564, -0.15549601, ...,  1.7911589 ,
        -0.06612705, -0.01098693],
       [-0.55842349,  0.18706564, -0.03216015, ..., -0.55829776,
        -0.06612705, -0.01098693],
       [-1.30252624, -0.49823908, -0.53800383, ..., -0.55829776,
        -0.06612705, -0.01098693],
       ...,
       [-0.55842349,  0.18706564,  0.18451095, ..., -0.55829776,
        -0.06612705, -0.01098693],
       [-1.30252624, -1.18354381, -1.0121802 , ..., -0.55829776,
        -0.06612705, -0.01098693],
       [ 0.92978203, -0.49823908, -1.01301355, ...,  1.7911589 ,
        -0.06612705, -0.01098693]])
```

In [49]:
```
# Training a Random Forest Model
from sklearn.ensemble import RandomForestRegressor
import numpy as pd
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [50]:
```
rf = RandomForestRegressor(n_estimators= 200, max_depth=15, random_state=42)
rf.fit(X_train, y_train)
```

Out[50]:
```
RandomForestRegressor(max_depth=15, n_estimators=200, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [51]:
```
y_pred = rf.predict(X_test)
```

In [52]:

```
print("Random Forest Results:")
print("MAE:", mean_absolute_error(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R²:", r2_score(y_test, y_pred))
```

```
Random Forest Results:
MAE: 1.4961055591369534
RMSE: 1.9876598310213827
R²: 0.9382126362735221
```

In [ ]:

```
from sklearn.model_selection import GridSearchCV

params = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 15, 20]
}
```

In [54]:

```
# Neural Network Architecture (Keras / TensorFlow)

import tensorflow as tf
from tensorflow.keras import layers, models
```

In [55]:

```
# Define architecture
nn = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(1)  # Regression → 1 output
])

# Compile model
nn.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

In [56]:
```
history = nn.fit(
    X_train_scaled, y_train,
    validation_split=0.2,
    epochs=50,
    batch_size=32,
    verbose=1
)

# Evaluate
y_pred_nn = nn.predict(X_test_scaled).flatten()

print("Neural Network Results:")
print("MAE:", mean_absolute_error(y_test, y_pred_nn))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_nn)))
print("R²:", r2_score(y_test, y_pred_nn))
```

```
Epoch 1/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 2s 472us/step - loss: 36.0531 - mae: 2.8030 - val_loss: 4.3845 - val_mae: 1.
4796
Epoch 2/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 448us/step - loss: 3.5551 - mae: 1.3298 - val_loss: 3.0257 - val_mae: 1.1
476
Epoch 3/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 450us/step - loss: 2.2302 - mae: 0.9914 - val_loss: 1.9532 - val_mae: 0.8
524
Epoch 4/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 460us/step - loss: 1.5394 - mae: 0.7937 - val_loss: 1.4698 - val_mae: 0.6
903
Epoch 5/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 452us/step - loss: 1.2327 - mae: 0.6914 - val_loss: 1.1798 - val_mae: 0.7
304
Epoch 6/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 451us/step - loss: 1.0237 - mae: 0.6334 - val_loss: 1.3014 - val_mae: 0.6
126
Epoch 7/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 452us/step - loss: 0.9957 - mae: 0.6017 - val_loss: 0.9908 - val_mae: 0.6
360
Epoch 8/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 463us/step - loss: 0.8670 - mae: 0.5722 - val_loss: 1.0972 - val_mae: 0.6
004
Epoch 9/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 451us/step - loss: 0.8317 - mae: 0.5504 - val_loss: 0.7868 - val_mae: 0.5
099
Epoch 10/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 452us/step - loss: 0.7470 - mae: 0.5303 - val_loss: 0.7365 - val_mae: 0.4
896
Epoch 11/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 452us/step - loss: 0.7141 - mae: 0.5226 - val_loss: 0.9662 - val_mae: 0.5
086
Epoch 12/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 452us/step - loss: 0.6683 - mae: 0.5054 - val_loss: 0.5785 - val_mae: 0.4
668
Epoch 13/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 454us/step - loss: 0.6368 - mae: 0.4881 - val_loss: 0.5725 - val_mae: 0.4
549
Epoch 14/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 456us/step - loss: 0.6075 - mae: 0.4781 - val_loss: 0.5805 - val_mae: 0.4
```

```
400
Epoch 15/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 453us/step - loss: 0.5814 - mae: 0.4731 - val_loss: 1.0712 - val_mae: 0.7
384
Epoch 16/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 489us/step - loss: 0.5567 - mae: 0.4591 - val_loss: 0.5209 - val_mae: 0.4
645
Epoch 17/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 471us/step - loss: 0.5317 - mae: 0.4524 - val_loss: 0.7702 - val_mae: 0.5
344
Epoch 18/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 460us/step - loss: 0.5232 - mae: 0.4514 - val_loss: 0.4897 - val_mae: 0.4
449
Epoch 19/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 454us/step - loss: 0.4758 - mae: 0.4379 - val_loss: 0.6724 - val_mae: 0.4
433
Epoch 20/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 453us/step - loss: 0.4799 - mae: 0.4349 - val_loss: 0.5424 - val_mae: 0.4
191
Epoch 21/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 454us/step - loss: 0.4329 - mae: 0.4273 - val_loss: 0.4603 - val_mae: 0.4
115
Epoch 22/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 454us/step - loss: 0.4514 - mae: 0.4261 - val_loss: 0.4249 - val_mae: 0.3
880
Epoch 23/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 457us/step - loss: 0.4139 - mae: 0.4160 - val_loss: 0.5186 - val_mae: 0.4
048
Epoch 24/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 458us/step - loss: 0.4364 - mae: 0.4228 - val_loss: 0.4133 - val_mae: 0.4
286
Epoch 25/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 463us/step - loss: 0.4006 - mae: 0.4124 - val_loss: 0.5175 - val_mae: 0.4
021
Epoch 26/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 461us/step - loss: 0.3959 - mae: 0.4104 - val_loss: 0.3316 - val_mae: 0.3
808
Epoch 27/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 460us/step - loss: 0.3911 - mae: 0.4079 - val_loss: 0.4003 - val_mae: 0.3
863
Epoch 28/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 467us/step - loss: 0.3675 - mae: 0.4026 - val_loss: 0.4804 - val_mae: 0.4
```

```
322
Epoch 29/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 461us/step - loss: 0.3621 - mae: 0.3993 - val_loss: 0.3770 - val_mae: 0.3
894
Epoch 30/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 461us/step - loss: 0.3417 - mae: 0.3916 - val_loss: 0.4813 - val_mae: 0.4
080
Epoch 31/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 460us/step - loss: 0.3919 - mae: 0.4011 - val_loss: 0.4085 - val_mae: 0.3
880
Epoch 32/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 461us/step - loss: 0.3200 - mae: 0.3867 - val_loss: 0.3905 - val_mae: 0.3
735
Epoch 33/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 461us/step - loss: 0.3450 - mae: 0.3917 - val_loss: 0.4763 - val_mae: 0.3
864
Epoch 34/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 474us/step - loss: 0.3315 - mae: 0.3857 - val_loss: 0.3528 - val_mae: 0.3
985
Epoch 35/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 461us/step - loss: 0.3251 - mae: 0.3817 - val_loss: 0.3343 - val_mae: 0.3
594
Epoch 36/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 470us/step - loss: 0.3147 - mae: 0.3784 - val_loss: 0.4872 - val_mae: 0.5
155
Epoch 37/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 462us/step - loss: 0.3138 - mae: 0.3768 - val_loss: 0.3716 - val_mae: 0.3
940
Epoch 38/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 460us/step - loss: 0.3056 - mae: 0.3738 - val_loss: 0.4307 - val_mae: 0.4
267
Epoch 39/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 459us/step - loss: 0.2862 - mae: 0.3683 - val_loss: 0.3511 - val_mae: 0.3
631
Epoch 40/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 468us/step - loss: 0.2966 - mae: 0.3718 - val_loss: 0.3818 - val_mae: 0.3
666
Epoch 41/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 460us/step - loss: 0.2972 - mae: 0.3667 - val_loss: 0.3703 - val_mae: 0.3
716
Epoch 42/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 459us/step - loss: 0.3008 - mae: 0.3671 - val_loss: 0.3043 - val_mae: 0.3
```

```
666
Epoch 43/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 471us/step - loss: 0.2842 - mae: 0.3641 - val_loss: 0.2913 - val_mae: 0.3
574
Epoch 44/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 466us/step - loss: 0.2829 - mae: 0.3650 - val_loss: 0.3054 - val_mae: 0.3
562
Epoch 45/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 461us/step - loss: 0.2847 - mae: 0.3608 - val_loss: 0.3715 - val_mae: 0.3
945
Epoch 46/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 459us/step - loss: 0.2712 - mae: 0.3565 - val_loss: 0.3526 - val_mae: 0.3
855
Epoch 47/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 459us/step - loss: 0.2702 - mae: 0.3556 - val_loss: 0.4198 - val_mae: 0.4
538
Epoch 48/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 461us/step - loss: 0.2798 - mae: 0.3586 - val_loss: 0.3838 - val_mae: 0.4
426
Epoch 49/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 469us/step - loss: 0.2582 - mae: 0.3522 - val_loss: 0.2821 - val_mae: 0.3
441
Epoch 50/50
2900/2900 ━━━━━━━━━━━━━━━━━━━━ 1s 465us/step - loss: 0.2721 - mae: 0.3542 - val_loss: 0.2557 - val_mae: 0.3
410
907/907 ━━━━━━━━━━━━━━━━━━━━ 0s 231us/step
Neural Network Results:
MAE: 0.3387588443357506
RMSE: 0.5009834246474352
R²: 0.9960747957728479
```

In [57]:
```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor

params = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 15, 20]
}

search = RandomizedSearchCV(
    RandomForestRegressor(random_state=42),
    param_distributions=params,
    n_iter=5,          # try only 5 random combos
    cv=3,
    scoring='r2',
    n_jobs=-1,
    random_state=42
)

search.fit(X_train, y_train)
print("Best RF params:", search.best_params_)
```

```
/Users/jagatheespandi/anaconda3/lib/python3.10/site-packages/pandas/core/arrays/masked.py:60: UserWarning:
Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/jagatheespandi/anaconda3/lib/python3.10/site-packages/pandas/core/arrays/masked.py:60: UserWarning:
Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/jagatheespandi/anaconda3/lib/python3.10/site-packages/pandas/core/arrays/masked.py:60: UserWarning:
Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/jagatheespandi/anaconda3/lib/python3.10/site-packages/pandas/core/arrays/masked.py:60: UserWarning:
Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/jagatheespandi/anaconda3/lib/python3.10/site-packages/pandas/core/arrays/masked.py:60: UserWarning:
Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/jagatheespandi/anaconda3/lib/python3.10/site-packages/pandas/core/arrays/masked.py:60: UserWarning:
Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/jagatheespandi/anaconda3/lib/python3.10/site-packages/pandas/core/arrays/masked.py:60: UserWarning:
Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/jagatheespandi/anaconda3/lib/python3.10/site-packages/pandas/core/arrays/masked.py:60: UserWarning:
Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (

Best RF params: {'n_estimators': 300, 'max_depth': 20}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: