In [97]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [98]:
```
dataset=pd.read_csv('dataset.csv')
```

In [99]:
```
dataset #EXPLORING DATA
```

Out[99]:

	market_id	created_at	actual_delivery_time	store_id	store_pri
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	
2	3.0	2015-01-22 20:39:28	2015-01-22 21:09:09	f0ade77b43923b38237db569b016ba25	
3	3.0	2015-02-03 21:21:45	2015-02-03 22:13:00	f0ade77b43923b38237db569b016ba25	
4	3.0	2015-02-15 02:40:36	2015-02-15 03:20:26	f0ade77b43923b38237db569b016ba25	
...	...	...	...	...	
197423	1.0	2015-02-17 00:19:41	2015-02-17 01:24:48	a914ecef9c12ffdb9bede64bb703d877	
197424	1.0	2015-02-13 00:01:59	2015-02-13 00:58:22	a914ecef9c12ffdb9bede64bb703d877	
197425	1.0	2015-01-24 04:46:08	2015-01-24 05:36:16	a914ecef9c12ffdb9bede64bb703d877	
197426	1.0	2015-02-01 18:18:15	2015-02-01 19:23:22	c81e155d85dae5430a8cee6f2242e82c	
197427	1.0	2015-02-08 19:24:33	2015-02-08 20:01:41	c81e155d85dae5430a8cee6f2242e82c	

197428 rows × 14 columns

In [100]:
```
dataset.head()
```

Out[100]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	
2	3.0	2015-01-22 20:39:28	2015-01-22 21:09:09	f0ade77b43923b38237db569b016ba25	
3	3.0	2015-02-03 21:21:45	2015-02-03 22:13:00	f0ade77b43923b38237db569b016ba25	
4	3.0	2015-02-15 02:40:36	2015-02-15 03:20:26	f0ade77b43923b38237db569b016ba25	

In [101]:
```
dataset.tail()
```

Out[101]:

	market_id	created_at	actual_delivery_time	store_id	store_prin
197423	1.0	2015-02-17 00:19:41	2015-02-17 01:24:48	a914ecef9c12ffdb9bede64bb703d877	
197424	1.0	2015-02-13 00:01:59	2015-02-13 00:58:22	a914ecef9c12ffdb9bede64bb703d877	
197425	1.0	2015-01-24 04:46:08	2015-01-24 05:36:16	a914ecef9c12ffdb9bede64bb703d877	
197426	1.0	2015-02-01 18:18:15	2015-02-01 19:23:22	c81e155d85dae5430a8cee6f2242e82c	
197427	1.0	2015-02-08 19:24:33	2015-02-08 20:01:41	c81e155d85dae5430a8cee6f2242e82c	

In [102]:
```
dataset.shape
```

Out[102]: (197428, 14)

In [103]:
```
dataset.describe()
```

Out[103]:

	market_id	order_protocol	total_items	subtotal	num_distinct_items	min_iter
count	196441.000000	196433.000000	197428.000000	197428.000000	197428.000000	197428
mean	2.978706	2.882352	3.196391	2682.331402	2.670791	686
std	1.524867	1.503771	2.666546	1823.093688	1.630255	522
min	1.000000	1.000000	1.000000	0.000000	1.000000	-86
25%	2.000000	1.000000	2.000000	1400.000000	1.000000	299
50%	3.000000	3.000000	3.000000	2200.000000	2.000000	595
75%	4.000000	4.000000	4.000000	3395.000000	3.000000	949
max	6.000000	7.000000	411.000000	27100.000000	20.000000	14700

In [104]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197428 entries, 0 to 197427
Data columns (total 14 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   market_id               196441 non-null  float64
 1   created_at              197428 non-null  object 
 2   actual_delivery_time    197421 non-null  object 
 3   store_id                197428 non-null  object 
 4   store_primary_category  192668 non-null  object 
 5   order_protocol          196433 non-null  float64
 6   total_items             197428 non-null  int64  
 7   subtotal                197428 non-null  int64  
 8   num_distinct_items      197428 non-null  int64  
 9   min_item_price          197428 non-null  int64  
 10  max_item_price          197428 non-null  int64  
 11  total_onshift_partners  181166 non-null  float64
 12  total_busy_partners     181166 non-null  float64
 13  total_outstanding_orders  181166 non-null  float64
dtypes: float64(5), int64(5), object(4)
memory usage: 21.1+ MB
```

In [105]:
```
dataset.isna().sum()
```

Out[105]:
```
market_id                 987
created_at                  0
actual_delivery_time        7
store_id                    0
store_primary_category   4760
order_protocol            995
total_items                 0
subtotal                    0
num_distinct_items          0
min_item_price              0
max_item_price              0
total_onshift_partners  16262
total_busy_partners     16262
total_outstanding_orders16262
dtype: int64
```

In [106]:
```
# Focusing on tragetting variables
```

In [107]:
```
# Converting argument to datetime(arranging dates and time in to correct sequer
dataset['created_at']=pd.to_datetime(dataset['created_at'])
dataset['actual_delivery_time']=pd.to_datetime(dataset['actual_delivery_time'])
```

In [108]:
```
#Adding new column delivery time

dataset['Delivery time']=(dataset['actual_delivery_time'] - dataset['created_at
```

In [109]:
```
dataset.head()
```

Out[109]:

·otocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_part
1.0	4	3441	4	557	1239	
2.0	1	1900	1	1400	1400	
1.0	1	1900	1	1900	1900	
1.0	6	6900	5	600	1800	
1.0	3	3900	3	1100	1600	

In [110]:
```
#Extracting various features from the data.
```

In [111]:
```
dataset['order_month']= dataset['created_at'].dt.month
dataset['order_hour']= dataset['created_at'].dt.hour
def order_time(i):
    if i in range(0,7):
        return 'Late Night'
    elif i in range(6,13):
        return 'Morning'
    elif i in range(12,19):
        return 'Afternoon'
    else:
        return 'Night'
```

In [112]:
```
dataset['time_of_day']= dataset['order_hour'].apply(order_time)
del dataset['order_hour']
dataset['order_weekday']= dataset['created_at'].dt.weekday
dataset['total_available_partners']= (dataset['total_onshift_partners'] - datas
```

In [113]:
```
dataset.drop(columns=['store_id','created_at','actual_delivery_time','total_ite
```

In [114]:
```
dataset
```

Out[114]:

ax_item_price	total_onshift_partners	total_busy_partners	total_outstanding_orders	Delivery time	orde
1239	33.0	14.0	21.0	62.983333	
1400	1.0	2.0	2.0	67.066667	
1900	1.0	0.0	0.0	29.683333	
1800	1.0	1.0	2.0	51.250000	
1600	6.0	6.0	9.0	39.833333	
...	...	...	...	...	
649	17.0	17.0	23.0	65.116667	
825	12.0	11.0	14.0	56.383333	
399	39.0	41.0	40.0	50.133333	
535	7.0	7.0	12.0	65.116667	
750	20.0	20.0	23.0	37.133333	

In [115]:
```
dataset=dataset.fillna(0)#FILLING NAN with 0
```

In [116]:
```
dataset.isna().sum()/dataset.shape[0]*100
```

Out[116]:
```
market_id                 0.0
store_primary_category    0.0
order_protocol            0.0
subtotal                  0.0
num_distinct_items        0.0
min_item_price            0.0
max_item_price            0.0
total_onshift_partners    0.0
total_busy_partners       0.0
total_outstanding_orders  0.0
Delivery time             0.0
order_month               0.0
time_of_day               0.0
order_weekday             0.0
total_available_partners  0.0
dtype: float64
```

In [117]:
```
dataset.isna().sum()
```

Out[117]:
```
market_id                 0
store_primary_category    0
order_protocol            0
subtotal                  0
num_distinct_items        0
min_item_price            0
max_item_price            0
total_onshift_partners    0
total_busy_partners       0
total_outstanding_orders  0
Delivery time             0
order_month               0
time_of_day               0
order_weekday             0
total_available_partners  0
dtype: int64
```

In [118]:
```
dataset.shape
```

Out[118]:
```
(197428, 15)
```

In [119]:
```
dataset.head()
```

Out[119]:

	market_id	store_primary_category	order_protocol	subtotal	num_distinct_items	min_item_pric
0	1.0	american	1.0	3441	4	55
1	2.0	mexican	2.0	1900	1	140
2	3.0	0	1.0	1900	1	190
3	3.0	0	1.0	6900	5	60
4	3.0	0	1.0	3900	3	110

In [120]:
```
uniqueValues=(dataset['store_primary_category']).unique()
```

In [121]:
```
uniqueValues
```

Out[121]:
```
array(['american', 'mexican', 0, 'indian', 'italian', 'sandwich', 'thai',
       'cafe', 'salad', 'pizza', 'chinese', 'singaporean', 'burger',
       'breakfast', 'mediterranean', 'japanese', 'greek', 'catering',
       'filipino', 'convenience-store', 'other', 'korean', 'vegan',
       'asian', 'barbecue', 'fast', 'dessert', 'smoothie', 'seafood',
       'vietnamese', 'cajun', 'steak', 'middle-eastern', 'soup',
       'vegetarian', 'persian', 'nepalese', 'sushi', 'latin-american',
       'hawaiian', 'chocolate', 'burmese', 'british', 'pasta', 'alcohol',
       'dim-sum', 'peruvian', 'turkish', 'malaysian', 'ethiopian',
       'afghan', 'bubble-tea', 'german', 'french', 'caribbean',
       'gluten-free', 'comfort-food', 'gastropub', 'pakistani',
       'moroccan', 'spanish', 'southern', 'tapas', 'russian', 'brazilian',
       'european', 'cheese', 'african', 'argentine', 'kosher', 'irish',
       'lebanese', 'belgian', 'indonesian', 'alcohol-plus-food'],
      dtype=object)
```

In [122]:
```
uniqueValues.shape
```

Out[122]:
```
(75,)
```

In [123]:
```
dataset.head()
```

Out[123]:

	market_id	store_primary_category	order_protocol	subtotal	num_distinct_items	min_item_pric
0	1.0	american	1.0	3441	4	55
1	2.0	mexican	2.0	1900	1	140
2	3.0	0	1.0	1900	1	190
3	3.0	0	1.0	6900	5	60
4	3.0	0	1.0	3900	3	110

In [124]:
```
uniqueValues.shape
```

Out[124]:
```
(75,)
```

In [125]:
```
dataset.isna().sum()
```

Out[125]:
```
market_id                 0
store_primary_category    0
order_protocol            0
subtotal                  0
num_distinct_items        0
min_item_price            0
max_item_price            0
total_onshift_partners    0
total_busy_partners       0
total_outstanding_orders  0
Delivery time             0
order_month               0
time_of_day               0
order_weekday             0
total_available_partners  0
dtype: int64
```

In [126]:
```
uniqueValues=(dataset['store_primary_category']).unique()
```

In [127]:
```
uniqueValues
```

Out[127]:
```
array(['american', 'mexican', 0, 'indian', 'italian', 'sandwich', 'thai',
       'cafe', 'salad', 'pizza', 'chinese', 'singaporean', 'burger',
       'breakfast', 'mediterranean', 'japanese', 'greek', 'catering',
       'filipino', 'convenience-store', 'other', 'korean', 'vegan',
       'asian', 'barbecue', 'fast', 'dessert', 'smoothie', 'seafood',
       'vietnamese', 'cajun', 'steak', 'middle-eastern', 'soup',
       'vegetarian', 'persian', 'nepalese', 'sushi', 'latin-american',
       'hawaiian', 'chocolate', 'burmese', 'british', 'pasta', 'alcohol',
       'dim-sum', 'peruvian', 'turkish', 'malaysian', 'ethiopian',
       'afghan', 'bubble-tea', 'german', 'french', 'caribbean',
       'gluten-free', 'comfort-food', 'gastropub', 'pakistani',
       'moroccan', 'spanish', 'southern', 'tapas', 'russian', 'brazilian',
       'european', 'cheese', 'african', 'argentine', 'kosher', 'irish',
       'lebanese', 'belgian', 'indonesian', 'alcohol-plus-food'],
      dtype=object)
```

In [128]:
```
dataset['TOP_MENU']=(dataset['store_primary_category'])#Creating the replica of
```

In [129]:
```
dataset
```

Out[129]:

	market_id	store_primary_category	order_protocol	subtotal	num_distinct_items	min_iten
0	1.0	american	1.0	3441	4	
1	2.0	mexican	2.0	1900	1	
2	3.0	0	1.0	1900	1	
3	3.0	0	1.0	6900	5	
4	3.0	0	1.0	3900	3	
...	...	...	...	...	...	
197423	1.0	fast	4.0	1389	3	
197424	1.0	fast	4.0	3010	4	
197425	1.0	fast	4.0	1836	3	
197426	1.0	sandwich	1.0	1175	1	
197427	1.0	sandwich	1.0	2605	4	

197428 rows × 16 columns

In [130]:
```
#Replacing 75 unique values in to numbers
dataset.TOP_MENU.replace(['american', 'mexican',0, 'indian', 'italian', 'sandwi
       'thai', 'cafe', 'salad', 'pizza', 'chinese', 'singaporean',
       'burger', 'breakfast', 'mediterranean', 'japanese', 'greek',
       'catering', 'filipino', 'convenience-store', 'other', 'korean',
       'vegan', 'asian', 'barbecue', 'fast', 'dessert', 'smoothie',
       'seafood', 'vietnamese', 'cajun', 'steak', 'middle-eastern',
       'soup', 'vegetarian', 'persian', 'nepalese', 'sushi',
       'latin-american', 'hawaiian', 'chocolate', 'burmese', 'british',
       'pasta', 'alcohol', 'dim-sum', 'peruvian', 'turkish', 'malaysian',
       'ethiopian', 'afghan', 'bubble-tea', 'german', 'french',
       'caribbean', 'gluten-free', 'comfort-food', 'gastropub',
       'pakistani', 'moroccan', 'spanish', 'southern', 'tapas', 'russian',
       'brazilian', 'european', 'cheese', 'african', 'argentine',
       'kosher', 'irish', 'lebanese', 'belgian', 'indonesian',
       'alcohol-plus-food'],[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
```

In [131]:
```
dataset.head()
```

Out[131]:

	market_id	store_primary_category	order_protocol	subtotal	num_distinct_items	min_item_pric
0	1.0	american	1.0	3441	4	55
1	2.0	mexican	2.0	1900	1	140
2	3.0	0	1.0	1900	1	190
3	3.0	0	1.0	6900	5	60
4	3.0	0	1.0	3900	3	110

In [132]:
```
dataset.tail()
```

Out[132]:

	market_id	store_primary_category	order_protocol	subtotal	num_distinct_items	min_iten
197423	1.0	fast	4.0	1389	3	
197424	1.0	fast	4.0	3010	4	
197425	1.0	fast	4.0	1836	3	
197426	1.0	sandwich	1.0	1175	1	
197427	1.0	sandwich	1.0	2605	4	

In [133]:
```
dataset.describe()
```

Out[133]:

	market_id	order_protocol	subtotal	num_distinct_items	min_item_price	max_ite
count	197428.000000	197428.000000	197428.000000	197428.000000	197428.000000	19742
mean	2.963815	2.867825	2682.331402	2.670791	686.218470	115
std	1.535490	1.513800	1823.093688	1.630255	522.038648	55
min	0.000000	0.000000	0.000000	1.000000	-86.000000	
25%	2.000000	1.000000	1400.000000	1.000000	299.000000	80
50%	3.000000	3.000000	2200.000000	2.000000	595.000000	109
75%	4.000000	4.000000	3395.000000	3.000000	949.000000	139
max	6.000000	7.000000	27100.000000	20.000000	14700.000000	1470

In [134]:
```
dataset['Delivery time'].describe()
```

Out[134]:
```
count    197428.000000
mean         48.469238
std         320.487931
min           0.000000
25%          35.066667
50%          44.333333
75%          56.350000
max      141947.650000
Name: Delivery time, dtype: float64
```

In [135]:
```
max_delivery_time = dataset['Delivery time'].max() #removing maximum value
```

In [136]:
```
max_delivery_time
```

Out[136]:
```
141947.65
```

In [137]:
```
dataset1= dataset[dataset['Delivery time'] < max_delivery_time]
```

In [138]:
```
print("Number of Rows Before Dropping:", len(dataset))
print("Number of Rows After Dropping:", len(dataset))
```

```
Number of Rows Before Dropping: 197428
Number of Rows After Dropping: 197428
```

In [139]:
```
# Checking and removing out liers if any
dataset1.select_dtypes(include= np.number).columns
```

Out[139]:
```
Index(['market_id', 'order_protocol', 'subtotal', 'num_distinct_items',
       'min_item_price', 'max_item_price', 'total_onshift_partners',
       'total_busy_partners', 'total_outstanding_orders', 'Delivery time',
       'order_month', 'order_weekday', 'total_available_partners', 'TOP_MEN
U'],
      dtype='object')
```

In [140]:
```
num_of_cols= ['market_id', 'order_protocol', 'subtotal', 'num_distinct_items',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders', 'Delivery time', 'order_month',
       'order_weekday','min_item_price','max_item_price' , 'TOP_MENU']
```

In [141]:
```
for i in num_of_cols:
    q1, q3= np.quantile(dataset1[i], [0.25, 0.75])
    iqr= q3 - q1
    ll= q1 -1.5*iqr
    ul= q3 +1.5*iqr
    dataset1= dataset1.loc[~((dataset1[i] < ll) | (dataset1[i]> ul))]
```

In [142]:
```
print("Number of Rows After Removing Outliers:", len(dataset1))
```

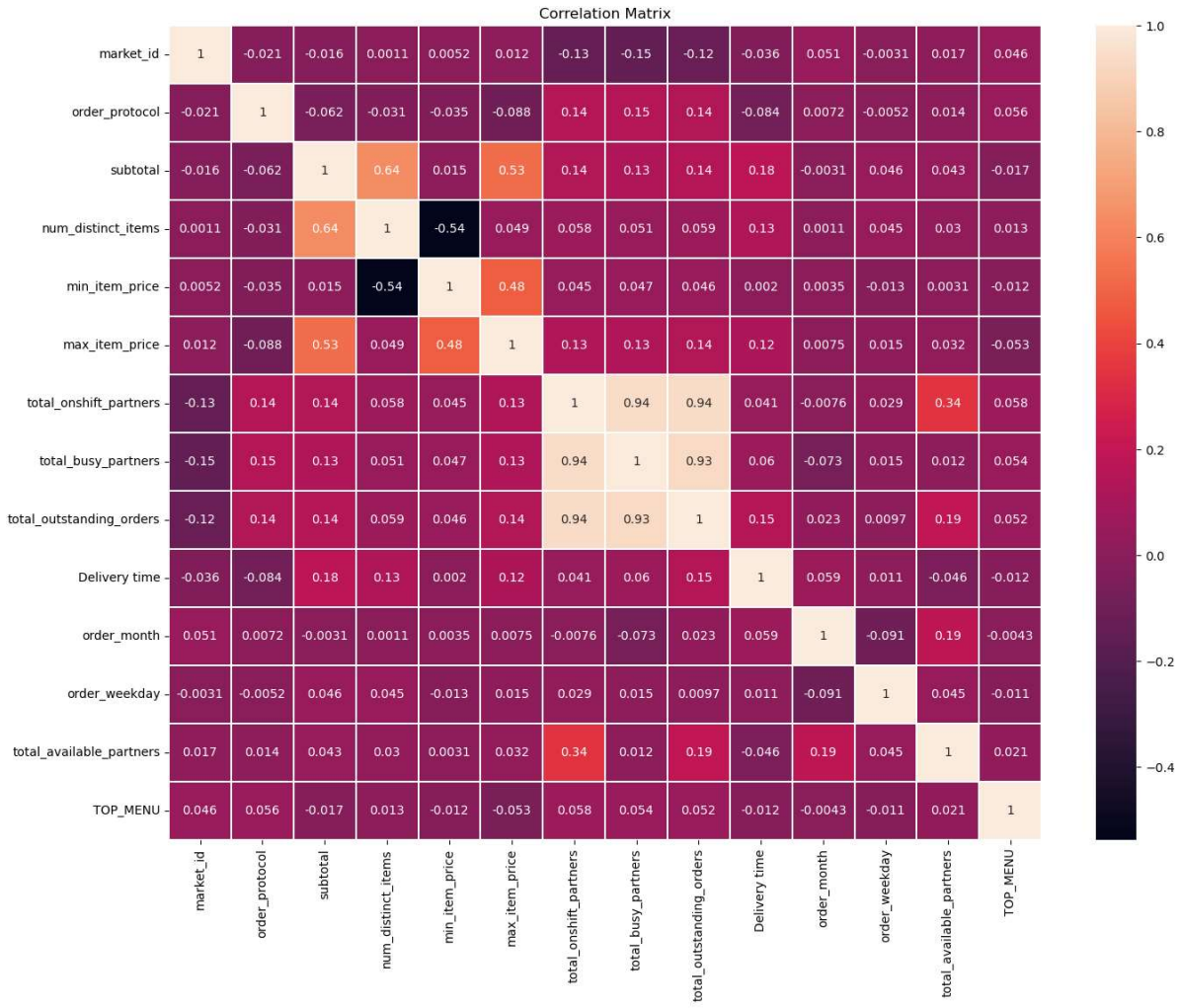
```
Number of Rows After Removing Outliers: 160121
```

In [143]:
```
dataset1['Delivery time'].describe()# min delivery time is 4 mins & maximum del
```

Out[143]:
```
count    160121.000000
mean         44.271152
std          14.211315
min           3.716667
25%          33.800000
50%          42.316667
75%          52.983333
max          85.583333
Name: Delivery time, dtype: float64
```

In [144]:
```
#Lets have a look on correlation matrix data on orange scale willbe highly core
sns.heatmap(dataset1.corr(),annot=True,linewidths=.19)
fig = plt.gcf()  # or by other means, like plt.subplots
figsize = fig.get_size_inches()
fig.set_size_inches(figsize * 2.5)
plt.title('Correlation Matrix')
plt.show()
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_10804\2220952400.py:2: FutureWarn
ing: The default value of numeric_only in DataFrame.corr is deprecated. In a
future version, it will default to False. Select only valid columns or specif
y the value of numeric_only to silence this warning.
  sns.heatmap(dataset1.corr(),annot=True,linewidths=.19)
```



In [145]:
```
correlation_matrix = dataset1.corr()## Calculate the correlation matrix
correlation_threshold = 0.4### Set the correlation threshold (e.g., 0.4 for hig
correlated_pairs = []## Create an empty list to store correlated pairs
# Iterate through the correlation matrix
for i in range(len(correlation_matrix.columns)):
    for j in range(i+1, len(correlation_matrix.columns)):
        if abs(correlation_matrix.iloc[i, j]) > correlation_threshold:
            correlated_pairs.append((correlation_matrix.columns[i], correlatior
            # Print the highly correlated pairs
for pair in correlated_pairs:
    print("Highly Correlated Pair:", pair)
```

```
Highly Correlated Pair: ('subtotal', 'num_distinct_items')
Highly Correlated Pair: ('subtotal', 'max_item_price')
Highly Correlated Pair: ('num_distinct_items', 'min_item_price')
Highly Correlated Pair: ('min_item_price', 'max_item_price')
Highly Correlated Pair: ('total_onshift_partners', 'total_busy_partners')
Highly Correlated Pair: ('total_onshift_partners', 'total_outstanding_order
s')
Highly Correlated Pair: ('total_busy_partners', 'total_outstanding_orders')

C:\Users\Owner\AppData\Local\Temp\ipykernel_10804\1695188212.py:1: FutureWarn
ing: The default value of numeric_only in DataFrame.corr is deprecated. In a
future version, it will default to False. Select only valid columns or specif
y the value of numeric_only to silence this warning.
  correlation_matrix = dataset1.corr()## Calculate the correlation matrix
```

In [146]:
```
dataset1.describe()
```

Out[146]:

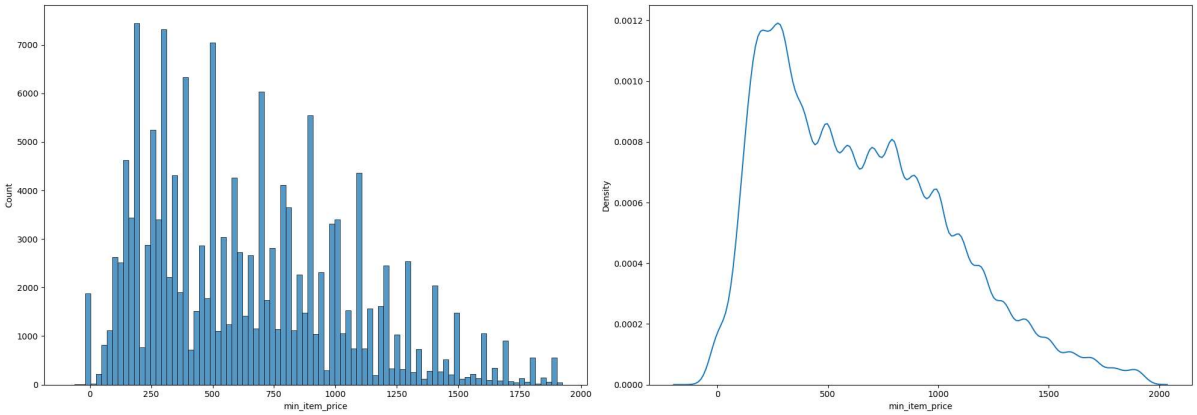
	market_id	order_protocol	subtotal	num_distinct_items	min_item_price	max_ite
count	160121.000000	160121.000000	160121.000000	160121.000000	160121.000000	16012
mean	2.982844	2.873446	2286.407461	2.450303	644.183024	104
std	1.550700	1.516710	1237.973496	1.267877	408.511823	37
min	0.000000	0.000000	0.000000	1.000000	-86.000000	5
25%	2.000000	1.000000	1317.000000	1.000000	299.000000	79
50%	3.000000	3.000000	2000.000000	2.000000	595.000000	100
75%	4.000000	4.000000	2997.000000	3.000000	900.000000	129
max	6.000000	7.000000	6387.000000	6.000000	1925.000000	205

In [147]:
```
dataset1.shape
```

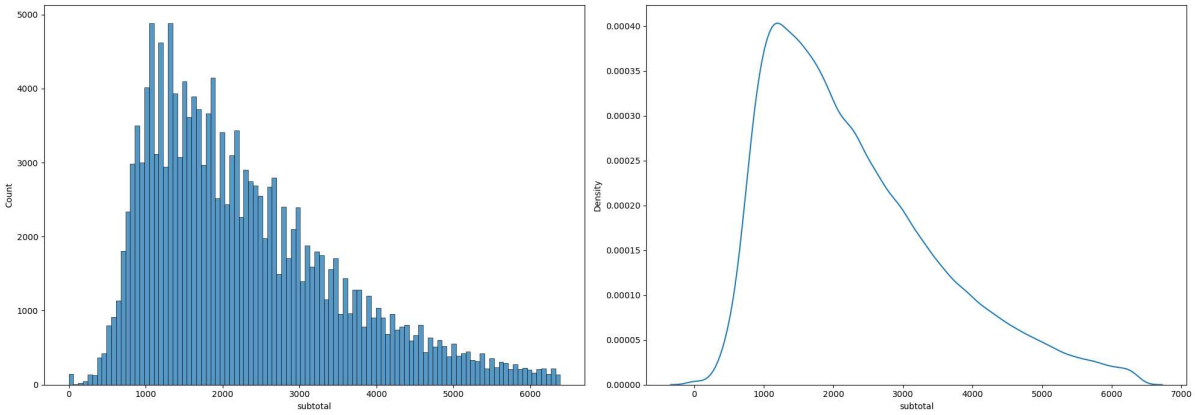
Out[147]:
```
(160121, 16)
```

In [148]:
```
def hist_kdeplot(dataset1, var:str):
    fig,axes = plt.subplots(nrows=1, ncols=2, figsize=(20,7))
    sns.histplot(dataset1, x = var, ax=axes[0])
    sns.kdeplot(dataset1, x = var, ax=axes[1])
    plt.tight_layout()
    plt.show()
```

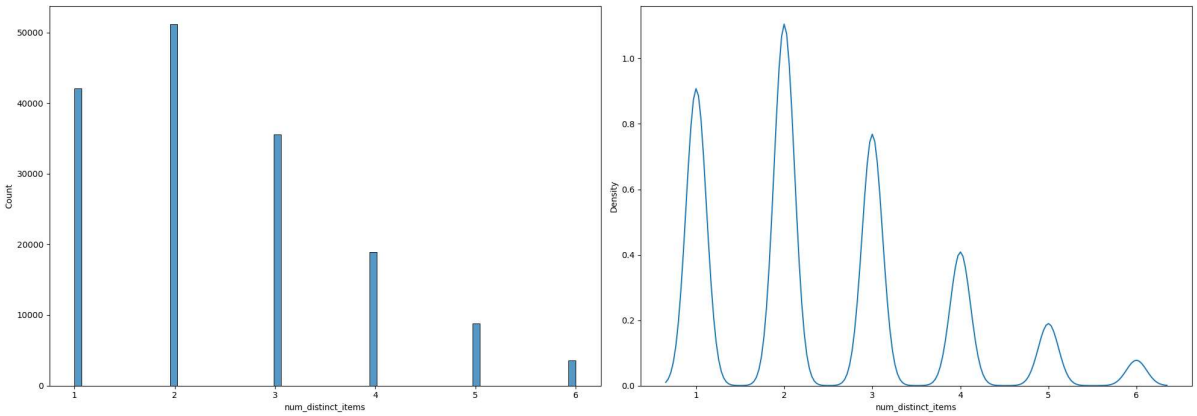
In [149]:
```
hist_kdeplot(dataset1,'min_item_price')
```



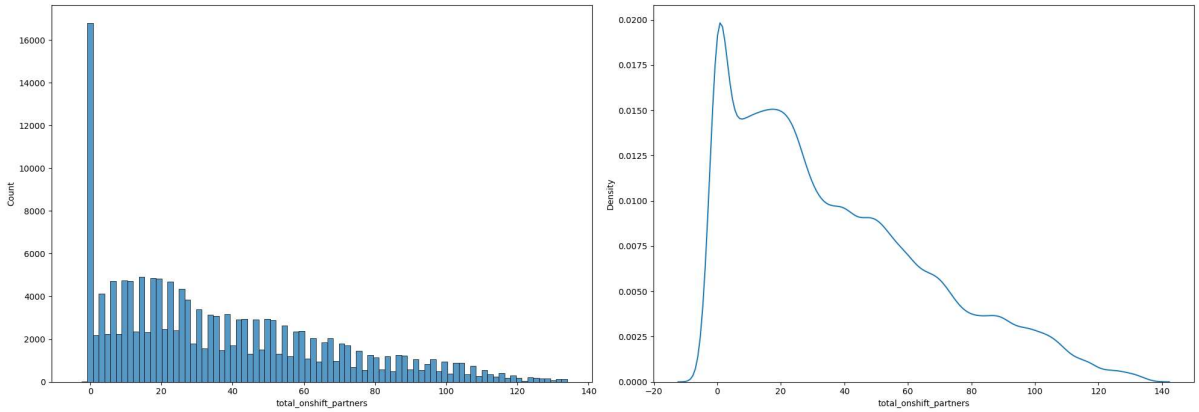
In [150]:
```
hist_kdeplot(dataset1,'subtotal')
```



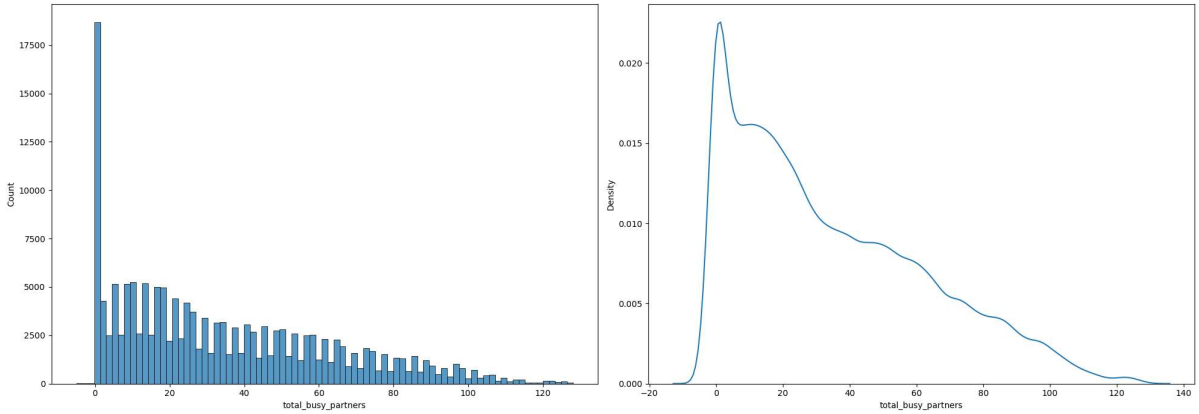
In [151]:
```
hist_kdeplot(dataset1,'num_distinct_items')
```



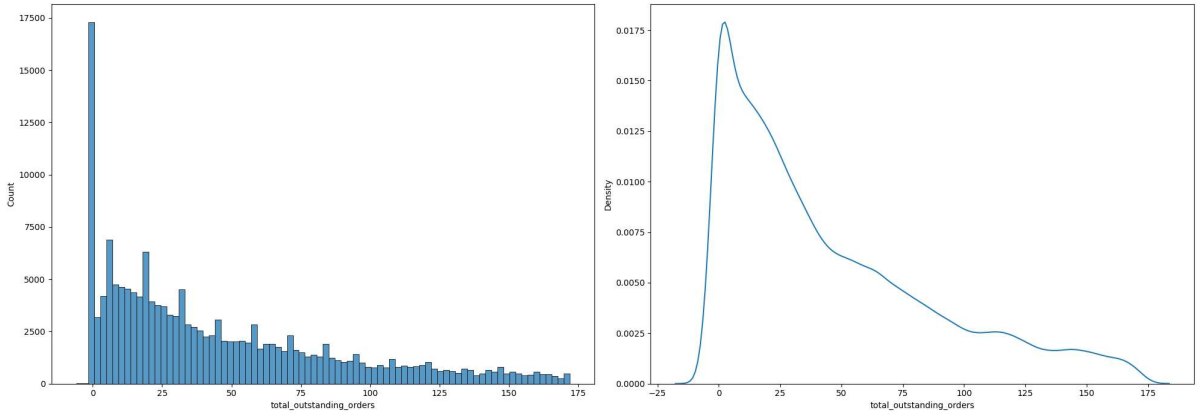
In [152]:
```
hist_kdeplot(dataset1,'total_onshift_partners')
```



In [153]:
```
hist_kdeplot(dataset1, 'total_busy_partners')
```



In [154]:
```
hist_kdeplot(dataset1, "total_outstanding_orders")
```

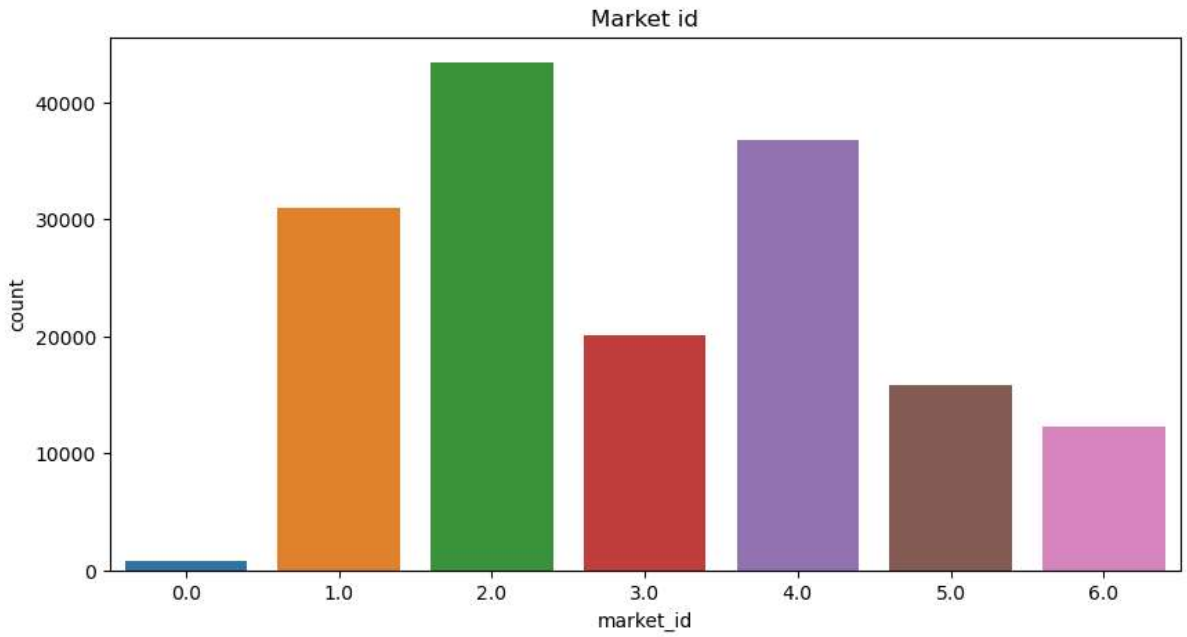


In [155]:
```
dataset1.head()
```

Out[155]:

	market_id	store_primary_category	order_protocol	subtotal	num_distinct_items	min_item_pric
0	1.0	american	1.0	3441	4	55
1	2.0	mexican	2.0	1900	1	140
2	3.0	0	1.0	1900	1	190
4	3.0	0	1.0	3900	3	110
5	3.0	0	1.0	5000	3	150

In [156]:
```
fig,ax = plt.subplots(figsize=(10,5))
sns.countplot(dataset1, x="market_id",ax=ax)
ax.set_title("Market id")
plt.show()
```



In [157]:
```
dataset1.describe()
```

Out[157]:

	market_id	order_protocol	subtotal	num_distinct_items	min_item_price	max_ite
count	160121.000000	160121.000000	160121.000000	160121.000000	160121.000000	16012
mean	2.982844	2.873446	2286.407461	2.450303	644.183024	104
std	1.550700	1.516710	1237.973496	1.267877	408.511823	37
min	0.000000	0.000000	0.000000	1.000000	-86.000000	5
25%	2.000000	1.000000	1317.000000	1.000000	299.000000	79
50%	3.000000	3.000000	2000.000000	2.000000	595.000000	100
75%	4.000000	4.000000	2997.000000	3.000000	900.000000	129
max	6.000000	7.000000	6387.000000	6.000000	1925.000000	205

In [158]:
```
dataset1.shape
```

Out[158]: (160121, 16)

In [159]:
```
dataset1
```

Out[159]:

	market_id	store_primary_category	order_protocol	subtotal	num_distinct_items	min_it
0	1.0	american	1.0	3441	4	
1	2.0	mexican	2.0	1900	1	
2	3.0	0	1.0	1900	1	
4	3.0	0	1.0	3900	3	
5	3.0	0	1.0	5000	3	
...	...	...	...	...	...	
197423	1.0	fast	4.0	1389	3	
197424	1.0	fast	4.0	3010	4	
197425	1.0	fast	4.0	1836	3	
197426	1.0	sandwich	1.0	1175	1	

In [160]:
```
X=dataset1.iloc[:,[0,2,3,4,5,6,7,8,9,11,13,14,15]].values
y=dataset1.iloc[:,10].values
```

In [161]:
```
X
```

Out[161]:
```
array([[ 1.000e+00,  1.000e+00,  3.441e+03, ...,  4.000e+00,  1.900e+01,
         0.000e+00],
       [ 2.000e+00,  2.000e+00,  1.900e+03, ...,  1.000e+00, -1.000e+00,
         1.000e+00],
       [ 3.000e+00,  1.000e+00,  1.900e+03, ...,  3.000e+00,  1.000e+00,
         2.000e+00],
       ...,
       [ 1.000e+00,  4.000e+00,  1.836e+03, ...,  5.000e+00, -2.000e+00,
         2.500e+01],
       [ 1.000e+00,  1.000e+00,  1.175e+03, ...,  6.000e+00,  0.000e+00,
         5.000e+00],
       [ 1.000e+00,  1.000e+00,  2.605e+03, ...,  6.000e+00,  0.000e+00,
         5.000e+00]])
```

In [162]:
```
y
```

Out[162]:
```
array([62.98333333, 67.06666667, 29.68333333, ..., 50.13333333,
       65.11666667, 37.13333333])
```

In [163]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
```

In [164]:
```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
```

Out[164]:
```
LinearRegression
LinearRegression()
```

In [165]:
```
y_pred = regressor.predict(X_test)
```

In [166]:
```
df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
```

In [167]:
```
df
```

Out[167]:

	Actual	Predicted
0	42.716667	41.129638
1	39.300000	37.546293
2	61.666667	48.651719
3	44.750000	42.608205
4	66.816667	51.649079
...	...	...
32020	29.300000	46.725482
32021	42.366667	52.594534
32022	33.300000	46.001630
32023	52.183333	45.497576
32024	33.316667	38.334100

32025 rows × 2 columns

In [168]:
```
y_pred
```

Out[168]:
```
array([41.12963845, 37.54629325, 48.65171929, ..., 46.00163045,
       45.49757624, 38.33410001])
```

In [169]:
```
y_test
```

Out[169]:
```
array([42.71666667, 39.3       , 61.66666667, ..., 33.3       ,
       52.18333333, 33.31666667])
```

In [170]:
```
from sklearn import metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:',metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_test,y_pr
print('R squared:',metrics.r2_score(y_test,y_pred))
```

```
Mean Absolute Error: 10.463418964099896
Mean Squared Error: 172.66735659686336
Root Mean Squared Error: 13.140295148772852
R squared: 0.13609955129248663
```

In [171]:
```
print('Train Score: ', regressor.score(X_train, y_train))
print('Test Score: ', regressor.score(X_test, y_test))
```

```
Train Score:  0.1472184201675436
Test Score:  0.13609955129248663
```

In [172]:
```
#XG BOOST REGRESSION
```

In [173]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in e:\anaconda\lib\site-packages (1.7.
6)
Requirement already satisfied: scipy in e:\anaconda\lib\site-packages (from x
gboost) (1.10.0)
Requirement already satisfied: numpy in e:\anaconda\lib\site-packages (from x
gboost) (1.23.5)
Note: you may need to restart the kernel to use updated packages.
```

In [174]:
```
X0=dataset1.iloc[:,[0,2,3,4,5,6,7,8,9,11,13,14]].values
y0=dataset1.iloc[:,10].values
```

In [175]:
```
from sklearn.model_selection import train_test_split
X0_train, X0_test, y0_train, y0_test = train_test_split(X0, y0, test_size=0.4,
import xgboost as xg
```

In [176]:
```
xgb_r = xg.XGBRegressor(objective ='reg:linear',
                  n_estimators = 10, seed = 123)
```

In [177]:
```
xgb_r.fit(X0_train, y0_train)
```

```
[15:56:07] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscali
ng-group-i-0fdc6d574b9c0d168-1\xgboost\xgboost-ci-windows\src\objective\regre
ssion_obj.cu:213: reg:linear is now deprecated in favor of reg:squarederror.
```

Out[177]:
```
XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=Non
e,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=No
ne,
             interaction_constraints=None, learning_rate=None, max_bin=Non
e,
             max_cat_threshold=None, max_cat_to_onehot=None,
```

In [178]:
```
y0_pred = xgb_r.predict(X0_test)
```

In [179]:
```
from sklearn import metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y0_test,y0_pred))
print('Mean Squared Error:',metrics.mean_squared_error(y0_test,y0_pred))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y0_test,y0_
print('R squared:',metrics.r2_score(y0_test,y0_pred))
```

```
Mean Absolute Error: 10.187751774422932
Mean Squared Error: 168.4340182710263
Root Mean Squared Error: 12.978213215655932
R squared: 0.16745957772201436
```

In [180]:
```
# RANDOM FOREST REGRESSOR
```

In [181]:
```
X1=dataset1.iloc[:,[0,2,3,4,5,6,7,8,9,11,13,14]].values
y1=dataset1.iloc[:,10].values
```

In [182]:
```
from sklearn.model_selection import train_test_split
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2,
```

In [183]:
```
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor()
regressor.fit(X1_train,y1_train)
```

Out[183]:
```
▾ RandomForestRegressor

RandomForestRegressor()
```

In [184]:
```
y1_pred=regressor.predict(X1_test)
```

In [185]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
mse=mean_squared_error(y1_test,y1_pred)
rmse=mse**.5
print("mse : ",mse)
print("rmse : ",rmse)
mae=mean_absolute_error(y1_test,y1_pred)
print("mae : ",mae)
```

```
mse :  166.82779232043177
rmse :  12.916183349597967
mae :  10.25042488350687
```

In [186]:
```
r2_score(y1_test,y1_pred)
```

Out[186]:
```
0.17619332666991772
```

In [ ]:
```

```