Problem Statement : Delivery time estimation using Artificial Neural Networks

In [1]:

```
import pandas as pd
import numpy as np
import datetime as dt
import category_encoders as ce
import numpy as np
import pandas as pd
import regex as re
import datetime as dt
import seaborn as sns
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (20, 6)
import warnings
warnings.filterwarnings("ignore")
from sklearn.neighbors import LocalOutlierFactor
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
```

In [ ]:

```

```

In [2]:

```
!gdown 1jql6ByYD4HHd9ESL8_v5PKn33PGHjE2k
```

```
/bin/bash: gdown: command not found
```

In [3]:

```
pip install category_encoders
```

```
Requirement already satisfied: category_encoders in /opt/conda/lib/python
3.7/site-packages (2.6.0)
Requirement already satisfied: scipy>=1.0.0 in /opt/conda/lib/python3.7/si
te-packages (from category_encoders) (1.7.3)
Requirement already satisfied: numpy>=1.14.0 in /opt/conda/lib/python3.7/s
ite-packages (from category_encoders) (1.21.6)
Requirement already satisfied: patsy>=0.5.1 in /opt/conda/lib/python3.7/si
te-packages (from category_encoders) (0.5.3)
Requirement already satisfied: statsmodels>=0.9.0 in /opt/conda/lib/python
3.7/site-packages (from category_encoders) (0.13.5)
Requirement already satisfied: scikit-learn>=0.20.0 in /opt/conda/lib/pyth
on3.7/site-packages (from category_encoders) (1.0.2)
Requirement already satisfied: pandas>=1.0.5 in /opt/conda/lib/python3.7/s
ite-packages (from category_encoders) (1.3.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/py
thon3.7/site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/si
te-packages (from pandas>=1.0.5->category_encoders) (2022.7.1)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packag
es (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/si
te-packages (from scikit-learn>=0.20.0->category_encoders) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/pyth
on3.7/site-packages (from scikit-learn>=0.20.0->category_encoders) (3.1.0)
Requirement already satisfied: packaging>=21.3 in /opt/conda/lib/python3.
7/site-packages (from statsmodels>=0.9.0->category_encoders) (23.0)
WARNING: Running pip as the 'root' user can result in broken permissions a
nd conflicting behaviour with the system package manager. It is recommende
d to use a virtual environment instead: https://pip.pypa.io/warnings/venv
(https://pip.pypa.io/warnings/venv)
WARNING: There was an error checking the latest version of pip.
Note: you may need to restart the kernel to use updated packages.
```

In [4]:

```
pip install seaborn
```

```
Requirement already satisfied: seaborn in /opt/conda/lib/python3.7/site-pa
ckages (0.12.2)
Requirement already satisfied: typing_extensions in /opt/conda/lib/python
3.7/site-packages (from seaborn) (4.4.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /opt/conda/lib/p
ython3.7/site-packages (from seaborn) (3.5.3)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in /opt/conda/lib/pyth
on3.7/site-packages (from seaborn) (1.21.6)
Requirement already satisfied: pandas>=0.25 in /opt/conda/lib/python3.7/si
te-packages (from seaborn) (1.3.5)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/s
ite-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.
7/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.0)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/pyth
on3.7/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python
3.7/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.38.0)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.
7/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python
3.7/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/si
te-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/si
te-packages (from pandas>=0.25->seaborn) (2022.7.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-p
ackages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.1
6.0)
WARNING: Running pip as the 'root' user can result in broken permissions a
nd conflicting behaviour with the system package manager. It is recommende
d to use a virtual environment instead: https://pip.pypa.io/warnings/venv
(https://pip.pypa.io/warnings/venv)
WARNING: There was an error checking the latest version of pip.
Note: you may need to restart the kernel to use updated packages.
```

Reading Data

In [5]:

```
import os
os.listdir('/kaggle/input')
```

Out[5]:

```
['newdataset']
```

In [2]:

```
df=pd.read_csv('dataset.csv')
```

In [3]:

```
df
```

Out[3]:

	market_id	created_at	actual_delivery_time	store_id	store
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

Checking the structure & characteristics of the dataset

In [4]:

```
df.describe()
```

Out[4]:

	market_id	order_protocol	total_items	subtotal	num_distinct_items	mir
count	196441.000000	196433.000000	197428.000000	197428.000000	197428.000000	19
mean	2.978706	2.882352	3.196391	2682.331402	2.670791	
std	1.524867	1.503771	2.666546	1823.093688	1.630255	
min	1.000000	1.000000	1.000000	0.000000	1.000000	
25%	2.000000	1.000000	2.000000	1400.000000	1.000000	
50%	3.000000	3.000000	3.000000	2200.000000	2.000000	
75%	4.000000	4.000000	4.000000	3395.000000	3.000000	
max	6.000000	7.000000	411.000000	27100.000000	20.000000	1

In [5]:

```
df.describe(include='object')
```

Out[5]:

	created_at	actual_delivery_time	store_id	store_primary_ca
count	197428	197421	197428	1
unique	180985	178110	6743	
top	2015-02-11 19:50:43	2015-02-11 20:40:45	d43ab110ab2489d6b9b2caa394bf920f	an
freq	6	5	937	

In [6]:

```
df.isnull().sum()
# There are null values in couple of columns
```

Out[6]:

```
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_orders   16262
dtype: int64
```

In [7]:

```
df.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
```

Data preprocessing

In [8]:

```
df['market_id'].value_counts(normalize=True)
# 28.02 % values are from market_id 2
```

Out[8]:

```
2.0    0.280278
4.0    0.242307
1.0    0.193631
3.0    0.118595
5.0    0.091631
6.0    0.073559
Name: market_id, dtype: float64
```

In [9]:

```
df['store_id'].value_counts(normalize=True)
```

Out[9]:

```
d43ab110ab2489d6b9b2caa394bf920f    0.004746
757b505cfd34c64c85ca5b5690ee5293    0.004371
faacbcd5bf1d018912c116bf2783e9a1    0.004128
cfecdb276f634854f3ef915e2e980c31    0.003875
45c48cce2e2d7fbdea1afc51c7c6ad26    0.003652
                                      ...
adad0f2b196a1ed3e3b9d9025c397132    0.000005
2e6d9c6052e99fcdfa61d9b9da273ca2    0.000005
25daeb9b3072e9c53f66a2196a92a011    0.000005
55285adfd78a019a3245917649e29b3c    0.000005
df263d996281d984952c07998dc54358    0.000005
Name: store_id, Length: 6743, dtype: float64
```

In [10]:

```
df['store_primary_category'].value_counts(normalize=True)
# There are 74 Unique values of stope_primary_category
```

Out[10]:

```
american             0.100686
pizza                0.089901
mexican              0.088749
burger               0.056875
sandwich             0.052214
                         ...
lebanese             0.000047
belgian              0.000010
indonesian           0.000010
chocolate            0.000005
alcohol-plus-food    0.000005
Name: store_primary_category, Length: 74, dtype: float64
```

In [11]:

```
df['order_protocol'].value_counts(normalize=True)
# 27.85% of the order protocols are from 1 followed by 3 and 5
```

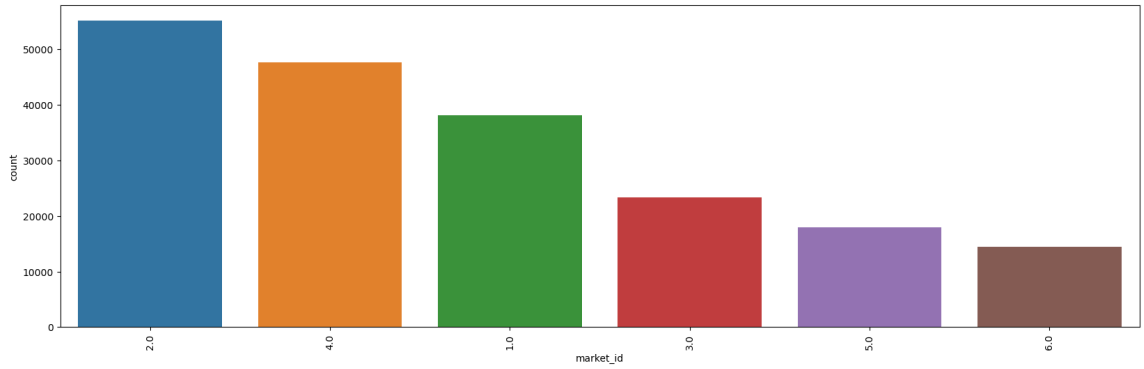
Out[11]:

```
1.0    0.278594
3.0    0.270825
5.0    0.225471
2.0    0.122444
4.0    0.098527
6.0    0.004042
7.0    0.000097
Name: order_protocol, dtype: float64
```

Data Visualization and cleaning

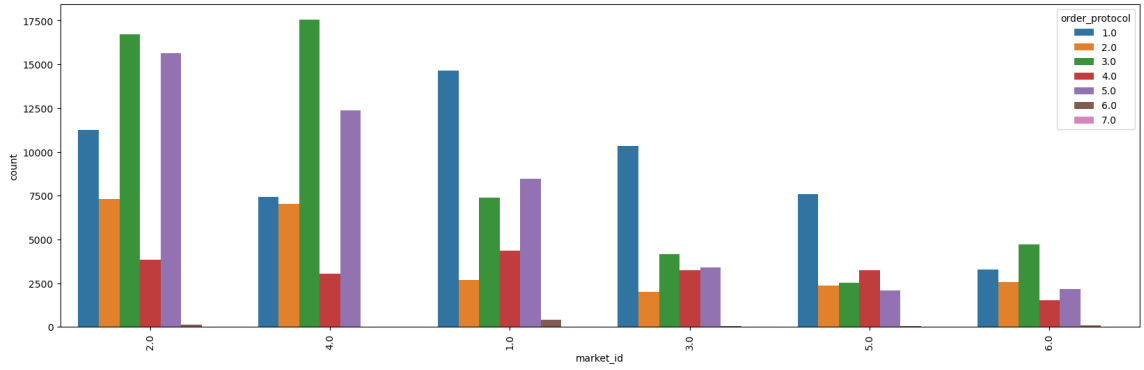
In [12]:

```
sns.countplot(data=df, x='market_id',order=df['market_id'].value_counts().index)
plt.xticks(rotation = 90)
plt.show()
# Count plot below shows number of market_id's
```



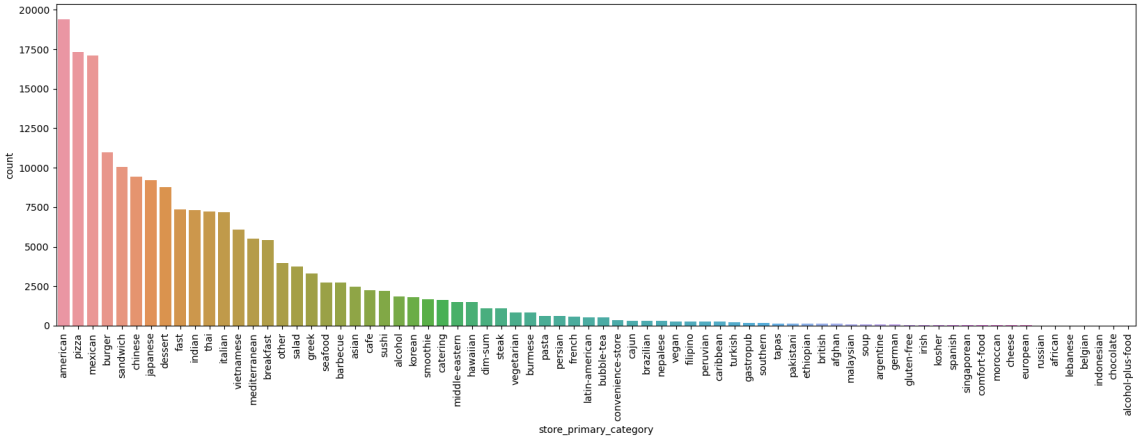
In [13]:

```
sns.countplot(data=df, x='market_id',order=df['market_id'].value_counts().index,hue='ord
plt.xticks(rotation = 90)
plt.show()
# order_protocol of 3 is highest with market_id 2 and 4
```



In [14]:

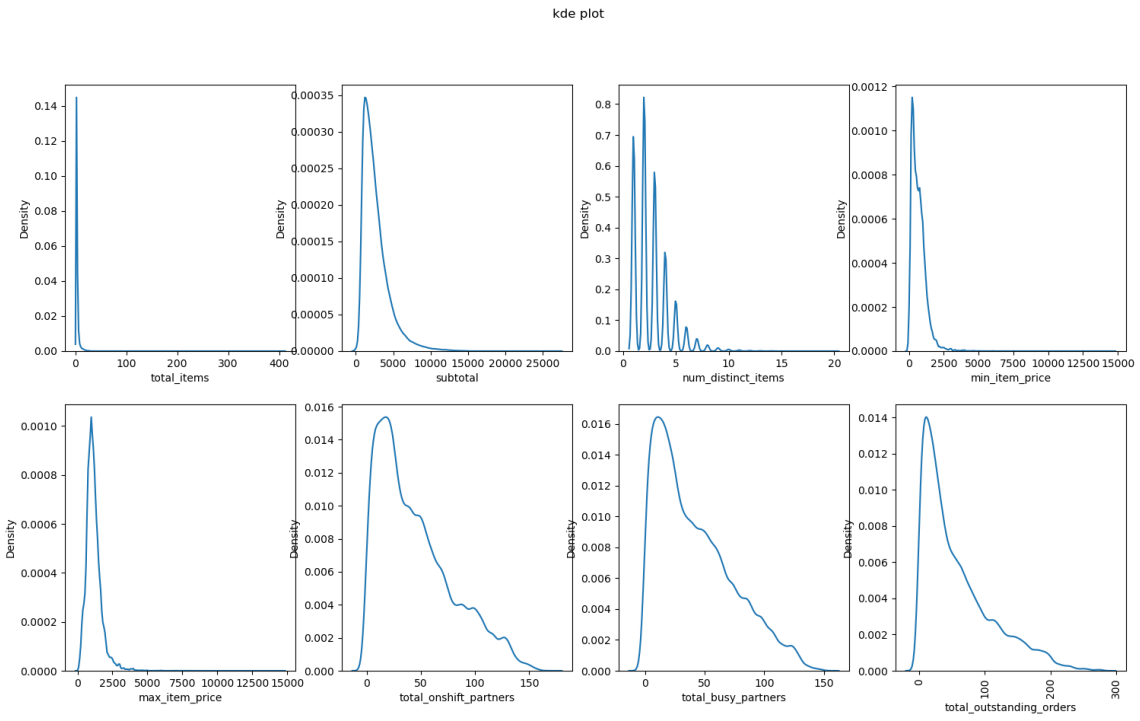
```
sns.countplot(data=df, x='store_primary_category',order=df['store_primary_category'].val
plt.xticks(rotation = 90)
plt.show()
# American, Pizza and Mexican are the top most items in the store_primary_category
```



In [15]:

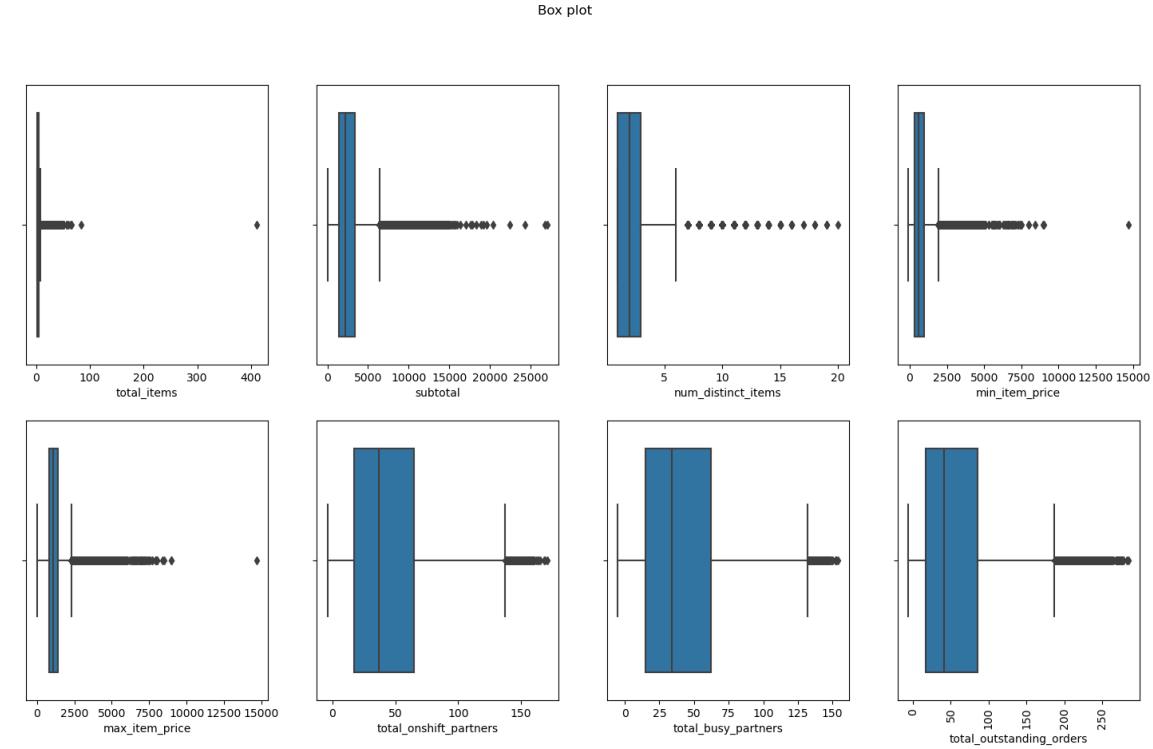
```
fig, axes = plt.subplots(2, 4, figsize=(18, 10))
plt.xticks(rotation = 90)
fig.suptitle('kde plot')
sns.kdeplot(ax=axes[0, 0], data=df, x='total_items')
sns.kdeplot(ax=axes[0, 1], data=df, x='subtotal')
sns.kdeplot(ax=axes[0, 2], data=df, x='num_distinct_items' )
sns.kdeplot(ax=axes[0, 3], data=df, x='min_item_price' )
sns.kdeplot(ax=axes[1, 0], data=df, x='max_item_price')
sns.kdeplot(ax=axes[1, 1], data=df, x='total_onshift_partners')
sns.kdeplot(ax=axes[1, 2], data=df, x='total_busy_partners')
sns.kdeplot(ax=axes[1, 3], data=df, x='total_outstanding_orders')
plt.show()

# kde plot for all the continuous variables
```



In [16]:

```
fig, axes = plt.subplots(2, 4, figsize=(18, 10))
plt.xticks(rotation = 90)
fig.suptitle('Box plot')
sns.boxplot(ax=axes[0, 0], data=df, x='total_items')
sns.boxplot(ax=axes[0, 1], data=df, x='subtotal')
sns.boxplot(ax=axes[0, 2], data=df, x='num_distinct_items' )
sns.boxplot(ax=axes[0, 3], data=df, x='min_item_price' )
sns.boxplot(ax=axes[1, 0], data=df, x='max_item_price')
sns.boxplot(ax=axes[1, 1], data=df, x='total_onshift_partners')
sns.boxplot(ax=axes[1, 2], data=df, x='total_busy_partners')
sns.boxplot(ax=axes[1, 3], data=df, x='total_outstanding_orders')
plt.show()
# Boxplot for all the continuous variables
# There are outliers in each variable
```

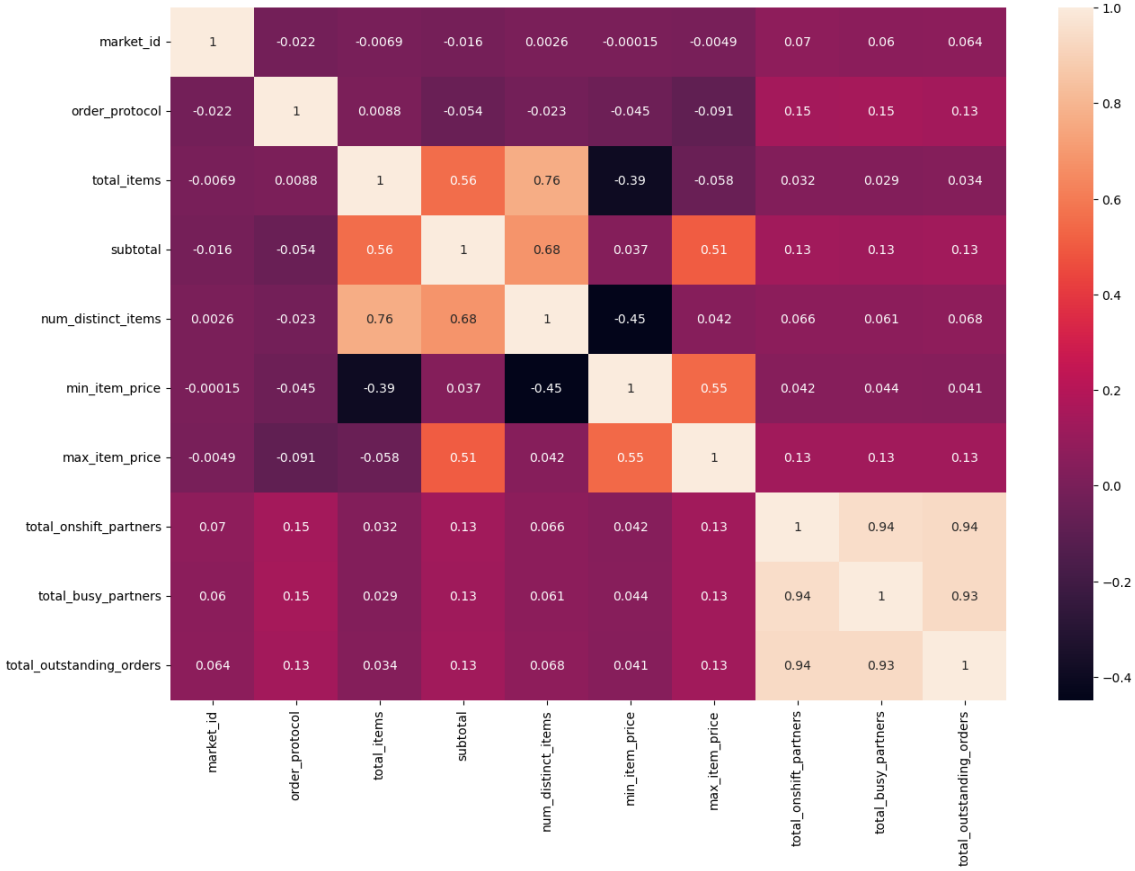


In [17]:

```
fig, ax = plt.subplots(figsize=(15,10))
sns.heatmap(df.corr(method ='pearson'),annot=True,ax=ax)
# Pearson corr
```

Out[17]:

<AxesSubplot:>

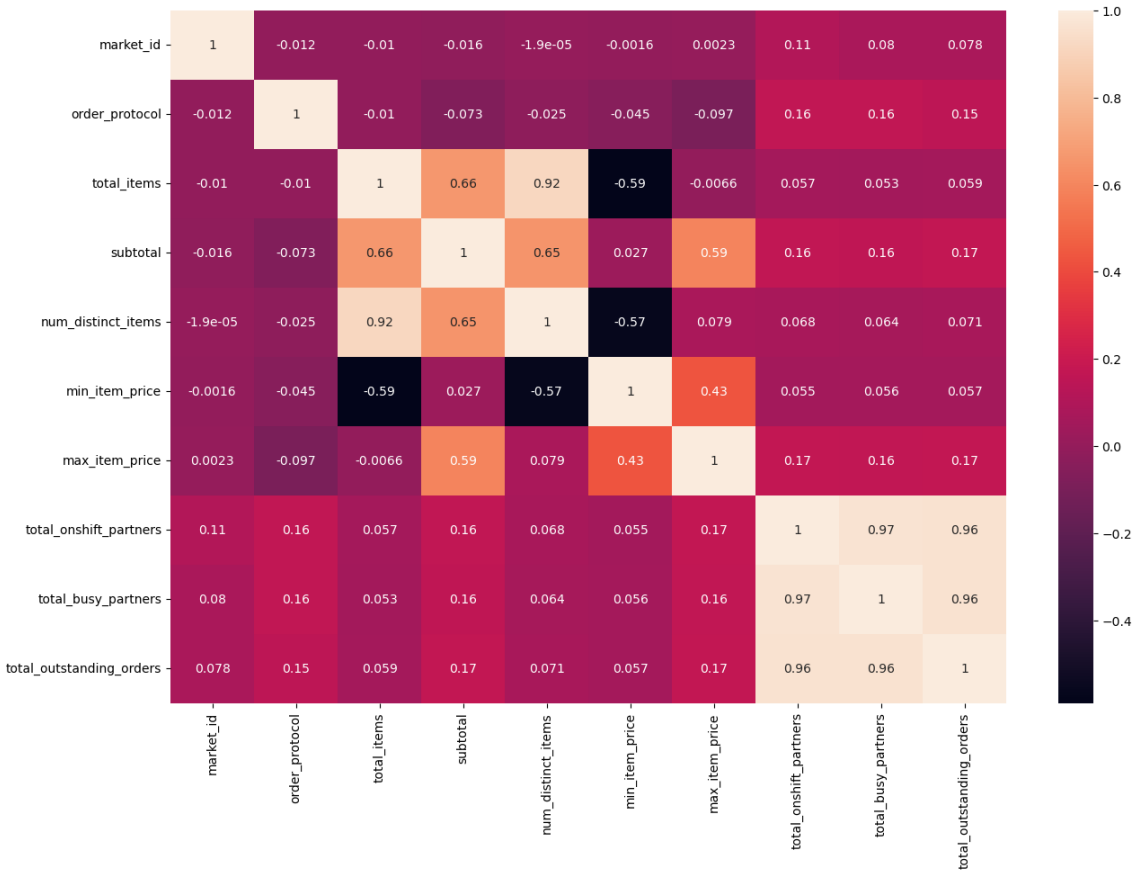


In [18]:

```
fig, ax = plt.subplots(figsize=(15,10))
sns.heatmap(df.corr(method ='spearman'),annot=True,ax=ax)
```

Out[18]:

<AxesSubplot:>



Missing values Imputation

In [19]:

```
dff=df[['store_id','store_primary_category']]
df1=pd.DataFrame(dff.groupby(['store_id','store_primary_category'])['store_primary_categ
if 'store_primary_category' in df1.columns:
    df1 = df1.rename(columns={'store_primary_category': 'count'})
df1 = df1.reset_index()
df1=df1.sort_values(by = ['store_id','store_primary_category','count'], ascending = [Tru
df1=df1.groupby(['store_id']).first().reset_index()
df=df.merge(df1, how='left', on=['store_id'])
for i in range(len(df)):
    if pd.isnull(df.iloc[i,4]):
        if pd.isnull(df.iloc[i,14])==False:
            df.iloc[i,4]=df.iloc[i,14]
df.drop(['store_primary_category_y','count'],axis=1,inplace=True)
df.rename(columns={'store_primary_category_x':'store_primary_category'},inplace=True)

# Filling missing store_primary_category with count of max occurance based on store_id
```

In [20]:

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

Out[20]:

```
market_id                    987
created_at                     0
actual_delivery_time           7
store_id                       0
store_primary_category       867
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_orders   16262
dtype: int64
```

In [21]:

```
dff=df[['store_id','market_id']]
df1=pd.DataFrame(dff.groupby(['store_id','market_id'])['market_id'].agg('count'))
if 'market_id' in df1.columns:
    df1 = df1.rename(columns={'market_id': 'count'})
df1 = df1.reset_index()
df1=df1.sort_values(by = ['store_id','market_id','count'], ascending = [True,True,False])
df1=df1.groupby(['store_id']).first().reset_index()
df=df.merge(df1, how='left', on=['store_id'])
for i in range(len(df)):
    if pd.isnull(df.iloc[i,0]):
        if pd.isnull(df.iloc[i,14])==False:
            df.iloc[i,0]=df.iloc[i,14]
df.drop(['market_id_y','count'],axis=1,inplace=True)
df.columns=['market_id', 'created_at', 'actual_delivery_time', 'store_id',
       'store_primary_category', 'order_protocol', 'total_items', 'subtotal',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders']
df.dropna(subset=['market_id', 'actual_delivery_time'],inplace=True)

# Filling missing market_id
```

In [22]:

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

Out[22]:

```
market_id                     0
created_at                    0
actual_delivery_time          0
store_id                      0
store_primary_category      866
order_protocol              992
total_items                   0
subtotal                      0
num_distinct_items            0
min_item_price                0
max_item_price                0
total_onshift_partners    16261
total_busy_partners       16261
total_outstanding_orders  16261
dtype: int64
```

In [23]:

```
dff=df[['store_id','order_protocol']]
df1=pd.DataFrame(dff.groupby(['store_id','order_protocol'])['order_protocol'].agg('count
if 'order_protocol' in df1.columns:
    df1 = df1.rename(columns={'order_protocol': 'count'})
df1 = df1.reset_index()
df1=df1.sort_values(by = ['store_id','order_protocol','count'], ascending = [True,True,F
df1=df1.groupby(['store_id']).first().reset_index()
df=df.merge(df1, how='left', on=['store_id'])
for i in range(len(df)):
    if pd.isnull(df.iloc[i,5]):
        if pd.isnull(df.iloc[i,14])==False:
            df.iloc[i,5]=df.iloc[i,14]
df.drop(['order_protocol_y','count'],axis=1,inplace=True)
df.columns=['market_id', 'created_at', 'actual_delivery_time', 'store_id',
       'store_primary_category', 'order_protocol', 'total_items', 'subtotal',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders']
# filling missing market_id
```

In [24]:

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

Out[24]:

```
market_id                     0
created_at                    0
actual_delivery_time          0
store_id                      0
store_primary_category      866
order_protocol                0
total_items                   0
subtotal                      0
num_distinct_items            0
min_item_price                0
max_item_price                0
total_onshift_partners    16261
total_busy_partners       16261
total_outstanding_orders  16261
dtype: int64
```

In [25]:

```
dff=df[['market_id','store_primary_category']]
df1=pd.DataFrame(dff.groupby(['market_id','store_primary_category'])['store_primary_cate
if 'store_primary_category' in df1.columns:
    df1 = df1.rename(columns={'store_primary_category': 'count'})
df1 = df1.reset_index()
df1=df1.sort_values(by = ['market_id','store_primary_category','count'], ascending = [Tr
df1=df1.groupby(['market_id']).first().reset_index()
df=df.merge(df1, how='left', on=['market_id'])
for i in range(len(df)):
    if pd.isnull(df.iloc[i,4]):
        if pd.isnull(df.iloc[i,14])==False:
            df.iloc[i,4]=df.iloc[i,14]
df.drop(['store_primary_category_y','count'],axis=1,inplace=True)
df.columns=['market_id', 'created_at', 'actual_delivery_time', 'store_id',
       'store_primary_category', 'order_protocol', 'total_items', 'subtotal',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders']
# filling missing store_primary_category
```

In [26]:

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

Out[26]:

```
market_id                     0
created_at                    0
actual_delivery_time          0
store_id                      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_partners    16261
total_busy_partners       16261
total_outstanding_orders  16261
dtype: int64
```

Feature Engineering

In [27]:

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

In [28]:

```
df['created_at_hour']=df['created_at'].dt.hour
df['created_at_date']=df['created_at'].dt.date
```

In [29]:

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

Out[29]:

```
market_id                     0
created_at                    0
actual_delivery_time          0
store_id                      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_partners    16261
total_busy_partners       16261
total_outstanding_orders  16261
created_at_hour               0
created_at_date               0
dtype: int64
```

In [30]:

```
dff=df[['market_id', 'created_at_hour',
       'store_primary_category','total_onshift_partners']]
df1=pd.DataFrame(dff.groupby(['market_id', 'created_at_hour',
       'store_primary_category'])['total_onshift_partners'].agg('mean'))
if 'total_onshift_partners' in df1.columns:
    df1 = df1.rename(columns={'total_onshift_partners': 'count'})
df1 = df1.reset_index()

df=df.merge(df1, how='left', on=['market_id', 'created_at_hour',
       'store_primary_category'])
for i in range(len(df)):
    if pd.isnull(df.iloc[i,11]):
        if pd.isnull(df.iloc[i,16])==False:
            df.iloc[i,11]=df.iloc[i,16]
df.drop('count',axis=1,inplace=True)
dff=df[['market_id', 'created_at_hour',
       'store_primary_category','total_busy_partners']]
df1=pd.DataFrame(dff.groupby(['market_id', 'created_at_hour',
       'store_primary_category'])['total_busy_partners'].agg('mean'))
if 'total_busy_partners' in df1.columns:
    df1 = df1.rename(columns={'total_busy_partners': 'count'})
df1 = df1.reset_index()

df=df.merge(df1, how='left', on=['market_id', 'created_at_hour',
       'store_primary_category'])
for i in range(len(df)):
    if pd.isnull(df.iloc[i,12]):
        if pd.isnull(df.iloc[i,16])==False:
            df.iloc[i,12]=df.iloc[i,16]
df.drop('count',axis=1,inplace=True)
dff=df[['market_id', 'created_at_hour',
       'store_primary_category','total_outstanding_orders']]
df1=pd.DataFrame(dff.groupby(['market_id', 'created_at_hour',
       'store_primary_category'])['total_outstanding_orders'].agg('mean'))
if 'total_outstanding_orders' in df1.columns:
    df1 = df1.rename(columns={'total_outstanding_orders': 'count'})
df1 = df1.reset_index()

df=df.merge(df1, how='left', on=['market_id', 'created_at_hour',
       'store_primary_category'])
for i in range(len(df)):
    if pd.isnull(df.iloc[i,13]):
        if pd.isnull(df.iloc[i,16])==False:
            df.iloc[i,13]=df.iloc[i,16]
df.drop('count',axis=1,inplace=True)

# Filling missing values of total_onshift_partners,total_busy_partners and total_outstan
```

In [31]:

```
df.isnull().sum()
# There are still some missing values
```

Out[31]:

```
market_id                     0
created_at                    0
actual_delivery_time          0
store_id                      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_partners     3121
total_busy_partners        3121
total_outstanding_orders   3121
created_at_hour               0
created_at_date               0
dtype: int64
```

In [32]:

```
3121/len(df)
# We are droping 1.58% of values
```

Out[32]:

```
0.015809095421896686
```

In [33]:

```
df.dropna(subset=['total_onshift_partners', 'total_busy_partners','total_outstanding_ord
```

In [34]:

```
df.isnull().sum()
# There are no null values
```

Out[34]:

```
market_id                  0
created_at                 0
actual_delivery_time       0
store_id                   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_partners     0
total_busy_partners        0
total_outstanding_orders   0
created_at_hour            0
created_at_date            0
dtype: int64
```

In [35]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 194297 entries, 0 to 197417
Data columns (total 16 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                194297 non-null  float64
 1   created_at               194297 non-null  datetime64[ns]
 2   actual_delivery_time     194297 non-null  datetime64[ns]
 3   store_id                 194297 non-null  object
 4   store_primary_category   194297 non-null  object
 5   order_protocol           194297 non-null  float64
 6   total_items              194297 non-null  int64
 7   subtotal                 194297 non-null  int64
 8   num_distinct_items       194297 non-null  int64
 9   min_item_price           194297 non-null  int64
 10  max_item_price           194297 non-null  int64
 11  total_onshift_partners   194297 non-null  float64
 12  total_busy_partners      194297 non-null  float64
 13  total_outstanding_orders 194297 non-null  float64
 14  created_at_hour          194297 non-null  int64
 15  created_at_date          194297 non-null  object
dtypes: datetime64[ns](2), float64(5), int64(6), object(3)
memory usage: 25.2+ MB
```

Delivery time in minutes

In [36]:

```
df['delivery_time_min']=((df['actual_delivery_time']-df['created_at'])/np.timedelta64(1,
```

In [37]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 194297 entries, 0 to 197417
Data columns (total 17 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                194297 non-null  float64
 1   created_at               194297 non-null  datetime64[ns]
 2   actual_delivery_time     194297 non-null  datetime64[ns]
 3   store_id                 194297 non-null  object
 4   store_primary_category   194297 non-null  object
 5   order_protocol           194297 non-null  float64
 6   total_items              194297 non-null  int64
 7   subtotal                 194297 non-null  int64
 8   num_distinct_items       194297 non-null  int64
 9   min_item_price           194297 non-null  int64
 10  max_item_price           194297 non-null  int64
 11  total_onshift_partners   194297 non-null  float64
 12  total_busy_partners      194297 non-null  float64
 13  total_outstanding_orders 194297 non-null  float64
 14  created_at_hour          194297 non-null  int64
 15  created_at_date          194297 non-null  object
 16  delivery_time_min        194297 non-null  float64
dtypes: datetime64[ns](2), float64(6), int64(6), object(3)
memory usage: 26.7+ MB
```

In [38]:

```
df['store_id'].nunique()
```

Out[38]:

```
6654
```

In [39]:

```
df['store_primary_category'].nunique()
```

Out[39]:

```
73
```

Encoding categorical Columns

In [40]:

```
encoder = ce.TargetEncoder(cols=['store_id'])
df_encoded = encoder.fit_transform(df['store_id'],df['delivery_time_min'])
df['store_id']=df_encoded
encoder = ce.TargetEncoder(cols=['store_primary_category'])
df_encoded = encoder.fit_transform(df['store_primary_category'],df['delivery_time_min'])
df['store_primary_category']=df_encoded
```

In [41]:

```
df['total_onshift_partners']=df['total_onshift_partners'].apply(lambda x : int(x))
df['total_busy_partners']=df['total_busy_partners'].apply(lambda x : int(x))
df['total_outstanding_orders']=df['total_outstanding_orders'].apply(lambda x : int(x))

# Converting to int values for all the float values as we had taken mean earlier
```

In [42]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 194297 entries, 0 to 197417
Data columns (total 17 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                194297 non-null  float64
 1   created_at               194297 non-null  datetime64[ns]
 2   actual_delivery_time     194297 non-null  datetime64[ns]
 3   store_id                 194297 non-null  float64
 4   store_primary_category   194297 non-null  float64
 5   order_protocol           194297 non-null  float64
 6   total_items              194297 non-null  int64
 7   subtotal                 194297 non-null  int64
 8   num_distinct_items       194297 non-null  int64
 9   min_item_price           194297 non-null  int64
 10  max_item_price           194297 non-null  int64
 11  total_onshift_partners   194297 non-null  int64
 12  total_busy_partners      194297 non-null  int64
 13  total_outstanding_orders 194297 non-null  int64
 14  created_at_hour          194297 non-null  int64
 15  created_at_date          194297 non-null  object
 16  delivery_time_min        194297 non-null  float64
dtypes: datetime64[ns](2), float64(5), int64(9), object(1)
memory usage: 26.7+ MB
```

Extracting weekday from data

In [43]:

```
df['created_at_weekday']=df['created_at'].dt.weekday
```

In [44]:

```
df
```

Out[44]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_category	order
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	50.395905	47.878758	
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	48.408865	44.533674	
2	3.0	2015-01-22 20:39:28	2015-01-22 21:09:09	48.408865	49.973629	
3	3.0	2015-02-03 21:21:45	2015-02-03 22:13:00	48.408865	49.973629	
4	3.0	2015-02-15 02:40:36	2015-02-15 03:20:26	48.408865	49.973629	
...	...	...	...	...	...	
197413	1.0	2015-02-17 00:19:41	2015-02-17 01:24:48	54.383918	44.039873	
197414	1.0	2015-02-13 00:01:59	2015-02-13 00:58:22	54.383918	44.039873	
197415	1.0	2015-01-24 04:46:08	2015-01-24 05:36:16	54.383918	44.039873	
197416	1.0	2015-02-01 18:18:15	2015-02-01 19:23:22	48.883704	44.656187	
197417	1.0	2015-02-08 19:24:33	2015-02-08 20:01:41	48.883704	44.656187	

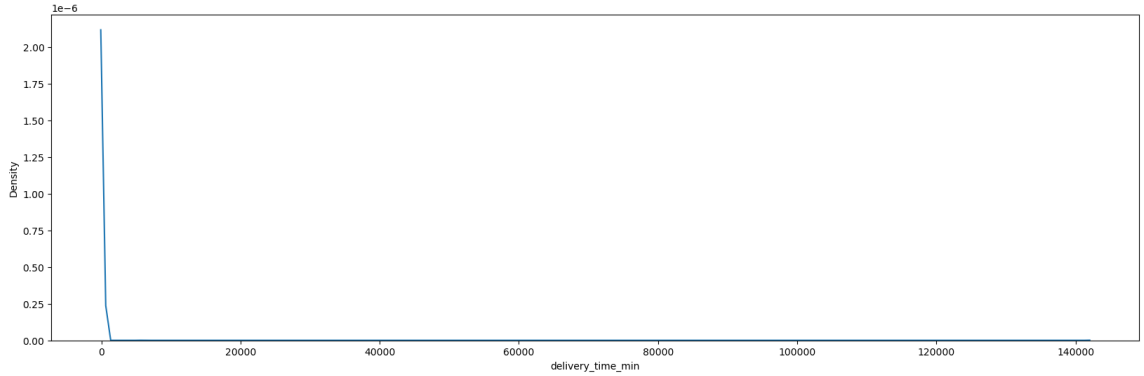
194297 rows × 18 columns

In [45]:

```
sns.kdeplot(df['delivery_time_min'])
```

Out[45]:

```
<AxesSubplot:xlabel='delivery_time_min', ylabel='Density'>
```

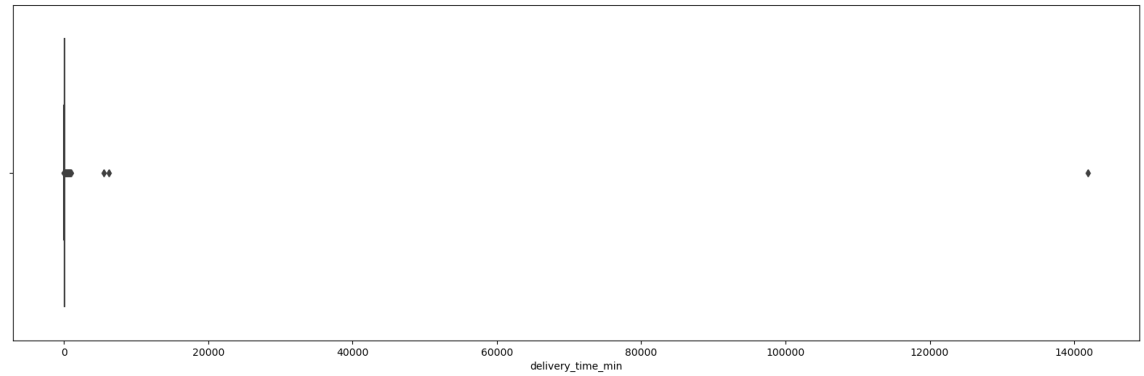


In [46]:

```
sns.boxplot(df['delivery_time_min'])
```

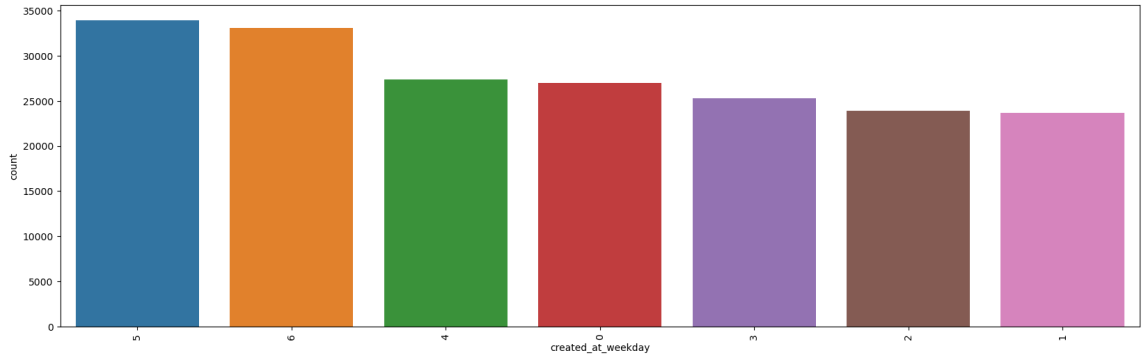
Out[46]:

```
<AxesSubplot:xlabel='delivery_time_min'>
```



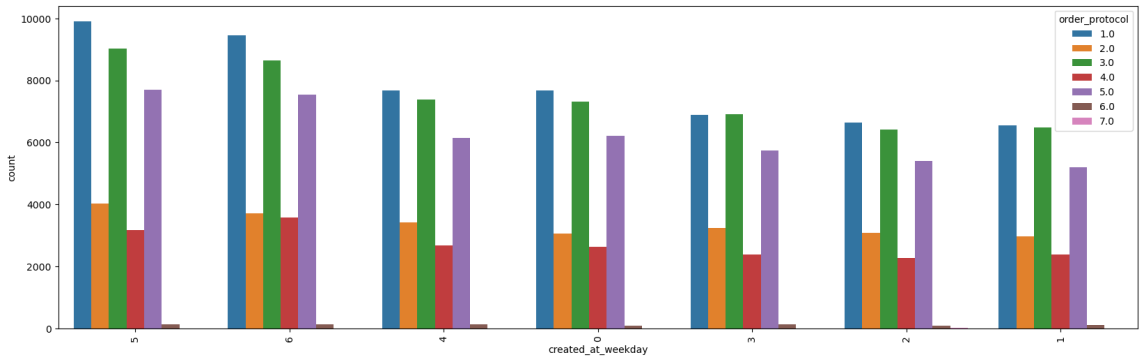
In [47]:

```
sns.countplot(data=df, x='created_at_weekday',order=df['created_at_weekday'].value_count
plt.xticks(rotation = 90)
plt.show()
# There more orders on weekday 5 and 6 i.e week day
```



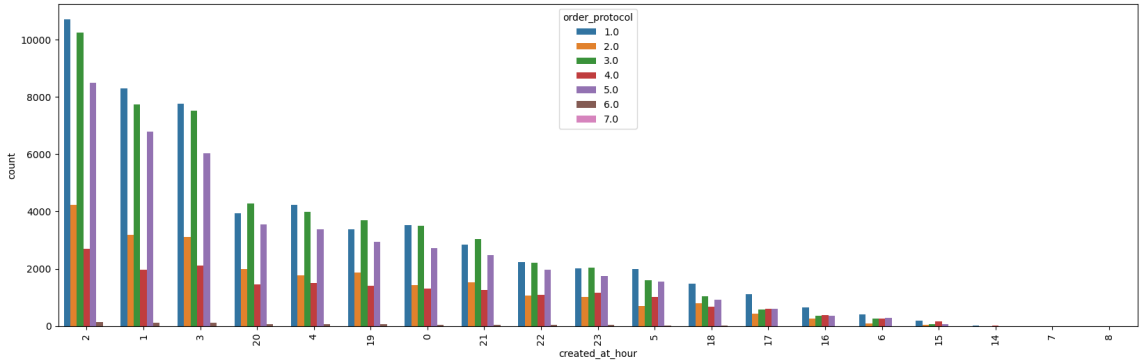
In [48]:

```
sns.countplot(data=df, x='created_at_weekday',order=df['created_at_weekday'].value_count
plt.xticks(rotation = 90)
plt.show()
# Least number of orders are on weekday 6
```



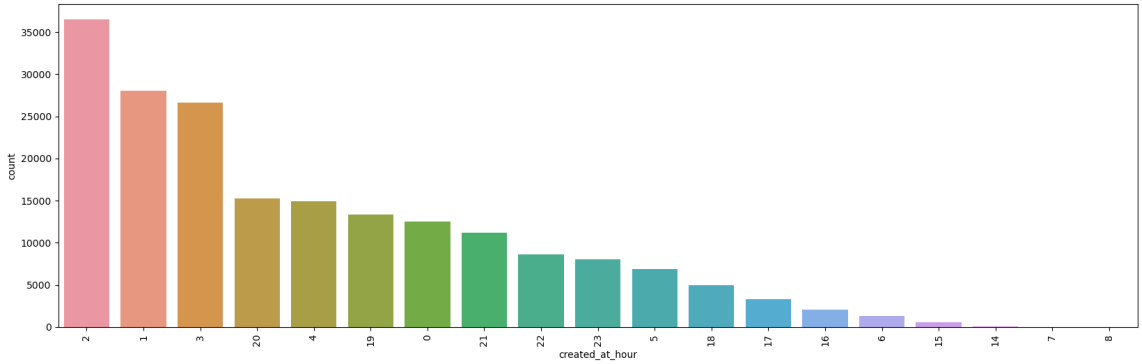
In [49]:

```
sns.countplot(data=df, x='created_at_hour',order=df['created_at_hour'].value_counts().in
plt.xticks(rotation = 90)
plt.show()
# Most of the orders are created at hours 2, 1 and 3
```



In [50]:

```
sns.countplot(data=df, x='created_at_hour',order=df['created_at_hour'].value_counts().in
plt.xticks(rotation = 90)
plt.show()
```



In [51]:

```
df.info()
```

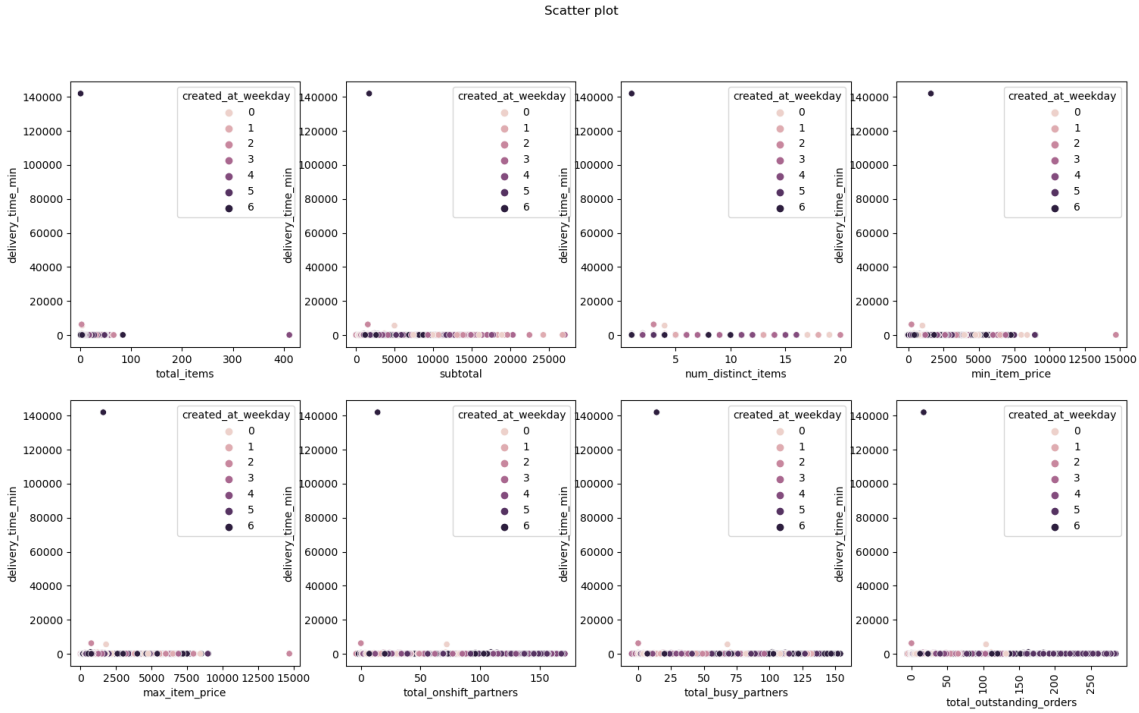
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 194297 entries, 0 to 197417
Data columns (total 18 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                194297 non-null  float64
 1   created_at               194297 non-null  datetime64[ns]
 2   actual_delivery_time     194297 non-null  datetime64[ns]
 3   store_id                 194297 non-null  float64
 4   store_primary_category   194297 non-null  float64
 5   order_protocol           194297 non-null  float64
 6   total_items              194297 non-null  int64
 7   subtotal                 194297 non-null  int64
 8   num_distinct_items       194297 non-null  int64
 9   min_item_price           194297 non-null  int64
 10  max_item_price           194297 non-null  int64
 11  total_onshift_partners   194297 non-null  int64
 12  total_busy_partners      194297 non-null  int64
 13  total_outstanding_orders 194297 non-null  int64
 14  created_at_hour          194297 non-null  int64
 15  created_at_date          194297 non-null  object
 16  delivery_time_min        194297 non-null  float64
 17  created_at_weekday       194297 non-null  int64
dtypes: datetime64[ns](2), float64(5), int64(10), object(1)
memory usage: 28.2+ MB
```

In [52]:

```
fig, axes = plt.subplots(2, 4, figsize=(18, 10))
plt.xticks(rotation = 90)
fig.suptitle('Scatter plot')
sns.scatterplot(ax=axes[0, 0], data=df,x='total_items',y='delivery_time_min',hue='create
sns.scatterplot(ax=axes[0, 1], data=df,x='subtotal',y='delivery_time_min',hue='created_a
sns.scatterplot(ax=axes[0, 2], data=df,x='num_distinct_items',y='delivery_time_min',hue=
sns.scatterplot(ax=axes[0, 3], data=df,x='min_item_price',y='delivery_time_min',hue='cre
sns.scatterplot(ax=axes[1, 0], data=df,x='max_item_price',y='delivery_time_min',hue='cre
sns.scatterplot(ax=axes[1, 1], data=df,x='total_onshift_partners',y='delivery_time_min',
sns.scatterplot(ax=axes[1, 2], data=df,x='total_busy_partners',y='delivery_time_min',hue
sns.scatterplot(ax=axes[1, 3], data=df,x='total_outstanding_orders',y='delivery_time_min
```

Out[52]:

```
<AxesSubplot:xlabel='total_outstanding_orders', ylabel='delivery_time_mi
n'>
```



In [53]:

```
df['created_at_date_month']=pd.to_datetime(df['created_at_date']).dt.month
df['created_at_date_year']=pd.to_datetime(df['created_at_date']).dt.year
df['created_at_date_day']=pd.to_datetime(df['created_at_date']).dt.day
# Extracting month year and day
```

In [54]:

```
df.columns
```

Out[54]:

```
Index(['market_id', 'created_at', 'actual_delivery_time', 'store_id',
       'store_primary_category', 'order_protocol', 'total_items', 'subtota
l',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders', 'created_at_hour', 'created_at_date',
       'delivery_time_min', 'created_at_weekday', 'created_at_date_month',
       'created_at_date_year', 'created_at_date_day'],
      dtype='object')
```

In [55]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 194297 entries, 0 to 197417
Data columns (total 21 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                194297 non-null  float64
 1   created_at               194297 non-null  datetime64[ns]
 2   actual_delivery_time     194297 non-null  datetime64[ns]
 3   store_id                 194297 non-null  float64
 4   store_primary_category   194297 non-null  float64
 5   order_protocol           194297 non-null  float64
 6   total_items              194297 non-null  int64
 7   subtotal                 194297 non-null  int64
 8   num_distinct_items       194297 non-null  int64
 9   min_item_price           194297 non-null  int64
 10  max_item_price           194297 non-null  int64
 11  total_onshift_partners   194297 non-null  int64
 12  total_busy_partners      194297 non-null  int64
 13  total_outstanding_orders 194297 non-null  int64
 14  created_at_hour          194297 non-null  int64
 15  created_at_date          194297 non-null  object
 16  delivery_time_min        194297 non-null  float64
 17  created_at_weekday       194297 non-null  int64
 18  created_at_date_month    194297 non-null  int64
 19  created_at_date_year     194297 non-null  int64
 20  created_at_date_day      194297 non-null  int64
dtypes: datetime64[ns](2), float64(5), int64(13), object(1)
memory usage: 32.6+ MB
```

In [56]:

```
dff=df[['market_id','store_id',
       'store_primary_category', 'order_protocol', 'total_items', 'subtotal',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders', 'created_at_hour',
       'delivery_time_min', 'created_at_weekday', 'created_at_date_month',
       'created_at_date_year', 'created_at_date_day']]
```

In [57]:

```
dff.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 194297 entries, 0 to 197417
Data columns (total 18 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   market_id                 194297 non-null  float64
 1   store_id                  194297 non-null  float64
 2   store_primary_category    194297 non-null  float64
 3   order_protocol            194297 non-null  float64
 4   total_items               194297 non-null  int64  
 5   subtotal                  194297 non-null  int64  
 6   num_distinct_items        194297 non-null  int64  
 7   min_item_price            194297 non-null  int64  
 8   max_item_price            194297 non-null  int64  
 9   total_onshift_partners    194297 non-null  int64  
 10  total_busy_partners       194297 non-null  int64  
 11  total_outstanding_orders  194297 non-null  int64  
 12  created_at_hour           194297 non-null  int64  
 13  delivery_time_min         194297 non-null  float64
 14  created_at_weekday        194297 non-null  int64  
 15  created_at_date_month     194297 non-null  int64  
 16  created_at_date_year      194297 non-null  int64  
 17  created_at_date_day       194297 non-null  int64  
dtypes: float64(5), int64(13)
memory usage: 28.2 MB
```

In [58]:

```
X=np.array(dff)
```

Outlier removal

In [59]:

```
clf = LocalOutlierFactor(n_neighbors=25,contamination=0.05)
```

In [60]:

```
Y=clf.fit_predict(X)
```

In [61]:

```
pd.Series(Y).value_counts()
```

Out[61]:

```
 1    184582
-1      9715
dtype: int64
```

In [62]:

```
df['out']=pd.Series(Y)
```

In [63]:

```
df1=df[df['out']==1]
# Removing 9715 rows
```

In [64]:

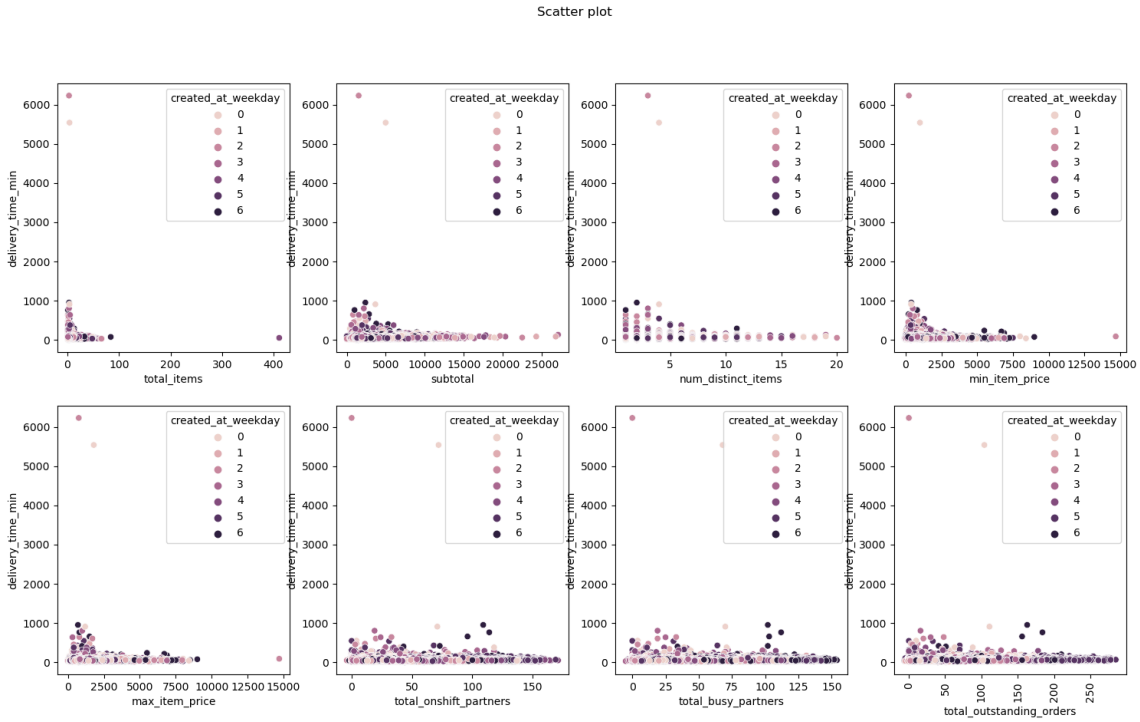
```
df1=df1[df1['delivery_time_min']<20000]
```

In [65]:

```
fig, axes = plt.subplots(2, 4, figsize=(18, 10))
plt.xticks(rotation = 90)
fig.suptitle('Scatter plot')
sns.scatterplot(ax=axes[0, 0], data=df1,x='total_items',y='delivery_time_min',hue='creat
sns.scatterplot(ax=axes[0, 1], data=df1,x='subtotal',y='delivery_time_min',hue='created_
sns.scatterplot(ax=axes[0, 2], data=df1,x='num_distinct_items',y='delivery_time_min',hue
sns.scatterplot(ax=axes[0, 3], data=df1,x='min_item_price',y='delivery_time_min',hue='cr
sns.scatterplot(ax=axes[1, 0], data=df1,x='max_item_price',y='delivery_time_min',hue='cr
sns.scatterplot(ax=axes[1, 1], data=df1,x='total_onshift_partners',y='delivery_time_min'
sns.scatterplot(ax=axes[1, 2], data=df1,x='total_busy_partners',y='delivery_time_min',hu
sns.scatterplot(ax=axes[1, 3], data=df1,x='total_outstanding_orders',y='delivery_time_mi
```

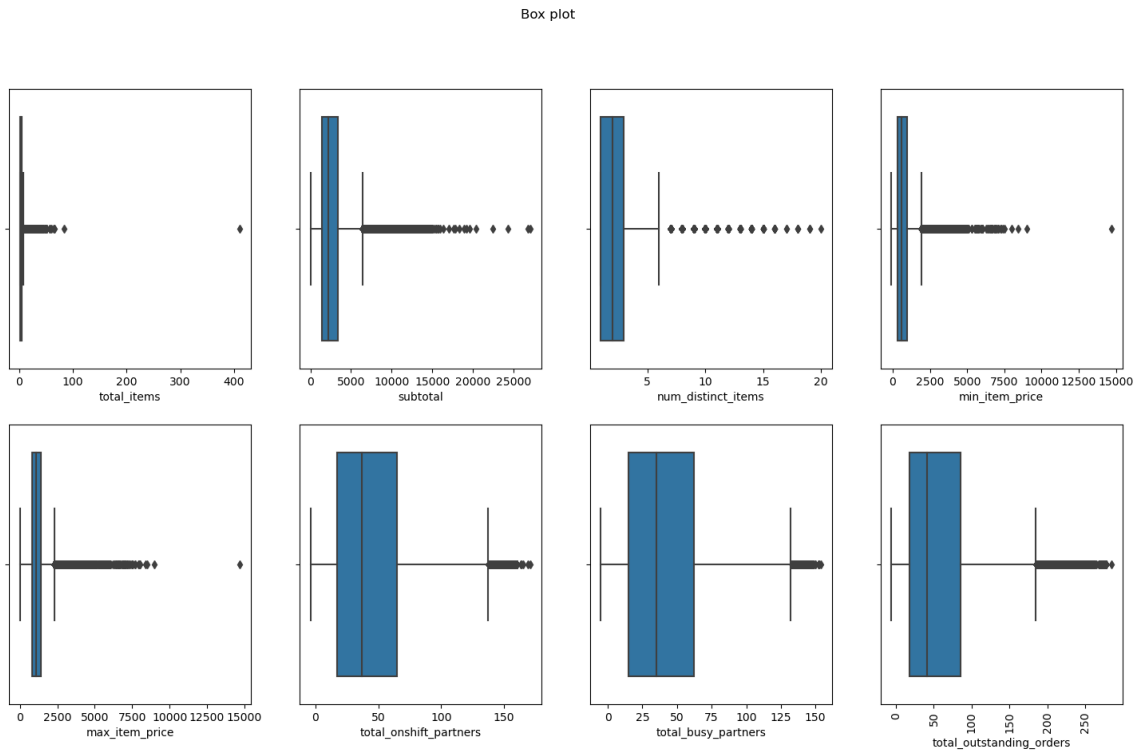
Out[65]:

```
<AxesSubplot:xlabel='total_outstanding_orders', ylabel='delivery_time_mi
n'>
```



In [66]:

```
fig, axes = plt.subplots(2, 4, figsize=(18, 10))
plt.xticks(rotation = 90)
fig.suptitle('Box plot')
sns.boxplot(ax=axes[0, 0], data=df1, x='total_items')
sns.boxplot(ax=axes[0, 1], data=df1, x='subtotal')
sns.boxplot(ax=axes[0, 2], data=df1, x='num_distinct_items' )
sns.boxplot(ax=axes[0, 3], data=df1, x='min_item_price' )
sns.boxplot(ax=axes[1, 0], data=df1, x='max_item_price')
sns.boxplot(ax=axes[1, 1], data=df1, x='total_onshift_partners')
sns.boxplot(ax=axes[1, 2], data=df1, x='total_busy_partners')
sns.boxplot(ax=axes[1, 3], data=df1, x='total_outstanding_orders')
plt.show()
```



In [67]:

```
df1.columns
```

Out[67]:

```
Index(['market_id', 'created_at', 'actual_delivery_time', 'store_id',
       'store_primary_category', 'order_protocol', 'total_items', 'subtota
l',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders', 'created_at_hour', 'created_at_date',
       'delivery_time_min', 'created_at_weekday', 'created_at_date_month',
       'created_at_date_year', 'created_at_date_day', 'out'],
      dtype='object')
```

In [68]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 181663 entries, 0 to 194296
Data columns (total 22 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   market_id                181663 non-null  float64
 1   created_at               181663 non-null  datetime64[ns]
 2   actual_delivery_time     181663 non-null  datetime64[ns]
 3   store_id                 181663 non-null  float64
 4   store_primary_category   181663 non-null  float64
 5   order_protocol           181663 non-null  float64
 6   total_items              181663 non-null  int64
 7   subtotal                 181663 non-null  int64
 8   num_distinct_items       181663 non-null  int64
 9   min_item_price           181663 non-null  int64
 10  max_item_price           181663 non-null  int64
 11  total_onshift_partners   181663 non-null  int64
 12  total_busy_partners      181663 non-null  int64
 13  total_outstanding_orders 181663 non-null  int64
 14  created_at_hour          181663 non-null  int64
 15  created_at_date          181663 non-null  object
 16  delivery_time_min        181663 non-null  float64
 17  created_at_weekday       181663 non-null  int64
 18  created_at_date_month    181663 non-null  int64
 19  created_at_date_year     181663 non-null  int64
 20  created_at_date_day      181663 non-null  int64
 21  out                      181663 non-null  float64
dtypes: datetime64[ns](2), float64(6), int64(13), object(1)
memory usage: 31.9+ MB
```

In [69]:

```
X=df1[['market_id', 'store_id',
       'store_primary_category', 'order_protocol', 'total_items', 'subtotal',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders', 'created_at_hour',  'created_at_weekday', 'created_at
       'created_at_date_year', 'created_at_date_day']]
y=df1['delivery_time_min']
```

In [70]:

```
X
```

Out[70]:

	market_id	store_id	store_primary_category	order_protocol	total_items	subtotal
0	1.0	50.395905	47.878758	1.0	4	3441
1	2.0	48.408865	44.533674	2.0	1	1900
2	3.0	48.408865	49.973629	1.0	1	1900
3	3.0	48.408865	49.973629	1.0	6	6900
4	3.0	48.408865	49.973629	1.0	3	3900
...	...	...	...	...	...	...
194292	6.0	46.855428	44.921455	2.0	1	1725
194293	6.0	46.855428	44.921455	2.0	1	1225
194294	6.0	46.855428	44.921455	2.0	1	1325
194295	1.0	46.855428	47.402817	3.0	1	1925
194296	6.0	46.855428	44.921455	2.0	1	1925

181663 rows × 17 columns

Train test split

In [71]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42
X_train, X_train_val, y_train, y_train_val = train_test_split(X_train, y_train, test_siz
```

In [72]:

```
len(X_train),len(X_train_val),len(X_test)
```

Out[72]:

```
(147146, 16350, 18167)
```

Random Forest

In [74]:

```
model=RandomForestRegressor(n_estimators=100, criterion='squared_error',
                            min_samples_split=2, min_samples_leaf=1, random_state=42)
```

In [75]:

```
model.fit(X_train,y_train)
```

Out[75]:

```
RandomForestRegressor(random_state=42)
```

In [76]:

```
model.score(X_train,  y_train)
```

Out[76]:

```
0.8839544982478807
```

In [77]:

```
model.score(X_train_val, y_train_val)
```

Out[77]:

```
0.3060957896075144
```

In [78]:

```
model.score(X_test,y_test)
```

Out[78]:

```
0.04430276036233449
```

In [79]:

```
y_train_pred=model.predict(X_train)
y_train_val_pred=model.predict(X_train_val)
y_test_pred=model.predict(X_test)
```

In [80]:

```
mae = mean_absolute_error(y_train, y_train_pred)
mse = mean_squared_error(y_train, y_train_pred)
rmse = np.sqrt(mse)
```

In [81]:

```
print(mae,mse,rmse)
```

```
4.077480348256855 66.88168821794162 8.178122536251314
```

In [82]:

```
mae = mean_absolute_error(y_train_val, y_train_val_pred)
mse = mean_squared_error(y_train_val, y_train_val_pred)
rmse = np.sqrt(mse)
```

In [83]:

```
print(mae,mse,rmse)
```

```
11.01597553715839 263.8691448959443 16.244049522700436
```

In [84]:

```
mae = mean_absolute_error(y_test, y_test_pred)
mse = mean_squared_error(y_test, y_test_pred)
rmse = np.sqrt(mse)
```

In [85]:

```
print(mae,mse,rmse)
```

```
11.343540791213092 2429.767955019698 49.292676484643216
```

In [86]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42
```

Data Scaling

In [87]:

```
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

In [88]:

```
X_train, X_train_val, y_train, y_train_val = train_test_split(X_train, y_train, test_siz
```

In [89]:

```
X_train.shape
```

Out[89]:

```
(147146, 17)
```

In [90]:

```
X_train_val.shape
```

Out[90]:

```
(16350, 17)
```

In [91]:

```
X_test.shape
```

Out[91]:

```
(18167, 17)
```

Neural Network

In [1]:

```
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Activation
from tensorflow.keras.activations import linear
from tensorflow.keras import Sequential
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.callbacks import LearningRateScheduler
```

In [2]:

```
def scheduler(epoch, lr):
  if epoch < 400:
    return lr
  else:
    return lr * tf.math.exp(-0.1)
```

In [3]:

```
L2Reg = tf.keras.regularizers.L1L2(l1=1e-7, l2=1e-7)
callback = tf.keras.callbacks.LearningRateScheduler(scheduler)
```

In [4]:

```
model = Sequential([
 Dense(32, activation="relu", input_shape=(17,), name="hidden_1",kernel_regularizer = L2

 Dense(64, activation="relu", name="hidden_2",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),


  Dense(128, activation="relu", name="hidden_3",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(256, activation="relu", name="hidden_4",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(512, activation="relu", name="hidden_5",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(1024, activation="relu", name="hidden_6",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(2028, activation="relu", name="hidden_7",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(4056, activation="relu", name="hidden_8",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

Dense(8052, activation="relu", name="hidden_9",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

Dense(4056, activation="relu", name="hidden_10",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(2028, activation="relu", name="hidden_11",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(1024, activation="relu", name="hidden_12",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(512, activation="relu", name="hidden_13",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(256, activation="relu", name="hidden_14",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),

  Dense(128, activation="relu", name="hidden_15",kernel_regularizer = L2Reg ),
 BatchNormalization(),
  Dropout(0.3),
```

```
 Dense(64, activation="relu", name="hidden_16",kernel_regularizer = L2Reg ),
BatchNormalization(),
 Dropout(0.3),

 Dense(32, activation="relu", name="hidden_17",kernel_regularizer = L2Reg ),
BatchNormalization(),
 Dropout(0.3),

 Dense(16, activation="relu", name="hidden_18",kernel_regularizer = L2Reg ),
BatchNormalization(),
 Dropout(0.2),

 Dense(8, activation="relu", name="hidden_19",kernel_regularizer = L2Reg ),
BatchNormalization(),
Dropout(0.2),


 Dense(4, activation="relu", name="hidden_20",kernel_regularizer = L2Reg ),
BatchNormalization(),
Dropout(0.2),


 Dense(2, activation="relu", name="hidden_21",kernel_regularizer = L2Reg ),
BatchNormalization(),



Dense(1, activation="linear", name="output")
])
```

In [5]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 hidden_1 (Dense)            (None, 32)                576

 hidden_2 (Dense)            (None, 64)                2112

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dropout (Dropout)           (None, 64)                0

 hidden_3 (Dense)            (None, 128)               8320

 batch_normalization_1 (Batc  (None, 128)              512
 hNormalization)

 dropout_1 (Dropout)         (None, 128)               0

 hidden_4 (Dense)            (None, 256)               33024

 batch_normalization_2 (Batc  (None, 256)              1024
 hNormalization)

 dropout_2 (Dropout)         (None, 256)               0

 hidden_5 (Dense)            (None, 512)               131584

 batch_normalization_3 (Batc  (None, 512)              2048
 hNormalization)

 dropout_3 (Dropout)         (None, 512)               0

 hidden_6 (Dense)            (None, 1024)              525312

 batch_normalization_4 (Batc  (None, 1024)             4096
 hNormalization)

 dropout_4 (Dropout)         (None, 1024)              0

 hidden_7 (Dense)            (None, 2028)              2078700

 batch_normalization_5 (Batc  (None, 2028)             8112
 hNormalization)

 dropout_5 (Dropout)         (None, 2028)              0

 hidden_8 (Dense)            (None, 4056)              8229624

 batch_normalization_6 (Batc  (None, 4056)             16224
 hNormalization)

 dropout_6 (Dropout)         (None, 4056)              0

 hidden_9 (Dense)            (None, 8052)              32666964

 batch_normalization_7 (Batc  (None, 8052)             32208
 hNormalization)

 dropout_7 (Dropout)         (None, 8052)              0
```

```
hidden_10 (Dense)           (None, 4056)              32662968

batch_normalization_8 (Batc  (None, 4056)             16224
hNormalization)

dropout_8 (Dropout)         (None, 4056)              0

hidden_11 (Dense)           (None, 2028)              8227596

batch_normalization_9 (Batc  (None, 2028)             8112
hNormalization)

dropout_9 (Dropout)         (None, 2028)              0

hidden_12 (Dense)           (None, 1024)              2077696

batch_normalization_10 (Bat  (None, 1024)             4096
chNormalization)

dropout_10 (Dropout)        (None, 1024)              0

hidden_13 (Dense)           (None, 512)               524800

batch_normalization_11 (Bat  (None, 512)              2048
chNormalization)

dropout_11 (Dropout)        (None, 512)               0

hidden_14 (Dense)           (None, 256)               131328

batch_normalization_12 (Bat  (None, 256)              1024
chNormalization)

dropout_12 (Dropout)        (None, 256)               0

hidden_15 (Dense)           (None, 128)               32896

batch_normalization_13 (Bat  (None, 128)              512
chNormalization)

dropout_13 (Dropout)        (None, 128)               0

hidden_16 (Dense)           (None, 64)                8256

batch_normalization_14 (Bat  (None, 64)               256
chNormalization)

dropout_14 (Dropout)        (None, 64)                0

hidden_17 (Dense)           (None, 32)                2080

batch_normalization_15 (Bat  (None, 32)               128
chNormalization)

dropout_15 (Dropout)        (None, 32)                0

hidden_18 (Dense)           (None, 16)                528

batch_normalization_16 (Bat  (None, 16)               64
chNormalization)
```

```
 dropout_16 (Dropout)        (None, 16)                0

 hidden_19 (Dense)           (None, 8)                 136

 batch_normalization_17 (Bat  (None, 8)                32
 chNormalization)

 dropout_17 (Dropout)        (None, 8)                 0

 hidden_20 (Dense)           (None, 4)                 36

 batch_normalization_18 (Bat  (None, 4)                16
 chNormalization)

 dropout_18 (Dropout)        (None, 4)                 0

 hidden_21 (Dense)           (None, 2)                 10

 batch_normalization_19 (Bat  (None, 2)                8
 chNormalization)

 output (Dense)              (None, 1)                 3

=================================================================
Total params: 87,441,549
Trainable params: 87,393,049
Non-trainable params: 48,500
_________________________________________________________________
```

In [6]:

```
from keras.utils.vis_utils import plot_model
plot_model(model, show_shapes=True, show_layer_names=True)
```

```
You must install pydot (`pip install pydot`) and install graphviz (see ins
tructions at https://graphviz.gitlab.io/download/) (https://graphviz.gitla
b.io/download/)) for plot_model to work.
```

In [118]:

```
model.compile(optimizer = tf.keras.optimizers.Adam(beta_1=0.9,beta_2=0.999),
 loss = tf.keras.losses.MeanAbsolutePercentageError(),
 metrics=['mean_absolute_error','mean_absolute_percentage_error','mean_squared_error'])
```

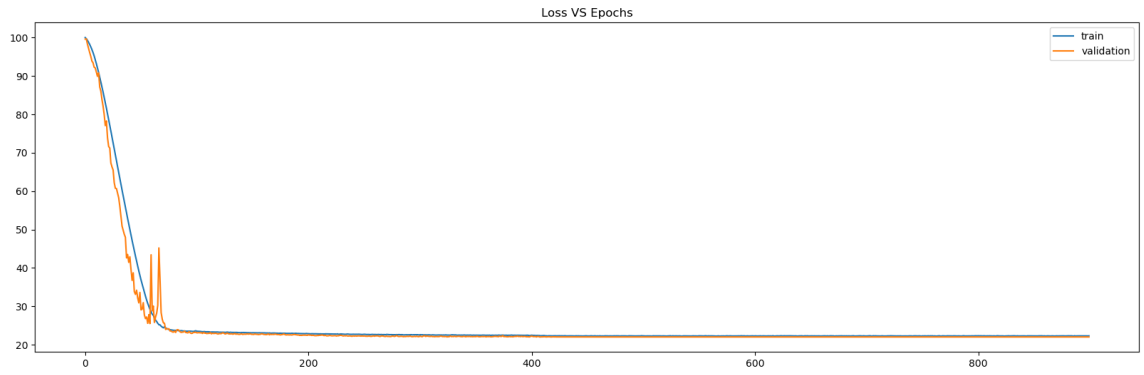
In [119]:

```
history = model.fit(X_train, y_train, validation_data = (X_train_val, y_train_val), epoc
```

```
Epoch 1/900
50/50 [==============================] - 32s 274ms/step - loss: 99.9924
- mean_absolute_error: 47.7042 - mean_absolute_percentage_error: 99.872
2 - mean_squared_error: 2853.6221 - val_loss: 99.5867 - val_mean_absolu
te_error: 47.8246 - val_mean_absolute_percentage_error: 99.4633 - val_m
ean_squared_error: 2667.5596 - lr: 0.0010
Epoch 2/900
50/50 [==============================] - 13s 258ms/step - loss: 99.6877
- mean_absolute_error: 47.5981 - mean_absolute_percentage_error: 99.563
7 - mean_squared_error: 2845.6909 - val_loss: 99.6468 - val_mean_absolu
te_error: 47.8593 - val_mean_absolute_percentage_error: 99.5224 - val_m
ean_squared_error: 2671.8210 - lr: 0.0010
Epoch 3/900
50/50 [==============================] - 13s 259ms/step - loss: 99.2764
- mean_absolute_error: 47.4694 - mean_absolute_percentage_error: 99.151
9 - mean_squared_error: 2837.8337 - val_loss: 98.2855 - val_mean_absolu
te_error: 47.4054 - val_mean_absolute_percentage_error: 98.1610 - val_m
ean_squared_error: 2642.1807 - lr: 0.0010
Epoch 4/900
```

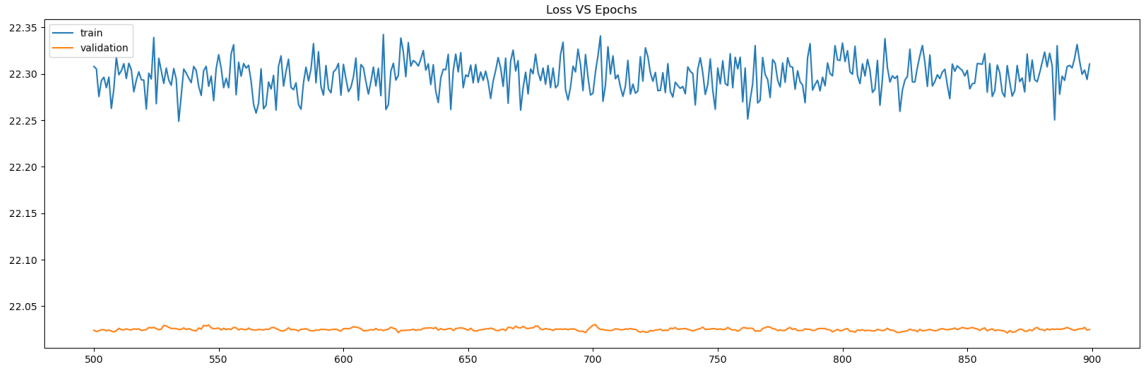
In [120]:

```
epochs = history.epoch
loss = history.history["loss"]
val_loss = history.history["val_loss"]
plt.plot(epochs, loss, label="train")
plt.plot(epochs, val_loss, label="validation")
plt.legend()
plt.title("Loss VS Epochs")
plt.show()
```



In [121]:

```
epochs = history.epoch[500:]
loss = history.history["loss"][500:]
val_loss = history.history["val_loss"][500:]
plt.plot(epochs, loss, label="train")
plt.plot(epochs, val_loss, label="validation")
plt.legend()
plt.title("Loss VS Epochs")
plt.show()
```



In [122]:

```
model.evaluate(X_train_val, y_train_val)
```

```
511/511 [==============================] - 4s 8ms/step - loss: 22.0249 - m
ean_absolute_error: 11.1394 - mean_absolute_percentage_error: 21.9483 - me
an_squared_error: 299.5685
```

Out[122]:

```
[22.024900436401367, 11.13938045501709, 21.948274612426758, 299.5685119628
906]
```

In [123]:

```
model.evaluate(X_train, y_train)
```

```
4599/4599 [==============================] - 36s 8ms/step - loss: 21.5963
- mean_absolute_error: 10.8487 - mean_absolute_percentage_error: 21.5198 -
mean_squared_error: 489.0732
```

Out[123]:

```
[21.5963134765625, 10.848658561706543, 21.5197811126709, 489.073211669921
9]
```

In [124]:

```
model.evaluate(X_test, y_test)
```

```
568/568 [==============================] - 5s 8ms/step - loss: 22.0226 - m
ean_absolute_error: 11.4178 - mean_absolute_percentage_error: 21.9460 - me
an_squared_error: 2468.3508
```

Out[124]:

```
[22.022571563720703, 11.417793273925781, 21.945960998535156, 2468.35083007
8125]
```

In [ ]:

```
```

In [ ]:

```
```