In [52]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

In [53]:
```
df = pd.read_csv('/content/sample_data/dataset.csv')
df.head()
```

Out[53]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_ca
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	ar
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	m
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 [54]:
```
df.head(20)
```

Out[54]:

	market_id	created_at	actual_delivery_time	store_id	store_primar
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	
5	3.0	2015-01-28 20:30:38	2015-01-28 21:08:58	f0ade77b43923b38237db569b016ba25	
6	3.0	2015-01-31 02:16:36	2015-01-31 02:43:00	f0ade77b43923b38237db569b016ba25	
7	3.0	2015-02-12 03:03:35	2015-02-12 03:36:20	f0ade77b43923b38237db569b016ba25	
8	2.0	2015-02-16 00:11:35	2015-02-16 00:38:01	f0ade77b43923b38237db569b016ba25	
9	3.0	2015-02-18 01:15:45	2015-02-18 02:08:57	f0ade77b43923b38237db569b016ba25	
10	3.0	2015-02-02 19:22:53	2015-02-02 20:09:19	f0ade77b43923b38237db569b016ba25	
11	3.0	2015-02-16 04:19:33	2015-02-16 06:34:00	f0ade77b43923b38237db569b016ba25	
12	3.0	2015-02-07 01:34:31	2015-02-07 02:17:14	f0ade77b43923b38237db569b016ba25	
13	3.0	2015-01-25 01:50:51	2015-01-25 02:28:53	f0ade77b43923b38237db569b016ba25	
14	1.0	2015-02-12 03:36:46	2015-02-12 04:14:39	ef1e491a766ce3127556063d49bc2f98	
15	1.0	2015-01-27 02:12:36	2015-01-27 03:02:24	ef1e491a766ce3127556063d49bc2f98	

	market_id	created_at	actual_delivery_time	store_id	store_prima
16	1.0	2015-02-06 00:42:42	2015-02-06 02:10:29	ef1e491a766ce3127556063d49bc2f98	
17	1.0	2015-02-08 02:04:17	2015-02-08 03:27:13	ef1e491a766ce3127556063d49bc2f98	
18	1.0	2015-01-31 04:35:54	2015-01-31 05:47:30	ef1e491a766ce3127556063d49bc2f98	
19	1.0	2015-01-31 02:21:23	2015-01-31 03:11:42	ce016f59ecc2366a43e1c96a4774d167	

In [55]:
```
df.market_id.unique().shape
```

Out[55]:
```
(7,)
```

In [56]:
```
df.store_id.unique().shape
```

Out[56]:
```
(6743,)
```

In [57]:
```
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
```

In [58]:
```
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
```

Removing null value rows

In [59]:
```
df.dropna(inplace=True)
```

Calulating Delivery Time based on order created and delivered time stamps

In [60]:
```
df['delivery_time'] = ((pd.to_datetime(df['actual_delivery_time'])-pd.to_datetime

df.head(10)
```

Out[60]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_c
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	a
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	
8	2.0	2015-02-16 00:11:35	2015-02-16 00:38:01	f0ade77b43923b38237db569b016ba25	
14	1.0	2015-02-12 03:36:46	2015-02-12 04:14:39	ef1e491a766ce3127556063d49bc2f98	
15	1.0	2015-01-27 02:12:36	2015-01-27 03:02:24	ef1e491a766ce3127556063d49bc2f98	
16	1.0	2015-02-06 00:42:42	2015-02-06 02:10:29	ef1e491a766ce3127556063d49bc2f98	
17	1.0	2015-02-08 02:04:17	2015-02-08 03:27:13	ef1e491a766ce3127556063d49bc2f98	
18	1.0	2015-01-31 04:35:54	2015-01-31 05:47:30	ef1e491a766ce3127556063d49bc2f98	
19	1.0	2015-01-31 02:21:23	2015-01-31 03:11:42	ce016f59ecc2366a43e1c96a4774d167	
20	1.0	2015-01-31 23:45:12	2015-02-01 00:14:05	ce016f59ecc2366a43e1c96a4774d167	

Extracting hour and dayofweek from order created timestamp

In [61]:
```
import datetime as dt
df['hour'] = pd.to_datetime(df['created_at']).dt.hour
df['dayofweek'] = pd.to_datetime(df['created_at']).dt.dayofweek
```

In [62]:
```
df.head()
```

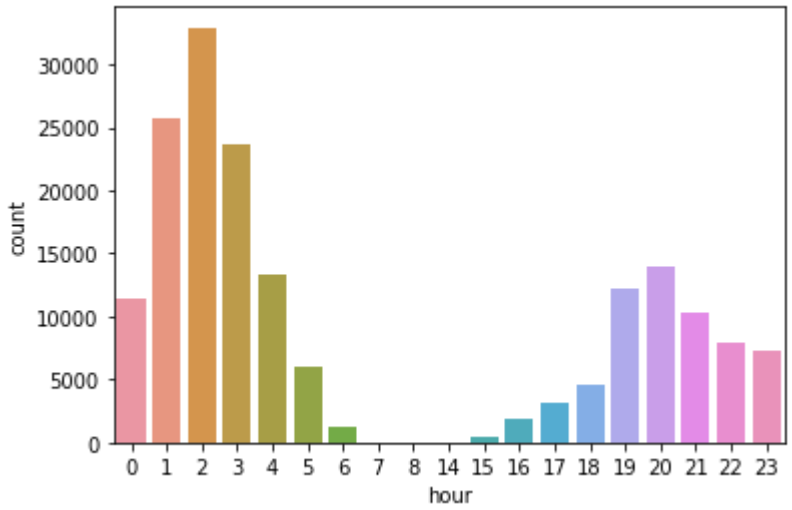
Out[62]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_c
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	a
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	
8	2.0	2015-02-16 00:11:35	2015-02-16 00:38:01	f0ade77b43923b38237db569b016ba25	
14	1.0	2015-02-12 03:36:46	2015-02-12 04:14:39	ef1e491a766ce3127556063d49bc2f98	
15	1.0	2015-01-27 02:12:36	2015-01-27 03:02:24	ef1e491a766ce3127556063d49bc2f98	

In [63]:
```
sns.countplot('hour',data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[63]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdbc7933e50>
```

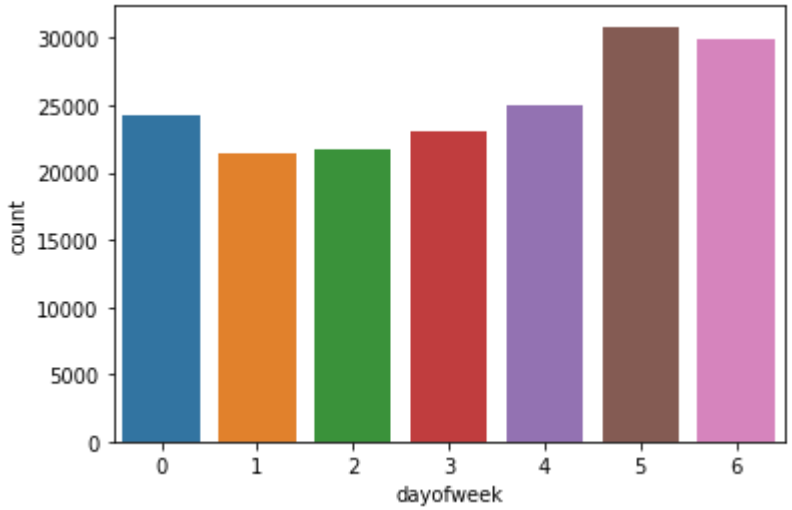


Most of the orders are placed in midnight that to after 12AM to 4Am and no orders are placed in between 7AM and 3PM

In [64]:
```
sns.countplot(df['dayofweek'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[64]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdbc7874250>
```



Saturdays and sundays have the most number of orders

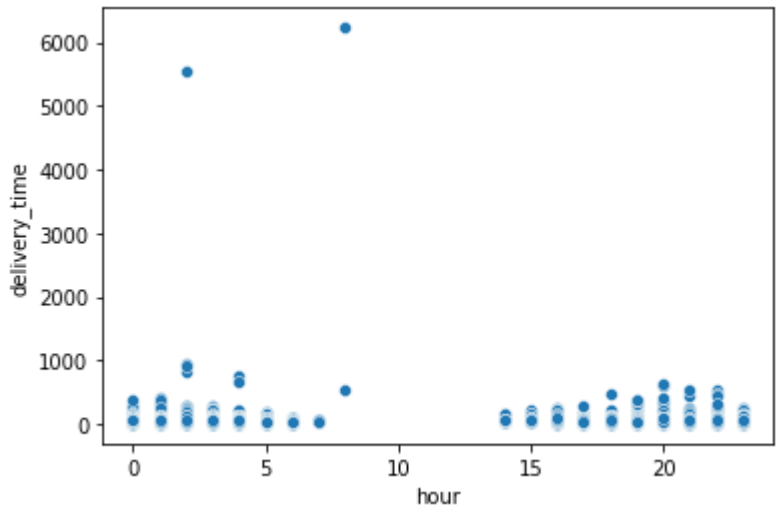
In [65]:
```
'Given data is from Data is from '+ str(pd.to_datetime(df['created_at']).dt.date.
```

Out[65]:
```
'Given data is from Data is from 2015-01-21 to 2015-02-18'
```

Given data is from Data is from 2015-01-21 to 2015-02-18

In [66]:
```
sns.scatterplot(y='delivery_time', x='hour',data=df)
```

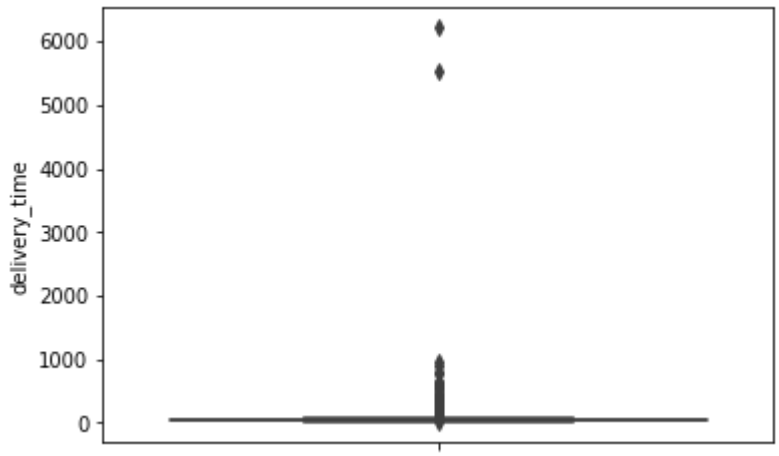
Out[66]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdbc77f9490>
```



Seems like there are lot of outliers in delivery_time

In [67]:
```
sns.boxplot(y=df['delivery_time'])
```

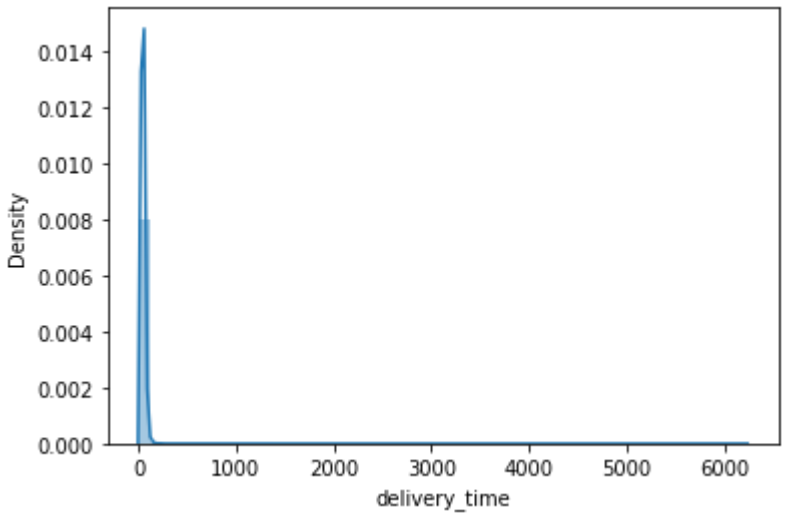
Out[67]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdbc7772f90>
```



In [68]:
```
sns.distplot(df['delivery_time'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
ning: `distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level function wit
h similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[68]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdbc7753d90>
```



Checking the data having deliverytime greater than 1000

In [69]:
```
df[df['delivery_time']>1000]
```

Out[69]:

	market_id	created_at	actual_delivery_time	store_id	store_primar
27189	1.0	2015-02-16 02:24:09	2015-02-19 22:45:31	d397c2b2be2178fe6247bd50fc97cff2	
185550	4.0	2015-01-28 08:34:06	2015-02-01 16:25:25	1679091c5a880faf6fb5e6087eb1b2dc	

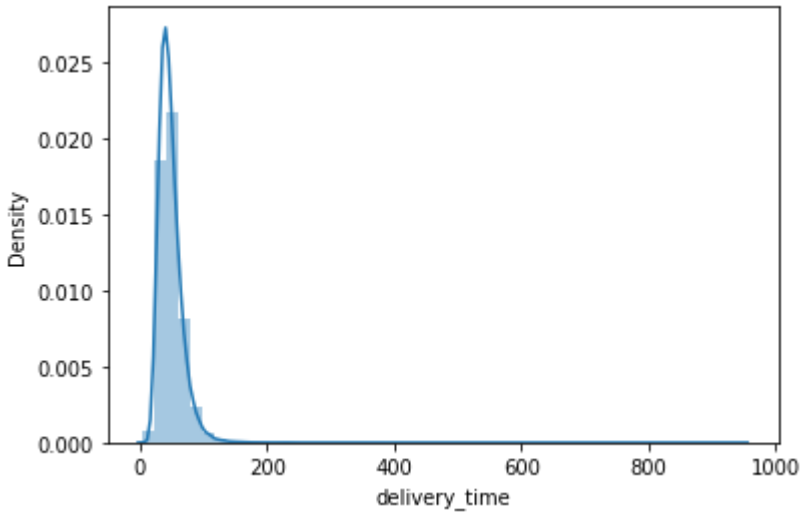
Filtering that data

In [70]:
```
df = df[df['delivery_time']<1000]
```

In [71]:
```
sns.distplot(df['delivery_time'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWar
ning: `distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level function wit
h similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[71]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdbc75dfb90>
```



still there are much outliers and hence checking the 1% outbound data

In [72]:
```
upper1 = np.percentile(df['delivery_time'],99,interpolation='midpoint')
lower1 = np.percentile(df['delivery_time'],1,interpolation='midpoint')
print(upper1,lower1)
```

```
108.0 19.0
```

In [73]:
```
df[df['delivery_time'] > upper1]
```

Out[73]:

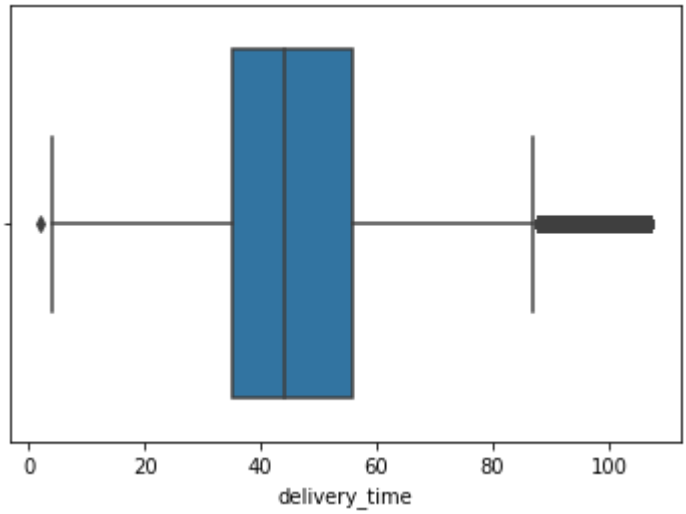
	market_id	created_at	actual_delivery_time	store_id	store_prima
458	1.0	2015-02-15 03:21:08	2015-02-15 05:14:14	a714ec6796f638ba4d5792f78dccd134	
459	1.0	2015-02-02 03:21:25	2015-02-02 05:32:27	a714ec6796f638ba4d5792f78dccd134	
641	6.0	2015-02-09 03:23:20	2015-02-09 05:14:22	9d7311ba459f9e45ed746755a32dcd11	
701	4.0	2015-02-02 01:47:52	2015-02-02 03:53:12	3f900db2608fb3eecb3ee77ba9ef5f60	
793	1.0	2015-02-05 03:35:13	2015-02-05 05:37:31	c56a4706337730e0e15da875405fa1c5	
...	...	...	...	...	
196932	1.0	2015-02-05 02:11:39	2015-02-05 04:04:25	1a21d8c9bbb99bca627434dbf4b98d01	
196949	1.0	2015-02-02 02:13:04	2015-02-02 04:31:26	1a21d8c9bbb99bca627434dbf4b98d01	
197045	4.0	2015-02-13 21:35:28	2015-02-14 00:03:27	17e62166fc8586dfa4d1bc0e1742c08b	
197353	1.0	2015-02-05 23:40:50	2015-02-06 01:33:00	a914ecef9c12ffdb9bede64bb703d877	
197414	1.0	2015-02-03 02:07:26	2015-02-03 04:22:00	a914ecef9c12ffdb9bede64bb703d877	

1709 rows × 17 columns

In [74]:
```
sns.boxplot(df[df['delivery_time']<upper1]['delivery_time'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[74]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdbc9b7c790>
```



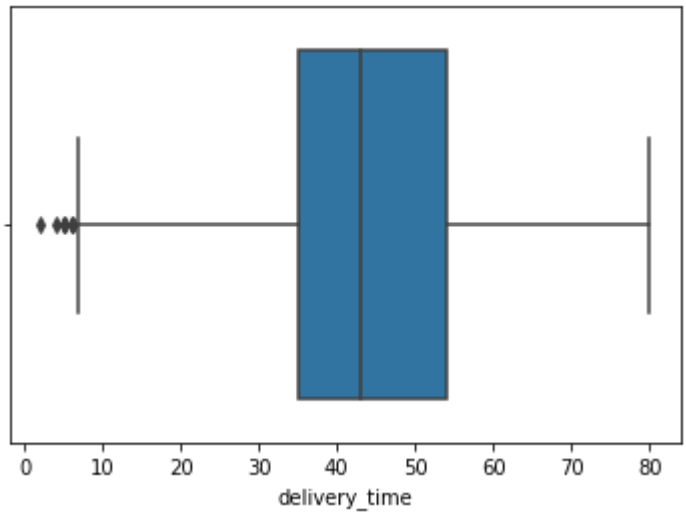
still there's much noice in upper limit, hence going with 5% outliers

In [75]:
```
upper5 = np.percentile(df['delivery_time'],95,interpolation='midpoint')
lower5 = np.percentile(df['delivery_time'],5,interpolation='midpoint')
print(upper5,lower5)
print(df[df['delivery_time'] > upper5].shape)
sns.boxplot(df[df['delivery_time']<upper5]['delivery_time'])
```

```
81.0 25.0
(8575, 17)

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[75]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdbc74ee110>
```

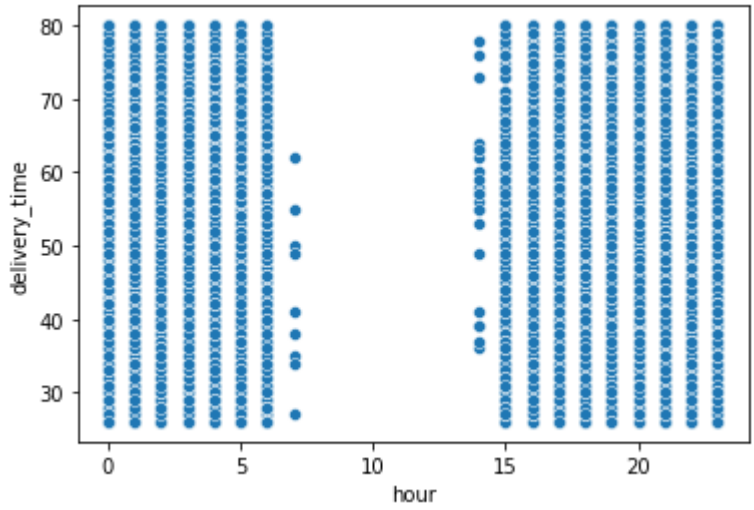


In [76]:
```
df = df[df['delivery_time']<81]
df = df[df['delivery_time']>25]
df.shape
```

Out[76]:
```
(156981, 17)
```

In [100]:
```
sns.scatterplot(y='delivery_time', x='hour',data=df)
```

Out[100]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdbcb599190>
```



Encoding Store Primary Category :

In [77]:
```
df['store_primary_category'].value_counts()
```

Out[77]:
```
american              16404
pizza                 14335
mexican               13710
burger                 8980
sandwich               7945
                      ...
russian                  10
lebanese                  8
belgian                   2
chocolate                 1
alcohol-plus-food         1
Name: store_primary_category, Length: 73, dtype: int64
```

In [78]:
```
pip install category_encoders
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) http
s://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pkg.dev/co
lab-wheels/public/simple/)
Requirement already satisfied: category_encoders in /usr/local/lib/python3.7/di
st-packages (2.5.1.post0)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.7/dist-p
ackages (from category_encoders) (1.3.5)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.7/d
ist-packages (from category_encoders) (0.12.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-pa
ckages (from category_encoders) (1.7.3)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.7/dist-p
ackages (from category_encoders) (1.21.6)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.
7/dist-packages (from category_encoders) (1.0.2)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.7/dist-pa
ckages (from category_encoders) (0.5.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-pa
ckages (from pandas>=1.0.5->category_encoders) (2022.4)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python
3.7/dist-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (f
rom patsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-pa
ckages (from scikit-learn>=0.20.0->category_encoders) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.
7/dist-packages (from scikit-learn>=0.20.0->category_encoders) (3.1.0)
```

In [79]:
```
from category_encoders import TargetEncoder
df = TargetEncoder(cols=['store_primary_category'],smoothing=1.0).fit(df,df['deli
```

```
/usr/local/lib/python3.7/dist-packages/category_encoders/target_encoder.py:124:
FutureWarning: Default parameter min_samples_leaf will change in version 2.6.Se
e https://github.com/scikit-learn-contrib/category_encoders/issues/327 (http
s://github.com/scikit-learn-contrib/category_encoders/issues/327)
  category=FutureWarning)
/usr/local/lib/python3.7/dist-packages/category_encoders/target_encoder.py:129:
FutureWarning: Default parameter smoothing will change in version 2.6.See http
s://github.com/scikit-learn-contrib/category_encoders/issues/327 (https://githu
b.com/scikit-learn-contrib/category_encoders/issues/327)
  category=FutureWarning)
```

In [80]:
```
df.head()
```

Out[80]:

	market_id	created_at	actual_delivery_time	store_id	store_primary_c
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	46
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	44
8	2.0	2015-02-16 00:11:35	2015-02-16 00:38:01	f0ade77b43923b38237db569b016ba25	47
14	1.0	2015-02-12 03:36:46	2015-02-12 04:14:39	ef1e491a766ce3127556063d49bc2f98	48
15	1.0	2015-01-27 02:12:36	2015-01-27 03:02:24	ef1e491a766ce3127556063d49bc2f98	48

In [81]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 156981 entries, 0 to 197427
Data columns (total 17 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   market_id                156981 non-null  float64
 1   created_at               156981 non-null  object 
 2   actual_delivery_time     156981 non-null  object 
 3   store_id                 156981 non-null  object 
 4   store_primary_category   156981 non-null  float64
 5   order_protocol           156981 non-null  float64
 6   total_items              156981 non-null  int64  
 7   subtotal                 156981 non-null  int64  
 8   num_distinct_items       156981 non-null  int64  
 9   min_item_price           156981 non-null  int64  
 10  max_item_price           156981 non-null  int64  
 11  total_onshift_partners   156981 non-null  float64
 12  total_busy_partners      156981 non-null  float64
 13  total_outstanding_orders 156981 non-null  float64
 14  delivery_time            156981 non-null  float64
 15  hour                     156981 non-null  int64  
 16  dayofweek                156981 non-null  int64  
dtypes: float64(7), int64(7), object(3)
memory usage: 25.6+ MB
```

Removing the columns of created_at,actual_delivery_time,store_id

In [82]:
```
df.drop(columns=['created_at','actual_delivery_time','store_id'],axis=1,inplace=T
```

In [83]:
```
df.columns
```

Out[83]:
```
Index(['market_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', 'delivery_time', 'hour', 'dayofweek'],
      dtype='object')
```

In [84]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 156981 entries, 0 to 197427
Data columns (total 14 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   market_id                 156981 non-null  float64
 1   store_primary_category    156981 non-null  float64
 2   order_protocol            156981 non-null  float64
 3   total_items               156981 non-null  int64  
 4   subtotal                  156981 non-null  int64  
 5   num_distinct_items        156981 non-null  int64  
 6   min_item_price            156981 non-null  int64  
 7   max_item_price            156981 non-null  int64  
 8   total_onshift_partners    156981 non-null  float64
 9   total_busy_partners       156981 non-null  float64
 10  total_outstanding_orders  156981 non-null  float64
 11  delivery_time             156981 non-null  float64
 12  hour                      156981 non-null  int64  
 13  dayofweek                 156981 non-null  int64  
dtypes: float64(7), int64(7)
memory usage: 22.0 MB
```

In [85]:
```
X = df.drop(columns='delivery_time')
y = df['delivery_time']
```

In [85]:
```

```

In [86]:
```
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_s
```

In [87]:
```
from sklearn.preprocessing import MinMaxScaler
st_x= MinMaxScaler()
X_train= st_x.fit_transform(X_train)
X_test= st_x.fit_transform(X_test)
```

In [88]:
```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(min_samples_split = 4, min_samples_leaf =2, max_dep
```

In [89]:
```
model.fit(X_train,y_train)
```

Out[89]:
```
RandomForestRegressor(max_depth=9, min_samples_leaf=2, min_samples_split=4)
```

In [90]:
```
y_pred = model.predict(X_test)
```

In [91]:
```
ys = {'actual' : y_test, 'predicted' : y_pred}
ys = pd.DataFrame(ys)
```

In [92]:
```
ys.head()
```

Out[92]:

	actual	predicted
111128	34.0	45.867276
69171	57.0	49.419719
187149	52.0	49.183729
33708	67.0	50.702173
129224	66.0	53.459807

In [93]:
```
from sklearn.metrics import mean_absolute_percentage_error, mean_squared_error
```

In [94]:
```
mape = mean_absolute_percentage_error(y_test,y_pred)
rmse = mean_squared_error(y_test,y_pred,squared=False)
mse = mean_squared_error(y_test,y_pred,squared=True)
print("mape : {0} , rmse :{1}, mse:{2}".format(mape,rmse,mse))
```

```
mape : 0.22001842942078237 , rmse :11.599389654357486, mse:134.5458403536155
```

Bulding a MLP

In [95]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from keras.layers import Activation,Dense
from tensorflow.keras.optimizers import Adam
model = Sequential()
```

In [96]:
```
model.add(Dense(16,activation='relu'))
model.add(Dense(256,activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dense(128,activation='relu'))
model.add(Dense(1,activation='linear'))
```

In [97]:
```
adam = Adam(learning_rate = 0.01)
model.compile(optimizer = adam, loss='mse', metrics = ['mse','mape'])
model.fit(X_train,y_train, batch_size=256, epochs = 20, validation_split = 0.2, v
```

```
Epoch 1/20
393/393 [==============================] - 7s 16ms/step - loss: 183.2137 - mse:
183.2137 - mape: 23.9974 - val_loss: 140.9263 - val_mse: 140.9263 - val_mape: 2
2.4107
Epoch 2/20
393/393 [==============================] - 10s 26ms/step - loss: 141.2934 - ms
e: 141.2934 - mape: 21.9287 - val_loss: 139.7464 - val_mse: 139.7464 - val_map
e: 21.8020
Epoch 3/20
393/393 [==============================] - 10s 26ms/step - loss: 139.0497 - ms
e: 139.0497 - mape: 21.7542 - val_loss: 141.7390 - val_mse: 141.7390 - val_map
e: 20.6591
Epoch 4/20
393/393 [==============================] - 10s 25ms/step - loss: 139.0758 - ms
e: 139.0758 - mape: 21.7158 - val_loss: 139.9463 - val_mse: 139.9463 - val_map
e: 22.9607
Epoch 5/20
393/393 [==============================] - 7s 17ms/step - loss: 137.5439 - mse:
137.5439 - mape: 21.6187 - val_loss: 137.6593 - val_mse: 137.6593 - val_mape: 2
0.9120
Epoch 6/20
393/393 [==============================] - 5s 14ms/step - loss: 136.2917 - mse:
136.2917 - mape: 21.5062 - val_loss: 137.4092 - val_mse: 137.4092 - val_mape: 2
0.9211
Epoch 7/20
393/393 [==============================] - 5s 14ms/step - loss: 137.1255 - mse:
137.1255 - mape: 21.5546 - val_loss: 136.2154 - val_mse: 136.2154 - val_mape: 2
1.6386
Epoch 8/20
393/393 [==============================] - 5s 14ms/step - loss: 135.7694 - mse:
135.7694 - mape: 21.4693 - val_loss: 142.4268 - val_mse: 142.4268 - val_mape: 2
0.4226
Epoch 9/20
393/393 [==============================] - 5s 14ms/step - loss: 136.2975 - mse:
136.2975 - mape: 21.4979 - val_loss: 138.4488 - val_mse: 138.4488 - val_mape: 2
1.0412
Epoch 10/20
393/393 [==============================] - 6s 14ms/step - loss: 135.4478 - mse:
135.4478 - mape: 21.4573 - val_loss: 135.9445 - val_mse: 135.9445 - val_mape: 2
1.2705
Epoch 11/20
393/393 [==============================] - 6s 14ms/step - loss: 135.2719 - mse:
135.2719 - mape: 21.4293 - val_loss: 146.6978 - val_mse: 146.6978 - val_mape: 2
4.3711
Epoch 12/20
393/393 [==============================] - 5s 14ms/step - loss: 134.4849 - mse:
134.4849 - mape: 21.3846 - val_loss: 135.2114 - val_mse: 135.2114 - val_mape: 2
1.8670
Epoch 13/20
393/393 [==============================] - 6s 14ms/step - loss: 134.5028 - mse:
134.5028 - mape: 21.3612 - val_loss: 143.2202 - val_mse: 143.2202 - val_mape: 2
3.8086
```

```
Epoch 14/20
393/393 [==============================] - 5s 14ms/step - loss: 134.4831 - mse:
134.4831 - mape: 21.3654 - val_loss: 137.1619 - val_mse: 137.1619 - val_mape: 2
2.6580
Epoch 15/20
393/393 [==============================] - 5s 14ms/step - loss: 134.1044 - mse:
134.1044 - mape: 21.3305 - val_loss: 140.1966 - val_mse: 140.1966 - val_mape: 2
3.3397
Epoch 16/20
393/393 [==============================] - 5s 13ms/step - loss: 134.3824 - mse:
134.3824 - mape: 21.3460 - val_loss: 140.6955 - val_mse: 140.6955 - val_mape: 2
3.5714
Epoch 17/20
393/393 [==============================] - 5s 13ms/step - loss: 134.8595 - mse:
134.8595 - mape: 21.3835 - val_loss: 135.0225 - val_mse: 135.0225 - val_mape: 2
1.7334
Epoch 18/20
393/393 [==============================] - 5s 14ms/step - loss: 134.0041 - mse:
134.0041 - mape: 21.3251 - val_loss: 135.4610 - val_mse: 135.4610 - val_mape: 2
1.6374
Epoch 19/20
393/393 [==============================] - 5s 13ms/step - loss: 133.6393 - mse:
133.6393 - mape: 21.3003 - val_loss: 137.9784 - val_mse: 137.9784 - val_mape: 2
2.7065
Epoch 20/20
393/393 [==============================] - 7s 18ms/step - loss: 134.0277 - mse:
134.0277 - mape: 21.3208 - val_loss: 136.6301 - val_mse: 136.6301 - val_mape: 2
2.3747
```

Out[97]:
```
<keras.callbacks.History at 0x7fdbcc0df810>
```

In [98]:
```
y_pred = model.predict(X_test)
```

In [99]:
```
mape = mean_absolute_percentage_error(y_test,y_pred)
rmse = mean_squared_error(y_test,y_pred,squared=False)
mse = mean_squared_error(y_test,y_pred,squared=True)
print("mape : {0} , rmse :{1}, mse:{2}".format(mape,rmse,mse))
```

```
mape : 0.22238478995437072 , rmse :11.593722234561115, mse:134.4143952521568
```

We can see there is not much difference between errors if we use RandomForest and MLPs for regression and tabular data.