Importing libraries and the dataset

In [457]:
```
# importing libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score, KFold
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
import seaborn as sns
import matplotlib.pyplot as plt
from pytz import timezone
from datetime import datetime, date, timedelta
from scipy import stats

sns.set(color_codes=True)
```

In [458]:
```
df=pd.read_excel('Analytics Sample Data.xlsx')
df.head()
```

Out[458]:

	Customer placed order datetime	Placed order with restaurant datetime	Driver at restaurant datetime	Delivered to consumer datetime	Driver ID	Restaurant ID	Consumer ID	Is New	Del Re
0	14 20:27:45	14 20:29:41	14 20:39:32	14 20:52:03	86	12	5	False	Palo Alto
1	07 20:16:28	07 20:17:32	07 20:36:00	07 20:49:02	325	66	5	False	Palo Alto
2	13 19:35:09	13 19:39:26	13 20:28:16	13 20:52:44	200	124	5	False	Palo Alto
3	22 19:47:53	22 19:56:08	22 20:01:20	22 20:18:01	154	5	14	False	Palo Alto
4	03 19:01:52	03 19:09:08	03 19:36:20	03 19:45:26	332	9	14	False	Palo Alto

In [459]:
```
df.columns = ['cust_place_time', 'rest_place_time', 'driver_rest_time',
              'cust_deliver_time', 'driver_id', 'rest_id', 'cust_id', 'is_new',
              'delivery_region', 'is_asap', 'order_tot', 'amt_discount', 'amt_tip', 'refund_amt']
```

Exploring the data

In [460]:
```
#checking data types
df.dtypes
```

Out[460]:
```
cust_place_time       object
rest_place_time       object
driver_rest_time      object
cust_deliver_time     object
driver_id              int64
rest_id                int64
cust_id                int64
is_new                  bool
delivery_region       object
is_asap                 bool
order_tot            float64
amt_discount         float64
amt_tip              float64
refund_amt           float64
dtype: object
```

In [461]:
```
# check count of nulls
df.isna().sum()
```

Out[461]:
```
cust_place_time         0
rest_place_time        40
driver_rest_time     4531
cust_deliver_time       0
driver_id               0
rest_id                 0
cust_id                 0
is_new                  0
delivery_region         0
is_asap                 0
order_tot               0
amt_discount            0
amt_tip                 0
refund_amt              0
dtype: int64
```

In [467]:
```
# if restaurant place time is missing, filling it with customer place ti
me, and leaving a missing label.
df['rest_place_time']=df['rest_place_time'].fillna(df['cust_place_time'
])
df.loc[pd.isna(df.driver_rest_time)==True, 'rest_place_time_missing'] =
True
```

In [468]:
```
# if driver arriving at restaurant time is missing, filling it with rest
aurant place time, and leaving a missing label.
df['driver_rest_time']=df['driver_rest_time'].fillna(df['rest_place_tim
e'])
df.loc[pd.isna(df.driver_rest_time)==True, 'driver_rest_time_missing'] =
True
```

In [470]:
```
# assuming this data set came from Jan 2018 temporarily (because it's da
ylight saving). Going to remove year and month later.
def convertTZ(df, column):
    df[column]=pd.to_datetime('2018-01-'+df[column].astype(str))
    df[column+'_pdt'] = pd.DatetimeIndex(df[column]).tz_localize('UTC').
tz_convert('US/Pacific')
    df.reset_index()
    #df[column+'_pdt']=df[column+'_pdt'].dt.time
for i in ['cust_place_time', 'rest_place_time', 'driver_rest_time', 'cus
t_deliver_time']:
    convertTZ(df, i)
```

In [471]:
```
# calculating time differences.
df['customer_delivery_time'] =df.cust_deliver_time_pdt-df.cust_place_tim
e_pdt
df['restaurant_notification']=df.rest_place_time_pdt-df.cust_place_time_
pdt
df['prep_and_pick']=df.driver_rest_time_pdt-df.rest_place_time_pdt
df['driver_delivery_time']=df.cust_deliver_time_pdt-df.driver_rest_time_
pdt
```

In [472]:
```
df.describe()
```

Out[472]:

	driver_id	rest_id	cust_id	order_tot	amt_discount	a
count	18078.000000	18078.000000	18078.000000	18078.000000	18078.000000	18078.0
mean	222.448169	108.721263	32010.313475	51.261496	1.356706	3.49202
std	105.219194	98.609766	42010.215766	50.599675	6.516667	3.70011
min	7.000000	2.000000	5.000000	0.000000	0.000000	0.00000
25%	141.000000	23.000000	4243.500000	26.660000	0.000000	1.40000
50%	227.000000	77.000000	10972.000000	38.630000	0.000000	2.55000
75%	314.000000	186.000000	56857.000000	57.885000	0.000000	4.46000
max	438.000000	409.000000	200449.000000	1604.130000	187.880000	120.300

In [473]:
```
# checking how many have negative value in time differences (because som
e orders are cross months)
df.loc[(df.customer_delivery_time<timedelta())|(df.prep_and_pick<timedel
ta())|(df.driver_delivery_time<timedelta())|(df.restaurant_notification<
timedelta())]['cust_place_time'].count()
```

Out[473]: 166

In [474]:
```
# removing these rows because it's less than 1% of the whole data set
df=df.loc[(df.customer_delivery_time>=timedelta())&(df.prep_and_pick>=ti
medelta())&(df.driver_delivery_time>=timedelta())]
```

In [478]:
```
# check if there are duplicates in the data set
df.drop_duplicates()['driver_id'].count()
# there are no duplicates.
```

Out[478]: 17913

In [475]:
```
df.describe()
```

Out[475]:

	driver_id	rest_id	cust_id	order_tot	amt_discount	a
count	17913.000000	17913.000000	17913.000000	17913.000000	17913.000000	17913.0
mean	222.561045	108.582594	32043.161447	51.150042	1.353117	3.48469
std	105.251022	98.444705	42046.808088	50.427191	6.517951	3.68997
min	7.000000	2.000000	5.000000	0.000000	0.000000	0.00000
25%	141.000000	23.000000	4243.000000	26.660000	0.000000	1.40000
50%	227.000000	77.000000	10977.000000	38.630000	0.000000	2.55000
75%	314.000000	186.000000	56862.000000	57.670000	0.000000	4.45000
max	438.000000	408.000000	200449.000000	1604.130000	187.880000	120.300

In [476]:
```
df.nunique()
```

Out[476]:
```
cust_place_time             17724
rest_place_time             17780
driver_rest_time            17678
cust_deliver_time           17725
driver_id                     293
rest_id                       313
cust_id                      6670
is_new                          2
delivery_region                 4
is_asap                         2
order_tot                    3705
amt_discount                  152
amt_tip                      1421
refund_amt                    366
rest_place_time_missing         1
driver_rest_time_missing        0
cust_place_time_pdt         17724
rest_place_time_pdt         17780
driver_rest_time_pdt        17678
cust_deliver_time_pdt       17725
customer_delivery_time       5604
restaurant_notification      3766
prep_and_pick                3040
driver_delivery_time         3580
dtype: int64
```

In [477]:
```
df.sum()
```

Out[477]:
```
driver_id       3.986736e+06
rest_id         1.945040e+06
cust_id         5.739892e+08
is_new          3.484000e+03
is_asap         1.432700e+04
order_tot       9.162507e+05
amt_discount    2.423838e+04
amt_tip         6.242132e+04
refund_amt      1.086938e+04
dtype: float64
```

In [479]:
```
df.mean()
```

Out[479]:
```
driver_id         222.561045
rest_id           108.582594
cust_id         32043.161447
is_new              0.194496
is_asap             0.799810
order_tot          51.150042
amt_discount        1.353117
amt_tip             3.484694
refund_amt          0.606787
dtype: float64
```

In [480]:
```
df.median()
```

Out[480]:
```
driver_id         227.00
rest_id            77.00
cust_id         10977.00
is_new              0.00
is_asap             1.00
order_tot          38.63
amt_discount        0.00
amt_tip             2.55
refund_amt          0.00
dtype: float64
```

In [481]:
```
# convert time differences to minutes.
df=df.drop(['cust_place_time', 'rest_place_time', 'driver_rest_time', 'c
ust_deliver_time'], axis=1)
df['customer_delivery_time'] =(df.cust_deliver_time_pdt-df.cust_place_ti
me_pdt).dt.total_seconds().div(60, fill_value=0)
df['prep_and_pick']=(df.driver_rest_time_pdt-df.rest_place_time_pdt).dt.
total_seconds().div(60, fill_value=0)
df['driver_delivery_time']=(df.cust_deliver_time_pdt-df.driver_rest_time
_pdt).dt.total_seconds().div(60, fill_value=0)
df['restaurant_notification']=(df.rest_place_time_pdt-df.cust_place_time
_pdt).dt.total_seconds().div(60, fill_value=0)
```

In [482]:
```
# extract more information from the time that the customer placed the or
der.
# please notice that we don't have the real data on which month this dat
a came from,
# we have no idea which weekday they correspond to.
df['weekday']=df['cust_place_time_pdt'].dt.weekday
df['day']=df['cust_place_time_pdt'].dt.day
df['hour']=df['cust_place_time_pdt'].dt.hour
```

In [483]:
```
# convert amount of discount to percentage of discount
df['perc_discount']=(df['amt_discount']/df['order_tot'])
df=df.drop(['amt_discount'], axis=1)
```

In [484]:
```
# convert amount of discount to percentage of discount
df['perc_refund']=(df['refund_amt']/df['order_tot']).replace(np.inf, 0)
df=df.drop(['refund_amt'], axis=1)
```

In [485]:
```
# check the clean data set again
df.head()
categorical_var_list=['is_new', 'delivery_region', 'is_asap']
```

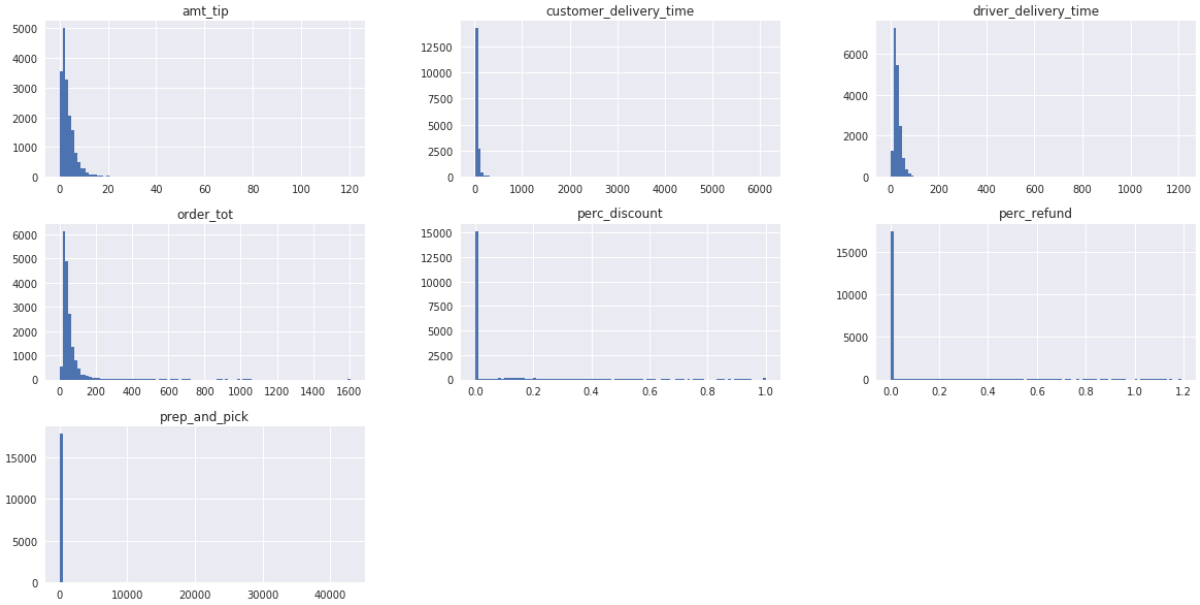
In [486]:
```
# checking the distribution of continuous variables
countinous_var_list=['order_tot', 'amt_tip', 'perc_refund', 'customer_de
livery_time', 'prep_and_pick', 'driver_delivery_time', 'perc_discount']
df[countinous_var_list].describe()
```

Out[486]:

	order_tot	amt_tip	perc_refund	customer_delivery_time	prep_and_pic
count	17913.000000	17913.000000	17912.000000	17913.000000	17913.000000
mean	51.150042	3.484694	0.011110	80.347239	17.620779
std	50.427191	3.689979	0.089223	242.024235	324.993734
min	0.000000	0.000000	0.000000	9.883333	0.000000
25%	26.660000	1.400000	0.000000	37.266667	0.000000
50%	38.630000	2.550000	0.000000	48.266667	10.516667
75%	57.670000	4.450000	0.000000	65.533333	19.316667
max	1604.130000	120.300000	1.190476	6122.450000	42960.166667

In [487]:
```
# display the distribution of continuous variables
%matplotlib inline
df[countinous_var_list].hist(bins=100, figsize=(20, 10))
```

Out[487]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fe746b90a10
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe746b88190
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe746cb5250
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe746c9c2d0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe74a6cf6d0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe746d85950
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe746e7fad0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe746ee3dd0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe74812b410
>]],
      dtype=object)
```



In [488]:
```
# check the distribution of categorical variables
for i in categorical_var_list:
    print i
    display(df[i].value_counts())
```

```
is_new

False    14429
True      3484
Name: is_new, dtype: int64

delivery_region

Palo Alto        11336
Mountain View     3716
San Jose          2835
None                26
Name: delivery_region, dtype: int64

is_asap

True     14327
False     3586
Name: is_asap, dtype: int64
```

In [490]:
```
# dropped 26 rows that don't have delivery region
df=df.loc[df.delivery_region!='None']
```

In [491]:
```
df.loc[df.is_new==True, 'new_dummy'] = 1
df.loc[df.is_new==False, 'new_dummy'] = 0
```

In [492]:
```
df.groupby('delivery_region')['new_dummy'].mean()
```

Out[492]:
```
delivery_region
Mountain View    0.191604
Palo Alto        0.186221
San Jose         0.232099
Name: new_dummy, dtype: float64
```

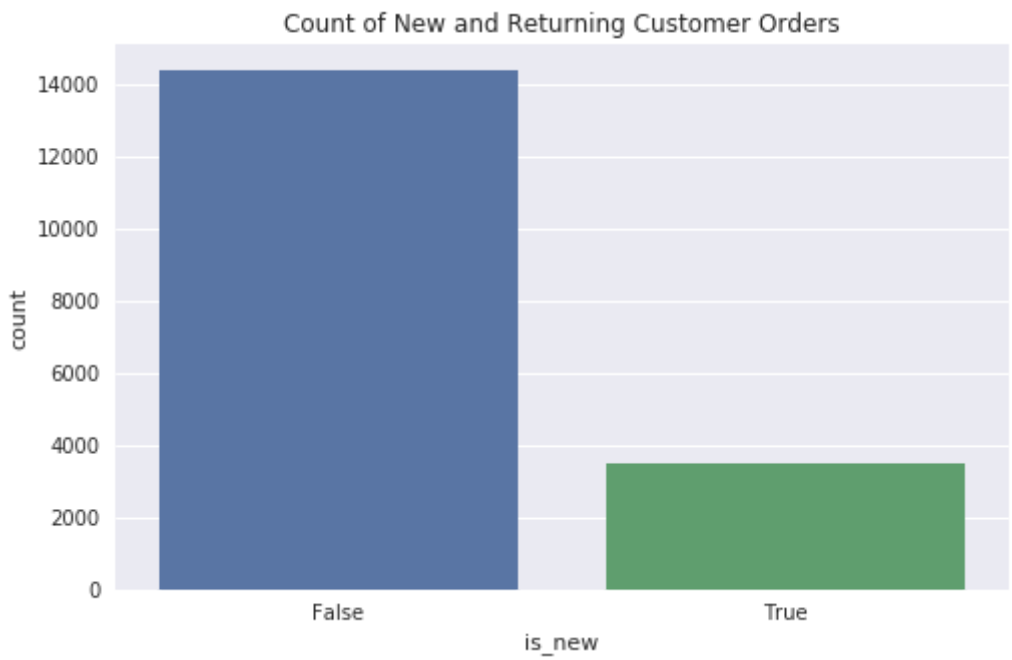
In [493]:
```
df['new_dummy'].mean()
```

Out[493]:
```
0.1946106110583105
```

Plotting the data for insights

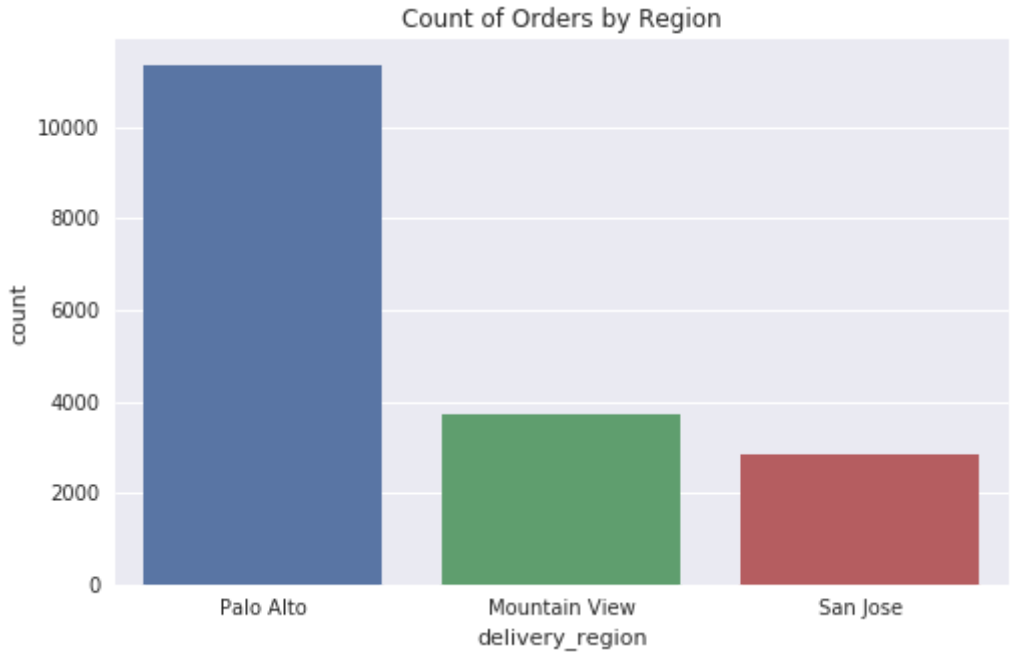
In [494]:
```
fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='is_new', data = df).set_title('Count of New and Returni
ng Customer Orders')
plt.plot()
```

Out[494]: []



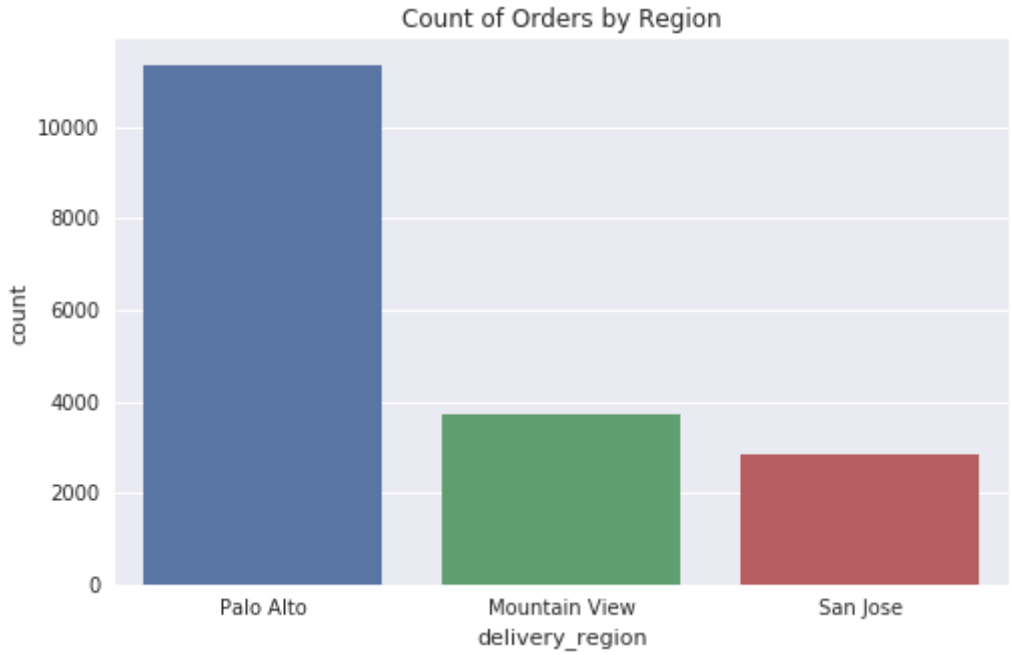
In [495]:
```
fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='delivery_region', data = df).set_title('Count of Orders
by Region')
plt.plot()
```

Out[495]: []

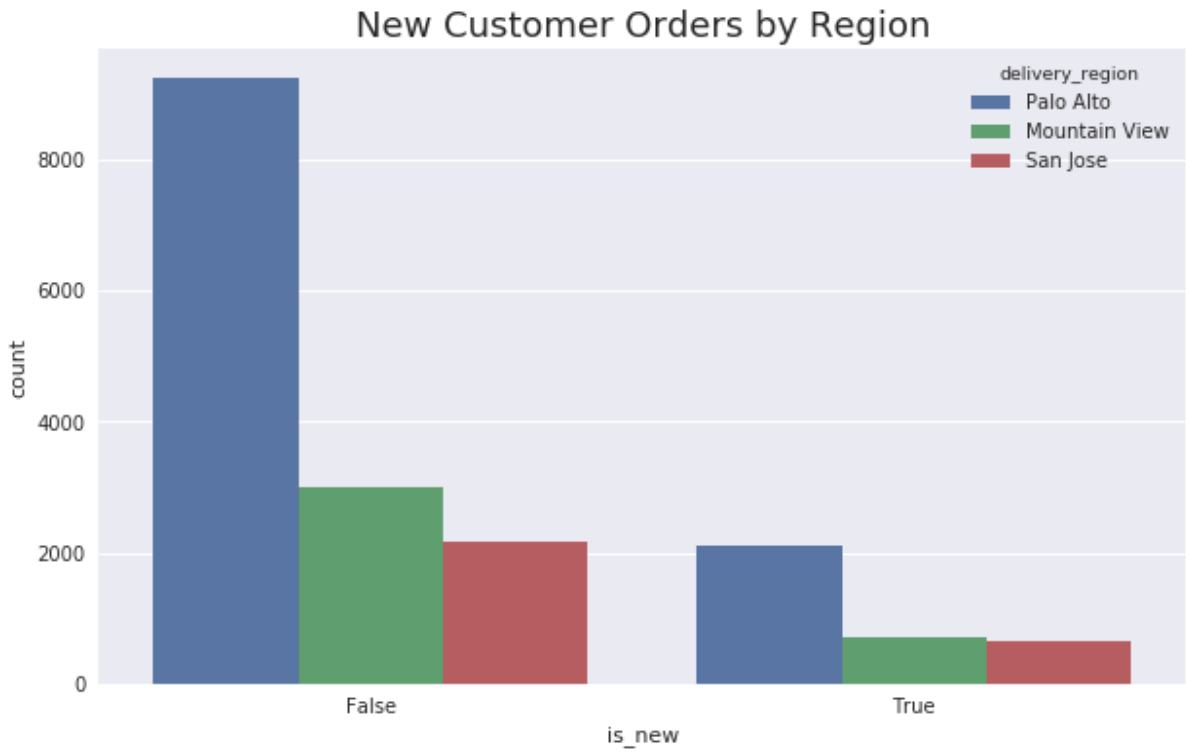


In [496]:
```
fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='delivery_region', data = df).set_title('Count of Orders
by Region')
plt.plot()
```

Out[496]: []

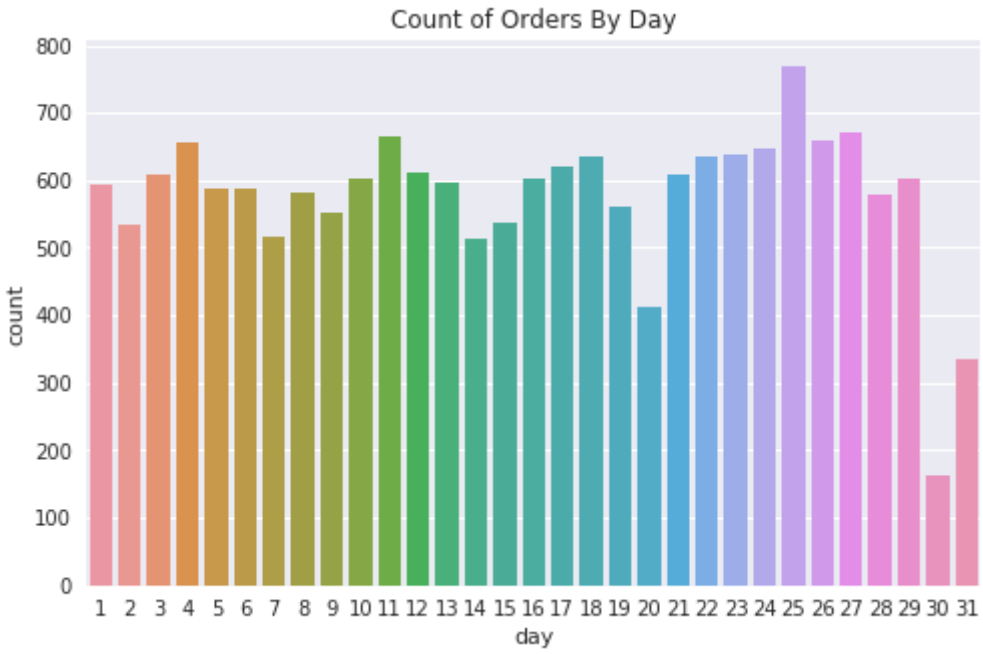


In [497]:
```
fig, ax = plt.subplots(figsize = (10,6))
sns.countplot(x = 'is_new', data = df, hue = 'delivery_region')
plt.title('New Customer Orders by Region', fontsize = 18);
```



In [498]:
```
fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='day', data = df).set_title('Count of Orders By Day')
plt.plot()
```

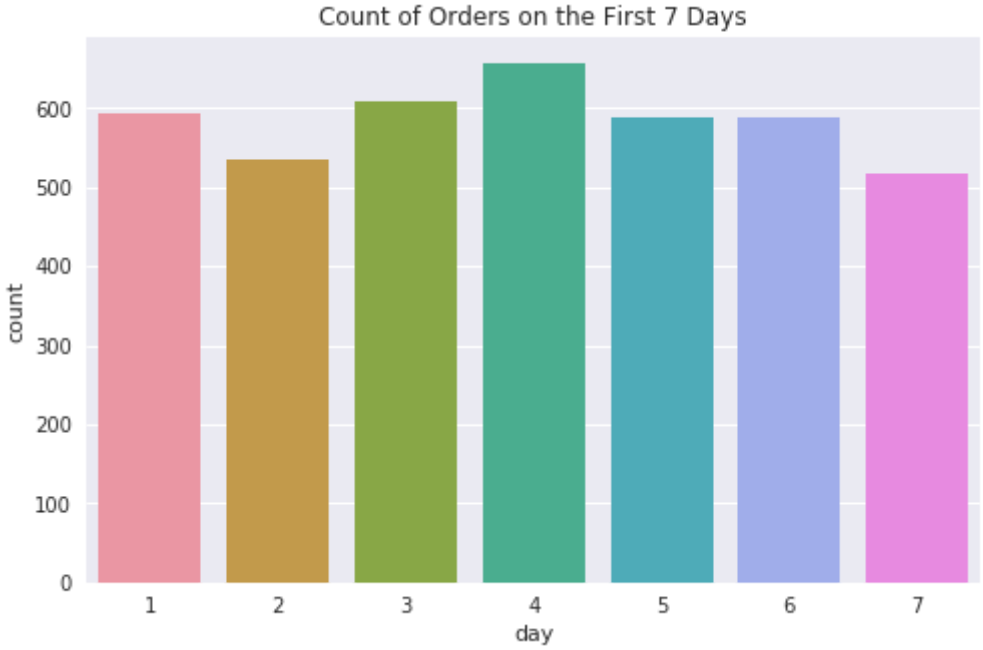
Out[498]: []



In [499]:
```
df_first_7_days=df.loc[df['day']<=7]

fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='day', data = df_first_7_days).set_title('Count of Order
s on the First 7 Days')
plt.plot()
```

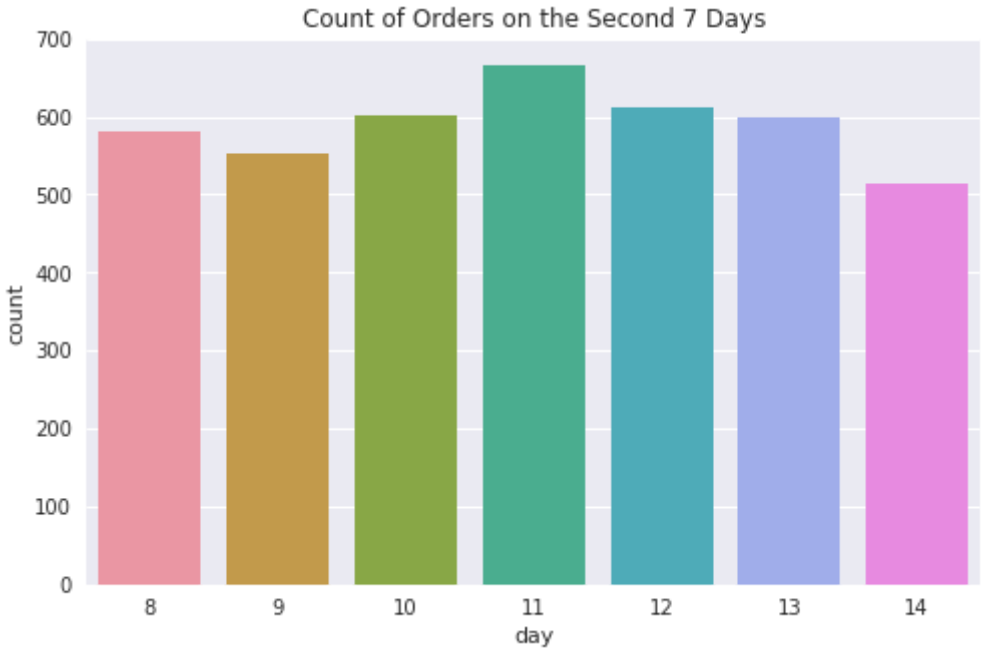
Out[499]: []



In [500]:
```
df_second_7_days=df.loc[ (df['day']>7) & (df['day']<=14)]

fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='day', data = df_second_7_days).set_title('Count of Orde
rs on the Second 7 Days')
plt.plot()
```

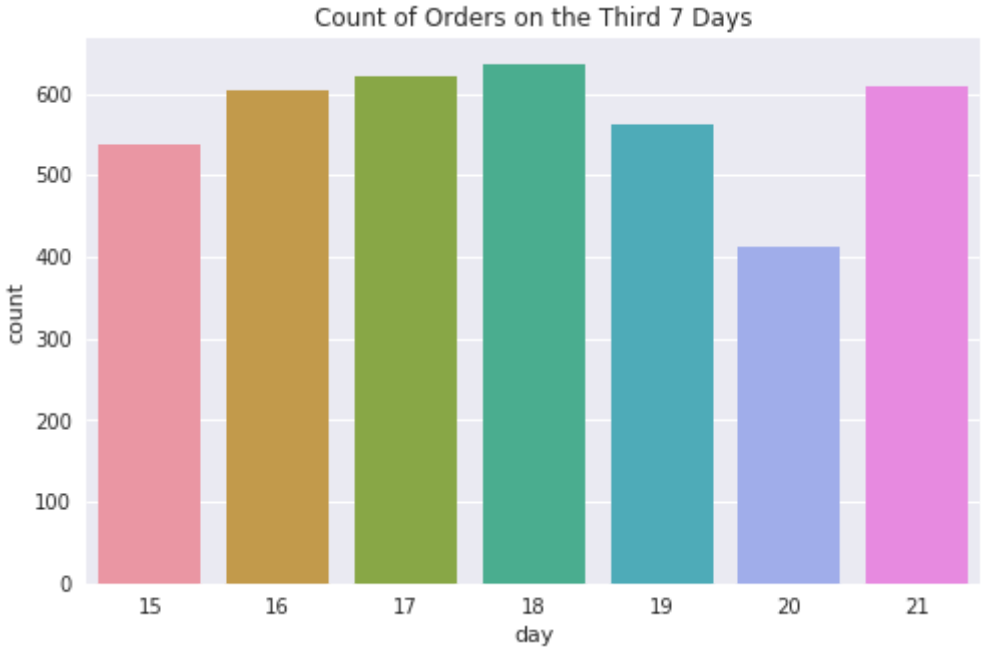
Out[500]: []



In [501]:
```
df_third_7_days=df.loc[ (df['day']>14) & (df['day']<=21)]

fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='day', data = df_third_7_days).set_title('Count of Order
s on the Third 7 Days')
plt.plot()
```

Out[501]: []

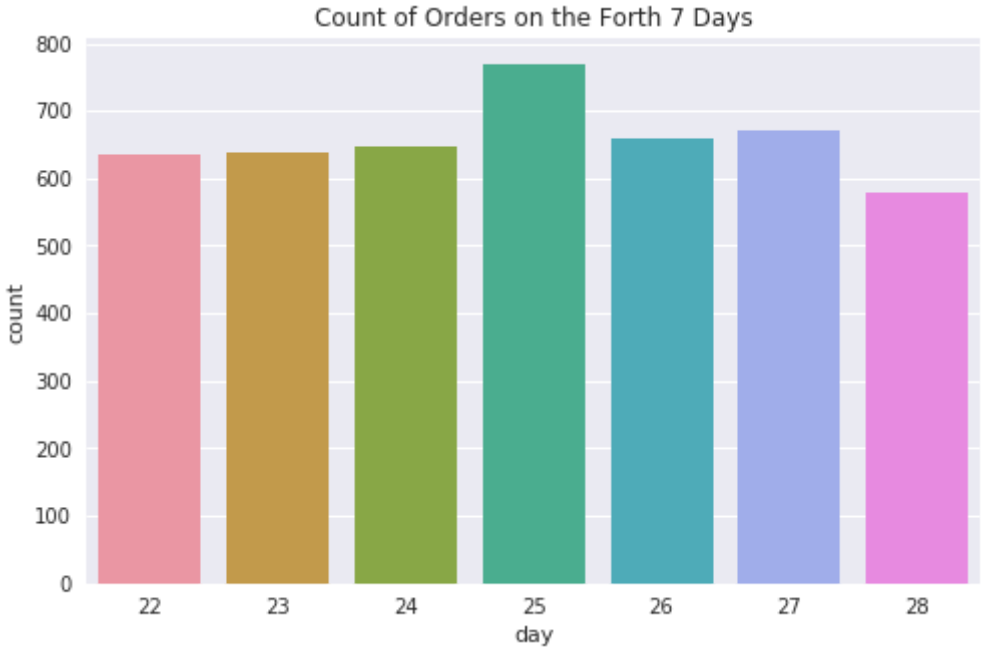


In [502]:
```
df_fourth_7_days=df.loc[ (df['day']>21) & (df['day']<=28)]

fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='day', data = df_fourth_7_days).set_title('Count of Orde
rs on the Forth 7 Days')

plt.plot()
```

Out[502]: []

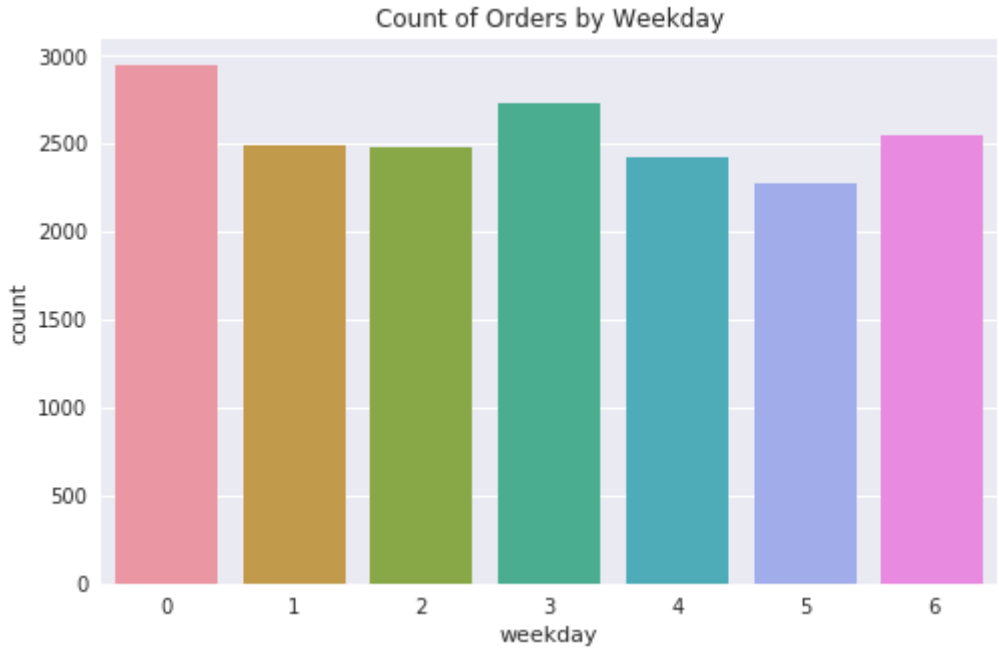


In [503]:
```
print df['weekday'].value_counts().sort_index()

fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='weekday', data = df).set_title('Count of Orders by Week
day')
plt.plot()
```

```
0    2948
1    2492
2    2476
3    2728
4    2420
5    2270
6    2553
Name: weekday, dtype: int64
```

Out[503]: []

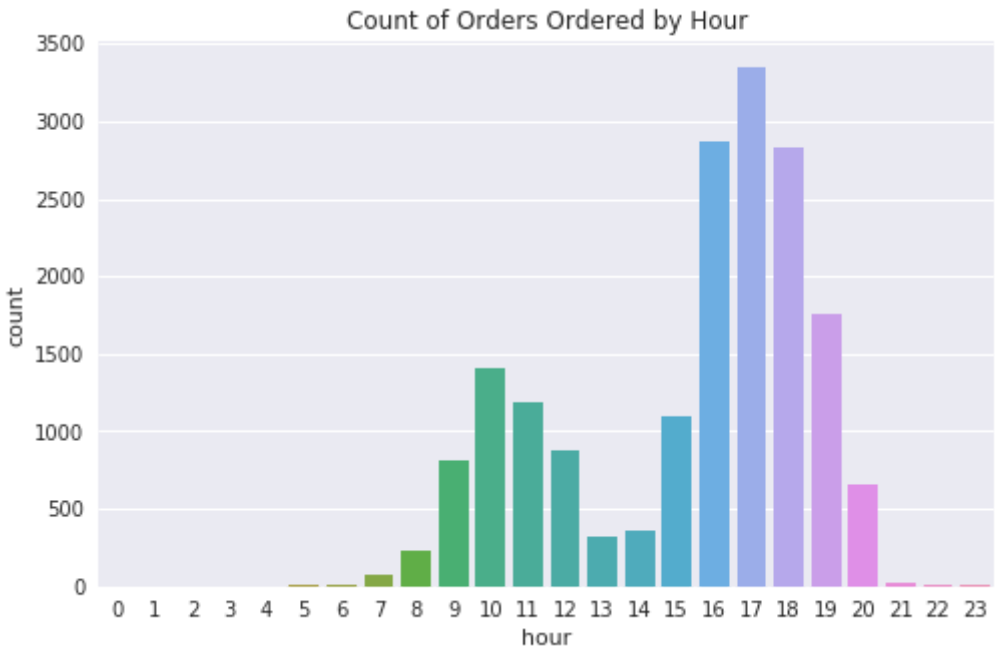


In [504]:
```
print df['hour'].value_counts().sort_index()

fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='hour', data = df).set_title('Count of Orders Ordered by
Hour')
plt.plot()
```

```
0         3
1         3
2         2
3         2
4         2
5         5
6        16
7        72
8       227
9       815
10     1401
11     1181
12      881
13      323
14      354
15     1101
16     2872
17     3348
18     2830
19     1757
20      651
21       21
22       11
23        9
Name: hour, dtype: int64
```

Out[504]: []



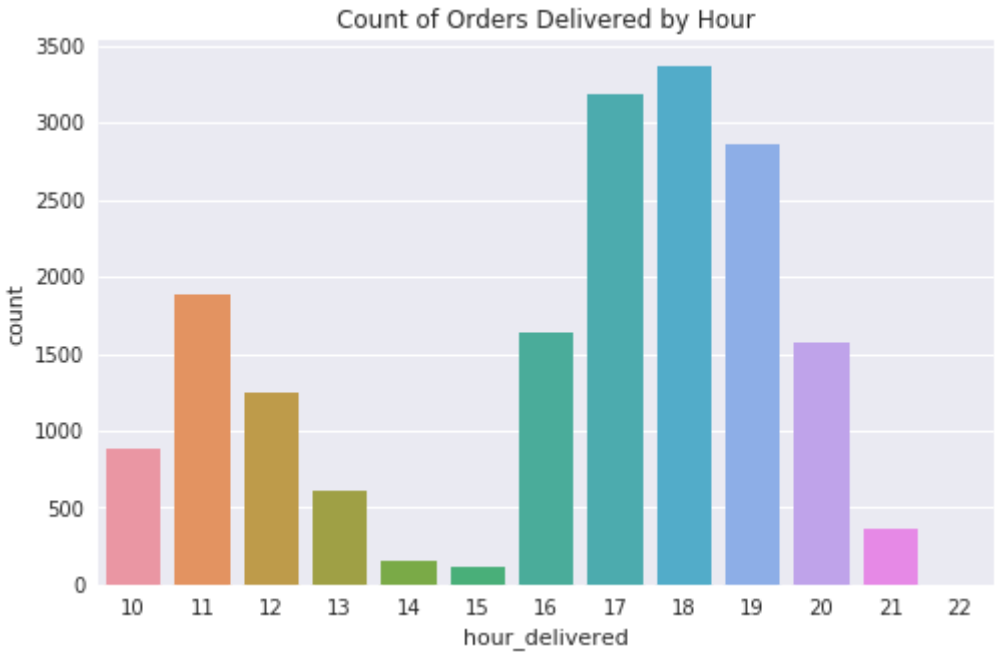
In [505]:
```
df['hour_delivered']=df['cust_deliver_time_pdt'].dt.hour
```

In [506]:
```
print df['hour_delivered'].value_counts().sort_index()

fig, ax = plt.subplots(figsize = (8,5))
sns.countplot(x='hour_delivered', data = df).set_title('Count of Orders
 Delivered by Hour')
plt.plot()
```

```
10     888
11    1884
12    1246
13     608
14     150
15     117
16    1641
17    3189
18    3367
19    2856
20    1576
21     362
22       3
Name: hour_delivered, dtype: int64
```

Out[506]: []



In [703]:
```
df.groupby('rest_id')['order_tot'].sum().mean()
```

Out[703]: 2908.6812779552683

In [697]:
```
df_cust=pd.DataFrame(df.groupby('cust_id')['order_tot'].sum())
```

In [698]:
```
df_cust
```

Out[698]:

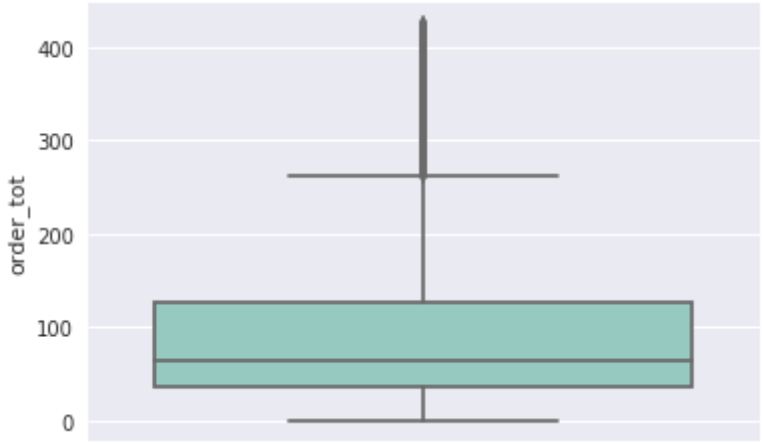
	order_tot
cust_id	
5	98.85
14	620.18
15	267.73
21	189.66
25	38.52
26	47.78
31	243.66
32	243.08
34	189.50
35	73.32
37	19.59
40	76.67
43	96.60
52	246.00
54	185.38
58	31.99
70	44.08
73	28.24
81	53.16
82	59.51
84	45.53
88	77.45
90	45.15
97	41.24
99	93.36
105	45.10
114	401.09
118	34.11
127	73.43
129	322.76
...	...

	order_tot
cust_id	
199183	27.64
199203	35.31
199264	30.47
199275	17.39
199300	52.16
199339	32.64
199342	26.66
199347	16.33
199355	30.36
199412	98.33
199425	78.21
199438	33.51
199439	40.74
199442	81.48
199445	30.36
199454	28.84
199484	19.54
199505	32.59
199532	16.82
199538	47.31
199572	214.66
199589	78.32
199614	25.02
199698	37.27
199817	450.36
199819	1024.20
200285	56.30
200354	20.14
200382	36.34
200449	25.58

6664 rows × 1 columns

In [700]:
```
sns.boxplot(y='order_tot', data=df_cust[df_cust.order_tot<425], palette="Set3")
```

Out[700]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6fe415790>



Investigate new vs existing consumers

In [718]:
```
df.loc[df.cust_id==15]
```

Out[718]:

	driver_id	rest_id	cust_id	is_new	delivery_region	is_asap	order_tot	amt_tip	rest_p
13	320	5	15	False	Palo Alto	True	33.08	1.65	NaN
14	305	20	15	False	Palo Alto	True	20.08	0.64	NaN
15	60	68	15	True	Palo Alto	True	49.88	2.01	NaN
16	156	86	15	True	Palo Alto	True	25.68	0.90	True
17	383	220	15	False	Palo Alto	True	31.50	1.17	True
18	330	232	15	True	Palo Alto	True	64.18	2.67	NaN
19	299	312	15	False	Palo Alto	True	43.33	1.71	NaN

7 rows × 25 columns

In [722]:
```
df.loc[df.is_new==True]['cust_id'].unique()
```

Out[722]:
```
array([    15,     21,     26, ..., 199572, 199817, 199819])
```

In [729]:
```
df.loc[df['cust_id'].isin(df.loc[df.is_new==True]['cust_id'].unique()), 'new_user']=1
```

In [730]:
```
print pd.DataFrame(df.loc[df.new_user==1].groupby('cust_id').size()).mean()
```

```
0    3.765534
dtype: float64
```

In [735]:
```
print pd.DataFrame(df.loc[df.new_user==1].groupby('cust_id').sum()).mean()
print pd.DataFrame(df.loc[df.new_user!=1].groupby('cust_id').sum()).mean()
```

```
driver_id                  836.821872
rest_id                    408.195112
is_new                       1.442005
is_asap                      2.966446
order_tot                  197.681582
amt_tip                     13.714362
customer_delivery_time     298.003480
restaurant_notification    133.383036
prep_and_pick               56.006331
driver_delivery_time       108.614112
day                         59.009114
hour                        57.158658
perc_discount                0.100937
perc_refund                  0.041895
new_dummy                    1.442005
hour_delivered              60.695112
new_user                     3.765534
dtype: float64
driver_id                  462.018588
rest_id                    225.169882
is_new                       0.000000
is_asap                      1.686118
order_tot                  101.932682
amt_tip                      6.826762
customer_delivery_time     167.516745
restaurant_notification     64.891824
prep_and_pick               41.828039
driver_delivery_time        60.796882
day                         32.468235
hour                        32.037647
perc_discount                0.086338
perc_refund                  0.022889
new_dummy                    0.000000
hour_delivered              33.958588
new_user                     0.000000
dtype: float64
```

Investigate user retention; performing t-tests

In [740]:
```
df=df.drop('ordered_first_week', axis=1)
```

In [741]:
```
df.loc[df['cust_id'].isin(df.loc[df.day<=7]['cust_id'].unique()), 'order
ed_first_week']=1
df.loc[df['cust_id'].isin(df.loc[(df.day>7)&(df.day<=14)]['cust_id'].uni
que()), 'ordered_second_week']=1
df.loc[df['cust_id'].isin(df.loc[(df.day>14)&(df.day<=21)]['cust_id'].un
ique()), 'ordered_third_week']=1
df.loc[df['cust_id'].isin(df.loc[(df.day>21)&(df.day<=28)]['cust_id'].un
ique()), 'ordered_fourth_week']=1
df.loc[df['cust_id'].isin(df.loc[(df.day>28)]['cust_id'].unique()), 'ord
ered_fifth_week']=1
```

In [746]:
```
print df.loc[(df['ordered_first_week']==1)]['cust_id'].count()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']==1
)]['cust_id'].count()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_third_week']==1
)]['cust_id'].count()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_fourth_week']==1
)]['cust_id'].count()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_fifth_week']==1
)]['cust_id'].count()
```

```
11202
8199
7814
8137
4250
```

In [759]:
```
#print df.loc[(df['ordered_first_week']==1)]['customer_delivery_time'].m
ean()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']==1
)]['customer_delivery_time'].mean()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']==1
)]['customer_delivery_time'].count()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']==1
)]['customer_delivery_time'].std()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']!=1
)]['customer_delivery_time'].mean()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']!=1
)]['customer_delivery_time'].count()
print df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']!=1
)]['customer_delivery_time'].std()
```

```
72.4883989917
8199
204.827303094
84.4735764236
3003
290.496397509
```

In [765]:
```
sns.distplot(df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']==1)&(df['customer_delivery_time']<100)]['customer_delivery_time'], label="retained_second_week")
sns.distplot(df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']!=1)&(df['customer_delivery_time']<100)]['customer_delivery_time'], label="churned_second_week")

plt.legend()
```

Out[765]: <matplotlib.legend.Legend at 0x7fe6fd9c4b10>

In [755]:
```
stats.ttest_ind(df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']==1)]['customer_delivery_time'],
                df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']!=1)]['customer_delivery_time'], equal_var=False)
```

Out[755]:
```
Ttest_indResult(statistic=-2.0794855912325243, pvalue=0.037634073793290
13)
```

In [ ]:
```
sns.distplot(df.groupby('cust_id')['order_tot'].sum(), label="total_rev")
plt.legend()
```

In [ ]:
```
stats.ttest_ind(df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']==1)&(df['ordered_third_week']==1)]['customer_delivery_time'],
                df.loc[(df['ordered_first_week']==1)&(df['ordered_second_week']!=1)]['customer_delivery_time'], equal_var=False)
```

In [731]:
```
print pd.DataFrame(df.loc[df.new_user!=1].groupby('cust_id').size()).mean()
```

```
0    2.069882
dtype: float64
```

In [716]:
```
print pd.DataFrame(df.loc[df.is_new==True].groupby('cust_id').size()).mean()
print pd.DataFrame(df.loc[df.is_new==False].groupby('cust_id').size()).mean()
```

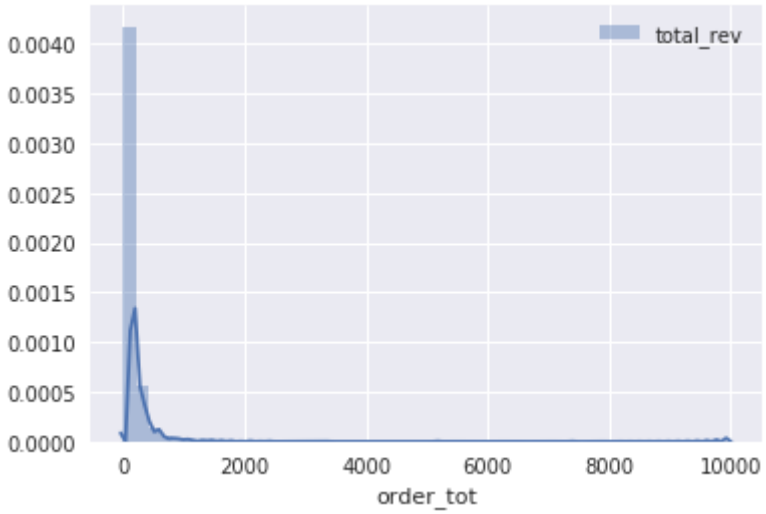
```
0    1.442005
dtype: float64
0    2.477386
dtype: float64
```

In [692]:
```
stats.ttest_ind(df[df.is_new==True]['order_tot'], df[df.is_new==False]['order_tot'], equal_var=False)
```

Out[692]:
```
Ttest_indResult(statistic=1.0555837917131625, pvalue=0.29120849235194474)
```

In [676]:
```
sns.distplot(df.groupby('cust_id')['order_tot'].sum(), label="total_rev")
plt.legend()
```

Out[676]:
```
<matplotlib.legend.Legend at 0x7fe720547610>
```



In [682]:
```
print df[df.is_new==True]['order_tot'].mean()
print df[df.is_new==True]['order_tot'].std()
print df[df.is_new==False]['order_tot'].mean()
print df[df.is_new==False]['order_tot'].std()
```

```
51.7215627693
51.8047537585
50.6993252811
49.0271205823
```

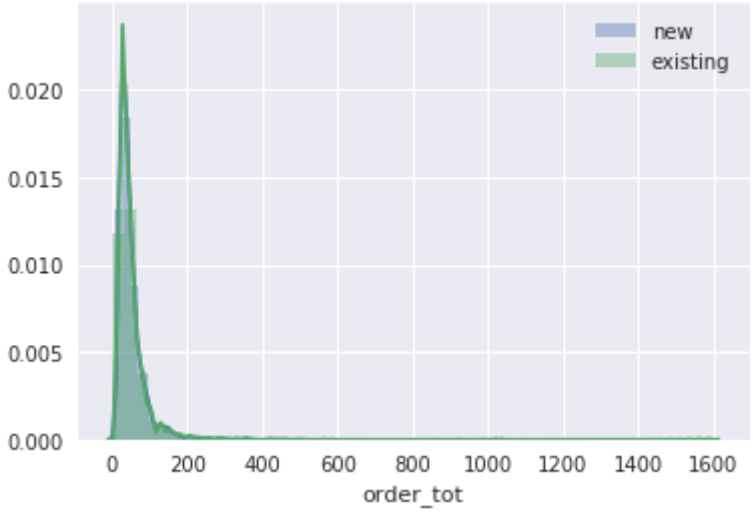
In [766]:
```
print df[df.is_new==True]['amt_tip'].mean()
print df[df.is_new==True]['amt_tip'].std()
print df[df.is_new==False]['amt_tip'].mean()
print df[df.is_new==False]['amt_tip'].std()
```

```
3.48557311117
3.93592886121
3.46986880466
3.59387842033
```

Investigating delivery time

In [683]:
```
sns.distplot(df[df.is_new==True]['order_tot'], label="new")
sns.distplot(df[df.is_new==False]['order_tot'], label="existing")
plt.legend()
```

Out[683]: <matplotlib.legend.Legend at 0x7fe72ccb3d50>



In [546]:
```
print df[df['is_asap'] == True]['customer_delivery_time'].quantile([.99, .95])
print df[df['is_asap'] == False]['customer_delivery_time'].quantile([.99, .95])
print df[df['is_asap'] == True]['customer_delivery_time'].mean()
print df[df['is_asap'] == False]['customer_delivery_time'].mean()
print df[df['is_asap'] == True]['customer_delivery_time'].median()
print df[df['is_asap'] == False]['customer_delivery_time'].median()
print df[df['is_asap'] == True]['customer_delivery_time'].std()
print df[df['is_asap'] == False]['customer_delivery_time'].std()
```

```
0.99    93.615333
0.95    75.973333
Name: customer_delivery_time, dtype: float64
0.99    2786.001833
0.95     918.206667
Name: customer_delivery_time, dtype: float64
46.8411693539
213.548632959
43.8
91.8416666667
39.0226734315
515.220504391
```

In [544]:
```
print df[df['is_asap'] == True]['driver_delivery_time'].quantile([.99, .95])
print df[df['is_asap'] == True]['driver_delivery_time'].mean()
print df[df['is_asap'] == True]['driver_delivery_time'].median()
print df[df['is_asap'] == True]['driver_delivery_time'].std()
```

```
0.99    70.958000
0.95    52.961667
Name: driver_delivery_time, dtype: float64
28.2964391243
25.8
13.342203621
```

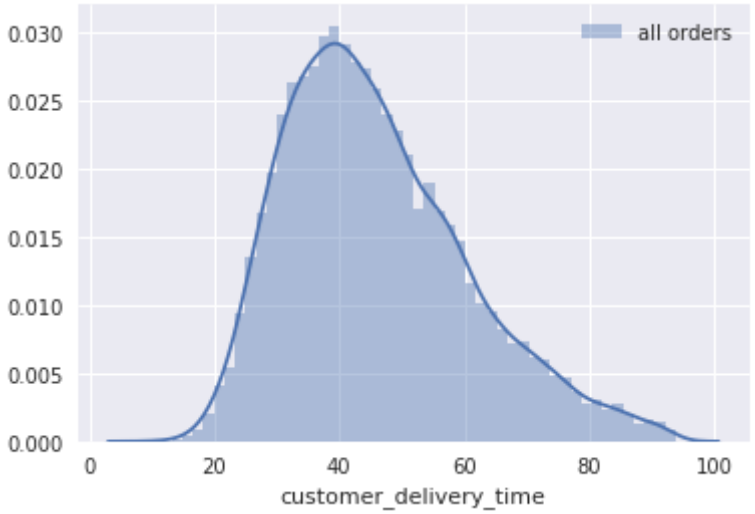
In [545]:
```
#print df[df['is_asap'] == True]['prep_and_pick'].quantile([.99, .95])
print df[df['is_asap'] == True]['prep_and_pick'].mean()
print df[df['is_asap'] == True]['prep_and_pick'].median()
print df[df['is_asap'] == True]['prep_and_pick'].std()

#print df[df['is_asap'] == True]['restaurant_notification'].quantile([.99, .95])
print df[df['is_asap'] == True]['restaurant_notification'].mean()
print df[df['is_asap'] == True]['restaurant_notification'].median()
print df[df['is_asap'] == True]['restaurant_notification'].std()
```

```
14.2190863405
9.65
358.975772857
4.32564388916
3.18333333333
360.229227696
```

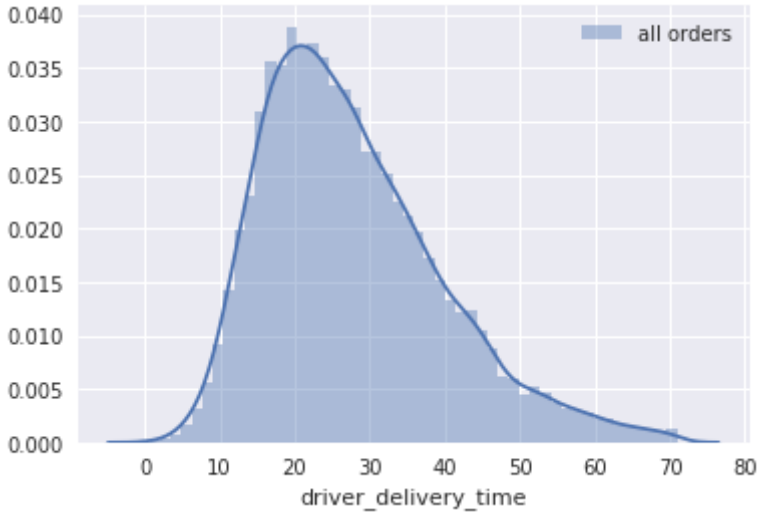
In [510]:
```
sns.distplot(df.loc[(df.customer_delivery_time<93.79)&(df.is_asap==True
)]['customer_delivery_time'], label="all orders")
plt.legend()
```

Out[510]: <matplotlib.legend.Legend at 0x7fe745946890>



In [511]:
```
sns.distplot(df.loc[(df.driver_delivery_time<70.99)&(df.is_asap==True)][
'driver_delivery_time'], label="all orders")
plt.legend()
```

Out[511]: <matplotlib.legend.Legend at 0x7fe74579cdd0>



In [767]:
```
print df[df['is_asap'] == False]['prep_and_pick'].quantile([.99])
print df[df['is_asap'] == False]['prep_and_pick'].mean()
print df[df['is_asap'] == False]['prep_and_pick'].median()
print df[df['is_asap'] == False]['prep_and_pick'].std()


print df[df['is_asap'] == False]['restaurant_notification'].quantile([0.
1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
print df[df['is_asap'] == False]['restaurant_notification'].mean()
print df[df['is_asap'] == False]['restaurant_notification'].median()
print df[df['is_asap'] == False]['restaurant_notification'].std()
```

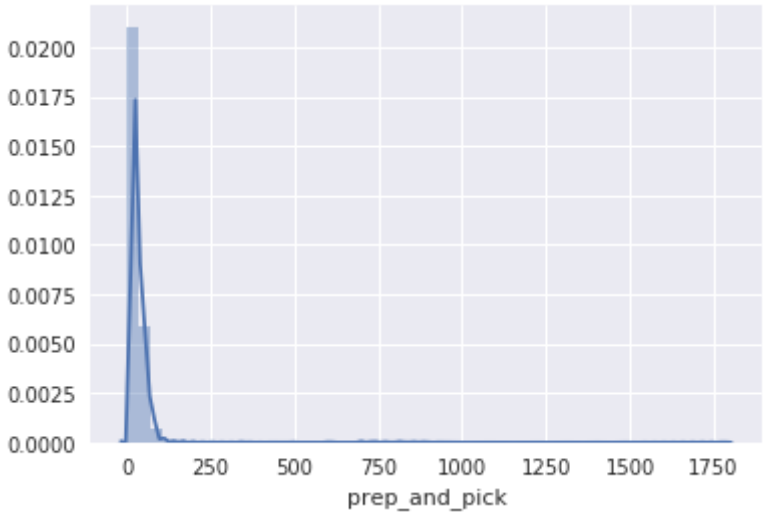
```
0.99    127.8295
Name: prep_and_pick, dtype: float64
30.6886516854
20.5666666667
110.134129174
0.1       3.215000
0.2      10.383333
0.3      18.750000
0.4      27.780000
0.5      37.558333
0.6      49.450000
0.7      66.693333
0.8     102.353333
0.9     201.228333
1.0    5868.916667
Name: restaurant_notification, dtype: float64
150.506573034
37.5583333333
493.766985633
```

In [548]:
```
print df[df['is_asap'] == False]['order_tot'].mean()
print df[df['is_asap'] == True]['order_tot'].mean()
```

```
78.8877191011
43.9433908006
```

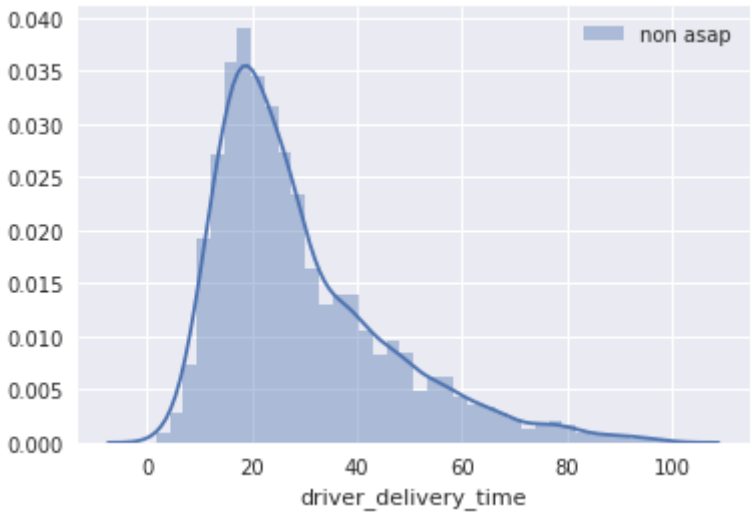
In [513]:
```
sns.distplot(df.loc[(df.prep_and_pick<2650)&(df.is_asap==False)]['prep_and_pick'], label="all orders")
```

Out[513]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe74569b210>



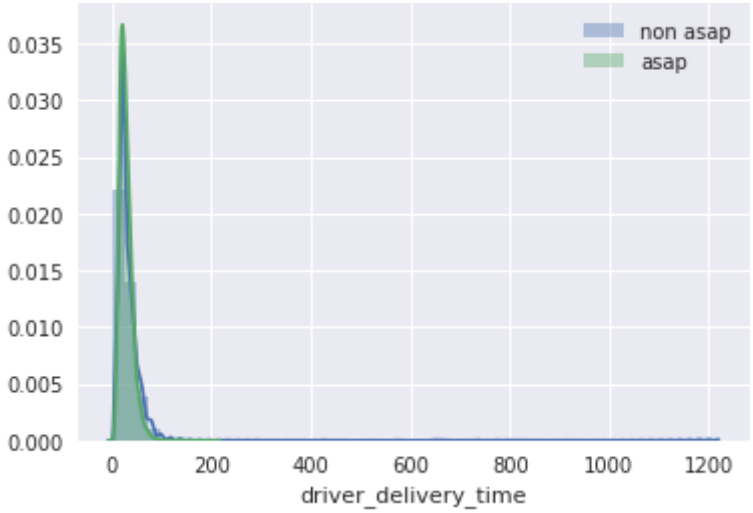
In [549]:
```
sns.distplot(df.loc[(df.driver_delivery_time<99.79)&(df.is_asap==False)]['driver_delivery_time'], label="non asap")
plt.legend()
```

Out[549]: <matplotlib.legend.Legend at 0x7fe744e04c10>



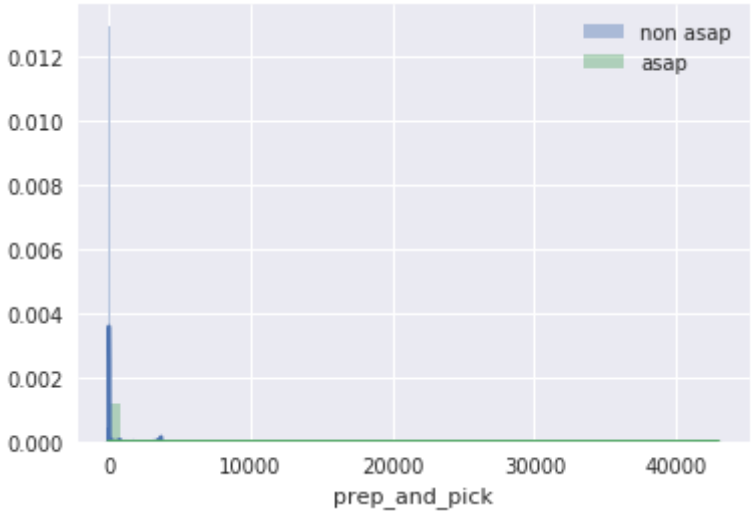
In [515]:
```
sns.distplot(df.loc[df.is_asap==False]['driver_delivery_time'], label="non asap")
sns.distplot(df.loc[df.is_asap==True]['driver_delivery_time'], label="asap")
plt.legend()
```

Out[515]: <matplotlib.legend.Legend at 0x7fe74555a3d0>



In [521]:
```
sns.distplot(df.loc[df.is_asap==False]['prep_and_pick'], label="non asap")
sns.distplot(df.loc[df.is_asap==True]['prep_and_pick'], label="asap")
plt.legend()
```

Out[521]: <matplotlib.legend.Legend at 0x7fe745002bd0>

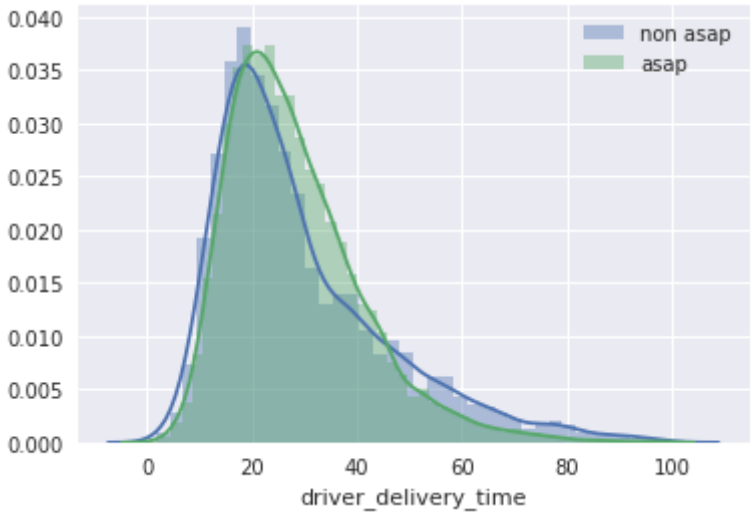


In [522]:
```
stats.ttest_ind(df.loc[df.is_asap==False]['prep_and_pick'], df.loc[df.is_asap==True]['prep_and_pick'], equal_var=False)
```

Out[522]:
```
Ttest_indResult(statistic=4.67673513683914, pvalue=2.936856256071042e-06)
```

In [516]:
```
sns.distplot(df.loc[(df.driver_delivery_time<100)&(df.is_asap==False)][
'driver_delivery_time'], label="non asap")
sns.distplot(df.loc[(df.driver_delivery_time<100)&(df.is_asap==True)]['d
river_delivery_time'], label="asap")
plt.legend()
```

Out[516]:
```
<matplotlib.legend.Legend at 0x7fe7455632d0>
```



In [525]:
```
df.loc[df.is_asap==True]['driver_delivery_time'].std()
```

Out[525]:
```
13.342203620962648
```

In [526]:
```
df.loc[df.is_asap==False]['driver_delivery_time'].std()
```

Out[526]:
```
46.30633577367681
```

In [527]:
```
print df.loc[df.is_asap==True]['prep_and_pick'].std()
print df.loc[df.is_asap==False]['prep_and_pick'].std()
```

```
358.975772857
110.134129174
```

In [517]:
```
stats.ttest_ind(df.loc[df.is_asap==False]['driver_delivery_time'], df.lo
c[df.is_asap==True]['driver_delivery_time'], equal_var=False)
```

Out[517]:
```
Ttest_indResult(statistic=5.174309761185881, pvalue=2.4088258372662247e
-07)
```

In [ ]:
```
df.loc[df.is_asap==True]['driver_delivery_time']
```

In [518]:
```
print df[df['is_asap'] == False]['driver_delivery_time'].quantile([.99])
print df[df['is_asap'] == False]['driver_delivery_time'].mean()
print df[df['is_asap'] == False]['driver_delivery_time'].median()
```

```
0.99    99.857167
Name: driver_delivery_time, dtype: float64
32.3534082397
24.7583333333
```

In [646]:
```
df_asap=df[df['is_asap'] == True]
```

In [533]:
```
print df_asap.groupby('delivery_region')['customer_delivery_time'].mean
()
print df_asap.groupby('delivery_region')['driver_delivery_time'].mean()
```

```
delivery_region
Mountain View    46.678463
Palo Alto        46.162330
San Jose         49.462170
Name: customer_delivery_time, dtype: float64
delivery_region
Mountain View    27.326451
Palo Alto        28.227745
San Jose         29.733900
Name: driver_delivery_time, dtype: float64
```

In [534]:
```
print df_asap.groupby('delivery_region')['customer_delivery_time'].std()
print df_asap.groupby('delivery_region')['driver_delivery_time'].std()
```

```
delivery_region
Mountain View    16.601881
Palo Alto        47.909483
San Jose         17.111176
Name: customer_delivery_time, dtype: float64
delivery_region
Mountain View    13.472428
Palo Alto        12.719439
San Jose         15.117002
Name: driver_delivery_time, dtype: float64
```

In [647]:
```
df_asap.isna().sum()
```

Out[647]:
```
driver_id                     0
rest_id                       0
cust_id                       0
is_new                        0
delivery_region               0
is_asap                       0
order_tot                     0
amt_tip                       0
rest_place_time_missing     10752
driver_rest_time_missing    14327
cust_place_time_pdt           0
rest_place_time_pdt           0
driver_rest_time_pdt          0
cust_deliver_time_pdt         0
customer_delivery_time        0
restaurant_notification       0
prep_and_pick                 0
driver_delivery_time          0
weekday                       0
day                           0
hour                          0
perc_discount                 2
perc_refund                   1
new_dummy                     0
hour_delivered                0
dtype: int64
```

Modeling

In [648]:
```
df_asap.loc[(df_asap.hour>=16)&(df_asap.hour<=19), 'dinner_peak'] = 1
df_asap.loc[(df_asap.hour<16)|(df_asap.hour>19), 'dinner_peak'] = 0
```

In [649]:
```
df_asap.loc[(df_asap.hour>=9)&(df_asap.hour<=12), 'lunch_peak'] = 1
df_asap.loc[(df_asap.hour<9)|(df_asap.hour>12), 'lunch_peak'] = 0
```

In [650]:
```
predictor_list=[
 'is_new',
 'delivery_region',
 'is_asap',
 'order_tot',
 'amt_tip',
 'weekday',
 'dinner_peak',
 'lunch_peak',
 'new_dummy']
X=df_asap[predictor_list]
```

In [652]:
```
X=pd.get_dummies(X, drop_first = True)
```

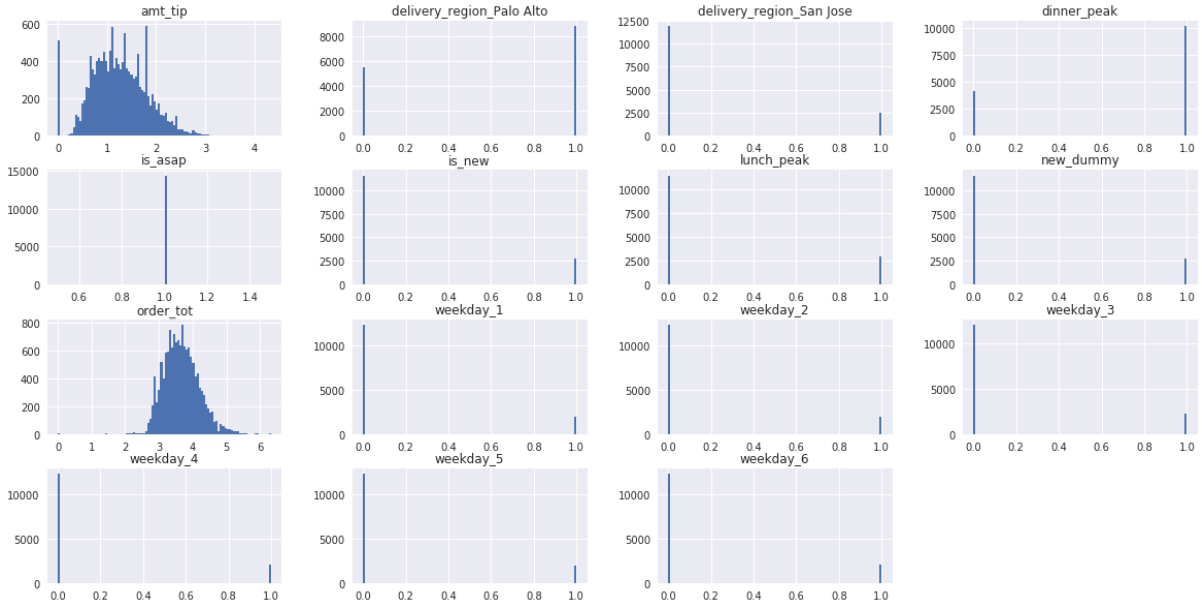
In [654]:
```
X.amt_tip=X.amt_tip.apply(np.log1p)
X.order_tot=X.order_tot.apply(np.log1p)
```

In [655]:
```
y=pd.DataFrame(df_asap['customer_delivery_time'])
```

In [656]:
```
y = y.apply(np.log1p)
```

In [657]:
```
X.hist(bins=100, figsize=(20, 10))
```

Out[657]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fe721a3aa90
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe778b92fd0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe72243f810
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe722038690
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe721ddc090
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe721701dd0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe72172ba10
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe7216c8d50
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe7216aaf50
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe7216bf510
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe721624ad0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe7215eab10
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe7215b15d0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe721569fd0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe721520d10
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe7214e7d50
>]],
      dtype=object)
```

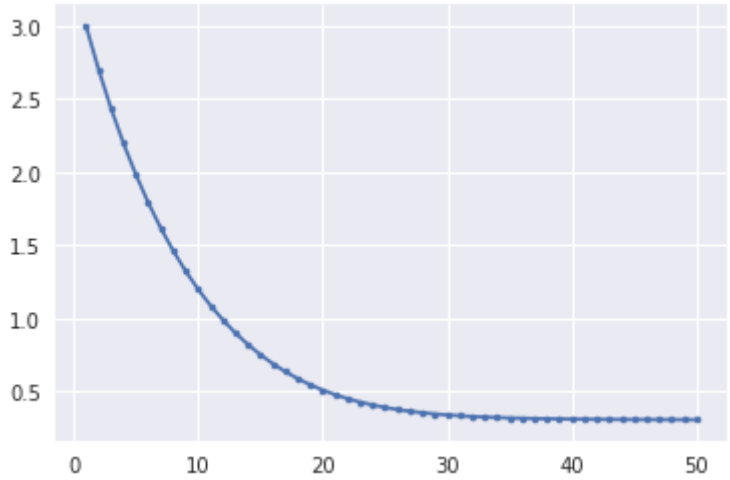


In [659]:
```
xgb_params = {
    'eta': 0.1,
    'max_depth' : 2,
    'subsample' : 0.7,
    'objective': 'reg:linear',
    'silent': 1,
    'seed': 0
}

dtrain = xgb.DMatrix(X, y, missing = np.nan, feature_names = X.columns.v
alues)
cv = xgb.cv(xgb_params, dtrain, num_boost_round = 50, nfold = 10)

plt.errorbar(np.arange(1,51), cv['test-rmse-mean'].values, cv['test-rmse
-std'].values, linestyle='-', marker='.')
```

Out[659]: <Container object of 3 artists>



In [660]:
```
model2 = xgb.train(xgb_params, dtrain, num_boost_round = 30)
```

In [661]:
```
fig, ax = plt.subplots(figsize = (12,5))
xgb.plot_importance(model2, max_num_features = 50, height = 0.5, ax = ax
)
```

Out[661]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe7213c8d50>