In [19]:
```
import pandas as pd
import numpy as np
import os
```

In [28]:
```
os.chdir = (r'\\10.0.7.226\ipba_group10')
```

In [29]:
```
os.getcwd()
```

Out[29]:
```
'C:\\Users\\IPBAB047'
```

Product Dataset - EDA

In [25]:
```
product = pd.read_csv(r'\\10.0.7.226\ipba_group10\product_dataset.c
sv')
```

In [30]:
```
product.head()
```

Out[30]:

	DBSKU	DEPARTMENT	CLASS	SUBCLASS	DEPARTMENT_NAME	CLASS_NAME	SU
0	2182204.0	12	3	32	Dept;1	Class;1	
1	2860882.0	12	3	31	Dept;1	Class;1	
2	2695858.0	12	5	50	Dept;1	Class;2	
3	675793.0	10	4	41	Dept;2	Class;3	
4	2864173.0	12	4	40	Dept;1	Class;3	

In [6]:
```
# number of rows(29342) and columns(7)
product.shape
```

Out[6]:
```
(29342, 7)
```

In [7]:
```
# info on dtypes
product.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29342 entries, 0 to 29341
Data columns (total 7 columns):
DBSKU                27232 non-null float64
DEPARTMENT           29342 non-null int64
CLASS                29342 non-null int64
SUBCLASS             29342 non-null int64
DEPARTMENT_NAME      29342 non-null object
CLASS_NAME           29342 non-null object
SUBCLASS_NAME;;;;    29342 non-null object
dtypes: float64(1), int64(3), object(3)
memory usage: 1.6+ MB
```

In [8]:
```
# product description is not relevant as the highlightened columns are all categorical
product.describe()
```

Out[8]:

	DBSKU	DEPARTMENT	CLASS	SUBCLASS
count	2.723200e+04	29342.000000	29342.000000	29342.000000
mean	1.372330e+06	10.730352	2.878093	28.343978
std	1.073834e+06	0.962975	2.490916	12.973040
min	1.000080e+05	10.000000	1.000000	5.000000
25%	4.617190e+05	10.000000	2.000000	20.000000
50%	7.814465e+05	10.000000	2.000000	21.000000
75%	2.632754e+06	12.000000	4.000000	40.000000
max	2.999987e+06	12.000000	99.000000	99.000000

In [9]:
```
# info on PRODUCT dataset' missing values
product.isnull().sum()
```

Out[9]:
```
DBSKU                2110
DEPARTMENT              0
CLASS                   0
SUBCLASS                0
DEPARTMENT_NAME         0
CLASS_NAME              0
SUBCLASS_NAME;;;;       0
dtype: int64
```

In [10]:
```
# Let's rename the messy SUBCLASS column by deleting the present se
micolon (;)
product = product.rename(columns = {"SUBCLASS_NAME;;;;":"SUBCLASS_N
AME"})
```

In [11]:
```
# Let's delete the semicolons present within the following columns,
and add an extra space between charachters
product['SUBCLASS_NAME'] = product['SUBCLASS_NAME'].str.replac
e(';',' ').astype(str)
product['CLASS_NAME'] = product['CLASS_NAME'].str.replace(';',' ').
astype(str)
product['DEPARTMENT_NAME'] = product['DEPARTMENT_NAME'].str.replac
e(';',' ').astype(str)
```

In [12]:
```
# Let's fill DBSKU NaNs with "NOT APPLICABLE"
product['DBSKU'].fillna(1.0, inplace = True)
```

In [13]:
```
# Let's take a look at the final result by showing the dataset's he
ad (first 2 values )
product.head(2)
```

Out[13]:

	DBSKU	DEPARTMENT	CLASS	SUBCLASS	DEPARTMENT_NAME	CLASS_NAME	SU
0	2182204.0	12	3	32	Dept 1	Class 1	
1	2860882.0	12	3	31	Dept 1	Class 1	

In [14]:
```
pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns',None)
```

In [15]:
```
# How many unique values there is in PRODUCT dataset
product.nunique(axis = 0, dropna=True)
```

Out[15]:
```
DBSKU              24293
DEPARTMENT             2
CLASS                  6
SUBCLASS              15
DEPARTMENT_NAME        2
CLASS_NAME             6
SUBCLASS_NAME          8
dtype: int64
```

In [16]:
```
# No missing values left uncovered as we created a new group within DBSKU called "NOT APPLICABLE"
product.isnull().sum()
```

Out[16]:
```
DBSKU              0
DEPARTMENT         0
CLASS              0
SUBCLASS           0
DEPARTMENT_NAME    0
CLASS_NAME         0
SUBCLASS_NAME      0
dtype: int64
```

In [17]:
```
product.head()
```

Out[17]:

	DBSKU	DEPARTMENT	CLASS	SUBCLASS	DEPARTMENT_NAME	CLASS_NAME	SU
0	2182204.0	12	3	32	Dept 1	Class 1	
1	2860882.0	12	3	31	Dept 1	Class 1	
2	2695858.0	12	5	50	Dept 1	Class 2	
3	675793.0	10	4	41	Dept 2	Class 3	
4	2864173.0	12	4	40	Dept 1	Class 3	

In [18]:
```
# Now DBSKU is an object and no longer a float
product.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29342 entries, 0 to 29341
Data columns (total 7 columns):
DBSKU              29342 non-null float64
DEPARTMENT         29342 non-null int64
CLASS              29342 non-null int64
SUBCLASS           29342 non-null int64
DEPARTMENT_NAME    29342 non-null object
CLASS_NAME         29342 non-null object
SUBCLASS_NAME      29342 non-null object
dtypes: float64(1), int64(3), object(3)
memory usage: 1.6+ MB
```

In [19]:
```
# DEPARTMENT - unique values
product['DEPARTMENT'].unique()
```

Out[19]:
```
array([12, 10], dtype=int64)
```

In [20]:
```
# CLASS - unique values
product['CLASS'].unique()
```

Out[20]:
```
array([ 3,  5,  4,  2,  1, 99], dtype=int64)
```

In [21]:
```
# SUBCLASS - unique values
product['SUBCLASS'].unique()
```

Out[21]:
```
array([32, 31, 50, 41, 40, 20, 21, 42,  5, 52, 51,  6, 30, 99, 3
7],
      dtype=int64)
```

In [22]:
```
product['DEPARTMENT_NAME'].unique()
```

Out[22]:
```
array(['Dept 1', 'Dept 2'], dtype=object)
```

In [23]:
```
product['CLASS_NAME'].unique()
```

Out[23]:
```
array(['Class 1', 'Class 2', 'Class 3', 'Class 4', 'Class 5', 'Cla
ss 6'],
      dtype=object)
```

In [24]:
```
product['SUBCLASS_NAME'].unique()
```

Out[24]:
```
array(['Sub Class 1', 'Sub Class 2', 'Sub Class 3', 'Sub Class 4',
       'Sub Class 5', 'Sub Class 6', 'Sub Class 7', 'Sub Class
8'],
      dtype=object)
```

product.to_csv(r'\\10.0.7.226\ipba_group10\product_new.csv')

Store Dataset - EDA

In [31]:
```
store = pd.read_csv(r'\\10.0.7.226\ipba_group10\store_dataset.csv')
```

In [32]:
```
# Let's have a glimpse of Store dataet's structure
store.head()
```

Out[32]:

	LOC_IDNT	CITY	STATE	STORE_TYPE	POSTAL_CD	STORE_SIZE
0	249	ST LOUIS	MO	Strip Store	63119	3963.0
1	401	PATCHOGUE	NY	Power Strip	11772	3378.0
2	644	NAPLES	FL	Outlet Strip	34114	3652.0
3	992	Carson	CA	NaN	90745	NaN
4	1270	CONCORD	NH	Regional Mall	3301	2535.0

In [27]:
```
# STORE dataset number of rows(1303) and columns(6)
store.shape
```

Out[27]:
```
(1303, 6)
```

In [28]:
```
# STORE dataset dtypes
store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 6 columns):
LOC_IDNT      1303 non-null int64
CITY          1303 non-null object
STATE         1303 non-null object
STORE_TYPE    1270 non-null object
POSTAL_CD     1303 non-null int64
STORE_SIZE    1119 non-null float64
dtypes: float64(1), int64(2), object(3)
memory usage: 61.2+ KB
```

In [29]:
```
# STORE_SIZE is the only column for which is worth checking it's de
scriptive structure
store['STORE_SIZE'].describe()
```

Out[29]:
```
count    1119.000000
mean     3345.776586
std       830.145647
min         1.000000
25%      3000.000000
50%      3375.000000
75%      3772.000000
max      6533.000000
Name: STORE_SIZE, dtype: float64
```

In [30]:
```
# Let's use groupby and aggregate function to check the mean size p
er store type
type_size = store.groupby('STORE_TYPE').agg({'STORE_SIZE': ['mea
n','median','min', 'max','count']})
print(type_size)
```

```
                      STORE_SIZE
                            mean  median     min     max count
STORE_TYPE
Downtown Store       3126.680000  3186.0     1.0  5225.0    25
Freestanding Store   4544.000000  3797.0  3745.0  6090.0     3
Lifestyle Center     3340.545455  3079.0  2395.0  4914.0    11
Mega Outlet Mall     3700.961538  3663.5  2225.0  4722.0    26
Mini Mall            2888.000000  3202.0     1.0  5550.0    11
NOT APPLICABLE          1.000000     1.0     1.0     1.0     1
No location format           NaN     NaN     NaN     NaN     0
Outlet Mall          3478.166667  3460.0     1.0  5558.0    30
Outlet Strip         3327.324176  3504.0     1.0  6204.0   182
Power Strip          3327.275542  3302.0  1785.0  6533.0   323
Regional Mall        3295.300000  3246.0     1.0  5601.0    60
Strip Store          3369.046620  3399.0     1.0  6000.0   429
Tourist Outlet Mall  2862.000000  2862.0  2405.0  3319.0     2
Tourist Outlet Strip 3417.466667  3231.0  2494.0  5143.0    15
```

In [31]:
```
# We have few missing values in STORE_TYPE(33) and STORE_SIZE(184)
store.isnull().sum()
```

Out[31]:
```
LOC_IDNT        0
CITY            0
STATE           0
STORE_TYPE     33
POSTAL_CD       0
STORE_SIZE    184
dtype: int64
```

In [32]:
```
# Let's check all the rows presenting missing values
store[store.isnull().any(axis=1)]
```

Out[32]:

	LOC_IDNT	CITY	STATE	STORE_TYPE	POSTAL_CD	STORE_SIZE
3	992	Carson	CA	NaN	90745	NaN
5	250	SUFFERN	NY	Strip Store	10901	NaN
8	993	Rancho Dominguez	CA	NaN	90221	NaN
13	925	KANSAS CITY	MO	Lifestyle Center	64153	NaN
27	931	SUFFERN	NY	Power Strip	10901	NaN
32	9931	Rancho Dominguez	CA	NaN	90221	NaN
37	9921	Carson	CA	NaN	90745	NaN
47	929	SUFFERN	NY	Downtown	10901	NaN

				Store		
52	927	GILBERT	AZ	Power Strip	85297	NaN
57	926	SUFFERN	NY	Power Strip	10901	NaN
71	587	NORTH MYRTLE BEACH	SC	Outlet Strip	29582	NaN
72	870	SUFFERN	NY	Outlet Strip	10901	NaN
82	4101	Suffern	NY	NaN	10901	NaN
88	4100	PATASKALA	OH	NOT APPLICABLE	43062	NaN
93	889	SUFFERN	NY	Strip Store	10901	NaN
103	886	STAMFORD	CT	Strip Store	6905	NaN
113	877	SUFFERN	NY	Downtown Store	10901	NaN
124	995	Pico Rivera	CA	NaN	90660	NaN
126	211	SUFFERN	NY	Power Strip	10901	NaN
130	9003	SUFFERN	NY	Power Strip	10901	NaN
134	907	SUFFERN	NY	Strip Store	10901	NaN
144	991	Suffern	NY	NaN	10901	NaN
149	9911	Suffern	NY	NaN	10901	NaN
153	638	SUFFERN	NY	Outlet Strip	10901	NaN
154	97	SUFFERN	NY	Strip Store	10901	NaN
159	897	SUFFERN	NY	Downtown Store	10901	NaN
160	9002	SUFFERN	NY	Power Strip	10901	NaN
161	172	SUFFERN	NY	Strip Store	10901	NaN
172	264	SUFFERN	NY	Strip Store	10901	NaN
179	921	SUFFERN	NY	Power Strip	10901	NaN
184	883	PITTSBURGH	PA	Regional Mall	15237	NaN
188	662	SUFFERN	NY	Downtown Store	10901	NaN
198	906	SUFFERN	NY	Downtown Store	10901	NaN
237	9861	Santa Fe Springs	CA	NaN	90670	NaN
256	840	WEST CALDWELL	NJ	Strip Store	7006	NaN
263	9004	SUFFERN	NY	Power Strip	10901	NaN
268	9001	SUFFERN	NY	Power Strip	10901	NaN
272	95	SUFFERN	NY	Strip Store	10901	NaN

273	933	BRIARCLIFF	NY	Power Strip	10510	NaN
288	9005	SUFFERN	NY	Power Strip	10901	NaN
292	89	SUFFERN	NY	Outlet Mall	10901	NaN
297	891	READING	PA	Outlet Mall	19610	NaN
302	902	OFALLON	MO	Strip Store	63368	NaN
303	9006	SUFFERN	NY	Power Strip	10901	NaN
319	675	SUFFERN	NY	Strip Store	10901	NaN
321	4150	GREENCASTLE	IN	NOT APPLICABLE	46135	NaN
323	2812	BURLINGTON	NC	Outlet Strip	27215	NaN
337	918	TUSTIN	CA	Power Strip	92782	NaN
341	272	SUFFERN	NY	Strip Store	10901	NaN
367	9007	SUFFERN	NY	Power Strip	10901	NaN
369	2813	SUFFERN	NY	Outlet Mall	10901	NaN
403	2822	SUFFERN	NY	Strip Store	10901	NaN
420	43	PORTCHESTER	NY	Strip Store	10573	NaN
448	2814	LINCOLN CITY	OR	Outlet Strip	97367	NaN
459	2877	SUFFERN	NY	Outlet Mall	10901	NaN
474	1360	NEW YORK	NY	Downtown Store	10011	NaN
488	2858	SUFFERN	NY	Strip Store	10901	NaN
490	718	SUFFERN	NY	Strip Store	10901	NaN
498	2823	SUFFERN	NY	Strip Store	10901	NaN
503	2821	SUFFERN	NY	Outlet Strip	10901	NaN
508	2887	SHELTON	CT	Strip Store	6484	NaN
509	49	SUFFERN	NY	Outlet Strip	10901	NaN
514	557	SUFFERN	NY	Outlet Mall	10901	NaN
520	2893	VACAVILLE	CA	Outlet Strip	95687	NaN
530	1361	E HARTFORD	CT	Outlet Strip	6118	NaN
538	2930	SUFFERN	NY	Freestanding Store	10901	NaN
546	9941	SANTA FE SRPINGS	CA	NaN	90670	NaN
551	917	VIERA	FL	Power Strip	32940	NaN
555	2918	SUFFERN	NY	Outlet Strip	10901	NaN
558	981	MAHWAH	NJ	NaN	7430	NaN

559	99721	Riverside	CA	NaN	92508	NaN
561	2917	SUFFERN	NY	Regional Mall	10901	NaN
565	9972	Riverside	CA	NaN	92508	NaN
593	2913	SUFFERN	NY	Strip Store	10901	NaN
597	9990	Pataskala	OH	NOT APPLICABLE	43062	NaN
599	2875	SUFFERN	NY	Outlet Strip	10901	NaN
616	2931	SUFFERN	NY	Strip Store	10901	NaN
630	994	SANTA FE SRPINGS	CA	NaN	90670	NaN
633	2856	SUFFERN	NY	Mini Mall	10901	NaN
637	171	SUFFERN	NY	Strip Store	10901	NaN
638	315	SUFFERN	NY	Strip Store	10901	NaN
644	2876	SUFFERN	NY	Outlet Strip	10901	NaN
645	482	CHARLOTTESVILLE	VA	Strip Store	22901	NaN
657	2844	SUFFERN	NY	Outlet Strip	10901	NaN
663	2907	SUFFERN	NY	Outlet Strip	10901	NaN
666	1367	NEWBURGH	NY	NOT APPLICABLE	12550	NaN
674	2919	SUFFERN	NY	Outlet Mall	10901	NaN
677	9811	MAHWAH	NJ	NaN	7430	NaN
678	988	RIVERSIDE	CA	NOT APPLICABLE	92508	NaN
686	2925	SUFFERN	NY	Outlet Strip	10901	NaN
689	9961	Gardena	CA	NaN	90248	NaN
692	2928	SUFFERN	NY	Outlet Strip	10901	NaN
695	983	PATASKALA	OH	NOT APPLICABLE	43062	NaN
703	2816	PIGEON FORGE	TN	Outlet Strip	37863	NaN
709	2927	SUFFERN	NY	Strip Store	10901	NaN
714	2898	WESTBOROUGH	MA	Strip Store	1581	NaN
724	2905	SUFFERN	NY	Outlet Strip	10901	NaN
729	2915	SUFFERN	NY	Outlet Strip	10901	NaN
733	4200	GREENCASTLE	IN	NaN	46135	NaN
739	997	Riverside	CA	NaN	92508	NaN
741	2916	SUFFERN	NY	Outlet Strip	10901	NaN
745	42001	GREENCASTLE	IN	NaN	46135	NaN

752	2910	SUFFERN	NY	Outlet Strip	10901	NaN
756	1373	EVERGREEN PARK	IL	Power Strip	60805	NaN
760	884	CHARLOTTE	NC	Strip Store	28262	NaN
763	2807	SUFFERN	NY	Strip Store	10901	NaN
769	2926	SUFFERN	NY	Outlet Strip	10901	NaN
774	2908	SUFFERN	NY	Outlet Mall	10901	NaN
785	2836	SUFFERN	NY	Strip Store	10901	NaN
796	2855	SUFFERN	NY	Outlet Strip	10901	NaN
806	2937	SUFFERN	NY	NOT APPLICABLE	10901	NaN
818	283	SUFFERN	NY	Strip Store	10901	NaN
823	2842	SUFFERN	NY	Strip Store	10901	NaN
831	99	MAHWAH	NJ	NOT APPLICABLE	7430	NaN
834	2857	SUFFERN	NY	Mini Mall	10901	NaN
838	-1	No city	-1	No location format	-1	NaN
839	2549	SUFFERN	NY	NOT APPLICABLE	10901	NaN
849	2895	SUFFERN	NY	Strip Store	10901	NaN
858	982	MAHWAH	NJ	NOT APPLICABLE	7430	NaN
859	9992	Riverside	CA	NOT APPLICABLE	92508	NaN
861	2848	SUFFERN	NY	Strip Store	10901	NaN
874	905	BRANSON	MO	Outlet Mall	65616	NaN
877	2888	SUFFERN	NY	Mega Outlet Mall	10901	NaN
883	600	SUFFERN	NY	Downtown Store	10901	NaN
886	216	SUFFERN	NY	Mini Mall	10901	NaN
889	904	SUFFERN	NY	Strip Store	10901	NaN
890	9008	SUFFERN	NY	Power Strip	10901	NaN
891	266	SUFFERN	NY	Mega Outlet Mall	10901	NaN
892	423	SUFFERN	NY	Strip Store	10901	NaN
895	26	SUFFERN	NY	Strip Store	10901	NaN
899	2626	SUFFERN	NY	NOT APPLICABLE	10901	NaN

909	2833	SUFFERN	NY	Strip Store	10901	NaN
911	702	SUFFERN	NY	Outlet Strip	10901	NaN
934	859	SUFFERN	NY	Downtown Store	10901	NaN
935	9871	Santa Fe Springs	CA	NaN	90670	NaN
941	996	Gardena	CA	NaN	90248	NaN
942	9901	Pataskala	OH	NaN	43062	NaN
950	2943	READING	PA	Outlet Mall	19610	NaN
955	2884	SUFFERN	NY	Strip Store	10901	NaN
962	888	SUFFERN	NY	Strip Store	10901	NaN
975	9841	Groveport	OH	NaN	43125	NaN
990	2902	PARAMUS	NJ	Strip Store	7652	NaN
993	980	BRONX	NY	NOT APPLICABLE	10454	NaN
1006	308	SUFFERN	NY	Power Strip	10901	NaN
1009	9851	Edison	NJ	NaN	8817	NaN
1016	2914	SUFFERN	NY	Strip Store	10901	NaN
1020	9997	RIVERSIDE	CA	NOT APPLICABLE	92508	NaN
1031	147	SUFFERN	NY	Strip Store	10901	NaN
1036	984	Groveport	OH	NaN	43125	NaN
1051	812	SUFFERN	NY	Downtown Store	10901	NaN
1075	821	SUFFERN	NY	Strip Store	10901	NaN
1079	2869	SUFFERN	NY	Outlet Strip	10901	NaN
1097	830	SUFFERN	NY	Strip Store	10901	NaN
1114	601	SUFFERN	NY	Power Strip	10901	NaN
1116	9000	SUFFERN	NY	Power Strip	10901	NaN
1118	2824	TANNERSVILLE	PA	Outlet Strip	18372	NaN
1125	462	SUFFERN	NY	Power Strip	10901	NaN
1130	317	PITTSBURGH	PA	Strip Store	15238	NaN
1136	299	SUFFERN	NY	Strip Store	10901	NaN
1155	2891	SUFFERN	NY	Strip Store	10901	NaN
1160	2896	SUFFERN	NY	Mini Mall	10901	NaN
1172	2514	MONTICELLO	NY	NOT APPLICABLE	12701	NaN

1181	9971	Riverside	CA	NaN	92508	NaN
1187	2827	SUFFERN	NY	Outlet Strip	10901	NaN
1192	333	SUFFERN	NY	Strip Store	10901	NaN
1194	858	SUFFERN	NY	Strip Store	10901	NaN
1195	987	Santa Fe Springs	CA	NaN	90670	NaN
1215	833	SUFFERN	NY	Strip Store	10901	NaN
1220	2854	SUCCASUNNA	NJ	Power Strip	7876	NaN
1226	552	SUFFERN	NY	Outlet Strip	10901	NaN
1228	985	Edison	NJ	NaN	8817	NaN
1237	986	Santa Fe Springs	CA	NaN	90670	NaN
1257	2852	SUFFERN	NY	Strip Store	10901	NaN
1263	2867	SUFFERN	NY	Strip Store	10901	NaN
1268	2805	SUFFERN	NY	Outlet Strip	10901	NaN
1270	683	MAX MEADOWS	VA	Outlet Strip	24360	NaN
1277	41011	Suffern	NY	NaN	10901	NaN
1282	85	SUFFERN	NY	NOT APPLICABLE	10901	NaN
1283	9951	Pico Rivera	CA	NaN	90660	NaN
1291	2921	SUFFERN	NY	Mega Outlet Mall	10901	NaN
1296	990	Pataskala	OH	NaN	43062	NaN
1297	989	PATASKALA	OH	NOT APPLICABLE	43062	NaN
1299	2819	SUFFERN	NY	Mega Outlet Mall	10901	NaN
1300	451	SUFFERN	NY	Outlet Strip	10901	NaN
1302	1286	NASHUA	NH	NaN	3063	2732.0

In [33]:
```
# Let's replace all the missing values in 'STORE_SIZE' with the mean size depending on the belonging 'STORE_TYPE' group
store['STORE_SIZE']=store['STORE_SIZE'].fillna(store.groupby('STORE_TYPE')['STORE_SIZE'].transform('mean'))
```

In [34]:
```
# Let's check AGAIN all the rows presenting missing values
store[store.isnull().any(axis=1)]
```

Out[34]:

	LOC_IDNT	CITY	STATE	STORE_TYPE	POSTAL_CD	STORE_SIZE
3	992	Carson	CA	NaN	90745	NaN

8	993	Rancho Dominguez	CA	NaN	90221	NaN
32	9931	Rancho Dominguez	CA	NaN	90221	NaN
37	9921	Carson	CA	NaN	90745	NaN
82	4101	Suffern	NY	NaN	10901	NaN
124	995	Pico Rivera	CA	NaN	90660	NaN
144	991	Suffern	NY	NaN	10901	NaN
149	9911	Suffern	NY	NaN	10901	NaN
237	9861	Santa Fe Springs	CA	NaN	90670	NaN
546	9941	SANTA FE SRPINGS	CA	NaN	90670	NaN
558	981	MAHWAH	NJ	NaN	7430	NaN
559	99721	Riverside	CA	NaN	92508	NaN
565	9972	Riverside	CA	NaN	92508	NaN
630	994	SANTA FE SRPINGS	CA	NaN	90670	NaN
677	9811	MAHWAH	NJ	NaN	7430	NaN
689	9961	Gardena	CA	NaN	90248	NaN
733	4200	GREENCASTLE	IN	NaN	46135	NaN
739	997	Riverside	CA	NaN	92508	NaN
745	42001	GREENCASTLE	IN	NaN	46135	NaN
838	-1	No city	-1	No location format	-1	NaN
935	9871	Santa Fe Springs	CA	NaN	90670	NaN
941	996	Gardena	CA	NaN	90248	NaN
942	9901	Pataskala	OH	NaN	43062	NaN
975	9841	Groveport	OH	NaN	43125	NaN
1009	9851	Edison	NJ	NaN	8817	NaN
1036	984	Groveport	OH	NaN	43125	NaN
1181	9971	Riverside	CA	NaN	92508	NaN
1195	987	Santa Fe Springs	CA	NaN	90670	NaN
1228	985	Edison	NJ	NaN	8817	NaN
1237	986	Santa Fe Springs	CA	NaN	90670	NaN
1277	41011	Suffern	NY	NaN	10901	NaN
1283	9951	Pico Rivera	CA	NaN	90660	NaN
1296	990	Pataskala	OH	NaN	43062	NaN

1302	1286	NASHUA	NH	NaN	3063	2732.0

In [35]:
```
# Let's drop the unnecessary row that brings no value with it 'no l
ocation format'
store.drop(store.loc[store['STORE_TYPE']=='No location format'].ind
ex, inplace=True)
```

In [36]:
```
# Let's recheck the missing values and do some treatment for the re
maining NaNs
store.isnull().sum()
```

Out[36]:
```
LOC_IDNT       0
CITY           0
STATE          0
STORE_TYPE    33
POSTAL_CD      0
STORE_SIZE    32
dtype: int64
```

In [37]:
```
# How many unique values there is in STORE dataset
store.nunique(axis = 0, dropna=True)
```

Out[37]:
```
LOC_IDNT      1302
CITY           925
STATE           49
STORE_TYPE      13
POSTAL_CD      982
STORE_SIZE     838
dtype: int64
```

In [38]:
```
# Let's fill the remaining NaNs contained in 'STORE_TYPE' with the
already existing category "NOT APPLICABLE"
store['STORE_TYPE'].fillna("NOT APPLICABLE", inplace = True)
```

In [39]:
```
# Unique values of STORE_TYPE
store['STORE_TYPE'].unique()
```

Out[39]:
```
array(['Strip Store', 'Power Strip', 'Outlet Strip', 'NOT APPLICAB
LE',
       'Regional Mall', 'Lifestyle Center', 'Mega Outlet Mall',
       'Outlet Mall', 'Tourist Outlet Mall', 'Downtown Store',
       'Tourist Outlet Strip', 'Freestanding Store', 'Mini Mall'],
      dtype=object)
```

In [40]:
```
# Count of STORES for TYPE
store['STORE_TYPE'].value_counts()
```

Out[40]:
```
Strip Store              480
Power Strip              345
Outlet Strip             213
Regional Mall             62
NOT APPLICABLE            51
Outlet Mall               39
Downtown Store            34
Mega Outlet Mall          30
Mini Mall                 15
Tourist Outlet Strip      15
Lifestyle Center          12
Freestanding Store         4
Tourist Outlet Mall        2
Name: STORE_TYPE, dtype: int64
```

In [41]:
```
# Let's replace again all the missing values in 'STORE_SIZE' with t
he mean size depending on the belonging 'STORE_TYPE' group
store['STORE_SIZE']=store['STORE_SIZE'].fillna(store.groupby('STORE
_TYPE')['STORE_SIZE'].transform('mean'))
```

In [42]:
```
# Now we don't have any missing value anymore, as we did some imput
ation.
store.isnull().sum()
```

Out[42]:
```
LOC_IDNT      0
CITY          0
STATE         0
STORE_TYPE    0
POSTAL_CD     0
STORE_SIZE    0
dtype: int64
```

In [43]:
```
# Unique values of STATE
store['STATE'].unique()
```

Out[43]:
```
array(['MO', 'NY', 'FL', 'CA', 'NH', 'MA', 'WV', 'IL', 'MD', 'NJ',
'AZ',
       'TX', 'IA', 'CT', 'KY', 'LA', 'TN', 'GA', 'OK', 'MI', 'WI',
'OH',
       'SC', 'KS', 'VA', 'PA', 'MN', 'DC', 'NV', 'IN', 'DE', 'AL',
'CO',
       'AR', 'NC', 'UT', 'RI', 'WY', 'MS', 'ND', 'WA', 'MT', 'ID',
'ME',
       'VT', 'OR', 'NM', 'NE', 'SD'], dtype=object)
```

In [44]:
```
# Count of ROWS per STATES
store['STATE'].value_counts()
```

Out[44]:
```
NY    242
CA     89
TX     74
NJ     60
PA     58
FL     46
IL     43
MO     41
MA     41
OH     39
NC     39
MI     39
VA     36
MD     32
CT     31
IN     26
CO     26
GA     25
TN     22
MN     19
AZ     19
SC     18
WI     18
LA     17
WA     15
KY     14
AL     13
OR     12
KS     12
IA     12
NH     12
OK     12
AR     11
NE     10
UT     10
MS     10
DE      9
NV      8
ME      7
WV      6
RI      5
ID      5
DC      5
ND      4
MT      3
SD      2
VT      2
WY      2
NM      1
Name: STATE, dtype: int64
```

store.to_csv(r'\\10.0.7.226\ipba_group10\store_new.csv')

Transaction Dataset - EDA

In [33]:
```
transaction = pd.read_csv(r'\\10.0.7.226\ipba_group10\transaction_dataset.csv')
```

In [34]:
```
transaction.head()
```

Out[34]:

	DAY_DT	LOC_INDT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES	TOTAL_
0	2015-09-26	1218	466896.0	0	NFP	16.80	
1	2015-08-02	1218	412445.0	0	NFP	29.99	
2	2015-10-21	1218	491738.0	0	FP	44.00	
3	2015-08-02	1218	414979.0	0	NFP	24.00	
4	2015-07-26	1218	458372.0	0	FP	48.00	

In [47]:
```
# TRANSACTION dataset number of rows (13053149) and columns (9)
transaction.shape
```

Out[47]:
```
(8862952, 9)
```

In [48]:
```
# TRANSACTION dataset Dtypes check
transaction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8862952 entries, 0 to 8862951
Data columns (total 9 columns):
DAY_DT             object
LOC_INDT           int64
DBSKU              float64
ONLINE_FLAG        float64
FULL_PRICE_IND     object
TOTAL_SALES        float64
TOTAL_UNITS        float64
TOTAL_SALES_PRFT   float64
TOTAL_COST         float64
dtypes: float64(6), int64(1), object(2)
memory usage: 608.6+ MB
```

In [49]:
```
# NFP = Loss    |   FP = Profit
transaction['FULL_PRICE_IND'].value_counts(dropna = False)
```

Out[49]:
```
NFP    6336768
FP     2526183
NaN          1
Name: FULL_PRICE_IND, dtype: int64
```

In [50]:
```
# TRANSACTION dataset's missing values
transaction.isnull().sum()
```

Out[50]:
```
DAY_DT                0
LOC_INDT              0
DBSKU               770
ONLINE_FLAG           1
FULL_PRICE_IND        1
TOTAL_SALES           1
TOTAL_UNITS           1
TOTAL_SALES_PRFT      1
TOTAL_COST            1
dtype: int64
```

In [51]:
```
# Let's fill DBSKU NaNs with "1.0"
transaction['DBSKU'].fillna(1.0, inplace = True)
```

In [52]:
```
# Let's take a look at NaNs value again. There is none.
transaction.isnull().sum()
```

Out[52]:
```
DAY_DT              0
LOC_INDT            0
DBSKU               0
ONLINE_FLAG         1
FULL_PRICE_IND      1
TOTAL_SALES         1
TOTAL_UNITS         1
TOTAL_SALES_PRFT    1
TOTAL_COST          1
dtype: int64
```

In [53]:
```
# DBSKU is not longer a float but an object
transaction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8862952 entries, 0 to 8862951
Data columns (total 9 columns):
DAY_DT               object
LOC_INDT             int64
DBSKU                float64
ONLINE_FLAG          float64
FULL_PRICE_IND       object
TOTAL_SALES          float64
TOTAL_UNITS          float64
TOTAL_SALES_PRFT     float64
TOTAL_COST           float64
dtypes: float64(6), int64(1), object(2)
memory usage: 608.6+ MB
```

In [54]:
```
# Let's see how many 0s are within the columns of interest, includi
ng TOTAL_SALES (our target variable)
transaction[(transaction['TOTAL_SALES']==0.0) & (transaction['TOTAL
_SALES_PRFT']==0.0) & (transaction['TOTAL_COST']==0.0)].count()
```

Out[54]:
```
DAY_DT               3006
LOC_INDT             3006
DBSKU                3006
ONLINE_FLAG          3006
FULL_PRICE_IND       3006
TOTAL_SALES          3006
TOTAL_UNITS          3006
TOTAL_SALES_PRFT     3006
TOTAL_COST           3006
dtype: int64
```

transaction.to_csv(r'C:\IPBAB047\transaction_new.csv')

Merged Datasets - EDA

In [55]:
```
# Merging dataset TRANSACTION on dataset PRODUCT using 'DBSKU' as a key, and performing an INNER join
prod_trans = pd.merge(transaction,
                     product[['DBSKU','DEPARTMENT','CLASS','SUBCLASS','DEPARTMENT_NAME','CLASS_NAME','SUBCLASS_NAME']],
                     on = 'DBSKU')
prod_trans.head()
```

Out[55]:

	DAY_DT	LOC_INDT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES	TOTAL_
0	2015-09-26	1218	466896.0	0.0	NFP	16.80	
1	2015-08-31	1218	466896.0	0.0	NFP	33.60	
2	2015-10-12	115	466896.0	0.0	NFP	13.44	
3	2015-08-29	728	466896.0	0.0	NFP	11.76	
4	2015-07-29	1070	466896.0	0.0	NFP	29.99	

In [56]:
```
# Let's see the number of rows (15070031) and columns (15) that the dataset (PRODUCT & TRANSACTION) holds
prod_trans.shape
```

Out[56]: (10697594, 15)

In [57]:
```
# As noticed above, we have more rows than the total rows of the longer dataset (TRANSACTION), let's delete duplicates
prod_trans = prod_trans.drop_duplicates(subset=None, keep='first')
```

In [58]:
```
# The shape is still larger than original, but much better looking now
prod_trans.shape
```

Out[58]: (8869496, 15)

In [59]:
```
prod_trans['DBSKU'].unique()
```

Out[59]: array([466896., 412445., 491738., ..., 649509., 121723., 618983.])

In [60]:
```
prod_trans.describe()
```

Out[60]:

	LOC_INDT	DBSKU	ONLINE_FLAG	TOTAL_SALES	TOTAL_UNITS	TOTAL_S
count	8.869496e+06	8.869496e+06	8.869496e+06	8.869496e+06	8.869496e+06	8
mean	7.840659e+02	1.080168e+06	2.174103e-02	4.307143e+01	1.124624e+00	2
std	6.912062e+02	8.100892e+05	1.458368e-01	4.088681e+01	9.915852e-01	2
min	2.000000e+00	1.000000e+00	0.000000e+00	-2.128000e+02	-5.000000e+00	-4
25%	3.070000e+02	5.241080e+05	0.000000e+00	2.999000e+01	1.000000e+00	1
50%	6.640000e+02	5.714300e+05	0.000000e+00	3.950000e+01	1.000000e+00	2
75%	1.156000e+03	2.116525e+06	0.000000e+00	4.822000e+01	1.000000e+00	3
max	4.150000e+03	2.999987e+06	1.000000e+00	7.260450e+03	1.610000e+02	4

In [61]:
```
# Let's check for NaNs...everything is clean
prod_trans.isnull().sum()
```

Out[61]:
```
DAY_DT              0
LOC_INDT            0
DBSKU               0
ONLINE_FLAG         0
FULL_PRICE_IND      0
TOTAL_SALES         0
TOTAL_UNITS         0
TOTAL_SALES_PRFT    0
TOTAL_COST          0
DEPARTMENT          0
CLASS               0
SUBCLASS            0
DEPARTMENT_NAME     0
CLASS_NAME          0
SUBCLASS_NAME       0
dtype: int64
```

In [62]:
```
# To be able to proceed to the next dataset merging we need to matc
h the characthers of the KEY columns
prod_trans.rename(columns = {"LOC_INDT":"LOC_IDNT"}, inplace = Tru
e)
```

In [63]:
```
# Loc name has been correctly changed
prod_trans.head(1)
```

Out[63]:

	DAY_DT	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES	TOTAL_
0	2015-09-26	1218	466896.0	0.0	NFP	16.8	

In [64]:
```
# Let's perform the final merging so to have all of the 3 datasets merged in 1 (PTS)
PTS = pd.merge(prod_trans,
                     store[['LOC_IDNT','CITY','STATE','STORE_TYPE','POSTAL_CD','STORE_SIZE']],
                     on = 'LOC_IDNT')
PTS.head()
```

Out[64]:

	DAY_DT	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES	TOTAL_
0	2015-09-26	1218	466896.0	0.0	NFP	16.80	
1	2015-08-31	1218	466896.0	0.0	NFP	33.60	
2	2015-09-08	1218	466896.0	0.0	NFP	21.00	
3	2015-08-21	1218	466896.0	0.0	NFP	14.70	
4	2015-08-02	1218	412445.0	0.0	NFP	29.99	

In [65]:
```
# Let's check the shape of the new formed DATASET
PTS.shape
```

Out[65]: (8869496, 20)

In [66]:
```
# We run again a code to eliminate eventual duplicates from the DATASET
PTS = PTS.drop_duplicates(subset=None, keep='first')
```

In [67]:
```
# The PTS dataset is clean. No NaNs
PTS.isnull().sum()
```

Out[67]:
```
DAY_DT             0
LOC_IDNT           0
DBSKU              0
ONLINE_FLAG        0
FULL_PRICE_IND     0
TOTAL_SALES        0
TOTAL_UNITS        0
TOTAL_SALES_PRFT   0
TOTAL_COST         0
DEPARTMENT         0
CLASS              0
SUBCLASS           0
DEPARTMENT_NAME    0
CLASS_NAME         0
SUBCLASS_NAME      0
CITY               0
STATE              0
STORE_TYPE         0
POSTAL_CD          0
STORE_SIZE         0
dtype: int64
```

In [68]:
```
# The unknown DBSKU, present in the PTS as 'NOT APPLICABLE' represe
nt the 0.06% oth the whole dataset
PTS.loc[PTS['DBSKU']==1.0].count() / PTS['DBSKU'].shape[0]
```

Out[68]:
```
DAY_DT             0.000825
LOC_IDNT           0.000825
DBSKU              0.000825
ONLINE_FLAG        0.000825
FULL_PRICE_IND     0.000825
TOTAL_SALES        0.000825
TOTAL_UNITS        0.000825
TOTAL_SALES_PRFT   0.000825
TOTAL_COST         0.000825
DEPARTMENT         0.000825
CLASS              0.000825
SUBCLASS           0.000825
DEPARTMENT_NAME    0.000825
CLASS_NAME         0.000825
SUBCLASS_NAME      0.000825
CITY               0.000825
STATE              0.000825
STORE_TYPE         0.000825
POSTAL_CD          0.000825
STORE_SIZE         0.000825
dtype: float64
```

In [69]:
```
# The unknown DBSKU, present in the PTS as 'NOT APPLICABLE' represe
nt the 2.5% oth the whole dataset
PTS.loc[PTS['STORE_TYPE']=='NOT APPLICABLE'].count() / PTS['STORE_T
YPE'].shape[0]
```

Out[69]:
```
DAY_DT              0.021741
LOC_IDNT            0.021741
DBSKU               0.021741
ONLINE_FLAG         0.021741
FULL_PRICE_IND      0.021741
TOTAL_SALES         0.021741
TOTAL_UNITS         0.021741
TOTAL_SALES_PRFT    0.021741
TOTAL_COST          0.021741
DEPARTMENT          0.021741
CLASS               0.021741
SUBCLASS            0.021741
DEPARTMENT_NAME     0.021741
CLASS_NAME          0.021741
SUBCLASS_NAME       0.021741
CITY                0.021741
STATE               0.021741
STORE_TYPE          0.021741
POSTAL_CD           0.021741
STORE_SIZE          0.021741
dtype: float64
```

In [70]:
```
## While exploring the transaction dataset we have checked the target variable "TOTAL_SALES" and noticed that often the value is 0
# as is the related cost and profit. Our target variable cannot have values = 0, furthermore these specific rows give us no information.
# I will drop them in full.
# Let's see how many 0s are within the columns of interest, including TOTAL_SALES (our target variable)
PTS[(PTS['TOTAL_SALES']==0.0) & (PTS['TOTAL_SALES_PRFT']==0.0) & (PTS['TOTAL_COST']==0.0)].count()
```

Out[70]:
```
DAY_DT              3040
LOC_IDNT            3040
DBSKU               3040
ONLINE_FLAG         3040
FULL_PRICE_IND      3040
TOTAL_SALES         3040
TOTAL_UNITS         3040
TOTAL_SALES_PRFT    3040
TOTAL_COST          3040
DEPARTMENT          3040
CLASS               3040
SUBCLASS            3040
DEPARTMENT_NAME     3040
CLASS_NAME          3040
SUBCLASS_NAME       3040
CITY                3040
STATE               3040
STORE_TYPE          3040
POSTAL_CD           3040
STORE_SIZE          3040
dtype: int64
```

In [71]:
```
# Let's proceed and drop the rows where TOTAL_SALES, TOTAL_SALES_PRFT, and TOTAL_COST + 0
PTS1 = PTS.drop(PTS[(PTS.TOTAL_SALES == 0.0) & (PTS.TOTAL_SALES_PRFT == 0.0) & (PTS.TOTAL_COST == 0.0)].index)
```

In [72]:
```
# I want to check how many TOTAL_SALES (price * total units) = 0 be
fore we drop those who had a correspondent value = 0
# for TOTAL_SALES_PRFT, and TOTAL_COST-----7789
PTS[(PTS['TOTAL_SALES']==0.0)].count()
```

Out[72]:
```
DAY_DT              4687
LOC_IDNT            4687
DBSKU               4687
ONLINE_FLAG         4687
FULL_PRICE_IND      4687
TOTAL_SALES         4687
TOTAL_UNITS         4687
TOTAL_SALES_PRFT    4687
TOTAL_COST          4687
DEPARTMENT          4687
CLASS               4687
SUBCLASS            4687
DEPARTMENT_NAME     4687
CLASS_NAME          4687
SUBCLASS_NAME       4687
CITY                4687
STATE               4687
STORE_TYPE          4687
POSTAL_CD           4687
STORE_SIZE          4687
dtype: int64
```

In [73]:
```
# To be sure that all values TOTAL_SALES = 0 are not the result of
wrongful data entry, I will simply add cost to total_sales_prft
PTS['TOTAL_SALES'] = PTS['TOTAL_COST'] + PTS['TOTAL_SALES_PRFT']
```

In [74]:
```
# In the dataset where I have dropped sales,profit,cost = 0 , I wil
l check how many sales only = 0 ------ 3259
PTS1[(PTS1['TOTAL_SALES']==0.0)].count()
```

Out[74]:
```
DAY_DT              1647
LOC_IDNT            1647
DBSKU               1647
ONLINE_FLAG         1647
FULL_PRICE_IND      1647
TOTAL_SALES         1647
TOTAL_UNITS         1647
TOTAL_SALES_PRFT    1647
TOTAL_COST          1647
DEPARTMENT          1647
CLASS               1647
SUBCLASS            1647
DEPARTMENT_NAME     1647
CLASS_NAME          1647
SUBCLASS_NAME       1647
CITY                1647
STATE               1647
STORE_TYPE          1647
POSTAL_CD           1647
STORE_SIZE          1647
dtype: int64
```

In [75]:
```
# We can see here how the rows look when total_sales = 0
# We basically have a cost per transaction, and a loss in profit fo
r the same amount
# This could mean that these transaction are the result of purchase
s done by promo cards.
PTS1[PTS1.TOTAL_SALES == 0.0]
```

Out[75]:

	DAY_DT	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES
1661	2015-11-02	1218	451781.0	0.0	NFP	0.0
2447	2015-08-15	1218	436816.0	0.0	NFP	0.0
19550	2016-10-29	115	532440.0	0.0	NFP	0.0
42338	2016-11-05	1070	2129247.0	0.0	NFP	0.0
43900	2017-02-22	1070	570481.0	0.0	NFP	0.0
53476	2017-03-31	1216	546119.0	0.0	NFP	0.0
59585	2016-06-24	188	501056.0	0.0	NFP	0.0
	2016-					

73014	11-27	9	580282.0	0.0	NFP	0.0
83837	2017-02-02	1246	2139212.0	0.0	NFP	0.0
84066	2017-06-30	1246	594747.0	0.0	NFP	0.0
86774	2016-05-27	1153	2936708.0	0.0	NFP	0.0
88046	2015-10-23	1153	2999367.0	0.0	NFP	0.0
100682	2016-06-22	591	549923.0	0.0	FP	0.0
102394	2017-01-27	591	2133355.0	0.0	NFP	0.0
104462	2016-05-30	661	496315.0	0.0	NFP	0.0
105743	2016-01-02	661	2980813.0	0.0	NFP	0.0
110638	2017-02-19	661	2138446.0	0.0	NFP	0.0
131942	2016-07-28	491	2125831.0	0.0	NFP	0.0
133917	2016-04-15	491	2126201.0	0.0	NFP	0.0
137434	2017-02-04	491	2133074.0	0.0	NFP	0.0
139054	2017-02-04	491	2146050.0	0.0	NFP	0.0
149103	2017-03-08	1341	567750.0	0.0	NFP	0.0
155773	2015-08-24	254	2962365.0	0.0	NFP	0.0
170254	2015-12-21	1244	489831.0	0.0	NFP	0.0
189222	2016-06-11	615	326462.0	0.0	NFP	0.0
207962	2016-10-30	1203	543462.0	0.0	NFP	0.0
209951	2016-05-29	1277	501536.0	0.0	NFP	0.0
212633	2016-10-25	1277	2129577.0	0.0	NFP	0.0
213413	2017-02-07	1277	2139618.0	0.0	NFP	0.0
213585	2017-02-27	1277	584813.0	0.0	NFP	0.0

219285	2016-11-04	660	533224.0	0.0	NFP	0.0
220552	2016-09-11	660	2128876.0	0.0	NFP	0.0
223798	2015-11-04	1085	2998971.0	0.0	NFP	0.0
231597	2017-03-16	1079	2139212.0	0.0	NFP	0.0
235174	2016-12-23	1024	447409.0	0.0	NFP	0.0
267271	2016-03-13	1278	480236.0	0.0	NFP	0.0
285557	2015-08-22	375	455618.0	0.0	NFP	0.0
285762	2015-09-11	375	450163.0	0.0	NFP	0.0
285763	2015-09-14	375	450163.0	0.0	NFP	0.0
324157	2016-07-29	1269	2129619.0	0.0	NFP	0.0
358604	2016-05-01	71	512814.0	0.0	NFP	0.0
364955	2017-05-22	71	595488.0	0.0	NFP	0.0
368645	2016-02-20	258	501056.0	0.0	NFP	0.0
369147	2015-11-27	258	2992842.0	0.0	NFP	0.0
373860	2016-11-25	258	532424.0	0.0	NFP	0.0
380349	2016-05-13	1282	554410.0	0.0	NFP	0.0
406128	2016-05-07	737	506188.0	0.0	NFP	0.0
410277	2017-01-26	737	580282.0	0.0	NFP	0.0
426217	2017-07-29	400	618868.0	0.0	NFP	0.0
448368	2017-01-27	564	2146118.0	0.0	NFP	0.0
455789	2016-06-28	1038	538447.0	0.0	NFP	0.0
458612	2015-07-28	573	2998971.0	0.0	NFP	0.0
488915	2016-11-17	1243	2138453.0	0.0	NFP	0.0

489210	2017-03-01	1243	584789.0	0.0	NFP	0.0
506260	2016-05-24	260	491647.0	0.0	NFP	0.0
509169	2016-11-12	260	533935.0	0.0	NFP	0.0
515444	2015-12-29	1093	818823.0	0.0	NFP	0.0
554779	2017-02-18	655	580910.0	0.0	NFP	0.0
585167	2015-08-20	678	451070.0	0.0	NFP	0.0
587860	2016-10-24	678	539015.0	0.0	NFP	0.0
600662	2016-10-29	102	540674.0	0.0	NFP	0.0
607792	2015-09-21	713	329508.0	0.0	NFP	0.0
617534	2015-08-24	494	837997.0	0.0	NFP	0.0
619073	2016-12-22	494	543439.0	0.0	NFP	0.0
621973	2015-12-01	1219	502849.0	0.0	NFP	0.0
657127	2016-12-10	365	531889.0	0.0	NFP	0.0
664858	2017-07-07	365	591818.0	0.0	NFP	0.0
666749	2017-07-26	365	613315.0	0.0	NFP	0.0
667114	2015-09-03	652	2989426.0	0.0	NFP	0.0
668470	2016-05-04	652	501056.0	0.0	NFP	0.0
668778	2015-11-21	652	2984146.0	0.0	NFP	0.0
677424	2016-05-10	586	2104661.0	0.0	NFP	0.0
678552	2017-02-04	586	451724.0	0.0	NFP	0.0
682366	2017-02-04	586	592451.0	0.0	NFP	0.0
685377	2016-05-14	1175	249698.0	0.0	NFP	0.0
	2016-					

699291	08-20	1213	519397.0	0.0	NFP	0.0
711011	2016-05-16	1288	2124420.0	0.0	NFP	0.0
714341	2017-03-15	1288	580571.0	0.0	NFP	0.0
735505	2016-01-05	1148	2999987.0	0.0	NFP	0.0
740565	2016-07-15	1148	540146.0	0.0	NFP	0.0
747649	2015-08-21	195	472308.0	0.0	NFP	0.0
749517	2016-04-11	195	2101238.0	0.0	NFP	0.0
789485	2015-12-30	1166	2110452.0	0.0	NFP	0.0
806935	2016-02-18	1275	458117.0	0.0	NFP	0.0
807596	2015-11-14	1275	404020.0	0.0	NFP	0.0
814281	2017-02-23	1275	2142794.0	0.0	NFP	0.0
826959	2016-05-23	1028	2101238.0	0.0	NFP	0.0
827884	2016-05-09	1028	540278.0	0.0	NFP	0.0
830957	2017-02-15	1028	2133660.0	0.0	NFP	0.0
832687	2017-01-19	1028	581413.0	0.0	NFP	0.0
834647	2017-07-08	1028	2155051.0	0.0	NFP	0.0
846125	2017-05-23	329	594549.0	0.0	NFP	0.0
847623	2017-05-08	329	590992.0	0.0	NFP	0.0
851259	2016-11-05	356	519397.0	0.0	NFP	0.0
858041	2016-06-03	1064	491647.0	0.0	NFP	0.0
886787	2016-05-23	322	457119.0	0.0	NFP	0.0
893035	2016-04-08	20	278176.0	0.0	NFP	0.0
894967	2015-08-29	20	2991596.0	0.0	NFP	0.0

895536	2015-08-06	20	292144.0	0.0	NFP	0.0
895705	2015-12-21	20	493023.0	0.0	NFP	0.0
895719	2016-12-30	20	2105452.0	0.0	NFP	0.0
898200	2016-10-26	20	532549.0	0.0	NFP	0.0
901623	2016-10-31	20	540674.0	0.0	NFP	0.0
905734	2017-07-18	20	2150110.0	0.0	NFP	0.0
906471	2017-07-29	20	2152702.0	0.0	NFP	0.0
907382	2017-07-28	20	2154831.0	0.0	NFP	0.0
913496	2016-08-20	1025	538645.0	0.0	NFP	0.0
930326	2016-01-09	1327	2109017.0	0.0	NFP	0.0
940480	2017-06-30	1327	2151886.0	0.0	NFP	0.0
945317	2016-10-21	656	542902.0	0.0	NFP	0.0
946094	2016-07-16	656	2124859.0	0.0	NFP	0.0
981027	2017-07-27	1154	612192.0	0.0	NFP	0.0
1003114	2016-06-17	1080	2125260.0	0.0	NFP	0.0
1029518	2017-05-23	1323	600866.0	0.0	NFP	0.0
1043339	2017-02-24	1230	557058.0	0.0	NFP	0.0
1062433	2016-04-22	480	2116236.0	0.0	NFP	0.0
1064968	2016-11-02	480	533166.0	0.0	NFP	0.0
1114863	2016-04-14	851	509273.0	0.0	NFP	0.0
1118459	2017-06-07	851	593814.0	0.0	NFP	0.0
1121340	2016-05-21	908	484238.0	0.0	NFP	0.0
1125588	2016-06-23	908	506188.0	0.0	NFP	0.0

1126903	2017-01-10	908	500553.0	0.0	NFP	0.0
1127676	2017-01-10	908	509265.0	0.0	NFP	0.0
1139252	2016-10-24	1298	546077.0	0.0	NFP	0.0
1139879	2016-07-10	1298	543462.0	0.0	NFP	0.0
1178012	2017-05-13	199	2103440.0	0.0	NFP	0.0
1180413	2016-12-28	199	2129742.0	0.0	NFP	0.0
1185151	2015-09-05	1302	436840.0	0.0	NFP	0.0
1194601	2015-08-11	420	423350.0	0.0	NFP	0.0
1211023	2016-02-15	674	2109652.0	0.0	NFP	0.0
1211543	2015-11-10	674	2991562.0	0.0	NFP	0.0
1256339	2016-05-31	1151	499038.0	0.0	NFP	0.0
1259154	2016-08-16	1151	553149.0	0.0	NFP	0.0
1293503	2016-05-30	760	458828.0	0.0	NFP	0.0
1301120	2016-02-07	143	2110452.0	0.0	NFP	0.0
1301145	2016-04-22	143	472134.0	0.0	NFP	0.0
1301392	2015-11-29	143	445841.0	0.0	NFP	0.0
1301437	2016-02-16	143	479428.0	0.0	NFP	0.0
1301801	2015-11-29	143	441840.0	0.0	NFP	0.0
1321588	2015-08-24	580	441816.0	0.0	NFP	0.0
1324338	2015-11-28	580	412593.0	0.0	NFP	0.0
1350431	2017-06-18	259	613091.0	0.0	NFP	0.0
1380234	2016-12-29	1163	2126557.0	0.0	NFP	0.0
	2017-					

1386027	03-10	1163	600882.0	0.0	NFP	0.0
1418104	2015-08-27	221	421644.0	0.0	NFP	0.0
1423042	2017-02-19	221	554287.0	0.0	NFP	0.0
1429385	2016-11-15	575	524629.0	0.0	NFP	0.0
1440162	2016-08-11	570	521666.0	0.0	NFP	0.0
1440231	2016-08-27	570	525485.0	0.0	NFP	0.0
1441601	2016-11-05	570	542399.0	0.0	NFP	0.0
1442128	2016-08-09	570	531772.0	0.0	NFP	0.0
1451102	2016-07-06	1012	531673.0	0.0	NFP	0.0
1458053	2015-10-31	149	428672.0	0.0	NFP	0.0
1461235	2016-11-07	149	2130328.0	0.0	NFP	0.0
1485993	2016-07-23	496	531210.0	0.0	NFP	0.0
1492179	2017-03-25	496	551176.0	0.0	NFP	0.0
1496520	2017-06-27	496	2155051.0	0.0	NFP	0.0
1507209	2015-11-28	1211	2104265.0	0.0	NFP	0.0
1516336	2015-09-05	1050	404020.0	0.0	NFP	0.0
1517509	2015-12-21	1050	476523.0	0.0	NFP	0.0
1520882	2016-10-22	1050	534057.0	0.0	NFP	0.0
1524186	2017-06-07	1050	583997.0	0.0	NFP	0.0
1527267	2017-06-18	1050	607168.0	0.0	NFP	0.0
1527666	2017-05-03	1050	594101.0	0.0	NFP	0.0
1546231	2017-03-02	8	594747.0	0.0	NFP	0.0
1561473	2017-01-25	227	572719.0	0.0	NFP	0.0

1575859	2017-02-06	629	2127001.0	0.0	NFP	0.0
1584692	2016-12-09	1081	532481.0	0.0	NFP	0.0
1585907	2017-02-09	1081	2139618.0	0.0	NFP	0.0
1592292	2017-07-27	1223	618918.0	0.0	NFP	0.0
1596628	2016-01-31	34	2114447.0	0.0	NFP	0.0
1596651	2016-05-20	34	485102.0	0.0	NFP	0.0
1600678	2015-08-21	34	411165.0	0.0	NFP	0.0
1614863	2016-11-03	34	554287.0	0.0	NFP	0.0
1615418	2017-03-19	34	570341.0	0.0	NFP	0.0
1616942	2017-01-31	34	580381.0	0.0	NFP	0.0
1637514	2017-06-25	51	2147892.0	0.0	NFP	0.0
1656501	2016-11-08	287	531996.0	0.0	NFP	0.0
1659007	2016-07-29	287	538694.0	0.0	NFP	0.0
1670854	2017-02-19	287	2149948.0	0.0	NFP	0.0
1687553	2016-11-05	1138	458042.0	0.0	NFP	0.0
1703832	2017-02-14	606	581413.0	0.0	NFP	0.0
1704250	2017-02-14	606	583427.0	0.0	NFP	0.0
1708089	2017-05-05	606	605535.0	0.0	NFP	0.0
1715117	2016-12-19	619	2127134.0	0.0	NFP	0.0
1718307	2016-06-29	619	551051.0	0.0	NFP	0.0
1749416	2016-02-21	145	503151.0	0.0	NFP	0.0
1750497	2016-08-02	145	2124693.0	0.0	NFP	0.0
1755973	2016-08-23	145	2130195.0	0.0	NFP	0.0

1757434	2017-07-24	145	2143313.0	0.0	NFP	0.0
1762343	2016-05-24	372	2116418.0	0.0	NFP	0.0
1799486	2017-06-17	736	2147918.0	0.0	NFP	0.0
1802255	2016-03-01	1052	2112151.0	0.0	NFP	0.0
1802297	2016-05-15	1052	470963.0	0.0	NFP	0.0
1807093	2017-07-29	1052	609552.0	0.0	NFP	0.0
1809000	2015-10-24	533	2102772.0	0.0	NFP	0.0
1834618	2016-11-10	1251	531301.0	0.0	NFP	0.0
1836992	2017-02-16	1251	576538.0	0.0	NFP	0.0
1848344	2016-06-24	1003	2125260.0	0.0	NFP	0.0
1860280	2017-03-17	56	515619.0	0.0	NFP	0.0
1862821	2016-11-18	56	570101.0	0.0	NFP	0.0
1867230	2016-08-29	314	2973156.0	0.0	NFP	0.0
1883794	2016-05-21	144	2121491.0	0.0	NFP	0.0
1894070	2017-07-04	144	593830.0	0.0	NFP	0.0
1906398	2016-05-16	799	480434.0	0.0	NFP	0.0
1907724	2016-01-14	799	502559.0	0.0	NFP	0.0
1938556	2016-04-24	1139	2109645.0	0.0	NFP	0.0
1968866	2016-04-28	507	532747.0	0.0	NFP	0.0
1979972	2017-07-23	507	2152983.0	0.0	NFP	0.0
1988374	2017-04-01	430	569731.0	0.0	NFP	0.0
1988624	2017-01-24	430	584714.0	0.0	NFP	0.0
1993957	2015-10-31	686	463232.0	0.0	NFP	0.0

2000950	2017-01-25	686	570002.0	0.0	NFP	0.0
2006743	2015-12-15	915	427682.0	0.0	NFP	0.0
2007281	2015-08-16	915	457895.0	0.0	NFP	0.0
2014579	2016-06-16	915	551101.0	0.0	NFP	0.0
2025818	2016-02-17	1015	485003.0	0.0	NFP	0.0
2026947	2015-12-05	1015	455774.0	0.0	NFP	0.0
2027325	2016-05-07	1015	499145.0	0.0	NFP	0.0
2055913	2016-02-04	1293	477125.0	0.0	NFP	0.0
2067294	2016-07-08	873	519405.0	0.0	NFP	0.0
2088076	2016-03-15	1005	2101238.0	0.0	NFP	0.0
2096763	2017-03-11	1005	2149203.0	0.0	NFP	0.0
2107529	2015-08-10	391	387910.0	0.0	NFP	0.0
2112735	2017-03-26	1179	551838.0	0.0	NFP	0.0
2117545	2015-08-08	1193	453944.0	0.0	NFP	0.0
2125874	2016-10-19	1193	575647.0	0.0	NFP	0.0
2133165	2015-10-31	1127	2978163.0	0.0	NFP	0.0
2150426	2017-02-18	648	2132613.0	0.0	NFP	0.0
2162231	2016-12-27	1292	545244.0	0.0	NFP	0.0
2167740	2017-07-27	1292	621482.0	0.0	NFP	0.0
2169615	2016-01-09	732	486324.0	0.0	NFP	0.0
2175155	2016-07-02	1173	539049.0	0.0	NFP	0.0
2186161	2016-06-22	1204	2128421.0	0.0	FP	0.0
	2015-					

2189974	10-18	164	2998179.0	0.0	NFP	0.0
2192112	2016-01-11	164	2109652.0	0.0	NFP	0.0
2210976	2016-12-03	29	557462.0	0.0	NFP	0.0
2223855	2017-03-18	84	551879.0	0.0	NFP	0.0
2233214	2016-10-23	1108	2128793.0	0.0	NFP	0.0
2250182	2017-02-20	253	2143982.0	0.0	NFP	0.0
2255854	2016-01-14	166	458075.0	0.0	NFP	0.0
2284327	2015-09-08	217	2997262.0	0.0	NFP	0.0
2289648	2017-03-22	217	551176.0	0.0	NFP	0.0
2292962	2015-09-13	783	454199.0	0.0	NFP	0.0
2312728	2017-02-21	1324	2132084.0	0.0	NFP	0.0
2334121	2017-06-17	653	600858.0	0.0	NFP	0.0
2365268	2015-09-14	408	2999987.0	0.0	NFP	0.0
2365872	2015-11-27	408	2100594.0	0.0	NFP	0.0
2371793	2016-06-08	726	504621.0	0.0	NFP	0.0
2389347	2017-01-21	1057	583427.0	0.0	NFP	0.0
2395305	2016-06-22	641	546101.0	0.0	NFP	0.0
2396640	2017-02-15	641	2136960.0	0.0	NFP	0.0
2398000	2017-03-30	641	2150011.0	0.0	NFP	0.0
2400472	2016-05-19	1124	490771.0	0.0	NFP	0.0
2403563	2015-08-13	1124	421644.0	0.0	NFP	0.0
2408468	2017-02-18	1124	551176.0	0.0	NFP	0.0
2415036	2016-05-06	167	491647.0	0.0	NFP	0.0

2421349	2017-06-21	167	2150110.0	0.0	NFP	0.0
2429205	2015-10-25	86	447417.0	0.0	NFP	0.0
2429488	2015-09-06	86	436386.0	0.0	NFP	0.0
2459185	2016-05-13	73	501536.0	0.0	NFP	0.0
2463423	2016-08-24	73	537746.0	0.0	NFP	0.0
2463531	2016-10-31	73	545970.0	0.0	NFP	0.0
2464684	2017-01-16	73	569715.0	0.0	NFP	0.0
2481078	2015-10-29	554	459727.0	0.0	NFP	0.0
2492764	2017-07-27	554	594531.0	0.0	NFP	0.0
2502617	2016-05-29	771	542001.0	0.0	NFP	0.0
2521599	2017-06-12	594	2154674.0	0.0	NFP	0.0
2524248	2016-01-08	1294	471003.0	0.0	NFP	0.0
2526082	2016-05-17	1294	2109645.0	0.0	NFP	0.0
2527020	2016-07-20	1294	513895.0	0.0	NFP	0.0
2534091	2017-02-17	1294	570481.0	0.0	NFP	0.0
2540173	2015-09-11	1066	444315.0	0.0	NFP	0.0
2541495	2016-05-02	1066	501056.0	0.0	NFP	0.0
2541724	2016-02-29	1066	473983.0	0.0	NFP	0.0
2577424	2016-12-24	581	533497.0	0.0	NFP	0.0
2589660	2017-02-20	347	2137752.0	0.0	NFP	0.0
2603326	2016-01-04	401	486563.0	0.0	NFP	0.0
2608731	2016-09-10	401	538843.0	0.0	NFP	0.0
2613981	2017-03-12	401	2145656.0	0.0	NFP	0.0

2632423	2016-10-30	1058	538447.0	0.0	NFP	0.0
2634428	2017-03-04	1058	2142794.0	0.0	NFP	0.0
2637181	2015-09-06	1177	455832.0	0.0	NFP	0.0
2645080	2016-08-17	758	2111716.0	0.0	NFP	0.0
2655612	2017-02-05	758	582395.0	0.0	NFP	0.0
2676987	2015-11-24	396	454025.0	0.0	NFP	0.0
2677413	2015-07-28	396	432724.0	0.0	NFP	0.0
2677500	2016-01-23	396	486613.0	0.0	NFP	0.0
2682510	2017-02-15	396	565192.0	0.0	NFP	0.0
2702169	2017-01-20	1037	597062.0	0.0	NFP	0.0
2705795	2016-05-20	69	472134.0	0.0	NFP	0.0
2707697	2015-07-29	69	2991570.0	0.0	NFP	0.0
2713998	2017-04-18	69	605725.0	0.0	NFP	0.0
2720623	2016-10-23	204	533422.0	0.0	NFP	0.0
2724299	2016-04-23	599	489740.0	0.0	NFP	0.0
2743841	2015-12-05	371	490771.0	0.0	NFP	0.0
2744428	2016-01-10	371	479071.0	0.0	NFP	0.0
2744901	2015-08-05	371	463257.0	0.0	NFP	0.0
2744947	2015-07-26	371	434282.0	0.0	NFP	0.0
2745050	2015-12-26	371	512012.0	0.0	NFP	0.0
2745194	2016-05-08	371	2114454.0	0.0	NFP	0.0
2745986	2015-09-06	371	465021.0	0.0	NFP	0.0
	2016-					

2748491	07-11	371	532457.0	0.0	NFP	0.0
2749831	2016-07-22	371	533562.0	0.0	NFP	0.0
2750275	2016-07-11	371	535880.0	0.0	NFP	0.0
2758726	2016-04-24	16	505545.0	0.0	NFP	0.0
2765521	2017-05-08	16	605725.0	0.0	NFP	0.0
2770784	2016-02-07	160	2115030.0	0.0	NFP	0.0
2788791	2017-02-20	128	547919.0	0.0	NFP	0.0
2792239	2017-07-27	128	606228.0	0.0	NFP	0.0
2798384	2016-08-11	461	542399.0	0.0	NFP	0.0
2813800	2016-06-16	1242	1.0	0.0	NFP	0.0
2813801	2016-06-16	1242	1.0	0.0	NFP	0.0
2813802	2016-06-16	1242	1.0	0.0	NFP	0.0
2813803	2016-06-16	1242	1.0	0.0	NFP	0.0
2813804	2016-06-16	1242	1.0	0.0	NFP	0.0
2813805	2016-06-16	1242	1.0	0.0	NFP	0.0
2813806	2016-06-16	1242	1.0	0.0	NFP	0.0
2813807	2016-06-16	1242	1.0	0.0	NFP	0.0
2813808	2016-06-16	1242	1.0	0.0	NFP	0.0
2813809	2016-06-16	1242	1.0	0.0	NFP	0.0
2813810	2016-06-16	1242	1.0	0.0	NFP	0.0
2813811	2016-06-16	1242	1.0	0.0	NFP	0.0
2813812	2016-06-16	1242	1.0	0.0	NFP	0.0
2813813	2016-06-16	1242	1.0	0.0	NFP	0.0

2813814	2016-06-16	1242	1.0	0.0	NFP	0.0
2813815	2016-06-16	1242	1.0	0.0	NFP	0.0
2813816	2016-06-16	1242	1.0	0.0	NFP	0.0
2813817	2016-06-16	1242	1.0	0.0	NFP	0.0
2813818	2016-06-16	1242	1.0	0.0	NFP	0.0
2833830	2016-11-14	773	2133215.0	0.0	NFP	0.0
2833951	2017-04-08	773	556324.0	0.0	NFP	0.0
2864248	2016-08-12	30	507004.0	0.0	NFP	0.0
2874531	2016-02-05	668	460238.0	0.0	NFP	0.0
2882824	2016-07-29	668	541573.0	0.0	NFP	0.0
2897440	2016-01-20	280	504126.0	0.0	NFP	0.0
2897784	2015-11-19	280	457119.0	0.0	NFP	0.0
2898070	2015-09-24	280	489815.0	0.0	NFP	0.0
2898153	2015-11-17	280	472407.0	0.0	NFP	0.0
2898238	2016-02-06	280	470955.0	0.0	NFP	0.0
2899317	2015-09-24	280	425348.0	0.0	NFP	0.0
2899450	2015-11-04	280	454181.0	0.0	NFP	0.0
2899459	2015-09-13	280	436865.0	0.0	NFP	0.0
2899477	2015-08-20	280	468959.0	0.0	NFP	0.0
2899640	2015-09-24	280	457986.0	0.0	NFP	0.0
2899709	2015-08-20	280	2992255.0	0.0	NFP	0.0
2899918	2016-01-26	280	499681.0	0.0	NFP	0.0
2900046	2015-08-12	280	361360.0	0.0	NFP	0.0

2900138	2015-08-12	280	348276.0	0.0	NFP	0.0
2913830	2016-11-08	1171	570101.0	0.0	NFP	0.0
2916206	2015-09-06	1169	237263.0	0.0	NFP	0.0
2922514	2016-01-17	134	818823.0	0.0	NFP	0.0
2923172	2015-12-15	134	489989.0	0.0	NFP	0.0
2923728	2015-08-26	134	454165.0	0.0	NFP	0.0
2924175	2016-04-05	134	480343.0	0.0	NFP	0.0
2924509	2015-08-03	134	458018.0	0.0	NFP	0.0
2924886	2016-02-14	134	486712.0	0.0	NFP	0.0
2925072	2015-12-15	134	496828.0	0.0	NFP	0.0
2926306	2015-12-20	134	527317.0	0.0	NFP	0.0
2930489	2017-01-29	134	551838.0	0.0	NFP	0.0
2940000	2016-11-22	1136	532465.0	0.0	NFP	0.0
2942453	2017-03-11	1136	572354.0	0.0	NFP	0.0
2955081	2017-05-15	1291	2151761.0	0.0	NFP	0.0
2973373	2016-06-04	636	470724.0	0.0	NFP	0.0
2976314	2015-09-05	636	2972471.0	0.0	NFP	0.0
3018743	2017-05-02	1150	595470.0	0.0	NFP	0.0
3058398	2016-04-20	1176	468801.0	0.0	NFP	0.0
3061636	2016-11-02	1176	2126912.0	0.0	NFP	0.0
3079614	2016-03-22	1061	541516.0	0.0	NFP	0.0
3084512	2017-03-06	1061	2138909.0	0.0	NFP	0.0
3109270	2017-07-16	1164	608836.0	0.0	NFP	0.0

3117352	2016-10-27	903	569566.0	0.0	NFP	0.0
3126617	2015-08-13	392	2989426.0	0.0	NFP	0.0
3130722	2016-06-18	392	2114017.0	0.0	NFP	0.0
3130732	2016-04-21	392	387746.0	0.0	NFP	0.0
3143994	2017-01-05	1051	2128751.0	0.0	NFP	0.0
3163222	2015-10-11	518	451088.0	0.0	NFP	0.0
3170681	2017-07-02	518	594747.0	0.0	NFP	0.0
3177999	2016-11-06	422	553149.0	0.0	NFP	0.0
3179156	2017-06-22	422	2147892.0	0.0	NFP	0.0
3179381	2017-06-11	422	593194.0	0.0	NFP	0.0
3186170	2017-03-19	132	569731.0	0.0	NFP	0.0
3268454	2015-12-01	1032	2991604.0	0.0	NFP	0.0
3280714	2016-12-08	1200	458174.0	0.0	NFP	0.0
3280908	2016-08-18	1200	2129817.0	0.0	NFP	0.0
3282512	2016-11-04	1200	2138909.0	0.0	NFP	0.0
3293353	2015-08-29	237	384289.0	0.0	NFP	0.0
3320279	2017-05-23	603	2131953.0	0.0	NFP	0.0
3333125	2017-03-08	741	567743.0	0.0	NFP	0.0
3333280	2017-03-08	741	584813.0	0.0	NFP	0.0
3336962	2015-08-05	1257	421628.0	0.0	NFP	0.0
3352069	2016-05-17	429	484238.0	0.0	NFP	0.0
3366516	2017-02-15	76	554295.0	0.0	NFP	0.0
3396803	2017-	373	2147397.0	0.0	NFP	0.0

	06-02					
3405182	2016-10-22	1245	2130773.0	0.0	NFP	0.0
3406271	2017-04-15	1245	571893.0	0.0	NFP	0.0
3406289	2017-02-19	1245	571430.0	0.0	NFP	0.0
3409246	2015-10-25	582	470963.0	0.0	NFP	0.0
3411230	2015-11-05	582	464354.0	0.0	NFP	0.0
3424520	2016-01-04	168	2120238.0	0.0	NFP	0.0
3432007	2017-06-22	168	583997.0	0.0	NFP	0.0
3432514	2017-02-17	168	2143982.0	0.0	NFP	0.0
3433520	2017-06-21	168	609560.0	0.0	NFP	0.0
3434880	2017-07-12	168	617837.0	0.0	NFP	0.0
3435060	2017-07-19	168	605535.0	0.0	NFP	0.0
3435166	2017-07-07	168	618868.0	0.0	NFP	0.0
3435204	2017-06-21	168	626846.0	0.0	NFP	0.0
3435446	2017-07-07	168	618892.0	0.0	NFP	0.0
3435745	2017-07-19	168	621383.0	0.0	NFP	0.0
3435793	2017-07-12	168	618827.0	0.0	NFP	0.0
3436023	2017-06-21	168	640078.0	0.0	NFP	0.0
3436412	2017-07-12	168	632109.0	0.0	NFP	0.0
3436415	2017-07-11	168	632109.0	0.0	NFP	0.0
3447990	2016-07-09	241	536334.0	0.0	NFP	0.0
3458869	2016-10-27	664	538694.0	0.0	NFP	0.0
3480872	2017-07-15	1022	584789.0	0.0	NFP	0.0

3481247	2017-02-15	1022	587071.0	0.0	NFP	0.0
3482225	2017-02-15	1022	581975.0	0.0	NFP	0.0
3482357	2017-04-07	1022	580910.0	0.0	NFP	0.0
3499522	2017-01-21	444	576538.0	0.0	NFP	0.0
3517163	2016-02-19	795	525485.0	0.0	NFP	0.0
3537905	2016-05-07	757	484238.0	0.0	NFP	0.0
3546786	2016-10-02	717	524868.0	0.0	NFP	0.0
3556812	2016-08-26	1174	524868.0	0.0	NFP	0.0
3568374	2017-05-19	896	600759.0	0.0	NFP	0.0
3582658	2017-02-19	752	2134718.0	0.0	NFP	0.0
3624518	2017-03-25	572	540112.0	0.0	NFP	0.0
3635576	2017-06-20	572	2154765.0	0.0	NFP	0.0
3659734	2015-09-07	1013	2982058.0	0.0	NFP	0.0
3660405	2016-11-04	1013	511972.0	0.0	NFP	0.0
3660702	2017-02-18	1013	2115865.0	0.0	NFP	0.0
3661203	2017-02-16	1013	2131235.0	0.0	NFP	0.0
3662126	2017-02-20	1013	571430.0	0.0	NFP	0.0
3662245	2017-02-20	1013	2133207.0	0.0	NFP	0.0
3662259	2017-02-20	1013	2138834.0	0.0	NFP	0.0
3675160	2017-03-11	1234	594689.0	0.0	NFP	0.0
3684852	2016-08-18	309	513622.0	0.0	NFP	0.0
3689642	2017-02-20	309	2134726.0	0.0	NFP	0.0
3704729	2016-07-31	763	540625.0	0.0	NFP	0.0

3712513	2017-02-17	763	2142448.0	0.0	NFP	0.0
3714282	2017-05-20	763	2146860.0	0.0	NFP	0.0
3722801	2015-10-13	450	485011.0	0.0	NFP	0.0
3723966	2016-02-05	450	487199.0	0.0	NFP	0.0
3733905	2016-10-22	450	533521.0	0.0	NFP	0.0
3748885	2016-10-31	1192	545939.0	0.0	NFP	0.0
3766111	2017-02-11	1071	545244.0	0.0	NFP	0.0
3768581	2017-02-17	1071	565184.0	0.0	NFP	0.0
3769011	2017-02-27	1071	582395.0	0.0	NFP	0.0
3794467	2016-02-22	1140	499038.0	0.0	NFP	0.0
3812330	2016-04-20	685	494617.0	0.0	NFP	0.0
3814149	2015-07-29	685	413187.0	0.0	NFP	0.0
3834256	2016-09-14	500	542704.0	0.0	NFP	0.0
3846389	2017-02-20	348	2148429.0	0.0	NFP	0.0
3849910	2015-11-27	57	473983.0	0.0	NFP	0.0
3864537	2017-06-16	1135	2155135.0	0.0	NFP	0.0
3871581	2016-12-17	1165	588285.0	0.0	NFP	0.0
3877616	2016-11-05	376	504472.0	0.0	NFP	0.0
3879700	2016-11-28	376	540674.0	0.0	NFP	0.0
3884965	2016-04-10	10	499111.0	0.0	NFP	0.0
3888148	2016-10-27	10	533927.0	0.0	NFP	0.0
3892526	2015-09-10	55	384289.0	0.0	NFP	0.0
3896511	2017-07-28	55	600809.0	0.0	NFP	0.0

3908568	2015-09-05	1033	441998.0	0.0	NFP	0.0
3921322	2017-02-20	1033	2142448.0	0.0	NFP	0.0
3923963	2016-01-01	353	496315.0	0.0	NFP	0.0
3929394	2017-02-18	353	2139436.0	0.0	NFP	0.0
3937720	2016-06-23	303	552653.0	0.0	FP	0.0
3961585	2017-06-10	334	593814.0	0.0	NFP	0.0
3978949	2016-08-19	746	545905.0	0.0	NFP	0.0
3989490	2015-08-08	1221	399956.0	0.0	NFP	0.0
4002218	2015-10-05	676	2993022.0	0.0	NFP	0.0
4002930	2015-10-28	676	436915.0	0.0	NFP	0.0
4022954	2015-10-10	754	432351.0	0.0	NFP	0.0
4033928	2016-06-24	6	2124966.0	0.0	NFP	0.0
4038775	2016-08-06	6	540112.0	0.0	NFP	0.0
4058345	2017-05-22	1054	600759.0	0.0	NFP	0.0
4065289	2016-10-01	465	545236.0	0.0	NFP	0.0
4068439	2015-08-07	532	2999748.0	0.0	NFP	0.0
4090782	2017-04-11	359	2139204.0	0.0	NFP	0.0
4095982	2016-06-30	517	818823.0	0.0	NFP	0.0
4103609	2016-05-16	561	496810.0	0.0	NFP	0.0
4104201	2016-05-28	561	457242.0	0.0	NFP	0.0
4120230	2017-02-17	561	2138768.0	0.0	NFP	0.0
4129666	2016-08-25	352	537696.0	0.0	NFP	0.0
	2015-					

4135434	10-09	235	445841.0	0.0	NFP	0.0
4136601	2016-04-29	235	2113498.0	0.0	NFP	0.0
4136725	2015-10-31	235	445775.0	0.0	NFP	0.0
4148780	2016-05-26	1090	473470.0	0.0	NFP	0.0
4156649	2016-04-11	740	512830.0	0.0	NFP	0.0
4159104	2016-10-29	740	2118471.0	0.0	NFP	0.0
4161200	2016-11-20	740	551838.0	0.0	NFP	0.0
4161682	2017-02-10	740	2143123.0	0.0	NFP	0.0
4161922	2017-02-20	740	2137463.0	0.0	NFP	0.0
4167624	2015-09-05	21	431254.0	0.0	NFP	0.0
4177480	2017-01-01	21	557595.0	0.0	NFP	0.0
4181902	2017-07-27	21	604355.0	0.0	NFP	0.0
4186282	2015-12-08	556	436816.0	0.0	NFP	0.0
4188955	2015-08-29	584	2999482.0	0.0	NFP	0.0
4201560	2016-08-10	1321	2128751.0	0.0	NFP	0.0
4217471	2016-08-02	83	457283.0	0.0	NFP	0.0
4226798	2017-04-21	83	613224.0	0.0	NFP	0.0
4241661	2017-04-08	281	568428.0	0.0	NFP	0.0
4253276	2015-08-12	212	451781.0	0.0	NFP	0.0
4257749	2016-03-13	212	2121376.0	0.0	FP	0.0
4279204	2017-02-15	290	2143131.0	0.0	NFP	0.0
4280455	2016-10-21	1199	2936708.0	0.0	NFP	0.0
4283154	2015-12-22	1199	493023.0	0.0	NFP	0.0

4288919	2016-08-08	1199	546010.0	0.0	NFP	0.0
4297545	2016-06-19	354	2113555.0	0.0	NFP	0.0
4317425	2017-07-25	294	569251.0	0.0	NFP	0.0
4319959	2017-06-11	294	2154674.0	0.0	NFP	0.0
4330454	2016-10-29	267	2128694.0	0.0	NFP	0.0
4330813	2017-03-04	267	2138909.0	0.0	NFP	0.0
4330841	2017-04-14	267	572966.0	0.0	NFP	0.0
4347821	2017-03-18	60	584714.0	0.0	NFP	0.0
4367007	2015-12-05	1115	454199.0	0.0	NFP	0.0
4370622	2015-09-10	23	412361.0	0.0	NFP	0.0
4373021	2016-12-16	23	533166.0	0.0	NFP	0.0
4373902	2016-11-05	23	545905.0	0.0	NFP	0.0
4383174	2016-03-13	1256	2124404.0	0.0	NFP	0.0
4401785	2017-02-19	1004	554295.0	0.0	NFP	0.0
4405569	2016-03-09	1212	458372.0	0.0	NFP	0.0
4406160	2015-12-04	1212	455337.0	0.0	NFP	0.0
4407466	2015-09-06	1212	2972471.0	0.0	NFP	0.0
4414590	2015-08-18	551	159137.0	0.0	NFP	0.0
4416804	2015-08-25	551	451989.0	0.0	NFP	0.0
4456429	2017-01-29	486	2139501.0	0.0	NFP	0.0
4462084	2015-08-22	1240	440750.0	0.0	NFP	0.0
4469187	2016-11-23	1240	583807.0	0.0	NFP	0.0
4475057	2015-11-27	403	363788.0	0.0	NFP	0.0

4475611	2015-09-25	403	427260.0	0.0	NFP	0.0
4478292	2017-01-05	403	545434.0	0.0	NFP	0.0
4479927	2017-02-25	403	557603.0	0.0	NFP	0.0
4481257	2017-03-12	403	594689.0	0.0	NFP	0.0
4481264	2017-02-20	403	2143057.0	0.0	NFP	0.0
4481333	2017-03-12	403	567768.0	0.0	NFP	0.0
4482678	2015-08-14	474	428144.0	0.0	NFP	0.0
4530719	2016-11-08	614	565192.0	0.0	NFP	0.0
4541517	2015-09-08	1303	2936708.0	0.0	NFP	0.0
4566747	2016-04-15	470	485003.0	0.0	NFP	0.0
4574955	2017-03-15	470	560219.0	0.0	NFP	0.0
4606321	2017-02-15	307	571232.0	0.0	NFP	0.0
4615004	2016-09-15	90	535864.0	0.0	NFP	0.0
4615341	2016-06-10	90	501049.0	0.0	NFP	0.0
4621077	2016-09-15	90	2132514.0	0.0	NFP	0.0
4621454	2016-11-03	90	2133116.0	0.0	NFP	0.0
4623663	2017-06-16	90	2147850.0	0.0	NFP	0.0
4630141	2015-09-14	1147	139444.0	0.0	NFP	0.0
4655022	2016-06-18	306	513358.0	0.0	NFP	0.0
4659014	2017-02-15	306	551176.0	0.0	NFP	0.0
4661828	2017-06-11	306	601245.0	0.0	NFP	0.0
4663161	2017-07-26	306	612192.0	0.0	NFP	0.0
	2016-					

4685003	07-02	1317	513283.0	0.0	NFP	0.0
4689235	2017-02-19	1317	581975.0	0.0	NFP	0.0
4696640	2016-11-11	432	504597.0	0.0	NFP	0.0
4701134	2017-07-26	432	619148.0	0.0	NFP	0.0
4704876	2015-08-27	4	453985.0	0.0	NFP	0.0
4719156	2017-02-17	190	551176.0	0.0	NFP	0.0
4722910	2017-07-26	190	602789.0	0.0	NFP	0.0
4728230	2016-10-25	91	2124396.0	0.0	NFP	0.0
4731801	2017-01-21	91	2141028.0	0.0	NFP	0.0
4733239	2017-07-28	91	2152603.0	0.0	NFP	0.0
4735630	2015-08-06	1137	455766.0	0.0	NFP	0.0
4737039	2015-08-21	1137	460774.0	0.0	NFP	0.0
4742688	2016-08-29	1137	545913.0	0.0	NFP	0.0
4786935	2016-04-25	673	2124677.0	0.0	NFP	0.0
4798287	2016-10-30	402	513978.0	0.0	NFP	0.0
4804720	2016-09-02	402	2133355.0	0.0	NFP	0.0
4804838	2017-01-18	402	580571.0	0.0	NFP	0.0
4814383	2015-12-10	1264	437145.0	0.0	NFP	0.0
4824642	2016-10-25	780	2126896.0	0.0	NFP	0.0
4827157	2016-09-26	780	533471.0	0.0	NFP	0.0
4829018	2017-03-03	780	571893.0	0.0	NFP	0.0
4864070	2015-11-06	27	454025.0	0.0	NFP	0.0
4879598	2016-11-06	65	531889.0	0.0	NFP	0.0

4903015	2016-05-07	1094	501536.0	0.0	NFP	0.0
4905973	2016-11-25	1094	2136937.0	0.0	FP	0.0
4926884	2016-11-25	719	533166.0	0.0	NFP	0.0
4932887	2015-12-28	437	471003.0	0.0	NFP	0.0
4932995	2016-02-05	437	501536.0	0.0	NFP	0.0
4948617	2017-03-03	318	576538.0	0.0	NFP	0.0
4949422	2017-03-24	318	576934.0	0.0	NFP	0.0
4974314	2017-07-03	1156	2151639.0	0.0	NFP	0.0
5003920	2017-02-08	1162	2143123.0	0.0	NFP	0.0
5012217	2016-08-13	355	538553.0	0.0	NFP	0.0
5030786	2015-09-06	1330	447714.0	0.0	NFP	0.0
5060329	2016-12-03	647	584284.0	0.0	NFP	0.0
5068208	2016-11-10	772	540112.0	0.0	NFP	0.0
5069438	2016-11-15	772	552034.0	0.0	NFP	0.0
5078037	2017-07-22	350	569251.0	0.0	NFP	0.0
5083278	2016-01-31	797	481259.0	0.0	NFP	0.0
5084166	2016-01-05	797	512608.0	0.0	NFP	0.0
5087460	2016-06-30	797	513911.0	0.0	NFP	0.0
5098520	2015-08-09	730	466714.0	0.0	NFP	0.0
5107409	2016-01-11	1075	472076.0	0.0	NFP	0.0
5107447	2015-07-31	1075	442079.0	0.0	NFP	0.0
5111602	2017-02-20	1075	571448.0	0.0	NFP	0.0
5112025	2017-03-18	1075	580910.0	0.0	NFP	0.0

5114931	2016-01-23	169	487199.0	0.0	NFP	0.0
5125897	2016-04-16	1313	507046.0	0.0	NFP	0.0
5130244	2017-03-20	1313	2134726.0	0.0	NFP	0.0
5131155	2017-02-07	1313	2141028.0	0.0	NFP	0.0
5137444	2016-05-19	750	2117853.0	0.0	NFP	0.0
5143732	2016-04-07	1060	496539.0	0.0	NFP	0.0
5160420	2017-07-27	1284	610246.0	0.0	NFP	0.0
5169539	2017-07-29	1104	608786.0	0.0	NFP	0.0
5174930	2016-12-21	1172	524629.0	0.0	NFP	0.0
5175103	2016-07-31	1172	2125815.0	0.0	NFP	0.0
5179314	2017-02-24	1172	584813.0	0.0	NFP	0.0
5193123	2017-01-11	344	2131375.0	0.0	NFP	0.0
5195447	2017-03-24	344	576819.0	0.0	NFP	0.0
5202343	2015-12-23	1157	482992.0	0.0	NFP	0.0
5218222	2016-04-27	475	2110668.0	0.0	NFP	0.0
5218909	2016-11-02	475	385708.0	0.0	NFP	0.0
5219303	2016-05-18	475	2113555.0	0.0	NFP	0.0
5224397	2017-07-16	475	605469.0	0.0	NFP	0.0
5250749	2017-02-16	693	2140178.0	0.0	NFP	0.0
5251151	2017-01-09	693	584813.0	0.0	FP	0.0
5252083	2017-07-10	693	593822.0	0.0	NFP	0.0
5259743	2015-09-02	1082	636472.0	0.0	NFP	0.0
5260380	2016-03-14	1082	2127597.0	0.0	NFP	0.0

5271845	2015-08-31	798	455741.0	0.0	NFP	0.0
5271852	2015-11-06	798	455741.0	0.0	NFP	0.0
5295230	2016-08-29	540	2127134.0	0.0	NFP	0.0
5298786	2017-02-24	540	551200.0	0.0	NFP	0.0
5303234	2016-03-19	163	2110700.0	0.0	NFP	0.0
5305915	2016-08-20	163	2132654.0	0.0	NFP	0.0
5307680	2016-09-23	163	571430.0	0.0	NFP	0.0
5318766	2016-02-19	230	818823.0	0.0	NFP	0.0
5325261	2016-10-25	230	522698.0	0.0	NFP	0.0
5360894	2016-04-10	1207	459727.0	0.0	NFP	0.0
5366128	2016-11-29	1207	2134734.0	0.0	NFP	0.0
5366524	2017-01-19	1207	2143123.0	0.0	NFP	0.0
5373201	2015-11-16	15	2999730.0	0.0	NFP	0.0
5373734	2015-11-05	15	455774.0	0.0	NFP	0.0
5380919	2017-03-27	15	551176.0	0.0	NFP	0.0
5387147	2015-12-26	788	427682.0	0.0	NFP	0.0
5397651	2016-10-24	788	551093.0	0.0	NFP	0.0
5407268	2015-08-11	98	464388.0	0.0	NFP	0.0
5409237	2016-10-23	98	531855.0	0.0	NFP	0.0
5415177	2016-06-25	98	531830.0	0.0	NFP	0.0
5420650	2017-07-27	98	2138099.0	0.0	NFP	0.0
5422105	2017-06-13	98	2150144.0	0.0	NFP	0.0
5426361	2015-08-27	31	340505.0	0.0	NFP	0.0

5431055	2016-10-05	31	557017.0	0.0	NFP	0.0
5437507	2015-07-28	704	453928.0	0.0	NFP	0.0
5449813	2016-12-05	704	584615.0	0.0	NFP	0.0
5457266	2016-09-09	202	531996.0	0.0	NFP	0.0
5461311	2017-02-21	202	565291.0	0.0	NFP	0.0
5463269	2017-06-17	202	601245.0	0.0	NFP	0.0
5470438	2017-02-16	345	2136986.0	0.0	NFP	0.0
5484134	2016-05-30	761	532747.0	0.0	NFP	0.0
5487676	2017-03-09	761	584714.0	0.0	NFP	0.0
5491898	2016-06-29	567	545269.0	0.0	NFP	0.0
5499355	2016-12-30	1067	2126847.0	0.0	NFP	0.0
5509969	2016-08-20	1002	547786.0	0.0	NFP	0.0
5529523	2015-11-12	1273	446450.0	0.0	NFP	0.0
5531987	2016-08-25	1273	544916.0	0.0	NFP	0.0
5532468	2016-09-01	1273	542738.0	0.0	NFP	0.0
5533040	2016-08-25	1273	540674.0	0.0	NFP	0.0
5533177	2017-02-18	1273	551820.0	0.0	NFP	0.0
5533194	2016-09-02	1273	547786.0	0.0	NFP	0.0
5537870	2015-09-30	245	454025.0	0.0	NFP	0.0
5547024	2017-02-11	245	2139618.0	0.0	NFP	0.0
5557444	2016-04-27	1146	484238.0	0.0	NFP	0.0
5562636	2016-08-19	1146	533497.0	0.0	NFP	0.0
5576396	2017-	177	555987.0	0.0	NFP	0.0

	04-13					
5587349	2016-07-12	1247	2115337.0	0.0	NFP	0.0
5597464	2016-05-16	3	505545.0	0.0	NFP	0.0
5600099	2016-11-05	3	2125005.0	0.0	NFP	0.0
5605605	2016-10-24	3	2129577.0	0.0	NFP	0.0
5608613	2017-02-12	3	2143123.0	0.0	NFP	0.0
5615052	2015-08-23	390	433144.0	0.0	NFP	0.0
5619480	2016-11-26	390	531889.0	0.0	NFP	0.0
5638405	2016-05-24	200	537688.0	0.0	NFP	0.0
5646599	2016-09-09	200	584326.0	0.0	NFP	0.0
5686251	2017-01-09	24	584375.0	0.0	NFP	0.0
5687743	2017-05-12	24	591214.0	0.0	NFP	0.0
5692996	2016-06-24	301	539049.0	0.0	NFP	0.0
5708630	2015-08-12	544	2999532.0	0.0	NFP	0.0
5718059	2017-02-18	544	572966.0	0.0	NFP	0.0
5719386	2017-02-05	544	2146050.0	0.0	NFP	0.0
5719681	2017-02-26	544	576934.0	0.0	NFP	0.0
5722460	2016-03-05	1087	2102772.0	0.0	NFP	0.0
5722674	2016-03-05	1087	2114439.0	0.0	NFP	0.0
5722941	2016-03-05	1087	2109520.0	0.0	NFP	0.0
5723410	2016-03-05	1087	2109017.0	0.0	NFP	0.0
5723762	2016-03-05	1087	2109496.0	0.0	NFP	0.0
5723896	2016-03-05	1087	2100586.0	0.0	NFP	0.0

5728497	2016-04-28	770	489872.0	0.0	NFP	0.0
5730617	2015-11-24	770	428672.0	0.0	NFP	0.0
5735032	2016-12-31	770	2135210.0	0.0	NFP	0.0
5736435	2017-02-18	770	567735.0	0.0	NFP	0.0
5740439	2015-11-15	455	445171.0	0.0	NFP	0.0
5740587	2016-05-15	455	489740.0	0.0	NFP	0.0
5745777	2016-11-13	455	532531.0	0.0	NFP	0.0
5762988	2016-07-16	447	552943.0	0.0	NFP	0.0
5767603	2015-10-10	198	466730.0	0.0	NFP	0.0
5770974	2016-05-24	198	538728.0	0.0	NFP	0.0
5791736	2015-08-04	1261	2997098.0	0.0	NFP	0.0
5794835	2017-02-17	1261	2126946.0	0.0	NFP	0.0
5846582	2016-11-18	111	2133215.0	0.0	NFP	0.0
5853011	2015-11-25	117	385955.0	0.0	NFP	0.0
5860372	2017-02-19	117	2118448.0	0.0	NFP	0.0
5864084	2017-01-20	117	581421.0	0.0	NFP	0.0
5896268	2015-10-31	1114	436733.0	0.0	NFP	0.0
5902073	2016-11-08	1114	552646.0	0.0	NFP	0.0
5911558	2015-10-10	620	458240.0	0.0	NFP	0.0
5912715	2016-12-12	620	515601.0	0.0	NFP	0.0
5914646	2016-10-26	620	2124677.0	0.0	NFP	0.0
5915626	2017-03-12	620	582395.0	0.0	NFP	0.0
5923234	2016-08-06	1226	529610.0	0.0	NFP	0.0

5943288	2017-02-28	1043	2138909.0	0.0	NFP	0.0
5991452	2016-01-06	699	507418.0	0.0	NFP	0.0
6018841	2015-11-27	510	412510.0	0.0	NFP	0.0
6023036	2016-05-25	510	542639.0	0.0	NFP	0.0
6024421	2016-09-29	510	2128694.0	0.0	NFP	0.0
6051448	2016-11-30	232	2126581.0	0.0	NFP	0.0
6052876	2017-03-07	232	2139212.0	0.0	NFP	0.0
6056335	2017-06-29	524	413070.0	0.0	NFP	0.0
6064594	2015-12-02	1091	445841.0	0.0	NFP	0.0
6065770	2016-02-13	1091	496489.0	0.0	NFP	0.0
6071511	2017-02-26	1091	570762.0	0.0	NFP	0.0
6074425	2016-01-24	456	489856.0	0.0	NFP	0.0
6074949	2016-02-14	456	1.0	0.0	NFP	0.0
6074950	2016-02-14	456	1.0	0.0	NFP	0.0
6074951	2016-02-14	456	1.0	0.0	NFP	0.0
6074952	2016-02-14	456	1.0	0.0	NFP	0.0
6074953	2016-02-14	456	1.0	0.0	NFP	0.0
6074954	2016-02-14	456	1.0	0.0	NFP	0.0
6074955	2016-02-14	456	1.0	0.0	NFP	0.0
6074956	2016-02-14	456	1.0	0.0	NFP	0.0
6074957	2016-02-14	456	1.0	0.0	NFP	0.0
6074958	2016-02-14	456	1.0	0.0	NFP	0.0
6074959	2016-	456	1.0	0.0	NFP	0.0

	02-14					
6074960	2016-02-14	456	1.0	0.0	NFP	0.0
6074961	2016-02-14	456	1.0	0.0	NFP	0.0
6074962	2016-02-14	456	1.0	0.0	NFP	0.0
6074963	2016-02-14	456	1.0	0.0	NFP	0.0
6074964	2016-02-14	456	1.0	0.0	NFP	0.0
6074965	2016-02-14	456	1.0	0.0	NFP	0.0
6074966	2016-02-14	456	1.0	0.0	NFP	0.0
6074967	2016-02-14	456	1.0	0.0	NFP	0.0
6081230	2016-03-15	59	501536.0	0.0	NFP	0.0
6081377	2015-08-11	59	437053.0	0.0	NFP	0.0
6082385	2015-11-14	59	429902.0	0.0	NFP	0.0
6087560	2017-01-21	59	2143891.0	0.0	NFP	0.0
6107620	2015-11-14	631	436816.0	0.0	NFP	0.0
6113629	2017-01-15	631	2125229.0	0.0	NFP	0.0
6122054	2017-06-13	631	601690.0	0.0	NFP	0.0
6141281	2016-08-26	364	2135236.0	0.0	NFP	0.0
6158745	2016-05-23	692	2129577.0	0.0	NFP	0.0
6161546	2017-02-15	692	570762.0	0.0	NFP	0.0
6164938	2015-10-12	446	457093.0	0.0	NFP	0.0
6191703	2015-08-05	291	441840.0	0.0	NFP	0.0
6192268	2015-12-12	291	482703.0	0.0	NFP	0.0
6198971	2017-02-01	291	567586.0	0.0	NFP	0.0

6211443	2017-01-07	543	538447.0	0.0	NFP	0.0
6211784	2016-04-21	543	2124388.0	0.0	NFP	0.0
6212628	2017-01-07	543	538397.0	0.0	NFP	0.0
6222734	2016-10-24	1184	544486.0	0.0	NFP	0.0
6233235	2017-02-17	932	567578.0	0.0	NFP	0.0
6233482	2017-02-18	932	2143958.0	0.0	NFP	0.0
6240103	2015-11-04	1133	2997551.0	0.0	NFP	0.0
6265344	2015-10-29	239	417360.0	0.0	NFP	0.0
6270306	2017-01-18	239	580761.0	0.0	NFP	0.0
6272713	2016-09-21	753	159137.0	0.0	FP	0.0
6283923	2017-02-15	753	2132555.0	0.0	NFP	0.0
6286174	2017-02-18	753	583484.0	0.0	NFP	0.0
6289695	2016-04-28	1106	491647.0	0.0	NFP	0.0
6292300	2017-01-10	1106	550053.0	0.0	NFP	0.0
6294951	2016-05-20	1121	480236.0	0.0	NFP	0.0
6300083	2016-05-16	1121	537688.0	0.0	NFP	0.0
6303043	2017-02-27	1121	584789.0	0.0	NFP	0.0
6332853	2016-09-17	251	2105403.0	0.0	NFP	0.0
6338122	2016-08-10	251	533471.0	0.0	NFP	0.0
6340098	2017-02-18	251	2140889.0	0.0	NFP	0.0
6341620	2017-06-17	251	593822.0	0.0	NFP	0.0
6344512	2015-11-29	394	494906.0	0.0	NFP	0.0
6345282	2016-01-09	394	479071.0	0.0	NFP	0.0

6350674	2017-01-11	394	165951.0	0.0	NFP	0.0
6350901	2017-07-21	394	601336.0	0.0	NFP	0.0
6350972	2017-04-13	394	595488.0	0.0	NFP	0.0
6351149	2017-06-24	394	2146878.0	0.0	NFP	0.0
6351581	2017-07-08	394	588905.0	0.0	NFP	0.0
6352175	2017-07-29	394	611079.0	0.0	NFP	0.0
6354461	2015-12-16	175	455766.0	0.0	NFP	0.0
6360558	2017-02-18	175	568600.0	0.0	NFP	0.0
6369096	2016-08-20	635	537621.0	0.0	NFP	0.0
6373364	2016-08-20	635	551515.0	0.0	NFP	0.0
6375285	2017-02-14	635	2141028.0	0.0	NFP	0.0
6375568	2017-03-20	635	580381.0	0.0	NFP	0.0
6376055	2017-05-03	635	594523.0	0.0	NFP	0.0
6383047	2015-11-14	624	436188.0	0.0	NFP	0.0
6386997	2016-09-10	624	540138.0	0.0	NFP	0.0
6395066	2015-08-20	785	411082.0	0.0	NFP	0.0
6398009	2015-11-10	785	426528.0	0.0	NFP	0.0
6427473	2016-06-17	1262	535211.0	0.0	NFP	0.0
6428898	2017-02-09	1262	2129882.0	0.0	NFP	0.0
6433141	2015-11-07	1144	455618.0	0.0	NFP	0.0
6439533	2017-03-06	1144	582395.0	0.0	NFP	0.0
6442595	2016-03-12	1008	2108316.0	0.0	NFP	0.0
6446932	2017-	1008	2140889.0	0.0	NFP	0.0

	02-19					
6449830	2016-03-01	116	472076.0	0.0	NFP	0.0
6462149	2017-02-21	368	565283.0	0.0	NFP	0.0
6468740	2017-02-19	240	557017.0	0.0	NFP	0.0
6479618	2015-11-09	154	408948.0	0.0	NFP	0.0
6482376	2016-08-12	154	2115691.0	0.0	NFP	0.0
6482887	2017-03-18	154	584789.0	0.0	NFP	0.0
6486606	2016-05-01	1307	2117721.0	0.0	NFP	0.0
6503499	2015-11-10	634	427823.0	0.0	NFP	0.0
6505340	2016-11-20	634	538447.0	0.0	NFP	0.0
6522099	2016-11-14	93	524512.0	0.0	NFP	0.0
6543085	2015-09-17	682	454629.0	0.0	NFP	0.0
6554000	2017-02-08	682	2139618.0	0.0	NFP	0.0
6571172	2015-10-01	1249	447391.0	0.0	NFP	0.0
6573311	2015-12-24	1249	482703.0	0.0	NFP	0.0
6589310	2015-10-08	1314	458083.0	0.0	NFP	0.0
6593389	2017-02-05	1314	565143.0	0.0	NFP	0.0
6593420	2017-02-18	1314	572354.0	0.0	NFP	0.0
6597714	2016-03-03	96	501056.0	0.0	NFP	0.0
6600086	2015-08-22	96	416891.0	0.0	NFP	0.0
6604393	2016-10-21	96	545905.0	0.0	NFP	0.0
6622958	2017-03-02	610	2139618.0	0.0	NFP	0.0
6645904	2016-08-23	1180	540849.0	0.0	NFP	0.0

6667671	2016-08-10	155	538801.0	0.0	NFP	0.0
6678776	2015-08-07	4150	2103432.0	1.0	FP	0.0
6679169	2015-10-15	4150	483123.0	1.0	FP	0.0
6679738	2015-12-19	4150	385955.0	1.0	FP	0.0
6680138	2016-02-11	4150	818823.0	1.0	FP	0.0
6680170	2016-02-12	4150	818823.0	1.0	FP	0.0
6680233	2016-04-26	4150	818823.0	1.0	FP	0.0
6680436	2017-02-15	4150	818823.0	1.0	FP	0.0
6681164	2017-04-26	4150	2775163.0	1.0	FP	0.0
6682884	2016-04-25	4150	2919563.0	1.0	NFP	0.0
6684004	2016-04-25	4150	2989301.0	1.0	NFP	0.0
6684140	2016-10-19	4150	2989301.0	1.0	FP	0.0
6685253	2015-09-03	4150	482406.0	1.0	FP	0.0
6685271	2015-09-02	4150	482406.0	1.0	FP	0.0
6685275	2015-09-17	4150	482406.0	1.0	FP	0.0
6685717	2016-04-24	4150	2998955.0	1.0	NFP	0.0
6685736	2016-04-25	4150	2998955.0	1.0	NFP	0.0
6686520	2016-04-24	4150	2919951.0	1.0	NFP	0.0
6686524	2016-04-24	4150	2919951.0	1.0	FP	0.0
6686578	2016-04-25	4150	2919951.0	1.0	NFP	0.0
6687638	2015-11-22	4150	492090.0	1.0	FP	0.0
6688264	2016-10-10	4150	348698.0	1.0	FP	0.0
6689183	2016-04-24	4150	159137.0	1.0	NFP	0.0

6689641	2015-09-26	4150	2110502.0	1.0	FP	0.0
6689872	2015-10-29	4150	473710.0	1.0	FP	0.0
6692056	2015-10-06	4150	457119.0	1.0	NFP	0.0
6694430	2016-04-24	4150	427724.0	1.0	FP	0.0
6694655	2017-03-07	4150	427724.0	1.0	FP	0.0
6694851	2015-09-03	4150	486613.0	1.0	FP	0.0
6697057	2015-08-05	4150	434076.0	1.0	NFP	0.0
6697173	2016-01-14	4150	504241.0	1.0	FP	0.0
6697254	2015-08-08	4150	458067.0	1.0	FP	0.0
6697473	2016-04-24	4150	2973156.0	1.0	FP	0.0
6697479	2016-04-25	4150	2973156.0	1.0	NFP	0.0
6697832	2016-04-25	4150	2963793.0	1.0	NFP	0.0
6697855	2016-04-24	4150	2963793.0	1.0	FP	0.0
6699334	2016-10-20	4150	2103440.0	1.0	FP	0.0
6700567	2016-01-14	4150	2110460.0	1.0	NFP	0.0
6700623	2016-04-24	4150	2110460.0	1.0	NFP	0.0
6700721	2015-08-05	4150	2101055.0	1.0	FP	0.0
6700885	2015-11-13	4150	2108241.0	1.0	FP	0.0
6701103	2015-11-19	4150	482513.0	1.0	NFP	0.0
6701943	2016-04-03	4150	472399.0	1.0	FP	0.0
6702002	2016-11-02	4150	472399.0	1.0	FP	0.0
6702927	2016-04-25	4150	2989269.0	1.0	FP	0.0
	2016-					

6702939	04-24	4150	2989269.0	1.0	FP	0.0
6702971	2016-04-25	4150	2989269.0	1.0	NFP	0.0
6704082	2016-04-24	4150	2105403.0	1.0	FP	0.0
6704095	2016-04-24	4150	2105403.0	1.0	NFP	0.0
6704124	2016-04-25	4150	2105403.0	1.0	NFP	0.0
6704675	2015-10-15	4150	2110445.0	1.0	FP	0.0
6705118	2015-09-24	4150	458075.0	1.0	FP	0.0
6706155	2015-09-03	4150	457515.0	1.0	FP	0.0
6706290	2016-04-25	4150	457515.0	1.0	NFP	0.0
6706327	2016-04-24	4150	457515.0	1.0	FP	0.0
6706337	2016-04-24	4150	457515.0	1.0	NFP	0.0
6707600	2016-04-25	4150	472522.0	1.0	NFP	0.0
6708091	2015-10-11	4150	2108316.0	1.0	FP	0.0
6709417	2015-09-17	4150	489385.0	1.0	FP	0.0
6709798	2015-09-03	4150	487082.0	1.0	FP	0.0
6711053	2015-07-30	4150	455618.0	1.0	NFP	0.0
6713686	2016-04-24	4150	2116244.0	1.0	NFP	0.0
6713758	2015-09-24	4150	2109108.0	1.0	FP	0.0
6714203	2015-08-13	4150	457887.0	1.0	FP	0.0
6714321	2015-09-02	4150	493023.0	1.0	FP	0.0
6714779	2015-09-12	4150	470930.0	1.0	FP	0.0
6715207	2016-04-24	4150	2100313.0	1.0	NFP	0.0
6716445	2015-09-19	4150	491654.0	1.0	FP	0.0

6716779	2016-04-24	4150	506345.0	1.0	NFP	0.0
6718695	2015-08-08	4150	457853.0	1.0	FP	0.0
6718942	2015-11-06	4150	482109.0	1.0	FP	0.0
6719350	2015-09-02	4150	468983.0	1.0	NFP	0.0
6720049	2015-09-24	4150	458166.0	1.0	FP	0.0
6721673	2016-04-24	4150	2113662.0	1.0	NFP	0.0
6722503	2015-09-17	4150	507921.0	1.0	FP	0.0
6723064	2015-10-09	4150	2110684.0	1.0	FP	0.0
6723349	2015-11-19	4150	506311.0	1.0	FP	0.0
6723658	2015-09-23	4150	482224.0	1.0	FP	0.0
6723982	2015-12-23	4150	496877.0	1.0	NFP	0.0
6724539	2016-01-20	4150	431759.0	1.0	FP	0.0
6724580	2016-04-24	4150	431759.0	1.0	FP	0.0
6724582	2016-04-25	4150	431759.0	1.0	NFP	0.0
6724879	2015-11-06	4150	476523.0	1.0	FP	0.0
6725437	2016-04-25	4150	2991224.0	1.0	FP	0.0
6725473	2016-04-25	4150	2991224.0	1.0	NFP	0.0
6725481	2016-04-24	4150	2991224.0	1.0	FP	0.0
6726690	2016-04-25	4150	469528.0	1.0	NFP	0.0
6727479	2016-04-26	4150	431767.0	1.0	NFP	0.0
6727652	2016-04-24	4150	2114330.0	1.0	NFP	0.0
6727655	2016-04-25	4150	2114330.0	1.0	NFP	0.0
6728817	2016-	4150	2100271.0	1.0	FP	0.0

	04-24					
6728859	2016-04-24	4150	2100271.0	1.0	NFP	0.0
6728863	2016-04-25	4150	2100271.0	1.0	NFP	0.0
6729903	2016-04-25	4150	457283.0	1.0	FP	0.0
6729913	2016-04-24	4150	457283.0	1.0	FP	0.0
6729929	2016-04-26	4150	457283.0	1.0	FP	0.0
6729970	2016-04-25	4150	457283.0	1.0	NFP	0.0
6730255	2016-04-25	4150	2100248.0	1.0	NFP	0.0
6730262	2016-04-24	4150	2100248.0	1.0	NFP	0.0
6730479	2016-04-24	4150	2100263.0	1.0	FP	0.0
6730481	2016-04-25	4150	2100263.0	1.0	NFP	0.0
6730505	2016-04-24	4150	2100263.0	1.0	NFP	0.0
6731101	2016-02-05	4150	2110866.0	1.0	FP	0.0
6731143	2015-12-28	4150	2110866.0	1.0	FP	0.0
6731164	2016-04-24	4150	2110866.0	1.0	NFP	0.0
6731173	2016-04-25	4150	2110866.0	1.0	NFP	0.0
6731203	2016-04-29	4150	2110866.0	1.0	FP	0.0
6731739	2015-11-18	4150	504167.0	1.0	FP	0.0
6732181	2016-04-24	4150	2110825.0	1.0	NFP	0.0
6732591	2016-04-24	4150	2991208.0	1.0	FP	0.0
6732625	2016-04-25	4150	2991208.0	1.0	NFP	0.0
6732695	2015-09-19	4150	2921809.0	1.0	FP	0.0
6733652	2015-11-18	4150	504142.0	1.0	FP	0.0

6734059	2015-08-31	4150	420992.0	1.0	FP	0.0
6734150	2016-04-25	4150	420992.0	1.0	NFP	0.0
6734169	2016-04-24	4150	420992.0	1.0	FP	0.0
6734434	2016-04-24	4150	457788.0	1.0	FP	0.0
6734754	2016-04-24	4150	2100230.0	1.0	FP	0.0
6734792	2016-04-25	4150	2100230.0	1.0	NFP	0.0
6734928	2016-04-25	4150	2987214.0	1.0	NFP	0.0
6735104	2015-12-31	4150	507673.0	1.0	FP	0.0
6735192	2016-04-24	4150	507673.0	1.0	NFP	0.0
6735211	2016-04-24	4150	507673.0	1.0	FP	0.0
6735227	2016-04-25	4150	507673.0	1.0	NFP	0.0
6735234	2016-04-26	4150	507673.0	1.0	NFP	0.0
6735564	2015-09-02	4150	469601.0	1.0	FP	0.0
6735627	2015-08-05	4150	2103499.0	1.0	FP	0.0
6736171	2016-04-25	4150	507566.0	1.0	NFP	0.0
6736201	2016-04-24	4150	507566.0	1.0	FP	0.0
6736498	2015-08-06	4150	468736.0	1.0	FP	0.0
6736551	2016-06-08	4150	451724.0	1.0	FP	0.0
6736954	2016-04-24	4150	431817.0	1.0	NFP	0.0
6737098	2017-01-26	4150	431817.0	1.0	NFP	0.0
6737177	2015-12-20	4150	431874.0	1.0	FP	0.0
6737303	2016-10-19	4150	431874.0	1.0	NFP	0.0
6737557	2016-04-25	4150	2991190.0	1.0	NFP	0.0

6737571	2016-04-25	4150	2991190.0	1.0	FP	0.0
6737574	2016-04-24	4150	2991190.0	1.0	FP	0.0
6737630	2016-08-04	4150	2991190.0	1.0	NFP	0.0
6737667	2017-01-11	4150	2991190.0	1.0	NFP	0.0
6737771	2015-12-14	4150	356030.0	1.0	FP	0.0
6737819	2016-04-24	4150	356030.0	1.0	FP	0.0
6737925	2016-04-24	4150	2110833.0	1.0	NFP	0.0
6737934	2016-04-25	4150	2110833.0	1.0	NFP	0.0
6737950	2016-04-24	4150	2110833.0	1.0	FP	0.0
6738358	2016-04-29	4150	356840.0	1.0	NFP	0.0
6738467	2016-04-24	4150	2110783.0	1.0	FP	0.0
6738492	2016-04-25	4150	2110783.0	1.0	NFP	0.0
6738517	2016-04-24	4150	2110783.0	1.0	NFP	0.0
6738772	2016-04-24	4150	488403.0	1.0	NFP	0.0
6738803	2016-04-25	4150	488403.0	1.0	NFP	0.0
6738959	2016-04-04	4150	467811.0	1.0	FP	0.0
6739087	2016-04-25	4150	2111096.0	1.0	NFP	0.0
6739091	2016-04-24	4150	2111096.0	1.0	FP	0.0
6739295	2016-04-25	4150	2110809.0	1.0	NFP	0.0
6739320	2016-04-24	4150	2110809.0	1.0	NFP	0.0
6739333	2016-04-24	4150	2110809.0	1.0	FP	0.0
6739486	2016-04-24	4150	2110486.0	1.0	NFP	0.0
6739875	2016-01-19	4150	487678.0	1.0	FP	0.0

6739969	2015-10-02	4150	488536.0	1.0	FP	0.0
6740103	2015-10-08	4150	487843.0	1.0	FP	0.0
6740313	2016-04-24	4150	489351.0	1.0	NFP	0.0
6740690	2016-04-25	4150	2110999.0	1.0	NFP	0.0
6740697	2016-04-24	4150	2110999.0	1.0	NFP	0.0
6741005	2016-04-26	4150	2110817.0	1.0	FP	0.0
6741037	2016-04-24	4150	2110817.0	1.0	FP	0.0
6741438	2016-04-24	4150	487595.0	1.0	NFP	0.0
6741451	2016-04-26	4150	487595.0	1.0	NFP	0.0
6741524	2016-03-07	4150	2116558.0	1.0	FP	0.0
6741576	2016-04-25	4150	2116558.0	1.0	NFP	0.0
6741628	2016-04-24	4150	2116558.0	1.0	FP	0.0
6741784	2016-04-24	4150	2116541.0	1.0	FP	0.0
6741797	2016-04-26	4150	2116541.0	1.0	NFP	0.0
6741804	2016-04-25	4150	2116541.0	1.0	NFP	0.0
6742129	2016-01-14	4150	488676.0	1.0	FP	0.0
6742324	2016-03-23	4150	2112334.0	1.0	NFP	0.0
6742546	2016-04-24	4150	2100321.0	1.0	NFP	0.0
6742555	2016-04-24	4150	2100321.0	1.0	FP	0.0
6742565	2016-05-18	4150	2100321.0	1.0	FP	0.0
6742609	2016-11-23	4150	2100321.0	1.0	NFP	0.0
6742616	2016-08-11	4150	2100321.0	1.0	NFP	0.0
	2016-					

6743015	04-24	4150	2111005.0	1.0	NFP	0.0
6743028	2016-04-25	4150	2111005.0	1.0	NFP	0.0
6743034	2016-04-24	4150	2111005.0	1.0	FP	0.0
6743409	2016-04-24	4150	2110932.0	1.0	NFP	0.0
6743411	2015-11-18	4150	2114371.0	1.0	FP	0.0
6743421	2015-11-25	4150	2114371.0	1.0	FP	0.0
6743763	2015-11-18	4150	2110718.0	1.0	FP	0.0
6744045	2016-01-21	4150	513341.0	1.0	FP	0.0
6744060	2016-03-27	4150	513341.0	1.0	FP	0.0
6744079	2016-01-28	4150	524504.0	1.0	FP	0.0
6744122	2016-03-23	4150	524504.0	1.0	FP	0.0
6744923	2016-03-23	4150	519397.0	1.0	FP	0.0
6745086	2016-02-17	4150	533042.0	1.0	FP	0.0
6745150	2016-05-03	4150	533042.0	1.0	FP	0.0
6745292	2015-12-22	4150	2117721.0	1.0	FP	0.0
6745552	2016-02-16	4150	500561.0	1.0	NFP	0.0
6745883	2016-02-25	4150	532556.0	1.0	FP	0.0
6746021	2015-12-22	4150	2117804.0	1.0	FP	0.0
6746162	2016-11-05	4150	2117804.0	1.0	FP	0.0
6746172	2017-01-04	4150	2117804.0	1.0	FP	0.0
6746263	2016-02-25	4150	540278.0	1.0	FP	0.0
6746348	2016-03-10	4150	538702.0	1.0	FP	0.0
6746526	2016-03-31	4150	540153.0	1.0	FP	0.0

6746528	2016-03-23	4150	540153.0	1.0	FP	0.0
6746703	2016-03-14	4150	2124958.0	1.0	FP	0.0
6747143	2015-12-22	4150	511923.0	1.0	FP	0.0
6747805	2016-04-06	4150	2126987.0	1.0	FP	0.0
6747858	2016-04-04	4150	532283.0	1.0	FP	0.0
6747939	2016-01-13	4150	2118042.0	1.0	FP	0.0
6748501	2016-02-25	4150	535864.0	1.0	FP	0.0
6748839	2016-02-25	4150	532754.0	1.0	FP	0.0
6748977	2016-05-20	4150	2127597.0	1.0	FP	0.0
6749290	2016-02-25	4150	531921.0	1.0	FP	0.0
6749552	2016-02-25	4150	2127134.0	1.0	FP	0.0
6749913	2016-02-17	4150	531228.0	1.0	FP	0.0
6749968	2016-04-28	4150	531228.0	1.0	FP	0.0
6750031	2016-03-11	4150	2124404.0	1.0	FP	0.0
6750262	2016-02-04	4150	2124396.0	1.0	FP	0.0
6751357	2016-02-10	4150	2121608.0	1.0	FP	0.0
6751444	2016-03-24	4150	2117986.0	1.0	FP	0.0
6751627	2016-05-03	4150	520593.0	1.0	FP	0.0
6752128	2016-02-25	4150	2124719.0	1.0	FP	0.0
6752226	2016-04-24	4150	2124651.0	1.0	FP	0.0
6752373	2016-03-15	4150	538579.0	1.0	FP	0.0
6752656	2016-03-17	4150	525402.0	1.0	FP	0.0
6752900	2015-	4150	511832.0	1.0	FP	0.0

	12-22					
6753174	2016-01-27	4150	2121350.0	1.0	FP	0.0
6753303	2016-02-19	4150	532473.0	1.0	FP	0.0
6753518	2016-05-02	4150	531871.0	1.0	FP	0.0
6753649	2016-04-05	4150	519405.0	1.0	FP	0.0
6753806	2016-04-04	4150	538744.0	1.0	FP	0.0
6754365	2016-01-13	4150	513333.0	1.0	FP	0.0
6754592	2016-01-13	4150	513283.0	1.0	FP	0.0
6754668	2016-02-25	4150	541664.0	1.0	FP	0.0
6755162	2016-01-20	4150	524959.0	1.0	FP	0.0
6755331	2016-07-06	4150	505917.0	1.0	NFP	0.0
6755492	2016-04-04	4150	2124669.0	1.0	FP	0.0
6755968	2016-03-14	4150	2124818.0	1.0	FP	0.0
6756656	2016-02-04	4150	2123844.0	1.0	FP	0.0
6756840	2016-02-17	4150	525212.0	1.0	FP	0.0
6757293	2015-12-02	4150	504696.0	1.0	FP	0.0
6758397	2016-04-28	4150	2127076.0	1.0	FP	0.0
6758629	2016-05-03	4150	524215.0	1.0	FP	0.0
6758667	2016-07-10	4150	524215.0	1.0	NFP	0.0
6758738	2015-12-22	4150	511949.0	1.0	FP	0.0
6758902	2016-01-27	4150	2118455.0	1.0	FP	0.0
6759054	2016-03-09	4150	2128306.0	1.0	FP	0.0
6759246	2015-12-22	4150	2115857.0	1.0	FP	0.0

6759442	2016-01-13	4150	513192.0	1.0	FP	0.0
6759473	2016-03-30	4150	513192.0	1.0	FP	0.0
6759502	2016-02-23	4150	535740.0	1.0	FP	0.0
6759922	2015-12-02	4150	500553.0	1.0	FP	0.0
6760162	2016-03-30	4150	537720.0	1.0	FP	0.0
6760183	2016-04-21	4150	537720.0	1.0	FP	0.0
6760184	2016-03-29	4150	537720.0	1.0	FP	0.0
6760237	2016-05-01	4150	537720.0	1.0	FP	0.0
6760346	2016-03-27	4150	506840.0	1.0	NFP	0.0
6760893	2016-01-27	4150	525261.0	1.0	FP	0.0
6761465	2016-06-14	4150	513093.0	1.0	NFP	0.0
6762269	2016-03-31	4150	539056.0	1.0	FP	0.0
6762272	2016-03-24	4150	539056.0	1.0	FP	0.0
6762343	2016-02-19	4150	522698.0	1.0	FP	0.0
6762543	2016-03-24	4150	2126565.0	1.0	FP	0.0
6763731	2016-01-13	4150	513325.0	1.0	FP	0.0
6763848	2016-04-14	4150	509539.0	1.0	FP	0.0
6763961	2016-01-13	4150	2118059.0	1.0	FP	0.0
6764258	2016-03-23	4150	532291.0	1.0	FP	0.0
6764362	2016-03-20	4150	506105.0	1.0	FP	0.0
6764635	2016-05-05	4150	532382.0	1.0	FP	0.0
6764824	2016-05-10	4150	512392.0	1.0	NFP	0.0
6764991	2016-04-24	4150	534198.0	1.0	FP	0.0

6765389	2015-12-10	4150	2115873.0	1.0	FP	0.0
6765449	2015-12-29	4150	507020.0	1.0	FP	0.0
6765661	2016-02-19	4150	532457.0	1.0	FP	0.0
6765715	2016-05-26	4150	515593.0	1.0	FP	0.0
6765810	2016-10-20	4150	515593.0	1.0	FP	0.0
6765924	2016-02-23	4150	542415.0	1.0	FP	0.0
6766192	2016-02-25	4150	545269.0	1.0	FP	0.0
6766232	2016-03-23	4150	545269.0	1.0	FP	0.0
6766785	2016-04-13	4150	541599.0	1.0	NFP	0.0
6766802	2016-06-20	4150	541599.0	1.0	NFP	0.0
6766860	2016-06-21	4150	536896.0	1.0	FP	0.0
6767848	2016-06-03	4150	531624.0	1.0	FP	0.0
6768294	2016-02-04	4150	525279.0	1.0	FP	0.0
6768408	2016-04-04	4150	538447.0	1.0	FP	0.0
6769147	2016-02-19	4150	2124362.0	1.0	FP	0.0
6769518	2016-02-25	4150	531939.0	1.0	FP	0.0
6769596	2016-04-25	4150	531939.0	1.0	FP	0.0
6769613	2016-03-02	4150	542704.0	1.0	FP	0.0
6769614	2016-03-09	4150	542704.0	1.0	FP	0.0
6769784	2016-04-05	4150	537613.0	1.0	FP	0.0
6769809	2016-02-25	4150	2124909.0	1.0	FP	0.0
6770158	2016-03-02	4150	540682.0	1.0	FP	0.0
6770166	2016-04-05	4150	540682.0	1.0	FP	0.0

6770605	2016-06-07	4150	513937.0	1.0	FP	0.0
6770781	2015-12-22	4150	511840.0	1.0	FP	0.0
6770935	2016-03-23	4150	532481.0	1.0	FP	0.0
6771273	2016-03-24	4150	534057.0	1.0	FP	0.0
6771381	2016-02-17	4150	2124933.0	1.0	FP	0.0
6771471	2015-12-22	4150	2117762.0	1.0	FP	0.0
6771631	2016-03-28	4150	533166.0	1.0	FP	0.0
6772096	2016-02-17	4150	512368.0	1.0	FP	0.0
6772138	2016-06-12	4150	512368.0	1.0	FP	0.0
6772289	2016-03-31	4150	515619.0	1.0	FP	0.0
6772301	2016-04-29	4150	515619.0	1.0	FP	0.0
6772329	2016-04-27	4150	515619.0	1.0	FP	0.0
6772334	2016-03-27	4150	515619.0	1.0	FP	0.0
6772474	2016-03-09	4150	551333.0	1.0	FP	0.0
6772800	2016-02-17	4150	522748.0	1.0	FP	0.0
6773012	2016-07-10	4150	532275.0	1.0	NFP	0.0
6773127	2016-03-22	4150	2126995.0	1.0	FP	0.0
6773240	2016-05-05	4150	532440.0	1.0	FP	0.0
6773357	2016-02-25	4150	2126888.0	1.0	FP	0.0
6773362	2016-03-21	4150	2126888.0	1.0	FP	0.0
6773930	2016-06-07	4150	506139.0	1.0	NFP	0.0
6773932	2016-06-06	4150	506139.0	1.0	NFP	0.0
	2016-					

6773961	05-23	4150	506139.0	1.0	NFP	0.0
6774208	2016-06-03	4150	2128231.0	1.0	FP	0.0
6774763	2016-04-27	4150	533224.0	1.0	FP	0.0
6774772	2016-03-24	4150	533224.0	1.0	FP	0.0
6775280	2016-08-31	4150	512897.0	1.0	NFP	0.0
6775454	2016-07-19	4150	2125229.0	1.0	NFP	0.0
6775767	2016-02-25	4150	550921.0	1.0	FP	0.0
6776570	2016-02-25	4150	529677.0	1.0	FP	0.0
6776789	2016-03-09	4150	2128769.0	1.0	FP	0.0
6777213	2016-03-24	4150	551762.0	1.0	FP	0.0
6777266	2016-04-27	4150	531848.0	1.0	FP	0.0
6777319	2016-02-23	4150	532465.0	1.0	FP	0.0
6778104	2016-11-06	4150	541508.0	1.0	NFP	0.0
6778125	2016-03-17	4150	2127845.0	1.0	FP	0.0
6778738	2016-03-23	4150	542696.0	1.0	FP	0.0
6778892	2016-05-19	4150	514828.0	1.0	FP	0.0
6779055	2016-02-19	4150	522680.0	1.0	FP	0.0
6779463	2016-03-24	4150	552398.0	1.0	FP	0.0
6779690	2016-03-24	4150	529610.0	1.0	FP	0.0
6779937	2016-05-11	4150	543249.0	1.0	FP	0.0
6780298	2016-02-17	4150	2132142.0	1.0	FP	0.0
6780776	2016-08-31	4150	2132688.0	1.0	NFP	0.0
6780814	2016-03-20	4150	551812.0	1.0	FP	0.0

6781535	2016-02-04	4150	529339.0	1.0	FP	0.0
6781614	2016-04-28	4150	549253.0	1.0	FP	0.0
6781812	2016-03-24	4150	552448.0	1.0	FP	0.0
6781872	2016-02-17	4150	531889.0	1.0	FP	0.0
6781946	2016-03-24	4150	531889.0	1.0	FP	0.0
6782182	2016-03-24	4150	2124446.0	1.0	FP	0.0
6782257	2016-10-14	4150	550368.0	1.0	NFP	0.0
6782274	2016-09-09	4150	550368.0	1.0	NFP	0.0
6782577	2016-08-30	4150	553958.0	1.0	NFP	0.0
6782787	2016-03-17	4150	544791.0	1.0	FP	0.0
6783076	2016-02-17	4150	544890.0	1.0	FP	0.0
6783330	2016-06-16	4150	534842.0	1.0	FP	0.0
6783420	2016-03-24	4150	532424.0	1.0	FP	0.0
6784356	2016-05-04	4150	544510.0	1.0	FP	0.0
6785237	2016-07-06	4150	2132654.0	1.0	FP	0.0
6785455	2016-05-26	4150	544569.0	1.0	FP	0.0
6786387	2016-06-09	4150	557462.0	1.0	FP	0.0
6786659	2016-07-27	4150	2132050.0	1.0	FP	0.0
6787117	2016-06-27	4150	542985.0	1.0	FP	0.0
6787483	2016-04-11	4150	534230.0	1.0	FP	0.0
6787505	2016-05-06	4150	534230.0	1.0	FP	0.0
6787528	2016-04-10	4150	534230.0	1.0	FP	0.0
6787583	2016-	4150	2125377.0	1.0	FP	0.0

	04-24					
6788654	2016-04-07	4150	539015.0	1.0	FP	0.0
6788831	2016-04-27	4150	2129551.0	1.0	FP	0.0
6789015	2016-05-11	4150	546101.0	1.0	FP	0.0
6789164	2016-06-26	4150	557413.0	1.0	FP	0.0
6789193	2016-05-10	4150	533497.0	1.0	FP	0.0
6789653	2016-04-24	4150	540070.0	1.0	FP	0.0
6791298	2016-08-07	4150	542936.0	1.0	NFP	0.0
6791612	2016-05-10	4150	2126847.0	1.0	FP	0.0
6791758	2016-04-17	4150	531830.0	1.0	FP	0.0
6792775	2016-05-10	4150	544486.0	1.0	FP	0.0
6792944	2016-04-28	4150	533935.0	1.0	FP	0.0
6793711	2016-05-10	4150	2131243.0	1.0	FP	0.0
6794084	2016-09-04	4150	547877.0	1.0	FP	0.0
6794172	2016-12-11	4150	561571.0	1.0	NFP	0.0
6796878	2016-05-10	4150	2130328.0	1.0	FP	0.0
6798041	2016-05-12	4150	544783.0	1.0	FP	0.0
6800939	2016-06-15	4150	534719.0	1.0	FP	0.0
6801019	2016-03-31	4150	2124677.0	1.0	FP	0.0
6801280	2016-03-31	4150	2129155.0	1.0	FP	0.0
6801933	2016-05-09	4150	552513.0	1.0	FP	0.0
6803497	2016-05-05	4150	547760.0	1.0	FP	0.0
6804486	2016-04-07	4150	543314.0	1.0	FP	0.0

6805815	2016-06-10	4150	541482.0	1.0	FP	0.0
6805921	2016-08-26	4150	2134767.0	1.0	FP	0.0
6806823	2016-09-22	4150	552034.0	1.0	FP	0.0
6807419	2016-05-01	4150	533489.0	1.0	FP	0.0
6807816	2016-06-24	4150	543462.0	1.0	FP	0.0
6807904	2016-06-12	4150	536334.0	1.0	NFP	0.0
6808029	2016-09-16	4150	545244.0	1.0	NFP	0.0
6808581	2016-08-31	4150	533521.0	1.0	NFP	0.0
6808709	2016-12-17	4150	572412.0	1.0	FP	0.0
6808878	2016-07-13	4150	580175.0	1.0	FP	0.0
6809757	2016-11-06	4150	548198.0	1.0	NFP	0.0
6809791	2016-12-17	4150	548198.0	1.0	NFP	0.0
6809841	2016-04-29	4150	2126508.0	1.0	FP	0.0
6810000	2016-04-28	4150	543553.0	1.0	FP	0.0
6810022	2016-06-23	4150	543553.0	1.0	FP	0.0
6810954	2016-12-19	4150	2132571.0	1.0	NFP	0.0
6811029	2016-05-05	4150	546598.0	1.0	FP	0.0
6811728	2016-05-05	4150	556365.0	1.0	FP	0.0
6812229	2016-12-19	4150	552869.0	1.0	FP	0.0
6812757	2016-09-16	4150	2129890.0	1.0	NFP	0.0
6813038	2016-12-17	4150	2142364.0	1.0	FP	0.0
6814023	2016-09-07	4150	547901.0	1.0	FP	0.0
6814048	2016-08-03	4150	547901.0	1.0	FP	0.0

6814169	2016-06-24	4150	2134759.0	1.0	FP	0.0
6814347	2016-05-26	4150	2129601.0	1.0	FP	0.0
6814527	2016-03-31	4150	540849.0	1.0	FP	0.0
6814673	2016-07-24	4150	557041.0	1.0	FP	0.0
6815353	2016-07-06	4150	551127.0	1.0	FP	0.0
6815632	2016-05-06	4150	2128751.0	1.0	FP	0.0
6815800	2016-09-07	4150	551200.0	1.0	FP	0.0
6816191	2016-04-07	4150	545277.0	1.0	FP	0.0
6816361	2016-09-02	4150	551861.0	1.0	FP	0.0
6816521	2016-08-09	4150	552646.0	1.0	FP	0.0
6816800	2016-10-20	4150	2132506.0	1.0	NFP	0.0
6816999	2016-08-31	4150	551192.0	1.0	FP	0.0
6817199	2016-04-28	4150	555136.0	1.0	FP	0.0
6817219	2016-04-27	4150	555136.0	1.0	FP	0.0
6817402	2016-07-19	4150	558221.0	1.0	FP	0.0
6817412	2016-05-11	4150	558221.0	1.0	FP	0.0
6817845	2016-03-24	4150	551515.0	1.0	FP	0.0
6819119	2016-05-05	4150	2133546.0	1.0	FP	0.0
6819888	2016-05-05	4150	2133553.0	1.0	FP	0.0
6820558	2016-04-07	4150	2127852.0	1.0	FP	0.0
6821146	2016-09-22	4150	542480.0	1.0	NFP	0.0
6821155	2016-07-06	4150	557157.0	1.0	FP	0.0
6821817	2016-11-06	4150	556324.0	1.0	NFP	0.0

6822008	2016-06-13	4150	2130336.0	1.0	FP	0.0
6822615	2016-03-24	4150	551416.0	1.0	FP	0.0
6823590	2016-05-05	4150	556357.0	1.0	FP	0.0
6823988	2016-05-05	4150	556373.0	1.0	FP	0.0
6824352	2016-07-28	4150	560219.0	1.0	FP	0.0
6824550	2016-04-07	4150	535856.0	1.0	FP	0.0
6825301	2016-06-08	4150	451708.0	1.0	FP	0.0
6825383	2016-07-28	4150	560151.0	1.0	FP	0.0
6825496	2016-07-28	4150	560227.0	1.0	FP	0.0
6825999	2016-07-28	4150	2135376.0	1.0	FP	0.0
6826222	2016-07-28	4150	560185.0	1.0	FP	0.0
6826863	2016-10-27	4150	2143818.0	1.0	FP	0.0
6827902	2016-10-20	4150	2138453.0	1.0	FP	0.0
6828906	2016-09-01	4150	2140608.0	1.0	FP	0.0
6829235	2016-08-04	4150	554964.0	1.0	FP	0.0
6830021	2016-11-02	4150	580993.0	1.0	FP	0.0
6830432	2016-08-11	4150	2135897.0	1.0	FP	0.0
6832006	2016-10-20	4150	557611.0	1.0	FP	0.0
6832265	2016-08-04	4150	553453.0	1.0	FP	0.0
6832613	2016-08-25	4150	2132639.0	1.0	FP	0.0
6832985	2016-08-11	4150	552190.0	1.0	FP	0.0
6833575	2017-03-02	4150	568881.0	1.0	FP	0.0
	2016-					

6834494	09-08	4150	2138594.0	1.0	FP	0.0
6834806	2016-09-01	4150	564765.0	1.0	FP	0.0
6834821	2016-09-08	4150	564765.0	1.0	FP	0.0
6836187	2017-01-17	4150	569715.0	1.0	NFP	0.0
6836399	2016-12-08	4150	2135350.0	1.0	NFP	0.0
6838146	2016-09-08	4150	572610.0	1.0	FP	0.0
6838423	2016-11-08	4150	560045.0	1.0	FP	0.0
6838550	2016-09-08	4150	569723.0	1.0	FP	0.0
6839525	2016-09-28	4150	570101.0	1.0	FP	0.0
6840029	2016-09-08	4150	570747.0	1.0	FP	0.0
6840596	2016-10-13	4150	582528.0	1.0	FP	0.0
6840784	2016-09-15	4150	570762.0	1.0	FP	0.0
6841124	2016-12-09	4150	565267.0	1.0	NFP	0.0
6841674	2017-03-03	4150	547232.0	1.0	NFP	0.0
6842988	2016-08-04	4150	2133207.0	1.0	FP	0.0
6844222	2016-11-03	4150	2148429.0	1.0	FP	0.0
6844223	2016-11-17	4150	2148429.0	1.0	FP	0.0
6844711	2016-09-22	4150	2139329.0	1.0	FP	0.0
6845855	2016-11-01	4150	2143776.0	1.0	FP	0.0
6847186	2016-12-18	4150	578237.0	1.0	FP	0.0
6847911	2017-02-08	4150	2147371.0	1.0	FP	0.0
6848263	2016-09-22	4150	583484.0	1.0	FP	0.0
6848381	2016-10-13	4150	571836.0	1.0	FP	0.0

6848417	2016-11-03	4150	584615.0	1.0	FP	0.0
6848746	2016-09-01	4150	575688.0	1.0	FP	0.0
6849158	2016-11-17	4150	586214.0	1.0	FP	0.0
6849455	2016-09-22	4150	560144.0	1.0	FP	0.0
6850311	2016-12-28	4150	2143289.0	1.0	FP	0.0
6850355	2016-10-11	4150	580910.0	1.0	FP	0.0
6851614	2016-11-03	4150	581470.0	1.0	FP	0.0
6851863	2017-02-21	4150	2142604.0	1.0	FP	0.0
6852457	2016-11-03	4150	2143792.0	1.0	FP	0.0
6852469	2016-10-27	4150	2143792.0	1.0	FP	0.0
6853165	2016-10-20	4150	571422.0	1.0	FP	0.0
6853176	2016-09-08	4150	571422.0	1.0	FP	0.0
6853607	2016-11-10	4150	606251.0	1.0	FP	0.0
6854280	2016-12-26	4150	2149161.0	1.0	FP	0.0
6856005	2017-01-20	4150	604231.0	1.0	FP	0.0
6857147	2017-01-05	4150	592519.0	1.0	FP	0.0
6857324	2016-12-23	4150	2147355.0	1.0	FP	0.0
6857548	2016-12-23	4150	2146902.0	1.0	FP	0.0
6857730	2017-01-20	4150	593111.0	1.0	FP	0.0
6858375	2017-03-02	4150	609560.0	1.0	FP	0.0
6858419	2017-03-02	4150	591396.0	1.0	FP	0.0
6858451	2017-03-15	4150	606079.0	1.0	FP	0.0
6858478	2016-	4150	606079.0	1.0	FP	0.0

	12-23					
6858827	2017-02-09	4150	601286.0	1.0	FP	0.0
6859010	2016-12-08	4150	592501.0	1.0	FP	0.0
6859122	2016-12-08	4150	590471.0	1.0	FP	0.0
6859809	2016-12-01	4150	2147918.0	1.0	FP	0.0
6860262	2016-12-17	4150	595470.0	1.0	FP	0.0
6860516	2016-12-23	4150	606061.0	1.0	FP	0.0
6860910	2017-02-02	4150	602888.0	1.0	FP	0.0
6860961	2017-02-02	4150	2150573.0	1.0	FP	0.0
6861102	2017-01-05	4150	592535.0	1.0	FP	0.0
6861580	2016-12-23	4150	2147975.0	1.0	FP	0.0
6861790	2017-01-12	4150	591214.0	1.0	FP	0.0
6861867	2016-12-01	4150	593830.0	1.0	FP	0.0
6862017	2016-12-08	4150	2146084.0	1.0	FP	0.0
6862090	2016-12-17	4150	603951.0	1.0	FP	0.0
6862471	2017-03-02	4150	608554.0	1.0	FP	0.0
6862650	2017-01-10	4150	594275.0	1.0	FP	0.0
6862689	2017-04-21	4150	594275.0	1.0	NFP	0.0
6862916	2017-04-26	4150	600916.0	1.0	FP	0.0
6862930	2017-01-20	4150	602797.0	1.0	FP	0.0
6863092	2017-02-16	4150	603456.0	1.0	FP	0.0
6863204	2017-02-02	4150	615914.0	1.0	FP	0.0
6863449	2017-01-12	4150	590992.0	1.0	FP	0.0

6863655	2017-01-12	4150	2147132.0	1.0	FP	0.0
6863868	2017-02-02	4150	602870.0	1.0	FP	0.0
6863884	2017-02-01	4150	590430.0	1.0	FP	0.0
6864261	2017-02-02	4150	600197.0	1.0	FP	0.0
6864390	2017-01-05	4150	2153767.0	1.0	FP	0.0
6864460	2017-01-23	4150	613125.0	1.0	FP	0.0
6865322	2017-02-02	4150	2149682.0	1.0	FP	0.0
6865336	2017-02-09	4150	607028.0	1.0	FP	0.0
6865421	2017-02-02	4150	615559.0	1.0	FP	0.0
6865479	2017-01-20	4150	594242.0	1.0	FP	0.0
6865670	2017-02-09	4150	2146886.0	1.0	FP	0.0
6866526	2017-03-02	4150	2151761.0	1.0	FP	0.0
6866647	2017-02-09	4150	606368.0	1.0	FP	0.0
6866940	2017-01-13	4150	594226.0	1.0	FP	0.0
6867006	2017-02-02	4150	607184.0	1.0	FP	0.0
6867635	2017-02-23	4150	607044.0	1.0	FP	0.0
6867733	2017-01-05	4150	2154823.0	1.0	FP	0.0
6868121	2017-02-14	4150	615450.0	1.0	FP	0.0
6868181	2017-02-23	4150	2155069.0	1.0	FP	0.0
6869059	2017-02-09	4150	613323.0	1.0	FP	0.0
6869140	2017-03-02	4150	606574.0	1.0	FP	0.0
6869213	2017-02-02	4150	615138.0	1.0	FP	0.0
6869567	2017-02-09	4150	2153577.0	1.0	FP	0.0

6869902	2017-02-23	4150	628388.0	1.0	FP	0.0
6869918	2017-02-23	4150	2158865.0	1.0	FP	0.0
6870092	2017-03-02	4150	2158931.0	1.0	FP	0.0
6870231	2017-02-23	4150	629220.0	1.0	FP	0.0
6870984	2017-03-30	4150	613281.0	1.0	FP	0.0
6888227	2016-05-20	1120	2109660.0	0.0	NFP	0.0
6890152	2016-05-20	1120	2113928.0	0.0	NFP	0.0
6905373	2017-03-28	1310	571430.0	0.0	NFP	0.0
6913589	2017-02-27	435	544569.0	0.0	NFP	0.0
6915764	2017-02-21	435	2132514.0	0.0	NFP	0.0
6921622	2017-01-16	457	2131227.0	0.0	NFP	0.0
6929691	2017-02-17	189	551820.0	0.0	NFP	0.0
6930446	2017-03-09	189	2133652.0	0.0	NFP	0.0
6933573	2015-11-14	1210	412825.0	0.0	NFP	0.0
6957831	2016-05-23	857	531590.0	0.0	NFP	0.0
6959957	2017-03-17	857	582395.0	0.0	NFP	0.0
6966746	2016-06-12	381	548123.0	0.0	NFP	0.0
6971695	2015-11-14	1281	422758.0	0.0	NFP	0.0
6980335	2016-05-18	269	482844.0	0.0	NFP	0.0
6992031	2015-11-19	710	480434.0	0.0	NFP	0.0
6992319	2015-11-04	710	453969.0	0.0	NFP	0.0
6992574	2015-08-17	710	833699.0	0.0	NFP	0.0
6992747	2015-09-02	710	387720.0	0.0	NFP	0.0

6993038	2015-11-12	710	454629.0	0.0	NFP	0.0
7004473	2016-03-26	208	486621.0	0.0	NFP	0.0
7007179	2017-02-07	208	551184.0	0.0	NFP	0.0
7008021	2017-03-04	208	584284.0	0.0	NFP	0.0
7016823	2017-05-20	912	602797.0	0.0	NFP	0.0
7031987	2015-08-03	114	2973693.0	0.0	NFP	0.0
7034388	2016-12-26	114	540625.0	0.0	NFP	0.0
7036607	2017-02-18	114	2133355.0	0.0	NFP	0.0
7059754	2016-12-02	1189	2133652.0	0.0	NFP	0.0
7063579	2016-09-12	1072	513754.0	0.0	NFP	0.0
7065087	2017-02-17	1072	546143.0	0.0	NFP	0.0
7066166	2016-09-14	1072	583476.0	0.0	NFP	0.0
7069482	2015-09-01	388	2989434.0	0.0	NFP	0.0
7071717	2016-04-26	388	2114017.0	0.0	NFP	0.0
7071874	2016-05-18	388	2115717.0	0.0	NFP	0.0
7077667	2017-02-20	388	554287.0	0.0	NFP	0.0
7088708	2017-05-20	1183	600858.0	0.0	NFP	0.0
7118418	2016-05-04	1266	489740.0	0.0	NFP	0.0
7121061	2016-02-11	1266	480863.0	0.0	NFP	0.0
7131298	2015-11-29	1089	2996025.0	0.0	NFP	0.0
7132091	2016-01-29	1089	501536.0	0.0	NFP	0.0
7138152	2017-02-17	1089	571232.0	0.0	NFP	0.0
	2016-					

7151328	04-21	2	485003.0	0.0	NFP	0.0
7153808	2016-05-09	2	469460.0	0.0	NFP	0.0
7157966	2016-07-30	2	534719.0	0.0	NFP	0.0
7159645	2016-12-03	2	553453.0	0.0	NFP	0.0
7170813	2017-02-07	165	592659.0	0.0	NFP	0.0
7177864	2016-09-10	1231	542399.0	0.0	NFP	0.0
7184619	2015-11-18	706	2999730.0	0.0	NFP	0.0
7184882	2016-01-20	706	472076.0	0.0	NFP	0.0
7185409	2016-03-22	706	455501.0	0.0	NFP	0.0
7186341	2016-07-13	706	520593.0	0.0	NFP	0.0
7206271	2017-06-12	135	593830.0	0.0	NFP	0.0
7207942	2017-07-28	135	621383.0	0.0	NFP	0.0
7208618	2017-06-28	135	2154765.0	0.0	NFP	0.0
7210854	2015-09-12	1329	445924.0	0.0	NFP	0.0
7234577	2015-09-05	205	2981696.0	0.0	NFP	0.0
7234742	2015-11-27	205	472183.0	0.0	NFP	0.0
7239055	2017-02-21	205	557629.0	0.0	NFP	0.0
7258272	2015-10-29	297	454207.0	0.0	NFP	0.0
7296015	2017-06-17	72	607226.0	0.0	NFP	0.0
7300862	2016-10-27	1125	532440.0	0.0	NFP	0.0
7314476	2016-07-06	404	2999730.0	0.0	NFP	0.0
7316783	2016-02-04	404	524108.0	0.0	NFP	0.0
7330922	2017-03-02	100	2142620.0	0.0	NFP	0.0

7337398	2016-12-24	54	540625.0	0.0	NFP	0.0
7339858	2016-08-31	54	2140962.0	0.0	FP	0.0
7340709	2017-02-12	54	618306.0	0.0	NFP	0.0
7351204	2016-02-05	677	501056.0	0.0	NFP	0.0
7362983	2015-11-18	542	428672.0	0.0	NFP	0.0
7363288	2016-11-10	542	506188.0	0.0	NFP	0.0
7365392	2016-10-21	542	2124677.0	0.0	NFP	0.0
7375428	2017-02-17	405	570002.0	0.0	NFP	0.0
7382327	2016-11-01	349	2125161.0	0.0	NFP	0.0
7383454	2017-02-03	349	2142802.0	0.0	NFP	0.0
7387399	2015-08-28	1300	436337.0	0.0	NFP	0.0
7390060	2016-10-27	1300	542779.0	0.0	NFP	0.0
7394098	2016-06-14	44	2110668.0	0.0	NFP	0.0
7401218	2016-08-03	44	551184.0	0.0	NFP	0.0
7403520	2017-06-22	44	594747.0	0.0	NFP	0.0
7407122	2015-08-20	203	374512.0	0.0	NFP	0.0
7414702	2015-10-10	1068	454090.0	0.0	NFP	0.0
7415100	2015-11-15	1068	441238.0	0.0	NFP	0.0
7415264	2015-11-13	1068	437103.0	0.0	NFP	0.0
7415317	2015-11-12	1068	436733.0	0.0	NFP	0.0
7415319	2015-11-28	1068	436733.0	0.0	NFP	0.0
7415878	2015-11-25	1068	430447.0	0.0	NFP	0.0
7417972	2016-	1068	531897.0	0.0	NFP	0.0

	07-10					
7419198	2016-10-29	1068	542761.0	0.0	NFP	0.0
7429806	2016-10-24	1168	524959.0	0.0	NFP	0.0
7430062	2016-02-23	1168	525485.0	0.0	FP	0.0
7452159	2016-04-28	1350	473454.0	0.0	NFP	0.0
7452353	2015-11-29	1350	496489.0	0.0	FP	0.0
7452359	2015-11-27	1350	496489.0	0.0	NFP	0.0
7467932	2017-02-17	1347	570127.0	0.0	NFP	0.0
7470735	2017-06-22	1347	601245.0	0.0	NFP	0.0
7472321	2017-07-27	1347	2155481.0	0.0	NFP	0.0
7476980	2017-06-26	1011	608869.0	0.0	NFP	0.0
7481869	2016-11-12	1353	538413.0	0.0	NFP	0.0
7492929	2017-05-18	1311	2147025.0	0.0	NFP	0.0
7494159	2015-10-28	1287	447417.0	0.0	NFP	0.0
7500028	2016-12-01	1287	555862.0	0.0	NFP	0.0
7500230	2017-02-14	1287	2142802.0	0.0	NFP	0.0
7505378	2016-02-15	1312	2100636.0	0.0	NFP	0.0
7507385	2016-10-28	1312	529487.0	0.0	NFP	0.0
7508176	2016-08-19	1312	2126573.0	0.0	NFP	0.0
7508564	2016-12-30	1312	2126995.0	0.0	NFP	0.0
7519520	2016-07-30	1346	546069.0	0.0	NFP	0.0
7534311	2016-03-12	1055	513358.0	0.0	NFP	0.0
7544445	2017-02-11	579	581413.0	0.0	NFP	0.0

7544516	2017-03-23	579	584284.0	0.0	NFP	0.0
7546884	2015-11-22	229	437020.0	0.0	NFP	0.0
7551930	2015-09-01	1048	2978353.0	0.0	NFP	0.0
7552022	2015-11-03	1048	436121.0	0.0	NFP	0.0
7556371	2017-07-22	1048	2138099.0	0.0	NFP	0.0
7557421	2017-06-14	1048	2147025.0	0.0	NFP	0.0
7569266	2016-04-30	439	501056.0	0.0	NFP	0.0
7571135	2015-08-11	439	2101030.0	0.0	NFP	0.0
7594562	2016-02-11	1214	335315.0	0.0	NFP	0.0
7598679	2017-05-11	1214	606228.0	0.0	NFP	0.0
7599238	2017-05-16	1214	624080.0	0.0	NFP	0.0
7618658	2017-02-27	1134	2138909.0	0.0	NFP	0.0
7627422	2015-11-02	576	333153.0	0.0	NFP	0.0
7638282	2016-12-29	1332	2127670.0	0.0	NFP	0.0
7655411	2017-01-21	66	594689.0	0.0	NFP	0.0
7659035	2017-07-29	66	623108.0	0.0	NFP	0.0
7662930	2017-02-18	527	2131136.0	0.0	NFP	0.0
7666388	2015-08-02	319	472076.0	0.0	NFP	0.0
7679956	2015-11-25	1279	2101014.0	0.0	NFP	0.0
7704154	2017-02-25	1030	560219.0	0.0	NFP	0.0
7712720	2017-02-06	1083	2139212.0	0.0	NFP	0.0
7730632	2017-05-15	1280	2151456.0	0.0	NFP	0.0
7747495	2017-06-23	617	451724.0	0.0	NFP	0.0

7750330	2017-03-25	617	560219.0	0.0	NFP	0.0
7756071	2016-06-16	665	491647.0	0.0	NFP	0.0
7756207	2015-11-11	665	143271.0	0.0	NFP	0.0
7792266	2016-05-08	523	479360.0	0.0	NFP	0.0
7797819	2015-11-27	1123	2998948.0	0.0	NFP	0.0
7798742	2015-11-27	1123	2983403.0	0.0	NFP	0.0
7808559	2015-12-31	646	476523.0	0.0	NFP	0.0
7822213	2017-02-21	646	552646.0	0.0	NFP	0.0
7824911	2016-12-23	646	2137455.0	0.0	NFP	0.0
7827877	2017-06-18	646	2147934.0	0.0	NFP	0.0
7835273	2015-12-10	1073	2954404.0	0.0	NFP	0.0
7841055	2016-04-22	1034	501536.0	0.0	NFP	0.0
7862246	2016-05-25	110	545913.0	0.0	NFP	0.0
7864578	2016-04-02	61	480400.0	0.0	NFP	0.0
7865337	2016-01-30	61	501056.0	0.0	NFP	0.0
7865753	2015-11-12	61	458083.0	0.0	NFP	0.0
7867961	2016-08-17	61	2125377.0	0.0	NFP	0.0
7879134	2017-03-27	1235	580571.0	0.0	NFP	0.0
7884940	2015-09-01	1190	423350.0	0.0	NFP	0.0
7894785	2017-07-08	1190	2152413.0	0.0	NFP	0.0
7928820	2017-03-10	325	555987.0	0.0	NFP	0.0
7940192	2015-12-19	146	481424.0	0.0	NFP	0.0
7942969	2016-06-17	146	531590.0	0.0	NFP	0.0

7943308	2016-07-20	146	458174.0	0.0	NFP	0.0
7943624	2016-07-20	146	545442.0	0.0	NFP	0.0
7945499	2016-04-20	499	489831.0	0.0	NFP	0.0
7949000	2016-05-25	499	2127126.0	0.0	NFP	0.0
7954679	2017-02-21	499	2139204.0	0.0	NFP	0.0
7956443	2017-07-07	499	2147918.0	0.0	NFP	0.0
7965038	2017-05-12	1343	600759.0	0.0	NFP	0.0
7969458	2016-12-20	249	554287.0	0.0	NFP	0.0
7978102	2017-01-23	178	588285.0	0.0	NFP	0.0
7979379	2017-05-19	178	591834.0	0.0	NFP	0.0
7985350	2016-10-19	1345	2125153.0	0.0	NFP	0.0
7987538	2016-09-06	1345	2127704.0	0.0	NFP	0.0
7988722	2016-10-19	1345	2132514.0	0.0	NFP	0.0
7989705	2017-02-15	1345	568238.0	0.0	NFP	0.0
8002285	2016-02-15	399	489872.0	0.0	NFP	0.0
8007595	2017-01-27	399	596932.0	0.0	NFP	0.0
8008617	2015-08-15	767	2989426.0	0.0	NFP	0.0
8012656	2016-10-23	767	540278.0	0.0	NFP	0.0
8026290	2016-12-26	492	582098.0	0.0	NFP	0.0
8026332	2017-03-13	492	2138909.0	0.0	NFP	0.0
8028918	2015-08-25	703	398271.0	0.0	NFP	0.0
8044883	2016-04-28	92	2109645.0	0.0	NFP	0.0
	2017-					

8051781	02-18	92	594689.0	0.0	NFP	0.0
8062161	2017-07-28	1233	2150573.0	0.0	NFP	0.0
8070629	2016-08-24	1110	2129247.0	0.0	NFP	0.0
8070901	2017-02-21	1110	2132514.0	0.0	NFP	0.0
8073071	2015-11-04	442	436071.0	0.0	NFP	0.0
8088140	2017-01-20	497	385708.0	0.0	NFP	0.0
8100051	2017-03-25	1354	2133116.0	0.0	NFP	0.0
8107809	2017-04-10	506	552653.0	0.0	NFP	0.0
8149754	2015-08-30	131	2940338.0	0.0	NFP	0.0
8154053	2016-10-27	131	2129247.0	0.0	NFP	0.0
8155907	2017-05-19	131	2140608.0	0.0	NFP	0.0
8156622	2017-02-16	131	2142646.0	0.0	NFP	0.0
8169611	2017-02-04	756	2141010.0	0.0	NFP	0.0
8178794	2016-07-30	234	532481.0	0.0	NFP	0.0
8185422	2015-11-06	140	2100495.0	0.0	NFP	0.0
8196436	2016-11-27	623	533224.0	0.0	NFP	0.0
8214446	2017-02-04	1195	2140590.0	0.0	NFP	0.0
8219036	2016-05-16	82	479352.0	0.0	NFP	0.0
8219118	2015-08-26	82	2973024.0	0.0	NFP	0.0
8229969	2016-08-23	1039	543553.0	0.0	NFP	0.0
8230135	2016-12-09	1039	2132522.0	0.0	NFP	0.0
8230209	2016-12-21	1039	544809.0	0.0	NFP	0.0
8231316	2017-07-24	1039	569251.0	0.0	NFP	0.0

8237904	2016-02-18	777	491647.0	0.0	NFP	0.0
8243569	2017-02-04	777	560052.0	0.0	NFP	0.0
8244940	2016-12-21	777	2133868.0	0.0	NFP	0.0
8245313	2017-02-18	777	581561.0	0.0	NFP	0.0
8245326	2017-03-12	777	565283.0	0.0	NFP	0.0
8258850	2016-06-08	670	487199.0	0.0	NFP	0.0
8259765	2015-08-29	670	454066.0	0.0	NFP	0.0
8262312	2016-11-16	670	2129577.0	0.0	NFP	0.0
8284847	2017-02-15	1299	570481.0	0.0	NFP	0.0
8296709	2016-08-21	1358	511840.0	0.0	NFP	0.0
8311452	2017-02-18	1333	565192.0	0.0	NFP	0.0
8321396	2017-02-12	449	543421.0	0.0	NFP	0.0
8330683	2016-12-14	123	542944.0	0.0	NFP	0.0
8337181	2016-12-30	484	544478.0	0.0	NFP	0.0
8337678	2017-03-20	484	571414.0	0.0	NFP	0.0
8338514	2017-02-04	484	596932.0	0.0	NFP	0.0
8341289	2015-08-05	313	423350.0	0.0	NFP	0.0
8345198	2016-11-16	313	539015.0	0.0	NFP	0.0
8347627	2017-04-29	313	2147330.0	0.0	NFP	0.0
8359751	2016-11-09	590	532390.0	0.0	NFP	0.0
8362886	2017-03-10	590	544569.0	0.0	NFP	0.0
8413423	2017-02-15	1309	2135103.0	0.0	NFP	0.0
8420897	2017-	1186	557843.0	0.0	NFP	0.0

	02-18					
8428236	2016-08-17	1036	545996.0	0.0	NFP	0.0
8431983	2016-01-06	156	512012.0	0.0	NFP	0.0
8432369	2015-09-13	156	422758.0	0.0	NFP	0.0
8432380	2015-12-11	156	2110122.0	0.0	NFP	0.0
8440865	2017-02-18	1340	2135343.0	0.0	NFP	0.0
8440972	2017-01-26	1340	2142802.0	0.0	NFP	0.0
8448854	2017-07-27	1182	2156208.0	0.0	NFP	0.0
8500425	2016-08-24	526	512400.0	0.0	NFP	0.0
8524719	2016-02-15	39	470963.0	0.0	NFP	0.0
8531225	2016-04-28	1027	458380.0	0.0	NFP	0.0
8563783	2015-10-10	1263	455337.0	0.0	NFP	0.0
8565883	2016-11-19	1263	519397.0	0.0	NFP	0.0
8572329	2015-09-26	637	2641068.0	0.0	NFP	0.0
8572630	2015-09-15	637	2101048.0	0.0	NFP	0.0
8580921	2016-12-03	1100	537753.0	0.0	NFP	0.0
8581591	2017-02-14	1100	580282.0	0.0	NFP	0.0
8598585	2015-12-05	1296	489872.0	0.0	NFP	0.0
8616110	2017-02-27	1336	554949.0	0.0	NFP	0.0
8645507	2016-05-24	512	533562.0	0.0	FP	0.0
8652086	2016-05-07	1007	501056.0	0.0	NFP	0.0
8652555	2015-12-05	1007	2999441.0	0.0	NFP	0.0
8662472	2016-05-08	1320	2108233.0	0.0	NFP	0.0

8664875	2017-01-28	1320	2137257.0	0.0	NFP	0.0
8667194	2015-11-27	778	436998.0	0.0	NFP	0.0
8667234	2016-05-19	778	457887.0	0.0	NFP	0.0
8669867	2016-10-24	778	2126854.0	0.0	NFP	0.0
8673570	2016-11-05	2865	2118463.0	0.0	NFP	0.0
8689965	2017-04-09	2945	2150532.0	0.0	NFP	0.0
8710461	2017-02-01	2936	2133116.0	0.0	NFP	0.0
8710507	2017-01-04	2936	2138974.0	0.0	NFP	0.0
8712856	2015-08-01	2837	2982538.0	0.0	NFP	0.0
8718225	2016-01-01	2824	2118737.0	0.0	NFP	0.0
8722137	2017-02-18	2824	2140913.0	0.0	NFP	0.0
8729534	2016-02-13	2860	2121392.0	0.0	NFP	0.0
8731278	2017-02-05	2860	2137455.0	0.0	NFP	0.0
8738358	2016-10-31	2944	2115865.0	0.0	NFP	0.0
8748921	2017-07-28	2912	2154989.0	0.0	NFP	0.0
8761782	2016-12-01	1355	540138.0	0.0	NFP	0.0
8762722	2017-02-12	1355	546119.0	0.0	NFP	0.0
8762805	2016-12-30	1355	2128892.0	0.0	NFP	0.0
8764966	2017-02-17	1355	2138834.0	0.0	NFP	0.0
8764982	2017-02-12	1355	567578.0	0.0	NFP	0.0
8770230	2016-08-09	1356	458042.0	0.0	NFP	0.0
8777540	2016-08-20	1363	513895.0	0.0	NFP	0.0
8798979	2017-01-20	1362	580910.0	0.0	NFP	0.0

8798983	2017-01-19	1362	580910.0	0.0	NFP	0.0
8802499	2016-08-21	1364	2121335.0	0.0	NFP	0.0
8803377	2016-11-26	1364	531889.0	0.0	NFP	0.0
8811859	2017-04-18	1334	2142851.0	0.0	NFP	0.0
8821533	2017-02-20	1322	2142794.0	0.0	NFP	0.0
8838384	2017-03-30	1359	2142794.0	0.0	NFP	0.0

In [76]:
```
# Let's check if there is any negative values for TOTAL_SALES ----4
PTS1[(PTS1['TOTAL_SALES']<0.0)].count()
```

Out[76]:
```
DAY_DT              4
LOC_IDNT            4
DBSKU               4
ONLINE_FLAG         4
FULL_PRICE_IND      4
TOTAL_SALES         4
TOTAL_UNITS         4
TOTAL_SALES_PRFT    4
TOTAL_COST          4
DEPARTMENT          4
CLASS               4
SUBCLASS            4
DEPARTMENT_NAME     4
CLASS_NAME          4
SUBCLASS_NAME       4
CITY                4
STATE               4
STORE_TYPE          4
POSTAL_CD           4
STORE_SIZE          4
dtype: int64
```

In [77]:
```
# Let's take a closer look at them.
# This could simply be clothes returns
PTS1[PTS1.TOTAL_SALES < 0.0]
```

Out[77]:

	DAY_DT	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES
6733593	2016-02-14	4150	2112136.0	1.0	NFP	-39.20
6747913	2016-02-14	4150	2118042.0	1.0	NFP	-81.20
6766287	2016-02-14	4150	2115352.0	1.0	NFP	-212.80
6782052	2016-07-03	4150	2132720.0	1.0	NFP	-29.99

In [78]:
```
# Now let's create the df PTS2 where we will drop the rows where pr
ice = 0 (possibly promo cards)
PTS2 = PTS1.drop(PTS1[(PTS1.TOTAL_SALES == 0.0)].index)
```

In [79]:
```
# Now there is no more 0 value within variable TOTAL_SALES in our d
ataset
PTS2[(PTS2['TOTAL_SALES']==0.0)].count()
```

Out[79]:
```
DAY_DT              0
LOC_IDNT            0
DBSKU               0
ONLINE_FLAG         0
FULL_PRICE_IND      0
TOTAL_SALES         0
TOTAL_UNITS         0
TOTAL_SALES_PRFT    0
TOTAL_COST          0
DEPARTMENT          0
CLASS               0
SUBCLASS            0
DEPARTMENT_NAME     0
CLASS_NAME          0
SUBCLASS_NAME       0
CITY                0
STATE               0
STORE_TYPE          0
POSTAL_CD           0
STORE_SIZE          0
dtype: int64
```

In [80]:
```
PTS2['DAY_DT'].describe()
```

Out[80]:
```
count        8864809
unique           733
top       2016-03-26
freq           44410
Name: DAY_DT, dtype: object
```

In [81]:
```
# Let's transform the date column from Y/M/D to datetime and then a
llocate them onto three different columns
PTS2['DAY_DT'] = pd.to_datetime(PTS2['DAY_DT'])

PTS2['DAY'] = PTS2['DAY_DT'].dt.dayofweek
PTS2['MONTH'] = PTS2['DAY_DT'].dt.month
PTS2['YEAR'] = PTS2['DAY_DT'].dt.year
# Let's drop 'DAY_DT'
PTS2 = PTS2.drop(['DAY_DT'], axis=1)
```

In [82]:
```
# Now it's clear!
PTS2.head()
```

Out[82]:

	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES	TOTAL_UNITS	T
0	1218	466896.0	0.0	NFP	16.80	1.0	
1	1218	466896.0	0.0	NFP	33.60	2.0	
2	1218	466896.0	0.0	NFP	21.00	1.0	
3	1218	466896.0	0.0	NFP	14.70	1.0	
4	1218	412445.0	0.0	NFP	29.99	1.0	

In [83]:
```
PTS2.shape
```

Out[83]:
```
(8864809, 22)
```

In [84]:
```
# TOTAL PROFIT
PTS.TOTAL_SALES_PRFT.sum()
```

Out[84]:
```
205731168.30480006
```

In [85]:
```
# COSTS
PTS.TOTAL_COST.sum()
```

Out[85]:
```
176290681.5152003
```

In [86]:
```
# REVENUE
PTS.TOTAL_SALES.sum()
```

Out[86]: 382021849.81999946

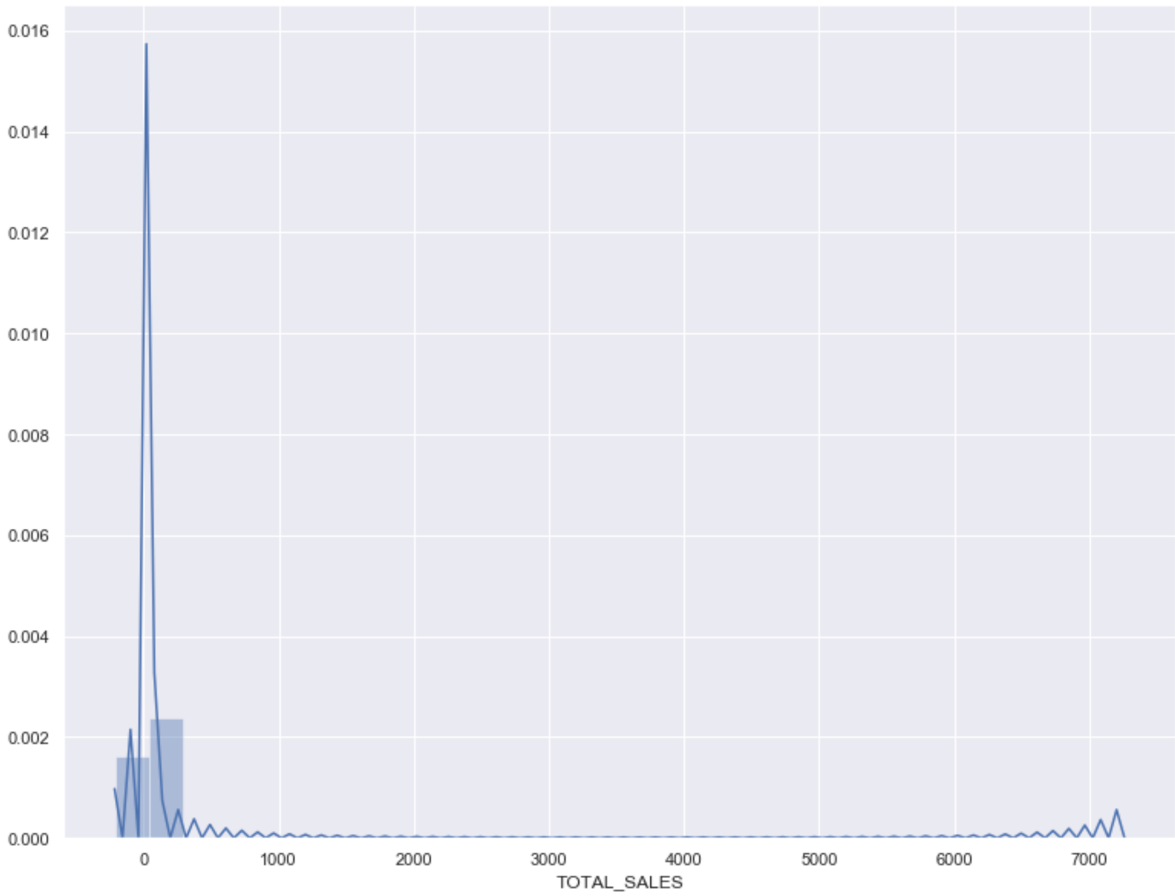
In [87]:
```
# % of costs on our total REVENUE
(PTS.TOTAL_COST.sum() / PTS.TOTAL_SALES.sum()) *100
```

Out[87]: 46.14675354256954

In [88]:
```
# TOTAL SALES MARGIN
(PTS.TOTAL_SALES.sum() - PTS.TOTAL_COST.sum()) / PTS.TOTAL_SALES.su
m()
```

Out[88]: 0.5385324645743046

In [89]:
```
# Before going ahead with the variables correlation, let's check th
e target variable 'TOTAL_SALES' distribution
# The distribution is high POSITIVELY SKEWED
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

sns.set(rc={'figure.figsize':(13,10)})
sns.distplot(PTS2['TOTAL_SALES'], bins=30)
plt.show(sns)
```



PTS2.to_csv(r'\\10.0.7.226\ipba_group10\merged_dataset.csv')

OUTLIERS -Treatment

In [1]:
```
import pandas as pd
import numpy as np
import os
```

In [37]:
```
os.chdir = (r'\\10.0.7.226\ipba_group10')
```

In [38]:
```
os.getcwd()
```

Out[38]:
```
'C:\\Users\\IPBAB047'
```

In [39]:
```
PTS2 = pd.read_csv(r'\\10.0.7.226\ipba_group10\merged_dataset.csv')
```

In [40]:
```
PTS2.columns
```

Out[40]:
```
Index(['Unnamed: 0', 'LOC_IDNT', 'DBSKU', 'ONLINE_FLAG', 'FULL_PRI
CE_IND',
       'TOTAL_SALES', 'TOTAL_UNITS', 'TOTAL_SALES_PRFT', 'TOTAL_CO
ST',
       'DEPARTMENT', 'CLASS', 'SUBCLASS', 'DEPARTMENT_NAME', 'CLAS
S_NAME',
       'SUBCLASS_NAME', 'CITY', 'STATE', 'STORE_TYPE', 'POSTAL_C
D',
       'STORE_SIZE', 'DAY', 'MONTH', 'YEAR'],
      dtype='object')
```

In [6]:
```
PTS2.drop('Unnamed: 0', axis=1, inplace=True)
```

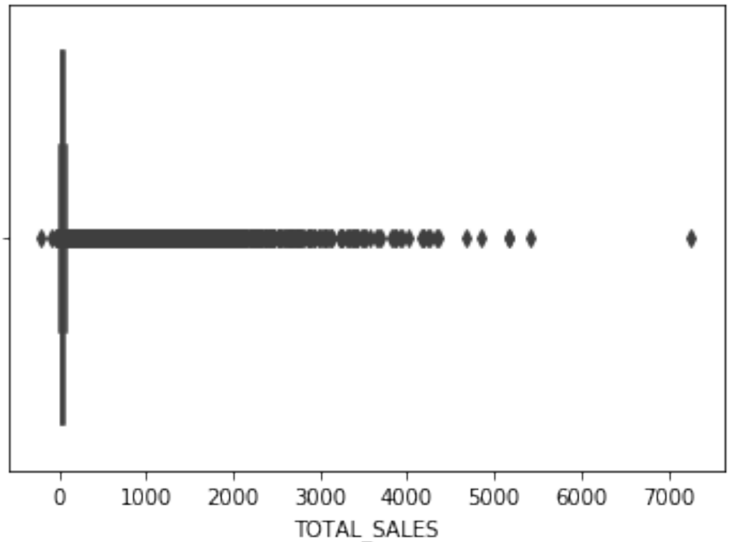
In [7]:
```
PTS2.head()
```

Out[7]:

	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES	TOTAL_UNITS	T(
0	1218	466896.0	0.0	NFP	16.80	1.0	
1	1218	466896.0	0.0	NFP	33.60	2.0	
2	1218	466896.0	0.0	NFP	21.00	1.0	
3	1218	466896.0	0.0	NFP	14.70	1.0	
4	1218	412445.0	0.0	NFP	29.99	1.0	

5 rows × 22 columns

In [8]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
# Let's check the boxplot for our variable (target variable) TOTAL_
SALES
# Not highly comprhensible
sns.boxplot(PTS2['TOTAL_SALES'])
plt.show(sns)
```

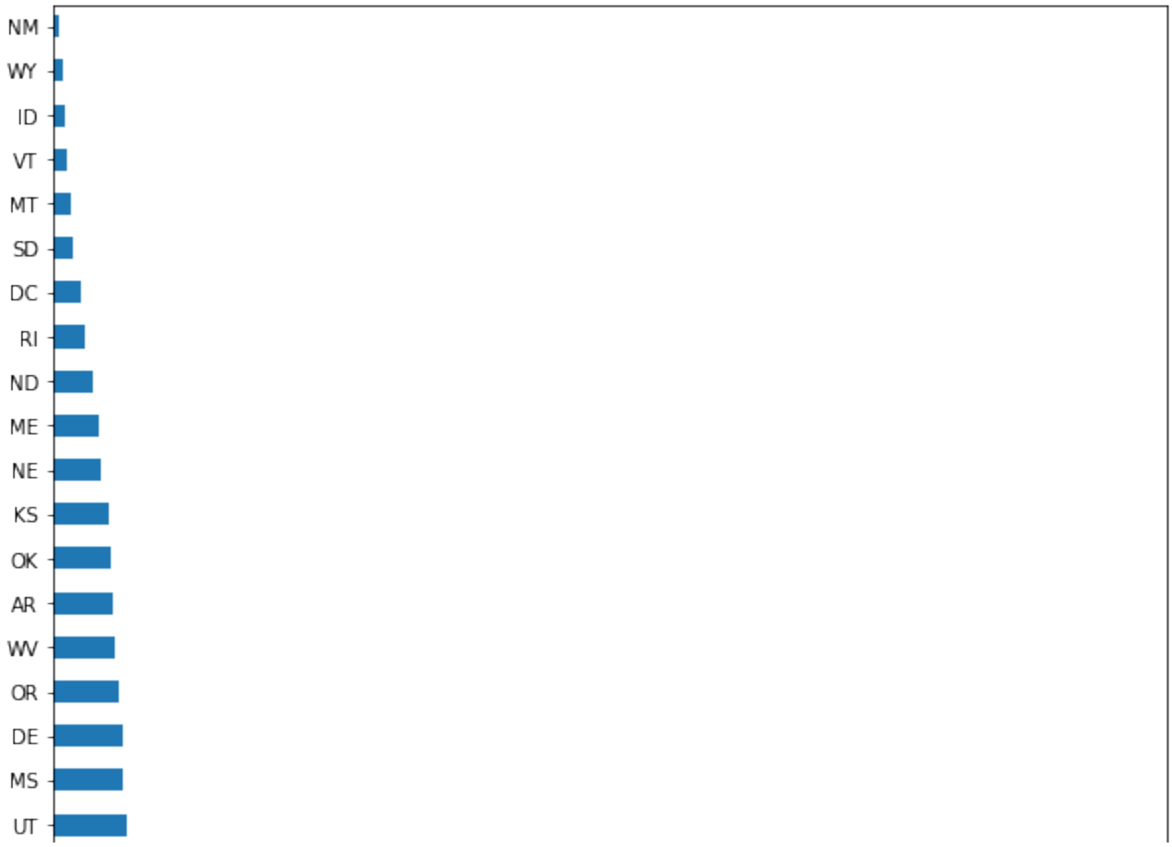


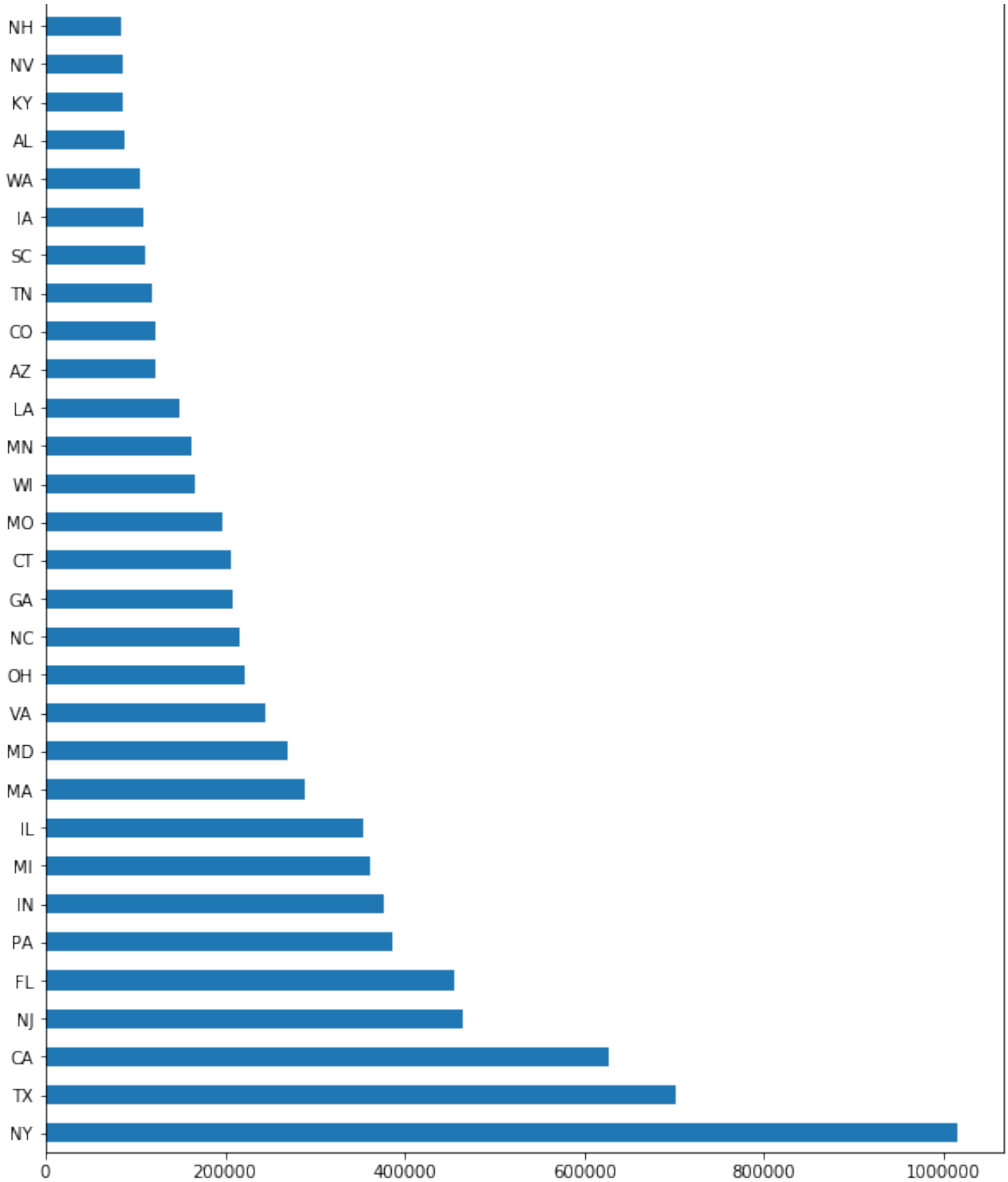
In [9]:
```
# How many transactions happen per STATE?
PTS2['STATE'].value_counts().head(20)
```

Out[9]:
```
NY    1016173
TX     702042
CA     627822
NJ     463954
FL     455096
PA     386292
IN     376191
MI     361231
IL     353432
MA     288887
MD     269163
VA     244073
OH     221005
NC     216596
GA     207816
CT     207507
MO     197978
WI     165540
MN     162962
LA     149233
Name: STATE, dtype: int64
```

In [10]:
```
# Let's see how many transactions happen per STATE
PTS2['STATE'].value_counts().plot(kind='barh', figsize=(10,20))
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x9804823d08>



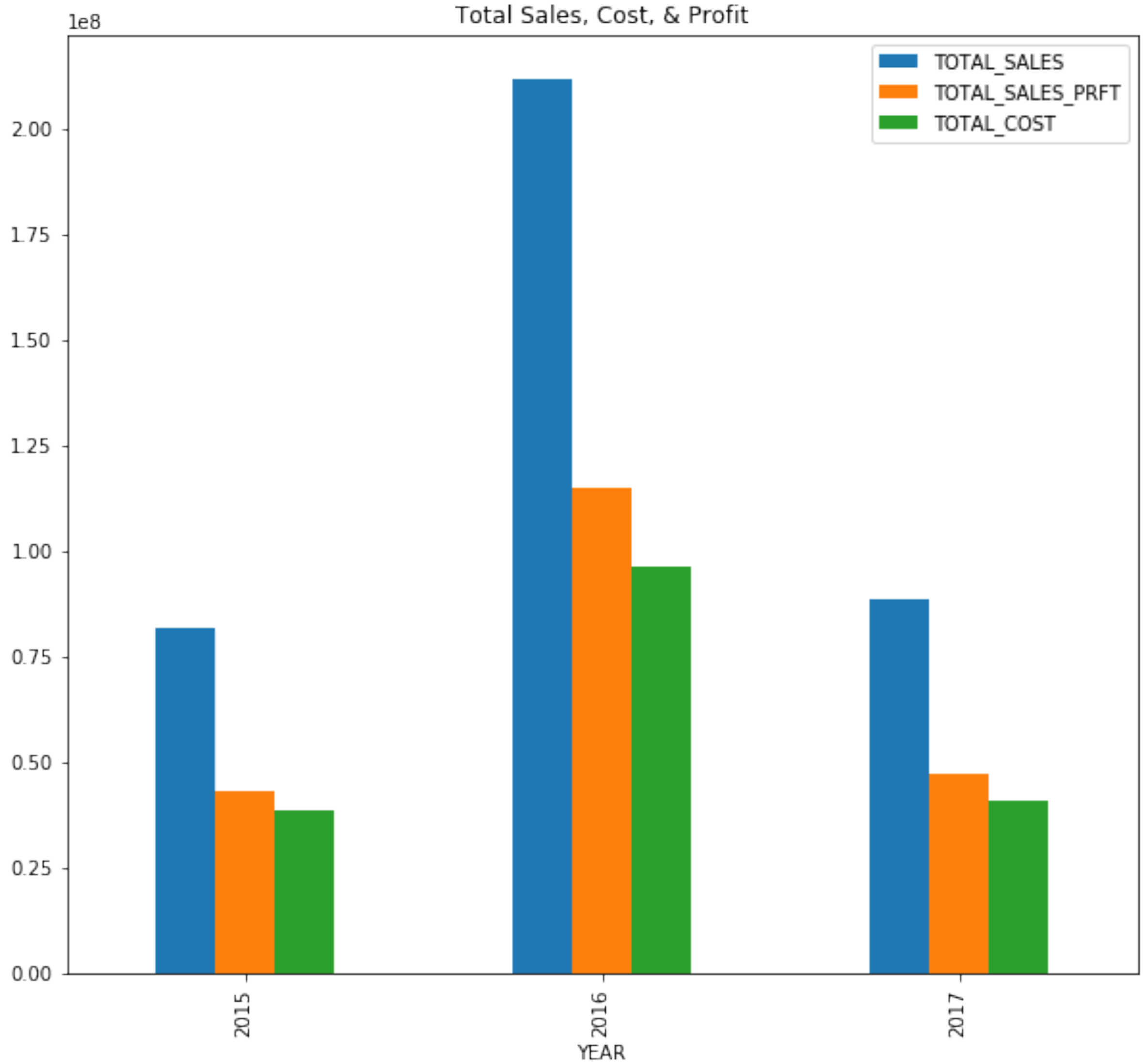


Let's see the distribution of sales during the years # In JULY-AUG seems to see a peak which is then intermittent every other month from February to June, with then another small peak in December # January is the poorest month in terms of sales (because of Christmas) PTS2['DAY_DT'].value_counts().sort_index().plot(figsize=(15,10))

```
In [ ]:
```

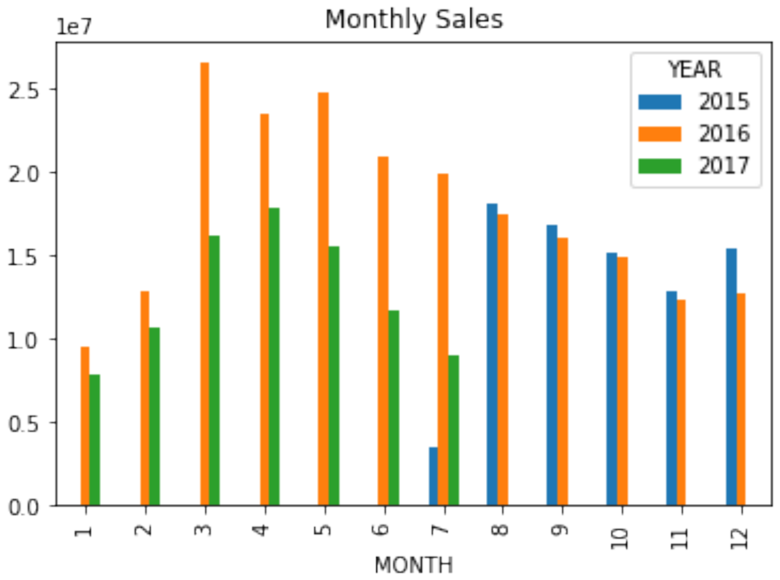
In [11]:
```
PTS2.groupby(['YEAR'])['TOTAL_SALES','TOTAL_SALES_PRFT','TOTAL_COS
T'].sum().plot(kind='bar', title = 'Total Sales, Cost, & Profit',fi
gsize=(10,9))
```

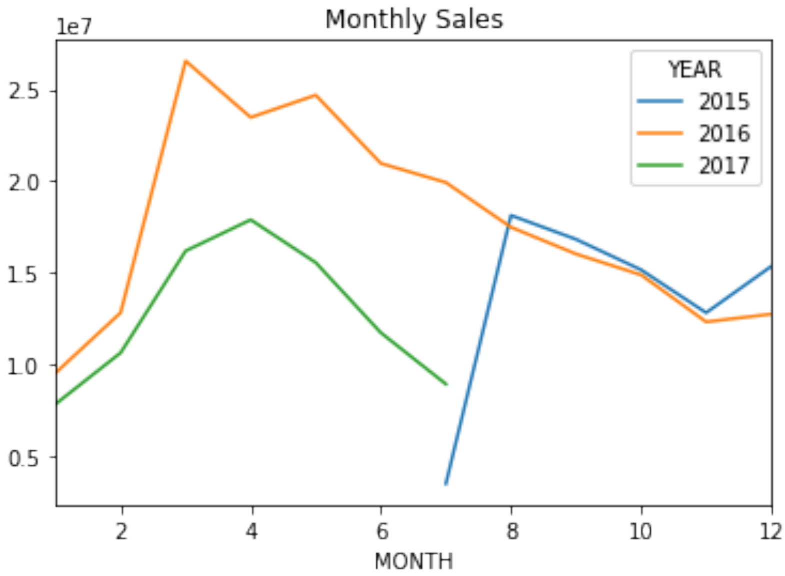
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x9804daa0c8>



In [12]:
```
PTS2.pivot_table(index='MONTH',columns='YEAR',values='TOTAL_SALES',
aggfunc=np.sum).plot(kind='bar',title='Monthly Sales')
PTS2.pivot_table(index='MONTH',columns='YEAR',values='TOTAL_SALES',
aggfunc=np.sum).plot(kind='line',title='Monthly Sales')
```

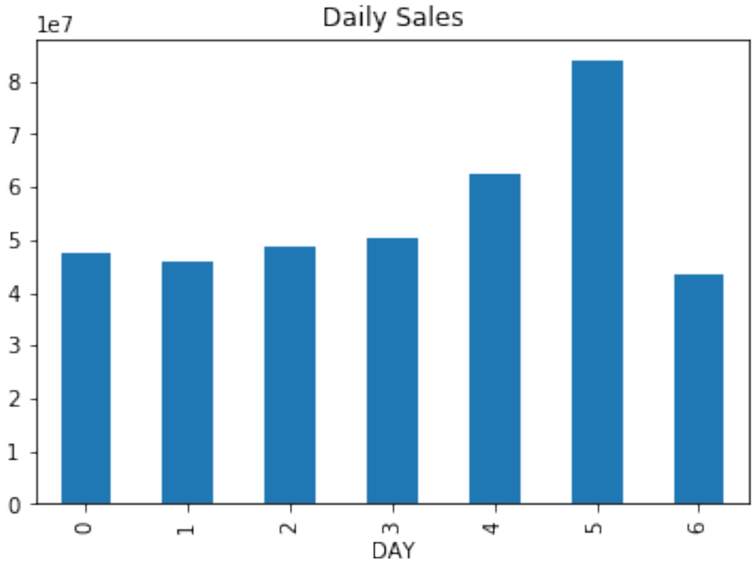
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x9804a6e088>





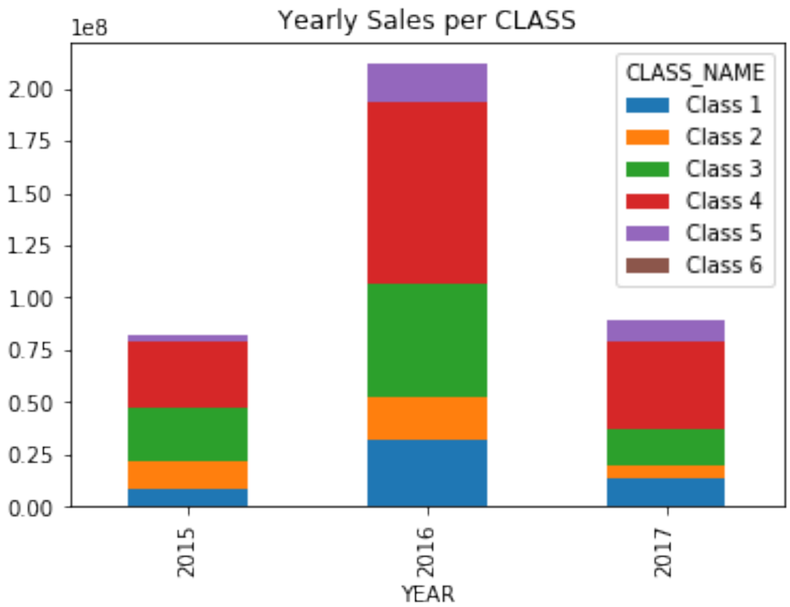
In [13]:
```
# Daily Sales Revenue
PTS2.groupby(['DAY'])['TOTAL_SALES'].sum().plot.bar(title = 'Daily Sales')
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x9804b17348>



In [14]:
```
# Yearly Sales by CLASS
PTS2.pivot_table(index='YEAR',columns='CLASS_NAME', values = 'TOTAL_SALES', aggfunc=np.sum).plot.bar(stacked=True, title='Yearly Sales per CLASS')
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x9804bc0a08>

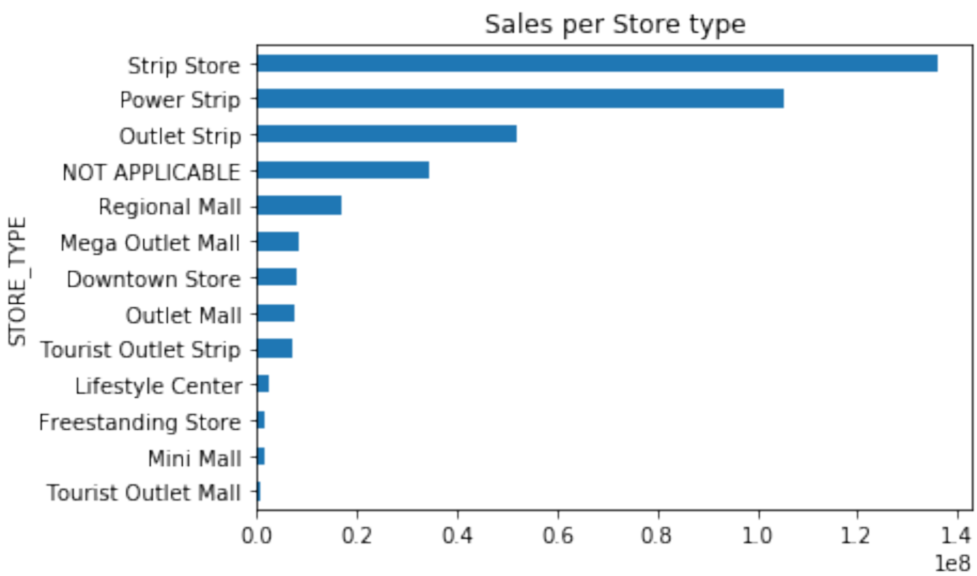


In [15]:
```
# Let's use groupby and aggregate function to check Sales per class
online_sales = PTS2.groupby(['CLASS']).agg({'TOTAL_SALES': ['mea
n','median','min', 'max','count']})
print(online_sales)
```

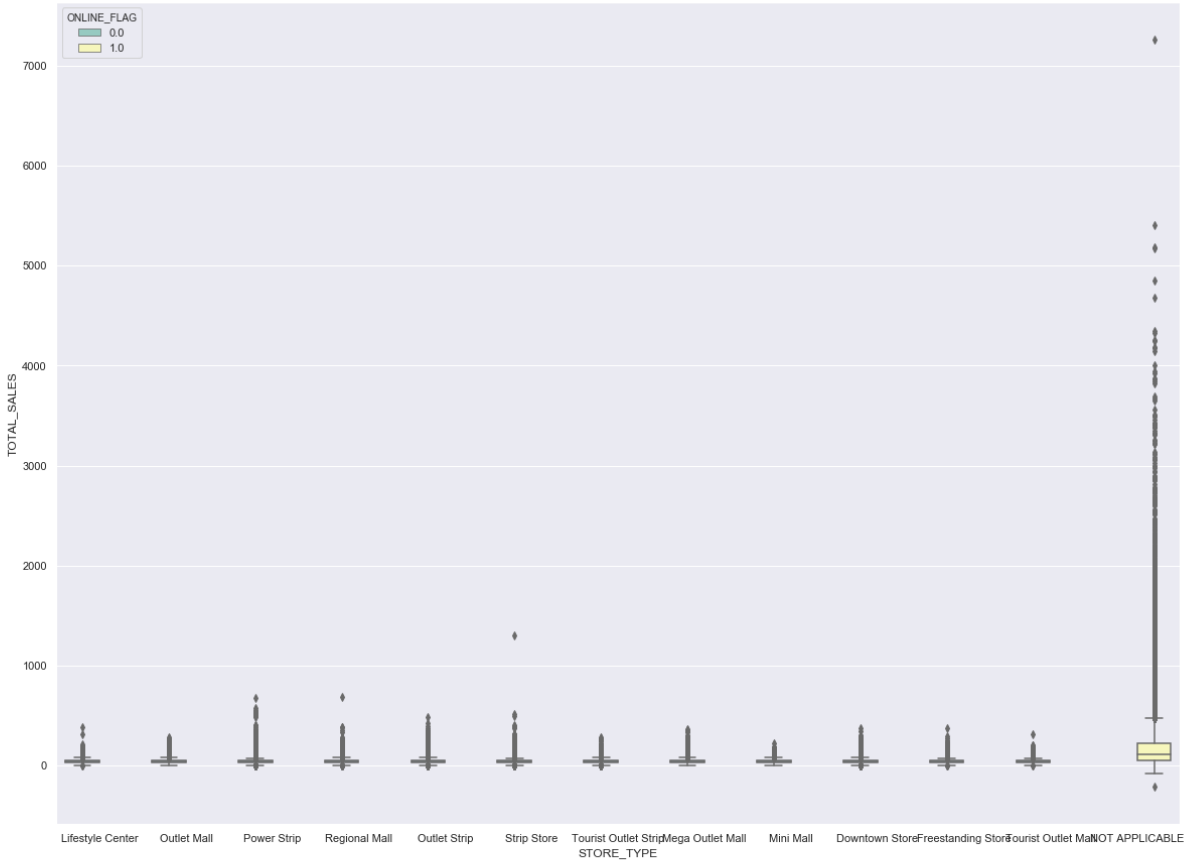
```
      TOTAL_SALES
             mean median     min      max    count
CLASS
1       42.254260  39.50    0.01  4179.90   709914
2       42.189061  39.22  -29.99  5177.76  3814913
3       41.795039  39.20 -212.80  7260.45  1304619
4       49.441494  46.00    0.01  3382.38  1948128
5       37.040563  36.80    0.04  2234.02  1084803
99      20.840851  17.00    0.40   510.03     2432
```

In [16]:
```
# TOTAL_SALES per store type
PTS2.groupby(['STORE_TYPE'])['TOTAL_SALES'].sum().sort_values(ascen
ding=True).plot.barh(title='Sales per Store type')
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x980371d0c8>



In [17]:
```
# Previously I have checked the boxplot for TOTAL_SALES, but I couldn't get a clear visual outlook from it,
# So now I am checking the boxplots of TOTAL_SALES for STORE_TYPE
# Most outliers reside where STORE_TYPE = NOT_APPLICABLE
# It is highly likely that STORE = NOT APPLICABLE means ONLINE_STORE
sns.set(rc={'figure.figsize':(20,15)})
ax = sns.boxplot(x="STORE_TYPE", y="TOTAL_SALES", hue='ONLINE_FLAG', data=PTS2, palette="Set3")
```



In [18]:
```
PTS2['STORE_TYPE'] = PTS2['STORE_TYPE'].str.replace('NOT APPLICABLE','Online Store')
```

In [19]:
```
PTS2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8864809 entries, 0 to 8864808
Data columns (total 22 columns):
LOC_IDNT             int64
DBSKU                float64
ONLINE_FLAG          float64
FULL_PRICE_IND       object
TOTAL_SALES          float64
TOTAL_UNITS          float64
TOTAL_SALES_PRFT     float64
TOTAL_COST           float64
DEPARTMENT           int64
CLASS                int64
SUBCLASS             int64
DEPARTMENT_NAME      object
CLASS_NAME           object
SUBCLASS_NAME        object
CITY                 object
STATE                object
STORE_TYPE           object
POSTAL_CD            int64
STORE_SIZE           float64
DAY                  int64
MONTH                int64
YEAR                 int64
dtypes: float64(7), int64(8), object(7)
memory usage: 1.5+ GB
```

In [20]:
```
# Let's use groupby and aggregate function to check the mean sales price per store type
online_sales = PTS2.groupby(['STORE_TYPE']).agg({'TOTAL_SALES': ['mean','median','min', 'max','count']})
print(online_sales)
```

```
                          TOTAL_SALES
                                 mean    median     min      max     coun
t
STORE_TYPE
Downtown Store              42.786805    40.770    0.01   378.00    18789
7
Freestanding Store          39.512671    39.000    0.60   378.36     4288
1
Lifestyle Center            39.143496    39.100    0.23   386.78     6092
2
Mega Outlet Mall            41.749411    40.000    0.01   367.50    19766
0
Mini Mall                   38.709461    38.450    0.10   220.68     3890
3
Online Store               178.907234   111.425 -212.80  7260.45    19164
4
Outlet Mall                 42.127229    41.210    0.01   280.00    17933
2
Outlet Strip                40.815010    40.000    0.01   480.00   126920
9
Power Strip                 39.558208    39.500    0.01   672.00   266557
0
Regional Mall               40.472389    39.980    0.01   691.20    41875
5
Strip Store                 39.806737    39.500    0.01  1297.51   342090
1
Tourist Outlet Mall         42.311518    40.500    0.98   318.00     2038
8
Tourist Outlet Strip        41.408703    40.000    0.01   280.00    17074
7
```

In [21]:

```
# Descriptive Statistics of our Price
PTS2['TOTAL_SALES'].describe().apply(lambda x: format(x, 'f'))
```

Out[21]:

```
count    8864809.000000
mean          43.094200
std           40.885621
min         -212.800000
25%           29.990000
50%           39.500000
75%           48.250000
max         7260.450000
Name: TOTAL_SALES, dtype: object
```

In [22]:

```
# TOTAL_SALES Expected Values
# Expected Maximum Value is 75% value + (1.5*IQR)
print("Expected Max Value -->", 48+(1.5*20))
# Expected Minimum Value is 25% value - (1.5*IQR)
print("Expected Min Value -->", 28-(1.5*20))
```

```
Expected Max Value --> 78.0
Expected Min Value --> -2.0
```

In [23]:
```
# TOTAL_SALES_PRFT Expected Values
# Expected Maximum Value is 75% value + (1.5*IQR)
print("Expected Max Value -->", 30.36+(1.5*20.29))
# Expected Minimum Value is 25% value - (1.5*IQR)
print("Expected Min Value -->", 10.07-(1.5*20.29))
```

```
Expected Max Value --> 60.795
Expected Min Value --> -20.365
```

In [24]:
```
# TOTAL_COST Expected Values
# Expected Maximum Value is 75% value + (1.5*IQR)
print("Expected Max Value -->", 20+(1.5*5))
# Expected Minimum Value is 25% value - (1.5*IQR)
print("Expected Min Value -->", 15-(1.5*5))
```

```
Expected Max Value --> 27.5
Expected Min Value --> 7.5
```

In [25]:
```
# Let's see from the 10th percentile to the 90th, how theprice rang
e looks like
PTS2['TOTAL_SALES'].quantile([.1,.2,.3,.4,.5,.6,.7,.8,.9])
```

Out[25]:
```
0.1    20.00
0.2    26.00
0.3    31.96
0.4    36.17
0.5    39.50
0.6    43.60
0.7    47.05
0.8    51.60
0.9    59.50
Name: TOTAL_SALES, dtype: float64
```

In [26]:
```
# Let's see from the 90th to the 99th percentile, how theprice rang
e looks like
PTS2['TOTAL_SALES'].quantile([.91,.92,.93,.94,.95,.96,.97,.98,.99])
```

Out[26]:
```
0.91     60.0000
0.92     62.0000
0.93     64.0000
0.94     67.1400
0.95     68.4000
0.96     74.9100
0.97     84.0000
0.98    100.0000
0.99    138.8692
Name: TOTAL_SALES, dtype: float64
```

In [27]:
```
# Let's see from 0.1 the 1st percentile, how theprice range looks l
ike
PTS2['TOTAL_SALES'].quantile([.01,.02,.03,.04,.05,.06,.07,.08,.09])
```

Out[27]:
```
0.01    10.20
0.02    12.96
0.03    14.40
0.04    15.40
0.05    16.67
0.06    17.60
0.07    18.40
0.08    19.42
0.09    19.99
Name: TOTAL_SALES, dtype: float64
```

In [ ]:

In [28]:
```
# Let's create a random sample to be able to manage a more size-to-
code Dataset
PTS2_sample = PTS2.sample(frac=0.1, replace=False, random_state=1)
```

In [29]:
```
PTS2_sample.shape
```

Out[29]:
```
(886481, 22)
```

In [30]:
```
PTS2_sample.head()
```

Out[30]:

	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES	TOTAL_UI
1071942	519	460923.0	0.0	NFP	17.25	
7511745	1346	534180.0	0.0	NFP	37.40	
7111738	141	555375.0	0.0	NFP	38.40	
6705496	4150	2105957.0	1.0	FP	291.79	
5828618	679	2131243.0	0.0	NFP	44.50	

5 rows × 22 columns

In [31]:
```
PTS2_sample.info()
```

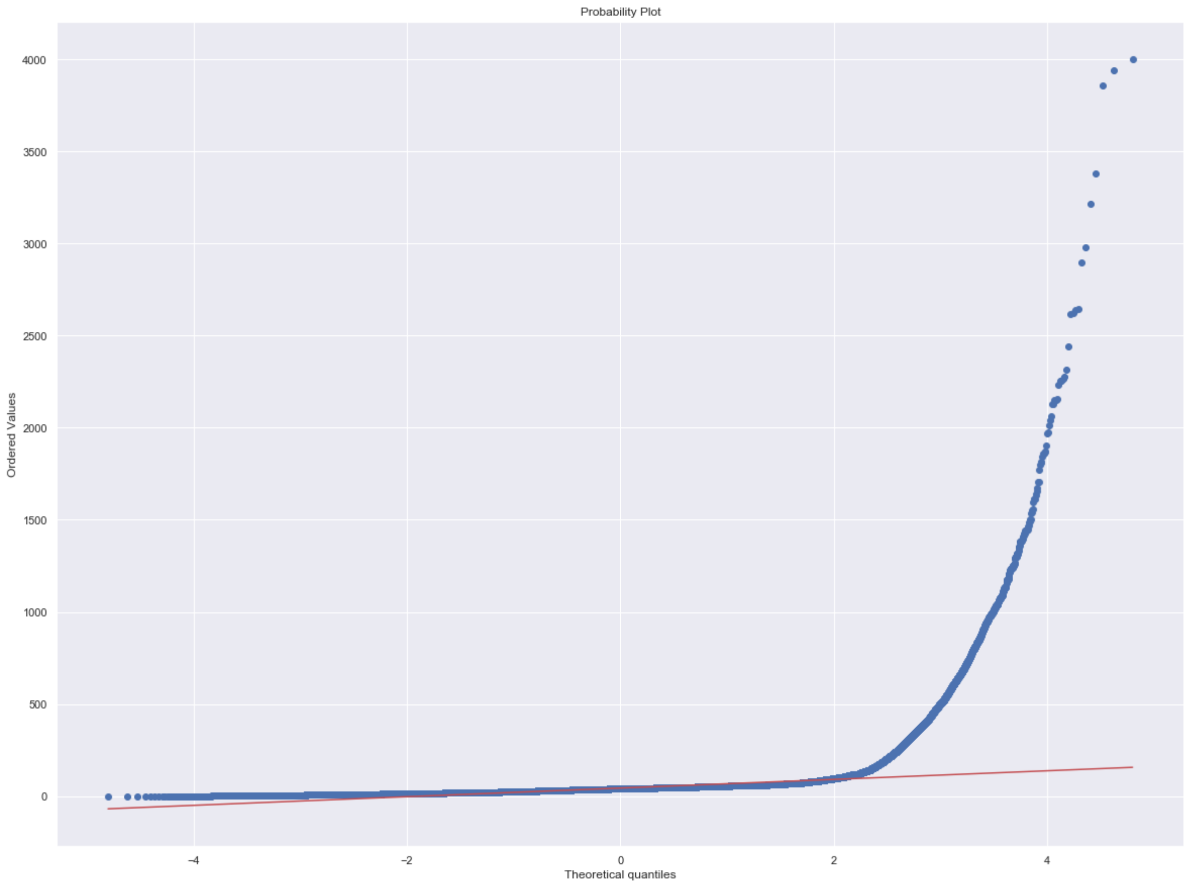
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 886481 entries, 1071942 to 4738873
Data columns (total 22 columns):
LOC_IDNT             886481 non-null int64
DBSKU                886481 non-null float64
ONLINE_FLAG          886481 non-null float64
FULL_PRICE_IND       886481 non-null object
TOTAL_SALES          886481 non-null float64
TOTAL_UNITS          886481 non-null float64
TOTAL_SALES_PRFT     886481 non-null float64
TOTAL_COST           886481 non-null float64
DEPARTMENT           886481 non-null int64
CLASS                886481 non-null int64
SUBCLASS             886481 non-null int64
DEPARTMENT_NAME      886481 non-null object
CLASS_NAME           886481 non-null object
SUBCLASS_NAME        886481 non-null object
CITY                 886481 non-null object
STATE                886481 non-null object
STORE_TYPE           886481 non-null object
POSTAL_CD            886481 non-null int64
STORE_SIZE           886481 non-null float64
DAY                  886481 non-null int64
MONTH                886481 non-null int64
YEAR                 886481 non-null int64
dtypes: float64(7), int64(8), object(7)
memory usage: 155.6+ MB
```

In [32]:
```
# LET'S CHECK SOME RATIO FOR THE ORIGINAL DATASET VS. SAMPLED DATASET (10% its original size)
```

In [33]:
```
PTS2_sample['TOTAL_SALES'].describe().apply(lambda x: format(x, 'f'))
```

Out[33]:
```
count    886481.000000
mean         43.128205
std          41.118505
min           0.010000
25%          29.990000
50%          39.500000
75%          48.370000
max        4002.750000
Name: TOTAL_SALES, dtype: object
```

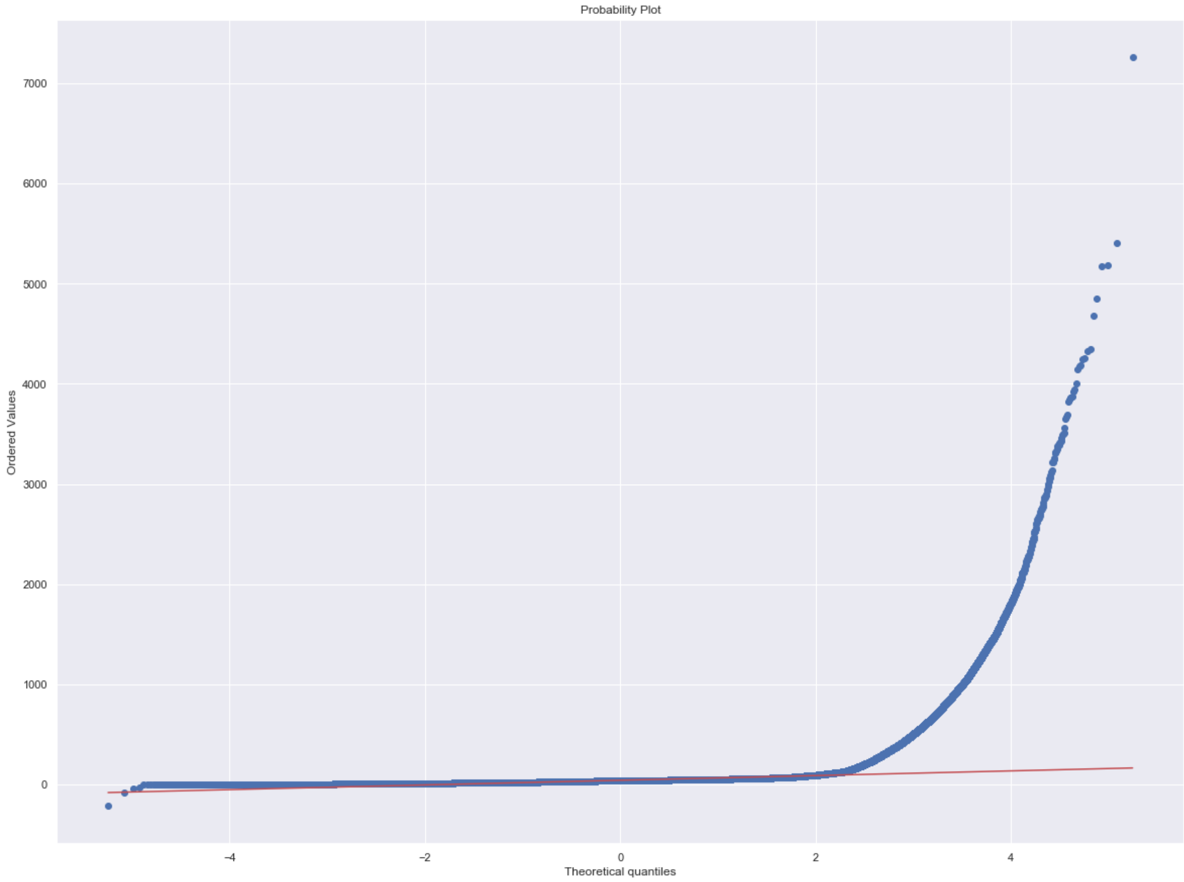
In [34]:
```
# TOTAL_SALES QQplot
from scipy import stats

fig = plt.figure()
res = stats.probplot(PTS2_sample['TOTAL_SALES'], plot=plt)
plt.show()
```



In [35]:
```
PTS2['TOTAL_SALES'].describe().apply(lambda x: format(x, 'f'))
```

Out[35]:
```
count    8864809.000000
mean          43.094200
std           40.885621
min         -212.800000
25%           29.990000
50%           39.500000
75%           48.250000
max         7260.450000
Name: TOTAL_SALES, dtype: object
```

In [36]:
```
# TOTAL_SALES QQplot - Original dataset

fig = plt.figure()
res = stats.probplot(PTS2['TOTAL_SALES'], plot=plt)
plt.show()
```



In [37]:
```
PTS2_sample['TOTAL_SALES_PRFT'].describe().apply(lambda x: format(
x, 'f'))
```

Out[37]:
```
count    886481.000000
mean         23.230413
std          25.664029
min       -1329.000000
25%          11.300000
50%          23.050000
75%          31.000000
max        2552.510000
Name: TOTAL_SALES_PRFT, dtype: object
```

In [38]:
```
PTS2['TOTAL_SALES_PRFT'].describe().apply(lambda x: format(x, 'f'))
```

Out[38]:
```
count    8864809.000000
mean          23.212073
std           25.623458
min        -4378.260000
25%           11.300000
50%           23.000000
75%           31.000000
max         4656.610000
Name: TOTAL_SALES_PRFT, dtype: object
```

In [39]:
```
PTS2_sample['TOTAL_COST'].describe().apply(lambda x: format(x, '
f'))
```

Out[39]:
```
count    886481.000000
mean         19.897792
std          18.964798
min         -13.013600
25%          15.150000
50%          17.500000
75%          20.500000
max        1800.000000
Name: TOTAL_COST, dtype: object
```

In [40]:
```
PTS2['TOTAL_COST'].describe().apply(lambda x: format(x, 'f'))
```

Out[40]:
```
count    8864809.000000
mean          19.882127
std           19.137562
min         -102.500000
25%           15.150000
50%           17.500000
75%           20.500000
max         5796.000000
Name: TOTAL_COST, dtype: object
```

In [41]:
```
PTS2_sample.isnull().sum()
```

Out[41]:
```
LOC_IDNT             0
DBSKU                0
ONLINE_FLAG          0
FULL_PRICE_IND       0
TOTAL_SALES          0
TOTAL_UNITS          0
TOTAL_SALES_PRFT     0
TOTAL_COST           0
DEPARTMENT           0
CLASS                0
SUBCLASS             0
DEPARTMENT_NAME      0
CLASS_NAME           0
SUBCLASS_NAME        0
CITY                 0
STATE                0
STORE_TYPE           0
POSTAL_CD            0
STORE_SIZE           0
DAY                  0
MONTH                0
YEAR                 0
dtype: int64
```

In [42]:
```
# Let's check the IQR for the following variables on the sampled da
taset
Q1s = PTS2_sample['TOTAL_SALES'].quantile(0.25)
Q3s = PTS2_sample['TOTAL_SALES'].quantile(0.75)
IQRs = Q3s -Q1s
print(IQRs)
print(Q1s)
print(Q3s)
```

```
18.38
29.99
48.37
```

In [43]:
```
# TOTAL_SALES Expected Values
# Expected Maximum Value is 75% value + (1.5*IQR)
print("Expected Max Value -->", 48+(1.5*20))
# Expected Minimum Value is 25% value - (1.5*IQR)
print("Expected Min Value -->", 28-(1.5*20))
```

```
Expected Max Value --> 78.0
Expected Min Value --> -2.0
```

In [44]:
```
# Let's get rid of the outliers for the variable TOTAL_SALES follow
ing the INTERQUARTILE RANGE's outliers detection method
PTS2_sample_out = PTS2_sample.loc[(PTS2_sample['TOTAL_SALES'] > -2)
& (PTS2_sample['TOTAL_SALES'] < 78)]
```

In [45]:
```
PTS2_sample_out.shape
```

Out[45]:
```
(853217, 22)
```

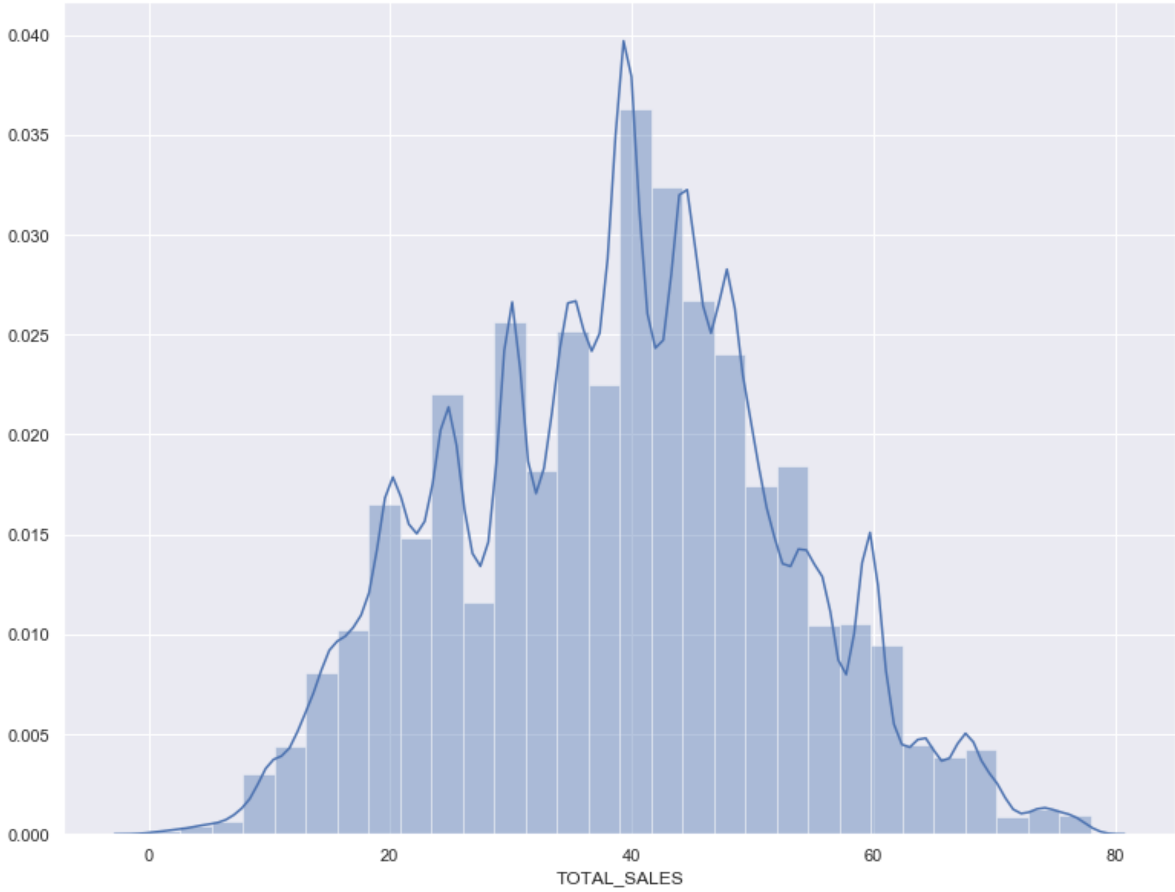
In [46]:
```
PTS2_sample_out['TOTAL_SALES'].describe()
```

Out[46]:
```
count    853217.000000
mean         38.704082
std          13.605004
min           0.010000
25%          29.620000
50%          39.500000
75%          48.000000
max          77.990000
Name: TOTAL_SALES, dtype: float64
```

In [47]:
```
# How many rows have been dropped?

print("Acual Number of Rows -->", PTS2_sample.shape[0])
print("Number of Rows after treatment -->", PTS2_sample_out.shape[
0])
print("Number of Records dropped -->", PTS2_sample.shape[0] - PTS2_
sample_out.shape[0])
```

```
Acual Number of Rows --> 886481
Number of Rows after treatment --> 853217
Number of Records dropped --> 33264
```

In [48]:
```
# Let's visualize the distribution of TOTAL_SALES after removing th
e outliers
sns.set(rc={'figure.figsize':(13,10)})
sns.distplot(PTS2_sample_out['TOTAL_SALES'], bins=30)
plt.show(sns)
```



In [ ]:

In [49]:
```
# Let's get rid of the outliers for the variable TOTAL_SALES_PRFT f
ollowing the INTERQUARTILE RANGE's outliers detection method
PTS2_sample_out1 = PTS2_sample_out.loc[(PTS2_sample_out['TOTAL_SALE
S_PRFT'] > -20.125) & (PTS2_sample_out['TOTAL_SALES_PRFT'] < 58.87
5)]
```

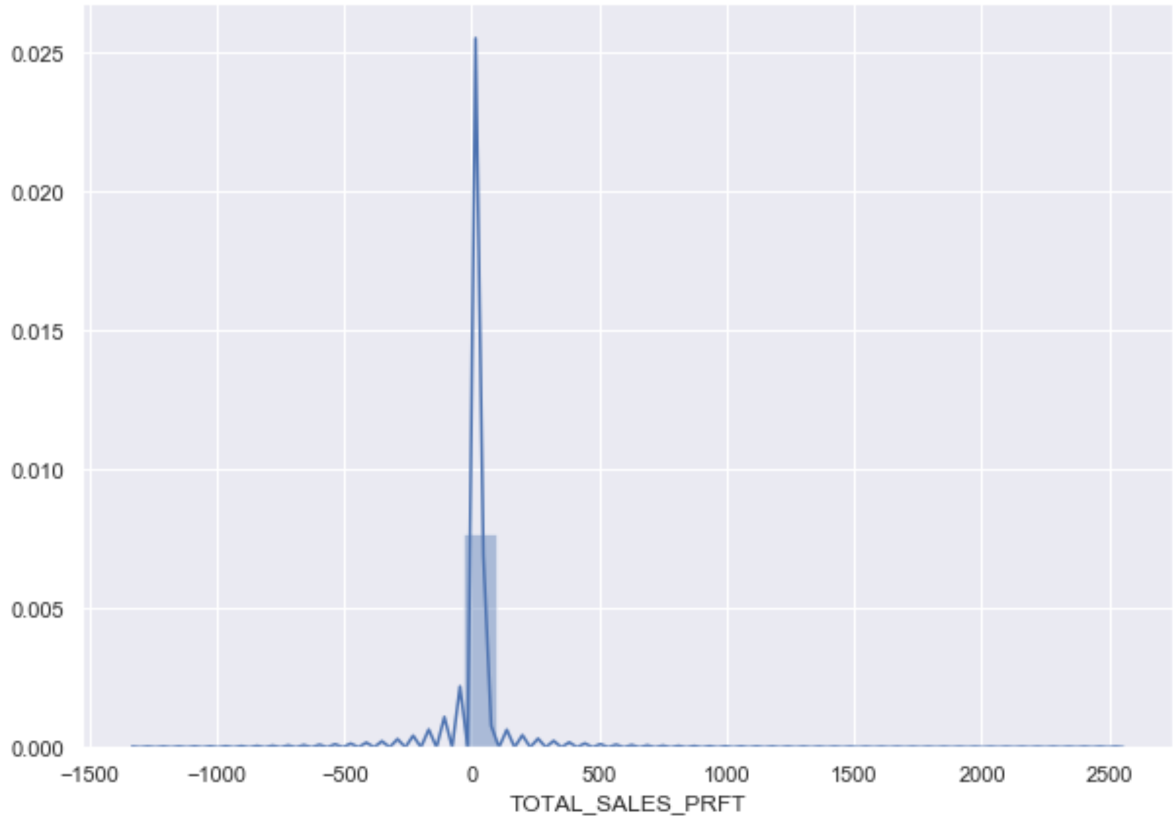
In [50]:
```
PTS2_sample_out1.shape
```

Out[50]:
```
(852910, 22)
```

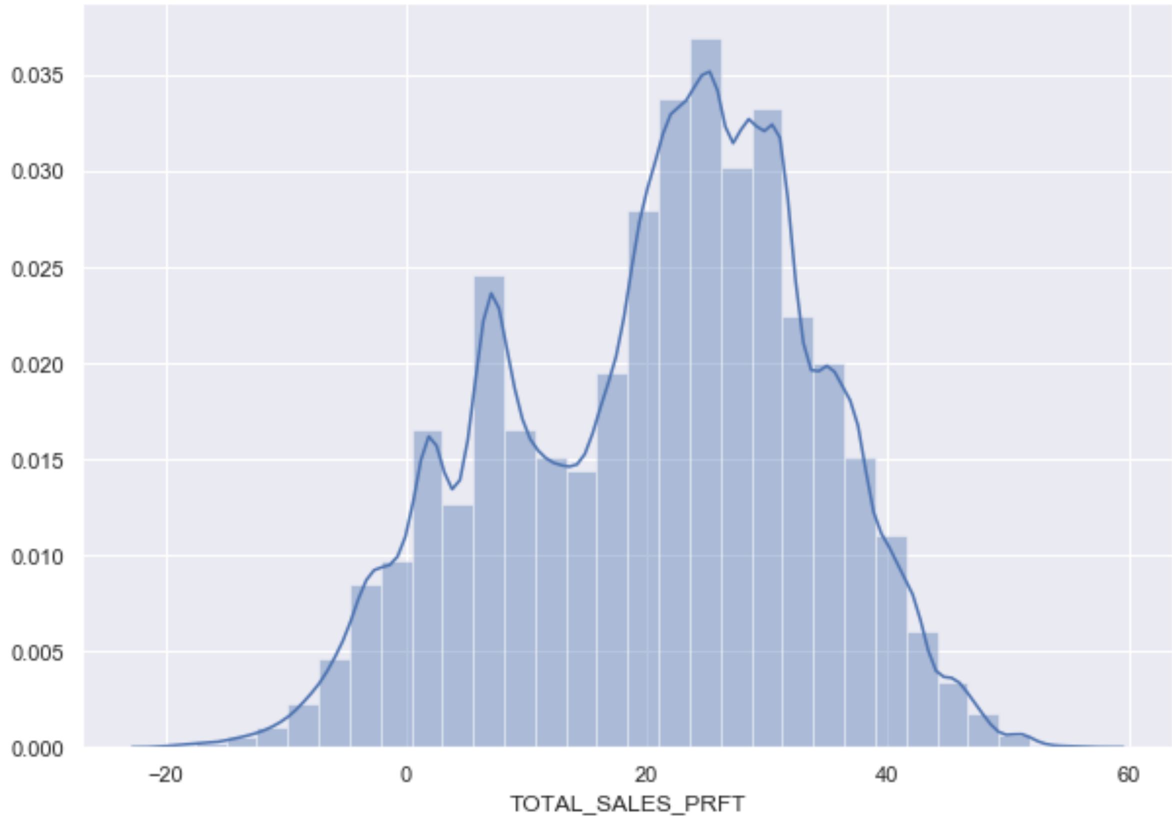
In [51]:
```
# How many rows have been dropped?

print("Acual Number of Rows -->", PTS2_sample.shape[0])
print("Number of Rows after treatment -->", PTS2_sample_out1.shape[
0])
print("Number of Records dropped -->", PTS2_sample.shape[0] - PTS2_
sample_out1.shape[0])
```

```
Acual Number of Rows --> 886481
Number of Rows after treatment --> 852910
Number of Records dropped --> 33571
```

In [58]:
```
# Let's visualize the distribution of TOTAL_SALES BEFORE removing t
he outliers
sns.set(rc={'figure.figsize':(10,7)})
sns.distplot(PTS2_sample['TOTAL_SALES_PRFT'], bins=30)
plt.show(sns)
```



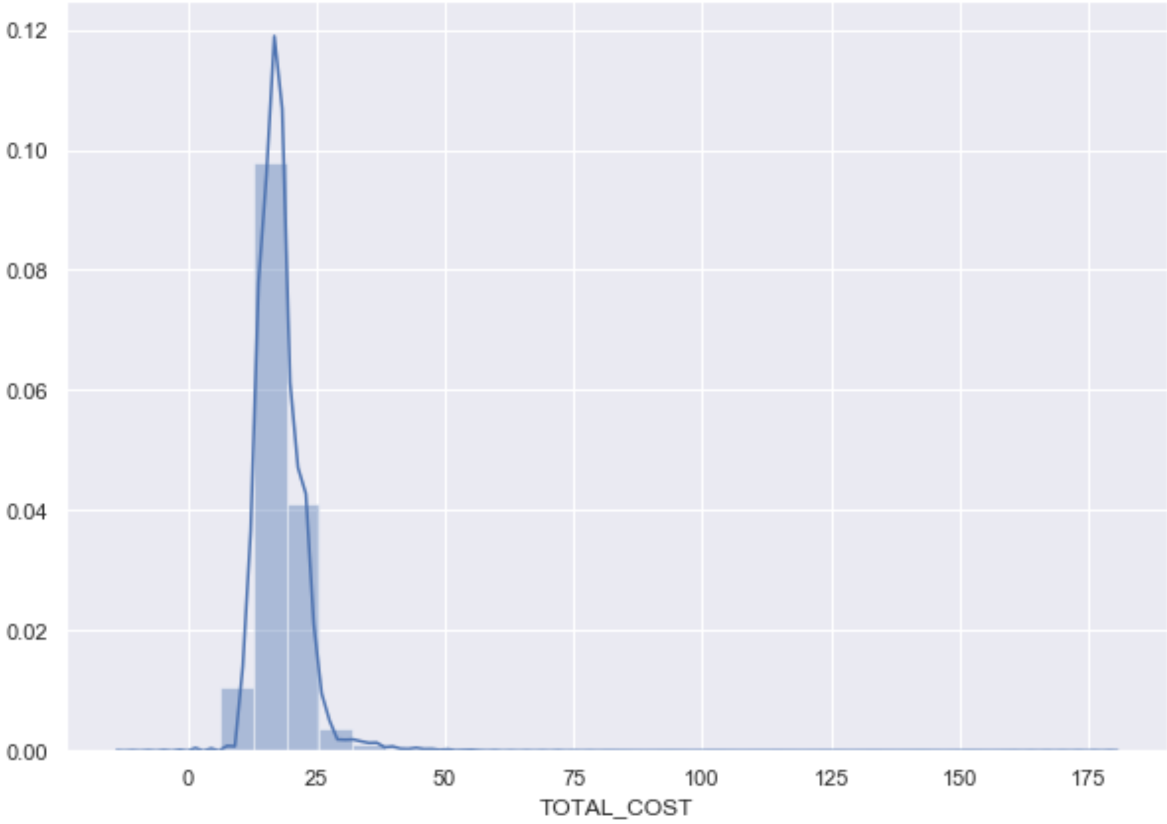
In [59]:
```
# Let's visualize the distribution of TOTAL_SALES after removing th
e outliers
sns.set(rc={'figure.figsize':(10,7)})
sns.distplot(PTS2_sample_out1['TOTAL_SALES_PRFT'], bins=30)
plt.show(sns)
```



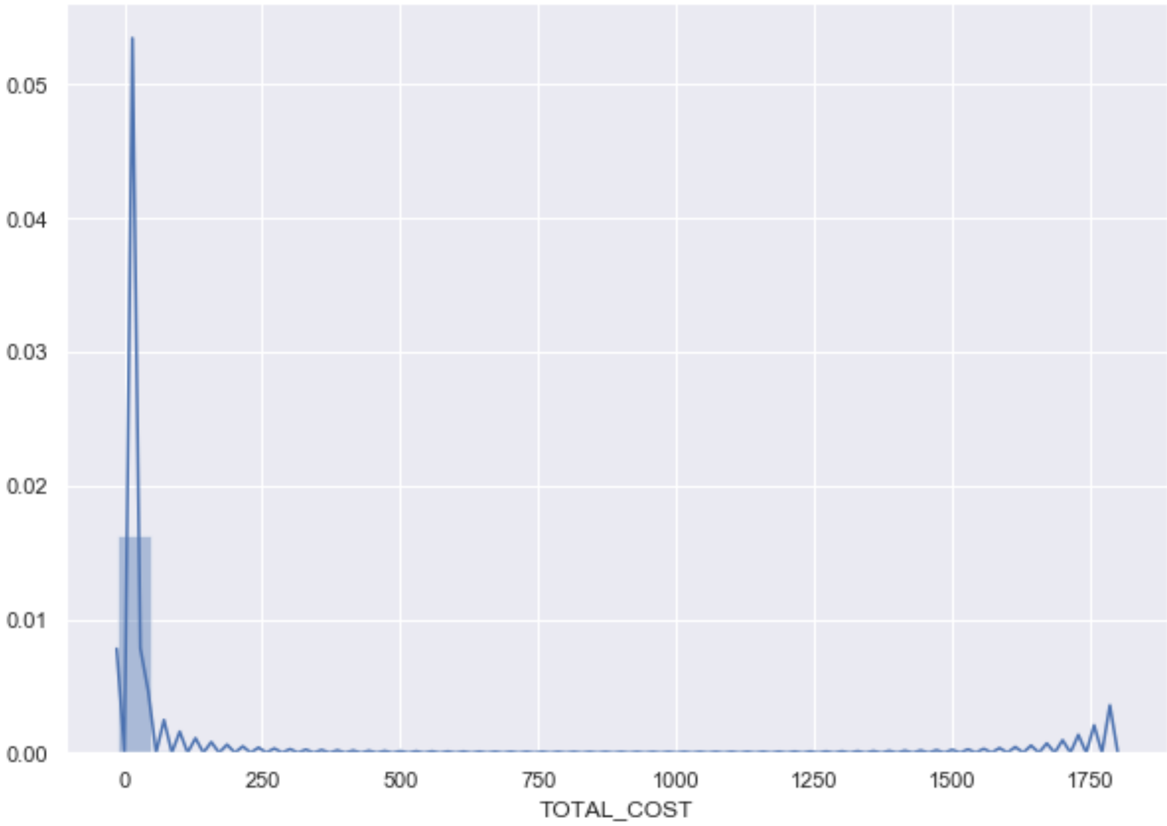
In [60]:
```
PTS2_sample_out1['TOTAL_SALES_PRFT'].describe()
```

Out[60]:
```
count    852910.000000
mean         20.719264
std          12.688727
min         -20.110000
25%          10.700000
50%          22.500000
75%          30.000000
max          56.940000
Name: TOTAL_SALES_PRFT, dtype: float64
```

In [61]:
```
# Let's visualize the distribution of TOTAL_COST when TOTAL_SALES o
utliers have been removed
sns.set(rc={'figure.figsize':(10,7)})
sns.distplot(PTS2_sample_out['TOTAL_COST'], bins=30)
plt.show(sns)
```



In [62]:
```
# Let's visualize the distribution of TOTAL_COST when TOTAL_SALES a
nd TOTAL_SALES_PRFT' outliers have been removed
sns.set(rc={'figure.figsize':(10,7)})
sns.distplot(PTS2_sample['TOTAL_COST'], bins=30)
plt.show(sns)
```



In [63]:
```
PTS2_sample['TOTAL_COST'].describe()
```

Out[63]:
```
count    886481.000000
mean         19.897792
std          18.964798
min         -13.013600
25%          15.150000
50%          17.500000
75%          20.500000
max        1800.000000
Name: TOTAL_COST, dtype: float64
```

In [64]:
```
# Let's check the IQR for the following variable (TOTAL_COST)
Q1c = PTS2_sample_out1['TOTAL_COST'].quantile(0.25)
Q3c = PTS2_sample_out1['TOTAL_COST'].quantile(0.75)
IQRc = Q3c -Q1c
print(IQRc)
print(Q1c)
print(Q3c)
```

```
5.0
15.0
20.0
```

In [65]:
```
# TOTAL_COST Expected Values
# Expected Maximum Value is 75% value + (1.5*IQR)
print("Expected Max Value -->", 19.75+(1.5*4.75))
# Expected Minimum Value is 25% value - (1.5*IQR)
print("Expected Min Value -->", 15-(1.5*4.75))
```

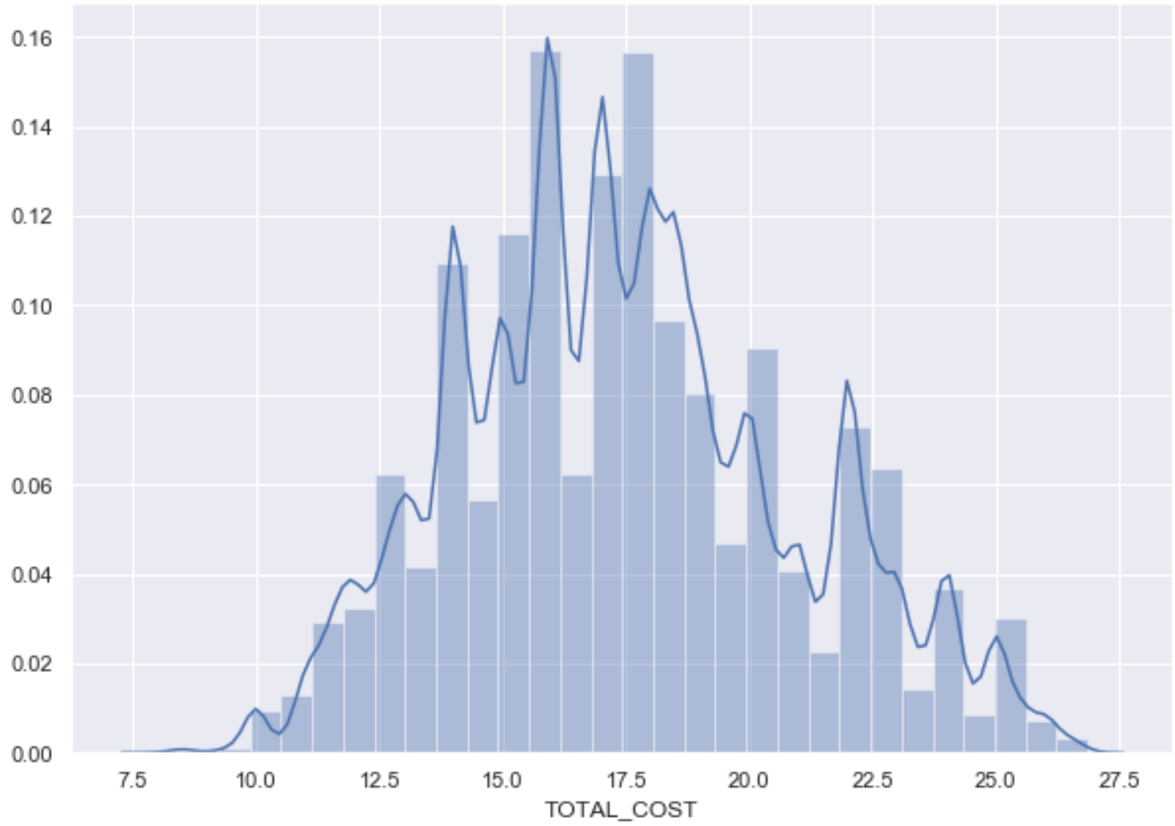
```
Expected Max Value --> 26.875
Expected Min Value --> 7.875
```

In [66]:
```
# Let's delete TOTAL_COST' outliers by using the IQR method again
PTS2_sample_out2 = PTS2_sample_out1.loc[(PTS2_sample_out1['TOTAL_CO
ST'] > 7.875) & (PTS2_sample_out1['TOTAL_COST'] < 26.875)]
```

In [67]:
```
PTS2_sample_out2.shape
```

Out[67]:
```
(825414, 22)
```

In [68]:
```
# Let's visually check the distribution of TOTAL_COST after deletio
n of outliers
sns.set(rc={'figure.figsize':(10,7)})
sns.distplot(PTS2_sample_out2['TOTAL_COST'], bins=30)
plt.show(sns)
```



In [69]:
```
PTS2_sample_out2['TOTAL_COST'].describe()
```

Out[69]:
```
count    825414.000000
mean         17.515583
std           3.411517
min           8.000000
25%          15.000000
50%          17.000000
75%          19.750000
max          26.854700
Name: TOTAL_COST, dtype: float64
```

In [70]:
```
# How many rows have been dropped?

print("Acual Number of Rows -->", PTS2_sample.shape[0])
print("Number of Rows after treatment -->", PTS2_sample_out2.shape[
0])
print("Number of Records dropped -->", PTS2_sample.shape[0] - PTS2_
sample_out2.shape[0])
```

```
Acual Number of Rows --> 886481
Number of Rows after treatment --> 825414
Number of Records dropped --> 61067
```

In [71]:
```
# Let's add to our dataset non-cumulative continuous variables such
as 'UNIT_PRICE', 'UNIT_SALES_PRFT', 'UNIT_COST'
```

In [72]:
```
PTS2_sample_out2['UNIT_PRICE'] = (PTS2_sample_out2['TOTAL_SALES'] /
PTS2_sample_out2['TOTAL_UNITS'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  """Entry point for launching an IPython kernel.
```

In [73]:
```
PTS2_sample_out2['UNIT_SALES_PRFT'] = (PTS2_sample_out2['TOTAL_SALE
S_PRFT'] / PTS2_sample_out2['TOTAL_UNITS'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  """Entry point for launching an IPython kernel.
```

In [74]:
```
PTS2_sample_out2['UNIT_COST'] = (PTS2_sample_out2['TOTAL_COST'] / PTS2_sample_out2['TOTAL_UNITS'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [75]:
```
PTS2_sample_out2.head()
```

Out[75]:

	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	TOTAL_SALES	TOTAL_UI
1071942	519	460923.0	0.0	NFP	17.25	
7511745	1346	534180.0	0.0	NFP	37.40	
7111738	141	555375.0	0.0	NFP	38.40	
5828618	679	2131243.0	0.0	NFP	44.50	
5947704	130	538835.0	0.0	FP	40.00	

5 rows × 25 columns

In [76]:
```
PTS2_sample_out2.shape
```

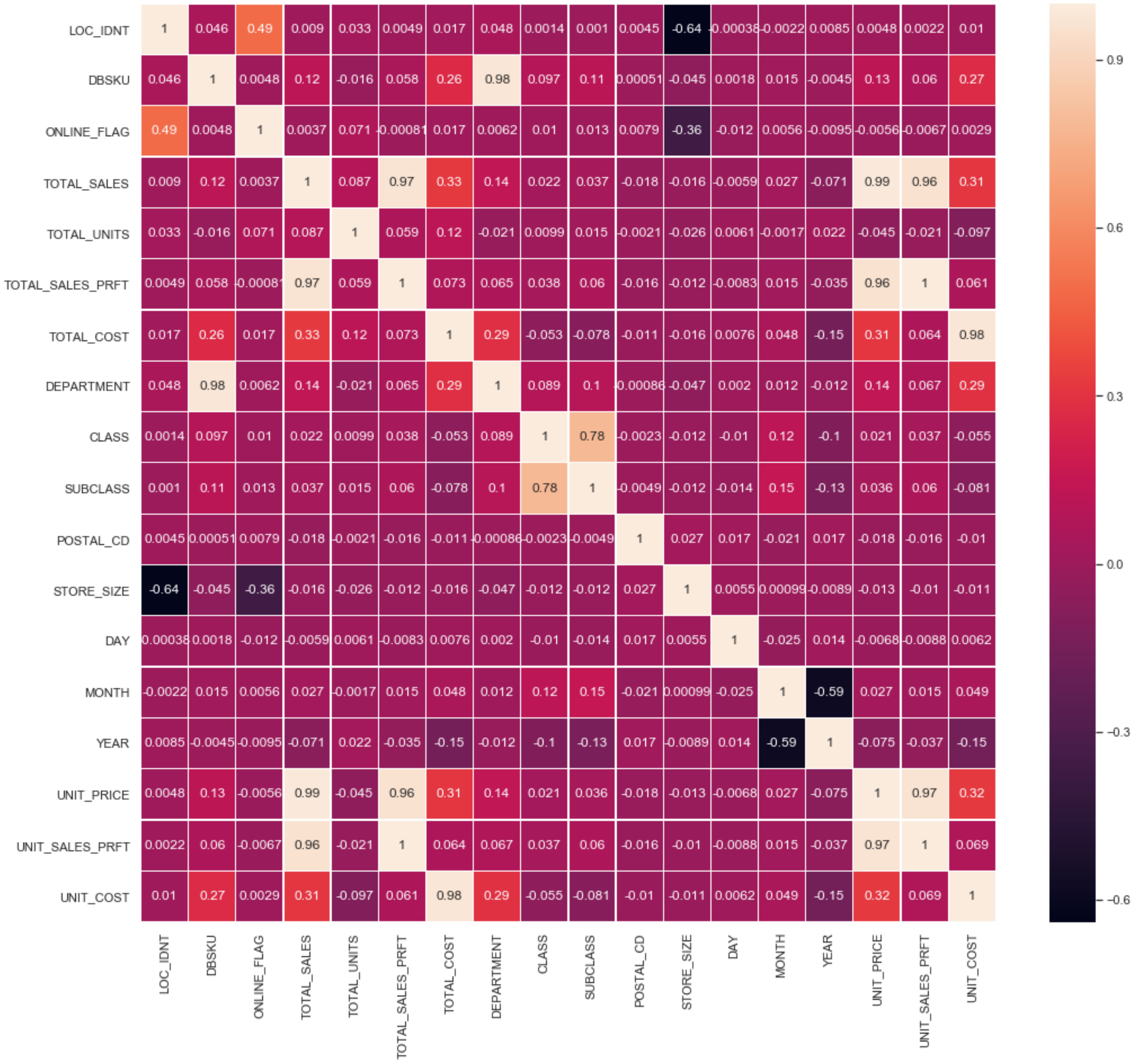
Out[76]:
```
(825414, 25)
```

VARIABLES CORRELATION

In [77]:
```
# Let's take a look at the heatmap for the sampled dataset where we have removed outliers from its target variable (TOTAL_SALES),
#TOTAL_SALES_PRFT, TOTAL_COST
# At this stage I still have not deleted the above mentioned variables as they represent a mere copy (considering what are we interested in),
# of the newly generated variables: UNIT_PRICE, UNIT_SALES_PRFT, UNIT_COST
import seaborn as sns
corr1 = PTS2_sample_out2.corr()
```

In [78]:
```
# The only highly correlated variable to our target (UNIT_PRICE | T
OTAL_SALES) is UNIT_SALES_PRFT | TOTAL_SALES_PRFT
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(17,15))
sns.heatmap(corr1, annot=True, linewidths=.3,ax=ax,
            xticklabels=corr1.columns.values,
            yticklabels=corr1.columns.values)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[78]: (18.0, 0.0)



In [ ]:

In [79]:
```
# Let's create a random sample to be able to manage a more size-to-code Dataset
# WE ARE SAMPLING A DATASET STARTING FROM THE PREVIOUSLY SAMPLED DATASET

PTS2_sample_small = PTS2_sample_out2.sample(frac=.04, replace=False, random_state=2)
```

In [80]:
```
PTS2_sample_small.shape
```

Out[80]:
```
(33017, 25)
```

In [81]:
```
# The descriptive statstics of our target variable looks incredibly similar to that of both the previously sampled dataset, and
# the original one
PTS2_sample_small['UNIT_PRICE'].describe()
```

Out[81]:
```
count    33017.000000
mean        38.290550
std         13.205841
min          0.010000
25%         29.400000
50%         39.290000
75%         48.000000
max         76.000000
Name: UNIT_PRICE, dtype: float64
```

In [98]:
```
PTS2_sample_small['TOTAL_SALES'].describe()
```

Out[98]:
```
count    33017.000000
mean        38.412249
std         13.275186
min          0.010000
25%         29.400000
50%         39.500000
75%         48.000000
max         77.800000
Name: TOTAL_SALES, dtype: float64
```

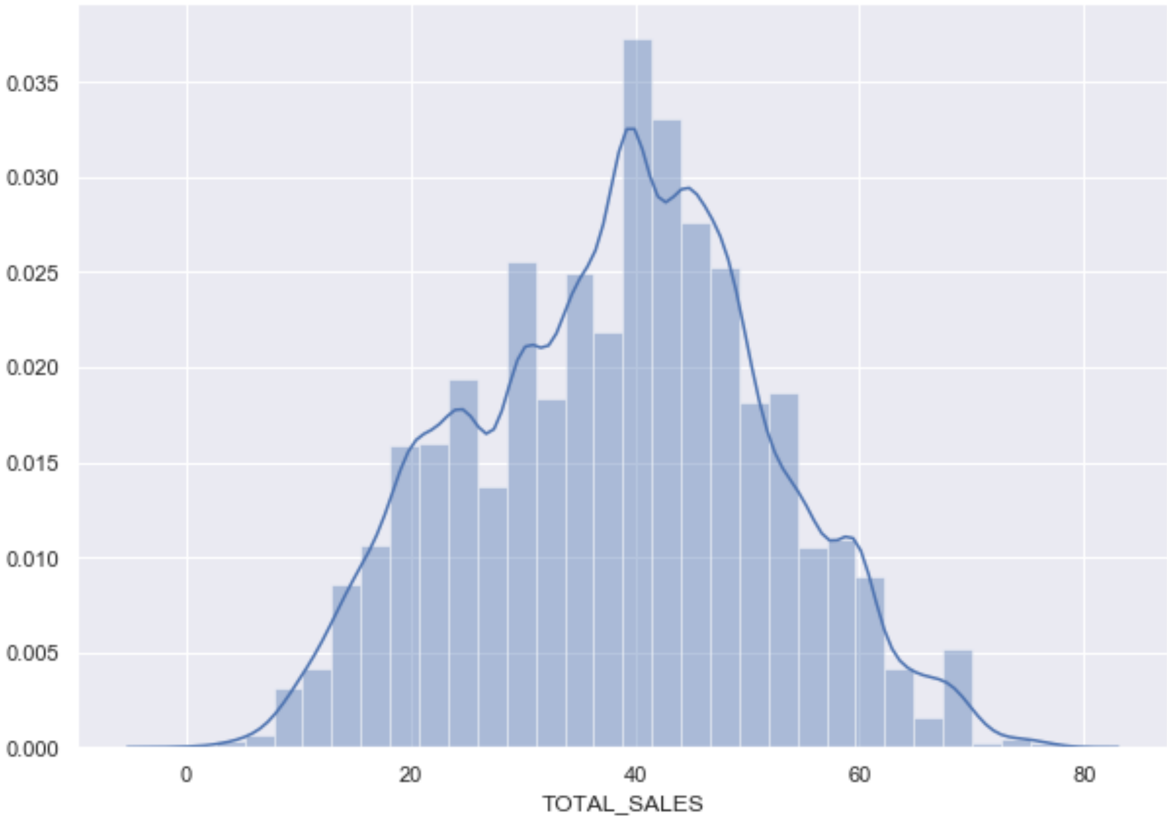
In [99]:
```
PTS2_sample_small['TOTAL_SALES_PRFT'].describe()
```

Out[99]:
```
count    33017.000000
mean        20.894482
std         12.574328
min        -20.010000
25%         11.100000
50%         22.700000
75%         30.000000
max         55.900000
Name: TOTAL_SALES_PRFT, dtype: float64
```

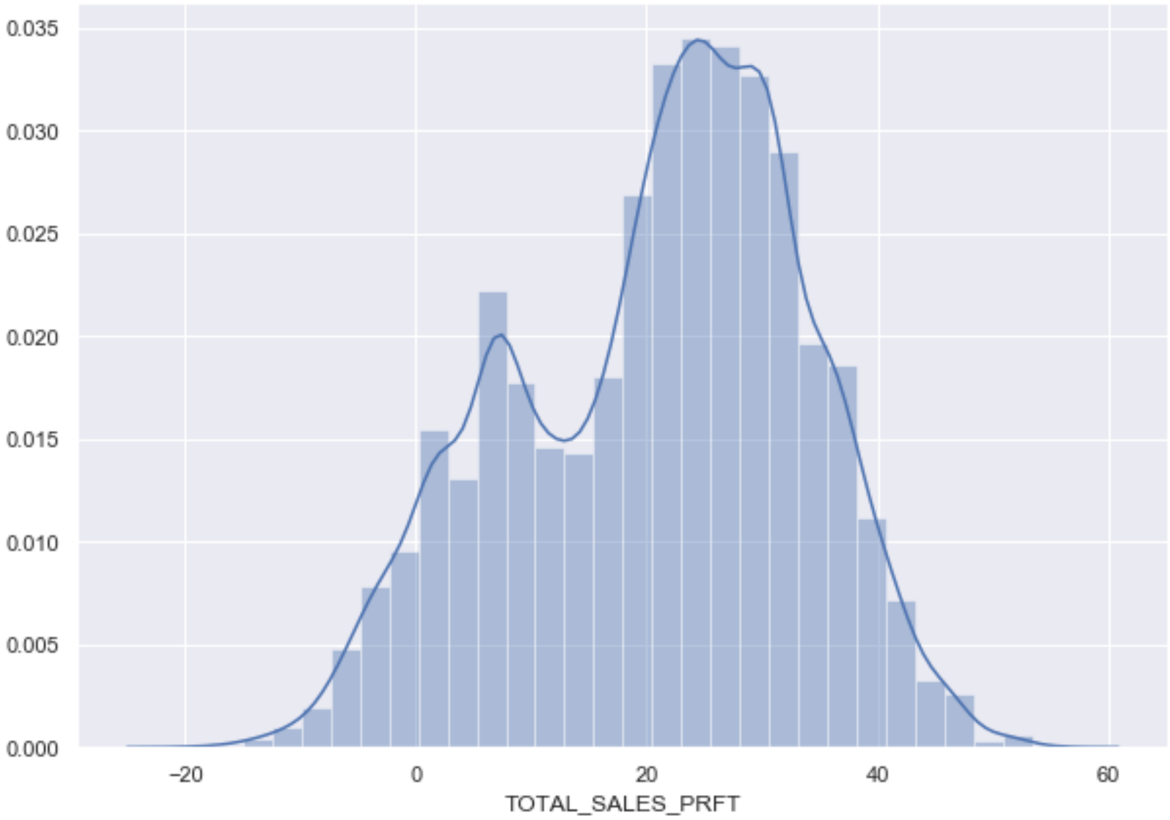
In [100]:
```
PTS2_sample_small['TOTAL_COST'].describe()
```

Out[100]:
```
count    33017.000000
mean        17.517767
std          3.403372
min          8.500000
25%         15.000000
50%         17.000000
75%         19.750000
max         26.800000
Name: TOTAL_COST, dtype: float64
```

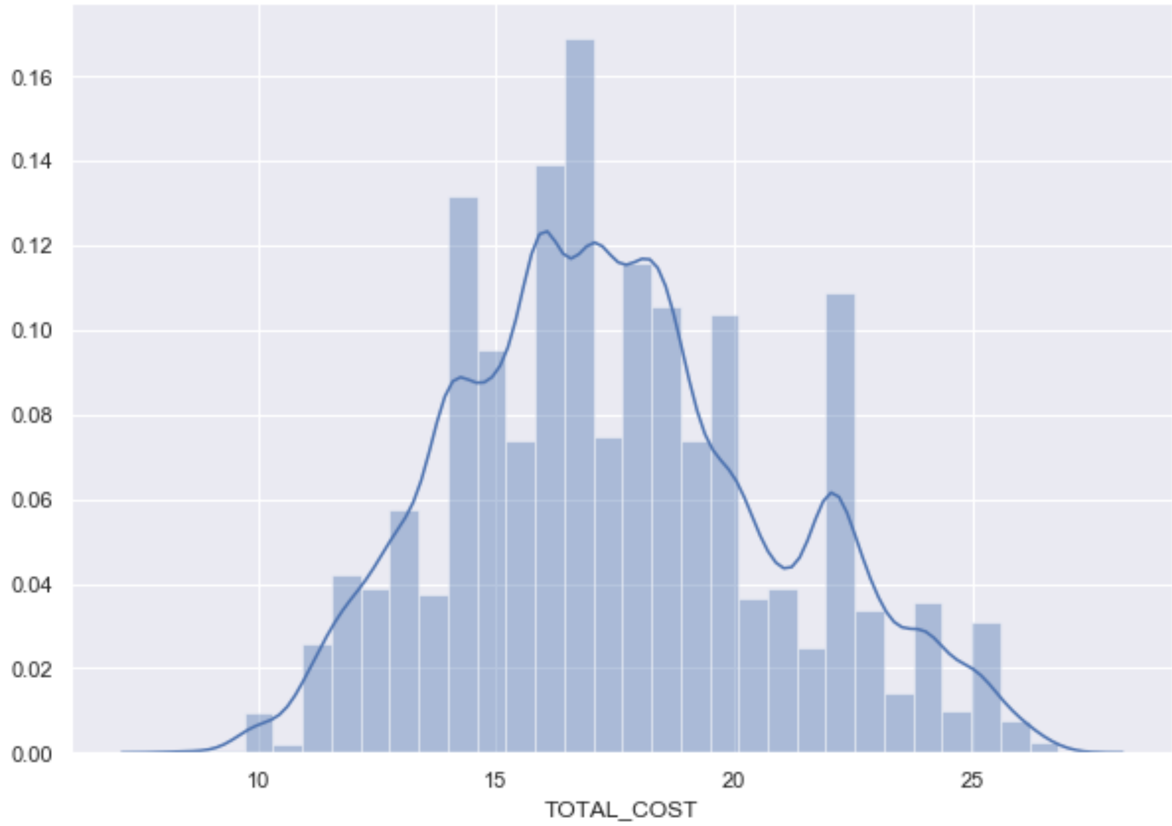
In [95]:
```
# Let's visualize the distribution of TOTAL_SALES BEFORE removing t
he outliers
sns.set(rc={'figure.figsize':(10,7)})
sns.distplot(PTS2_sample_small['TOTAL_SALES'], bins=30)
plt.show(sns)
```



In [96]:
```
# Let's visualize the distribution of TOTAL_SALES BEFORE removing t
he outliers
sns.set(rc={'figure.figsize':(10,7)})
sns.distplot(PTS2_sample_small['TOTAL_SALES_PRFT'], bins=30)
plt.show(sns)
```



In [97]:
```
# Let's visualize the distribution of TOTAL_SALES BEFORE removing t
he outliers
sns.set(rc={'figure.figsize':(10,7)})
sns.distplot(PTS2_sample_small['TOTAL_COST'], bins=30)
plt.show(sns)
```



In [82]:
```
# Determine index for continuous variables
num_feats=PTS2_sample_small.dtypes[PTS2_sample_small.dtypes!='objec
t'].index
```

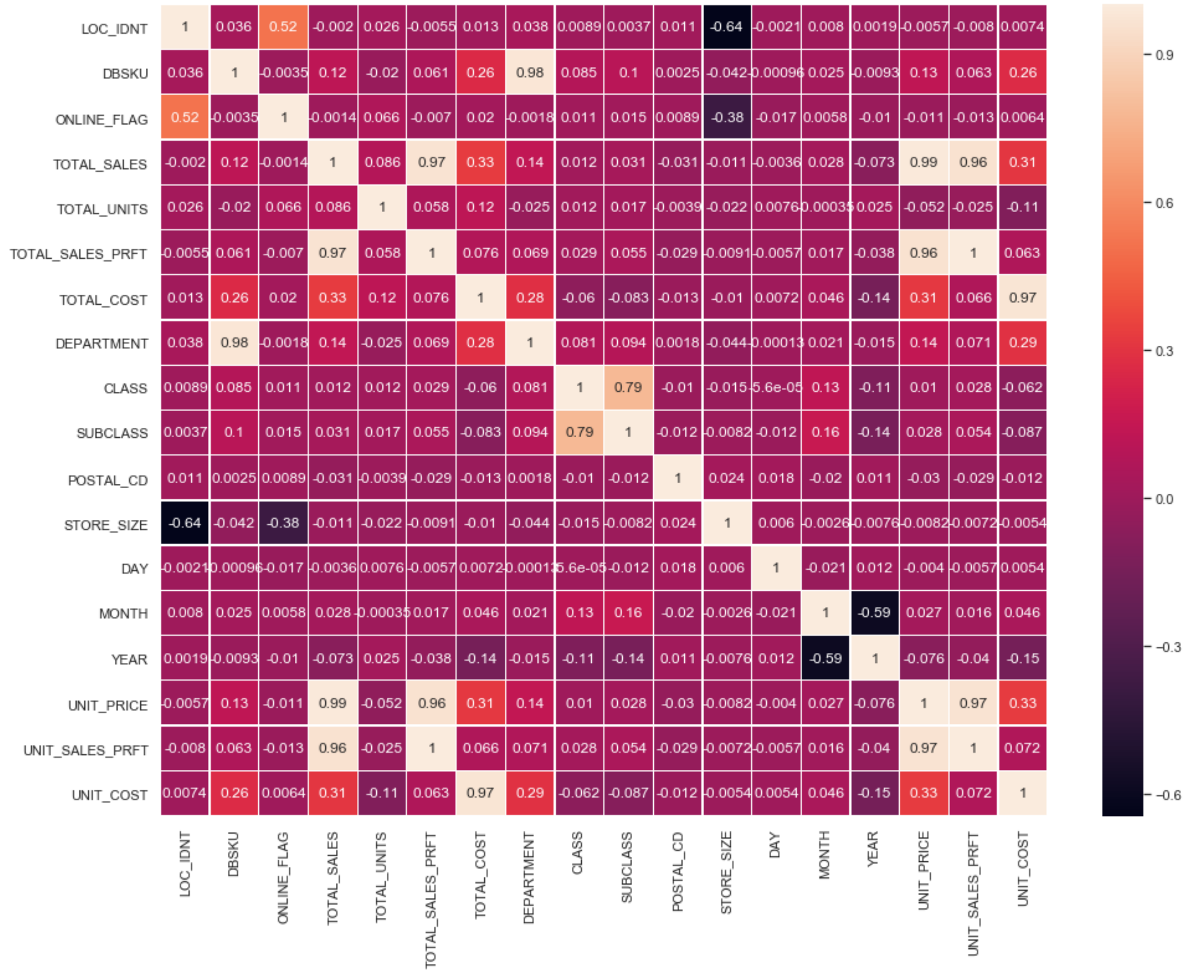
In [83]:
```
# Calculate skew and sort
skew_feats=PTS2_sample_small[num_feats].skew().sort_values
```

In [84]:
```
print(skew_feats)
```

```
<bound method Series.sort_values of LOC_IDNT              2.045489
DBSKU                 0.906313
ONLINE_FLAG          11.856305
TOTAL_SALES          -0.019856
TOTAL_UNITS          15.353540
TOTAL_SALES_PRFT     -0.323352
TOTAL_COST            0.295068
DEPARTMENT            0.779319
CLASS                26.221413
SUBCLASS              0.167883
POSTAL_CD             0.333549
STORE_SIZE           -0.827387
DAY                  -0.293262
MONTH                 0.083130
YEAR                 -0.051373
UNIT_PRICE           -0.030544
UNIT_SALES_PRFT      -0.329150
UNIT_COST             0.281997
dtype: float64>
```

In [85]:
```
# Build the correlation matrix based on the new sampled dataset
matrix = PTS2_sample_small.corr()
f, ax =plt.subplots(figsize=(16,12))
sns.heatmap(matrix, annot=True, linewidths=.3,ax=ax,
           xticklabels=matrix.columns.values,
           yticklabels=matrix.columns.values)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[85]: (18.0, 0.0)



In [86]:
```python
# Let's check with a function wheter what it is graphically represe
nted is true in terms of correlation

interesting_variables = matrix['UNIT_PRICE'].sort_values(ascending=
False)
# Filter out tatget variables (UNIT_PRICE) and variables with a low
correlation score (v such that -0.6 <= v <= 0.6)
interesting_variables = interesting_variables[abs(interesting_varia
bles) >= 0.6]
interesting_variables = interesting_variables[interesting_variable
s.index != 'UNIT_PRICE']
interesting_variables
```

Out[86]:
```
TOTAL_SALES        0.989435
UNIT_SALES_PRFT    0.966577
TOTAL_SALES_PRFT   0.959843
Name: UNIT_PRICE, dtype: float64
```

In [87]:
```
# < 0.6
interesting_variables = matrix['UNIT_PRICE'].sort_values(ascending=
False)
interesting_variables = interesting_variables[abs(interesting_varia
bles) <= -0.6]
interesting_variables = interesting_variables[interesting_variable
s.index != 'UNIT_PRICE']
interesting_variables
```

Out[87]: Series([], Name: UNIT_PRICE, dtype: float64)

In [88]:
```
# VIF - checking multicollinearity
# Let's define a simple function that we can feed afterwards to our
numeric dataset
from statsmodels.stats.outliers_influence import variance_inflation
_factor

def calc_vif(X):
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in r
ange(X.shape[1])]

    return(vif)
```

In [89]:
```
# Let's create an object to include only numeric variables (continu
ous and categorical)
PTS2_num = PTS2_sample_small._get_numeric_data()

PTS2_num.head()
```

Out[89]:

	LOC_IDNT	DBSKU	ONLINE_FLAG	TOTAL_SALES	TOTAL_UNITS	TOTAL_SALE
4339447	60	534198.0	0.0	27.60	1.0	
2418015	167	2134783.0	0.0	57.37	1.0	
5247633	693	2133033.0	0.0	15.60	1.0	
7806713	646	2124941.0	0.0	59.00	1.0	
3923468	353	482083.0	0.0	24.00	1.0	

In [90]:
```
# Let's feed our function and see the VIF related to each numeric v
ariable
X = PTS2_num.iloc[:,:-1]

calc_vif(X)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\stats\outli
ers_influence.py:185: RuntimeWarning: divide by zero encountered i
n double_scalars
  vif = 1. / (1. - r_squared_i)
```

Out[90]:

	variables	VIF
0	LOC_IDNT	5.506373e+00
1	DBSKU	6.583081e+01
2	ONLINE_FLAG	1.390359e+00
3	TOTAL_SALES	inf
4	TOTAL_UNITS	1.671388e+04
5	TOTAL_SALES_PRFT	inf
6	TOTAL_COST	inf
7	DEPARTMENT	3.181991e+03
8	CLASS	1.113812e+01
9	SUBCLASS	1.613637e+01
10	POSTAL_CD	3.164818e+00
11	STORE_SIZE	3.766893e+01
12	DAY	4.038214e+00
13	MONTH	5.183856e+00
14	YEAR	1.884556e+04
15	UNIT_PRICE	1.931270e+05
16	UNIT_SALES_PRFT	7.414365e+04

In [91]:
```
# Now, I will get rid of the variables with the higher VARIANCE, on
e by one, up until we see the VIF of the remaining
# variables go down in a range that generally stays below 5.
# We end up with these 3 variables

X = PTS2_num.drop(['TOTAL_SALES','TOTAL_SALES_PRFT',
                   'TOTAL_COST','UNIT_PRICE','UNIT_SALES_PRFT','UNIT
_COST','TOTAL_UNITS','DEPARTMENT','YEAR','STORE_SIZE','SUBCLASS'],a
xis=1)
calc_vif(X)
```

Out[91]:

	variables	VIF
0	LOC_IDNT	3.291721
1	DBSKU	2.581350
2	ONLINE_FLAG	1.337628
3	CLASS	3.777714
4	POSTAL_CD	2.730977
5	DAY	3.274382
6	MONTH	4.052754

In [92]:
```
from scipy.stats import spearmanr, kendalltau
import matplotlib.pyplot as plt
%matplotlib inline

# Calculate Spearman's correlation coefficient
coef, p =spearmanr(PTS2_sample_small.LOC_IDNT, PTS2_sample_small.PO
STAL_CD)
print('Spearmans correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p);

print('============================================')

# Calculate Kendall's correlation coefficient
coef, p =kendalltau(PTS2_sample_small.LOC_IDNT, PTS2_sample_small.P
OSTAL_CD)
print('Kendalls correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p)
```

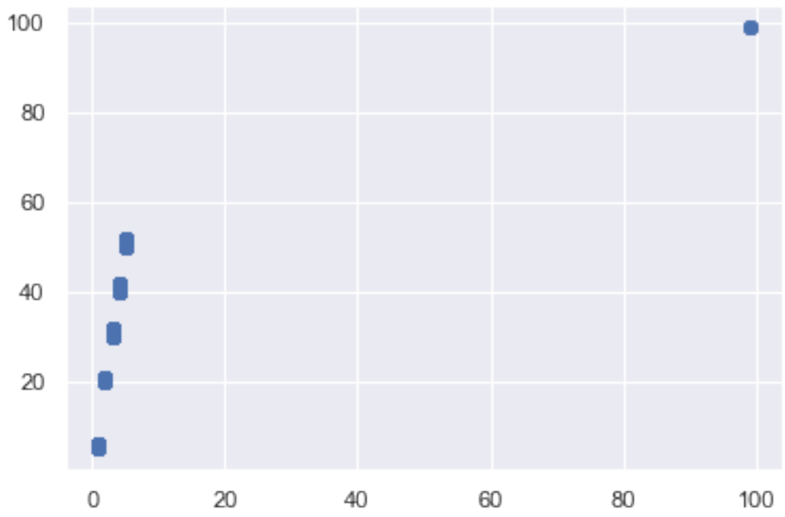
```
Spearmans correlation coefficient: 0.029
Samples are correlated (reject H0) p=0.000
============================================
Kendalls correlation coefficient: 0.019
Samples are correlated (reject H0) p=0.000
```

In [93]:
```
# CORRELATION - DEPARTMENT and DEPARTMENT_NAME -----WE CAN GET RID
OF DEPARTMENT_NAME (multicollinear)
from scipy.stats import spearmanr, kendalltau

# Calculate Spearman's correlation coefficient
coef, p =spearmanr(PTS2_sample_small.DEPARTMENT, PTS2_sample_small.
DEPARTMENT_NAME)
print('Spearmans correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p);

print('===========================================')

# Calculate Kendall's correlation coefficient
coef, p =kendalltau(PTS2_sample_small.DEPARTMENT, PTS2_sample_smal
l.DEPARTMENT_NAME)
print('Kendalls correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p)
```

```
Spearmans correlation coefficient: -1.000
Samples are correlated (reject H0) p=0.000
===========================================
Kendalls correlation coefficient: -1.000
Samples are correlated (reject H0) p=0.000
```

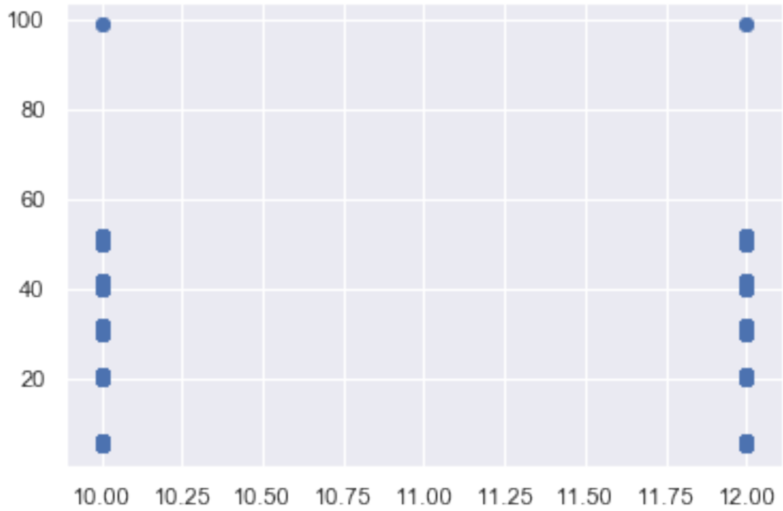
In [94]:
```
# CORRELATION - CLASS and SUBCLASS -------- WE CAN GET RID OF CLASS
(multicollinear)
from scipy.stats import spearmanr, kendalltau

# Calculate Spearman's correlation coefficient
coef, p =spearmanr(PTS2_sample_small.CLASS, PTS2_sample_small.SUBCL
ASS)
print('Spearmans correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p);

print('===========================================')

# Calculate Kendall's correlation coefficient
coef, p =kendalltau(PTS2_sample_small.CLASS, PTS2_sample_small.SUBC
LASS)
print('Kendalls correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p);

# plot
plt.scatter(PTS2_sample_small.CLASS, PTS2_sample_small.SUBCLASS)
plt.show()
```

```
Spearmans correlation coefficient: 0.971
Samples are correlated (reject H0) p=0.000
===========================================
Kendalls correlation coefficient: 0.927
Samples are correlated (reject H0) p=0.000
```



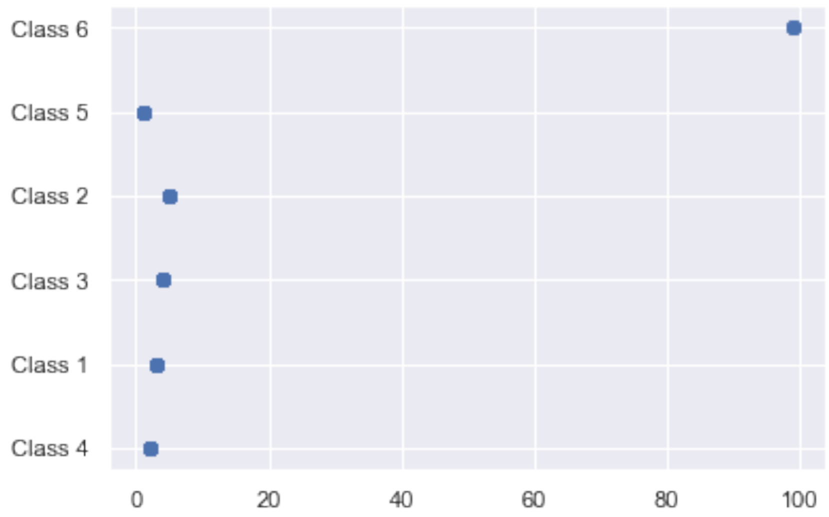
In [89]:
```
# CORRELATION - DEPARTMENT and SUBCLASS
from scipy.stats import spearmanr, kendalltau

# Calculate Spearman's correlation coefficient
coef, p =spearmanr(PTS2_sample_small.DEPARTMENT, PTS2_sample_small.
SUBCLASS)
print('Spearmans correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p);

print('===========================================')

# Calculate Kendall's correlation coefficient
coef, p =kendalltau(PTS2_sample_small.DEPARTMENT, PTS2_sample_smal
l.SUBCLASS)
print('Kendalls correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p);

# plot
plt.scatter(PTS2_sample_small.DEPARTMENT, PTS2_sample_small.SUBCLAS
S)
plt.show()
```

```
Spearmans correlation coefficient: 0.072
Samples are correlated (reject H0) p=0.000
============================================
Kendalls correlation coefficient: 0.063
Samples are correlated (reject H0) p=0.000
```



In [90]:
```
# CORRELATION - CLASS and CLASS_NAME
from scipy.stats import spearmanr, kendalltau

# Calculate Spearman's correlation coefficient
coef, p =spearmanr(PTS2_sample_small.CLASS, PTS2_sample_small.CLASS
_NAME)
print('Spearmans correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p);

print('===========================================')

# Calculate Kendall's correlation coefficient
coef, p =kendalltau(PTS2_sample_small.CLASS, PTS2_sample_small.CLAS
S_NAME)
print('Kendalls correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p);

# plot
plt.scatter(PTS2_sample_small.CLASS, PTS2_sample_small.CLASS_NAME)
plt.show()
```

```
Spearmans correlation coefficient: -0.837
Samples are correlated (reject H0) p=0.000
===========================================
Kendalls correlation coefficient: -0.723
Samples are correlated (reject H0) p=0.000
```



In [91]:
```
# CORRELATION - SUBCLASS and SUBCLASS_NAME
from scipy.stats import spearmanr, kendalltau

# Calculate Spearman's correlation coefficient
coef, p =spearmanr(PTS2_sample_small.SUBCLASS, PTS2_sample_small.SU
BCLASS_NAME)
print('Spearmans correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p);

print('===========================================')

# Calculate Kendall's correlation coefficient
coef, p =kendalltau(PTS2_sample_small.SUBCLASS, PTS2_sample_small.S
UBCLASS_NAME)
print('Kendalls correlation coefficient: %.3f' % coef)
# Interpret the significance
alpha = 0.05
if p > alpha:
    print('Samples are uncorrelated (fail to reject H0) P=%.3f' %
p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p)
```

```
Spearmans correlation coefficient: -0.308
Samples are correlated (reject H0) p=0.000
===========================================
Kendalls correlation coefficient: -0.266
Samples are correlated (reject H0) p=0.000
```

In [92]:
```
# LET'S now drop all of the not very useful numeric variables we go
t rid of by doing the VIF analysis, and store the
# remaining variables in the new df 'Pdrop'
```

In [93]:
```
Pdrop = PTS2_sample_small.drop(['TOTAL_SALES','TOTAL_SALES_PRFT',
                        'TOTAL_COST','CLASS','CLASS_NAME','DEPARTM
ENT_NAME','TOTAL_UNITS'], axis=1)
```

In [94]:
```
Pdrop.shape
```

Out[94]:
```
(33017, 18)
```

In [95]:
```
Pdrop.head()
```

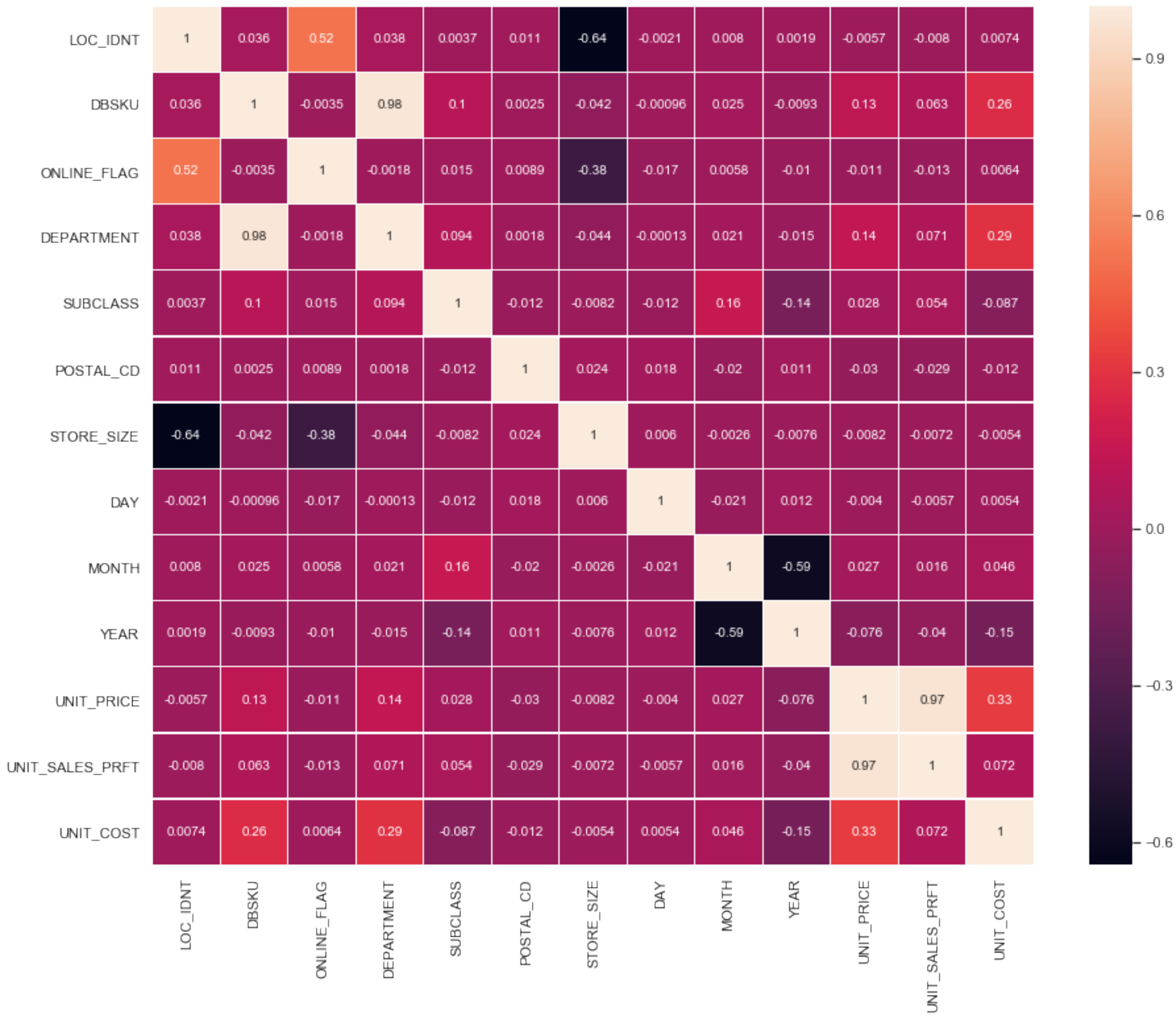
Out[95]:

	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	DEPARTMENT	SUBCLAS
4339447	60	534198.0	0.0	FP	10	2
2418015	167	2134783.0	0.0	NFP	12	2
5247633	693	2133033.0	0.0	NFP	12	3
7806713	646	2124941.0	0.0	NFP	12	2
3923468	353	482083.0	0.0	NFP	10	4

In [96]:
```
# Let's have another look to variables correlations using a heatmap
within the new dataset
corr2 = Pdrop.corr()
```

In [97]:
```
# Build the correlation matrix
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(15,12))
sns.heatmap(corr2, annot=True, linewidths=.3,ax=ax,
            xticklabels=corr2.columns.values,
            yticklabels=corr2.columns.values)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[97]: (13.0, 0.0)



In [98]:
```
# Again, UNIT_SALES_PRFT is the only higlhy (and positively) correl
ated variable to our target UNIT_PRICE

interesting_variables = corr2['UNIT_PRICE'].sort_values(ascending=F
alse)
# Filter out tatget variables (UNIT_PRICE) and variables with a low
correlation score (v such that -0.6 <= v <= 0.6)
interesting_variables = interesting_variables[abs(interesting_varia
bles) >= 0.6]
interesting_variables = interesting_variables[interesting_variable
s.index != 'UNIT_PRICE']
interesting_variables
```

Out[98]:
```
UNIT_SALES_PRFT    0.966577
Name: UNIT_PRICE, dtype: float64
```

In [99]:
```
# Let's now get dummies for the below indicated variables (mostly i
nteresting because of our project statement)
```

In [100]:
```
# Let's now get dummies for the below indicated variables (mostly i
nteresting because of our project statement)
Pdummies = pd.get_dummies(Pdrop, columns = ['SUBCLASS_NAME','SUBCLA
SS','DEPARTMENT','STORE_TYPE','ONLINE_FLAG','FULL_PRICE_IND',])
```

In [101]:
```
Pdummies.head()
```

Out[101]:

	LOC_IDNT	DBSKU	CITY	STATE	POSTAL_CD	STORE_SIZE	DAY
4339447	60	534198.0	POUGHKEEPSIE	NY	12601	3257.0	6
2418015	167	2134783.0	DOWNERS GROVE	IL	60516	3647.0	3
5247633	693	2133033.0	SEVIERVILLE	TN	37862	3174.0	3
7806713	646	2124941.0	BROOKLYN	NY	11234	2820.0	1
3923468	353	482083.0	GROVE CITY	OH	43123	3342.0	5

5 rows × 52 columns

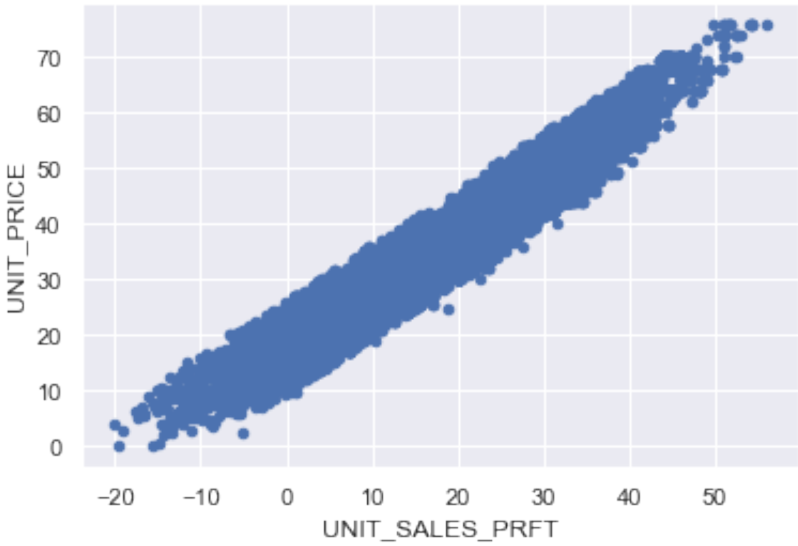
In [102]:
```
Pdummies.shape
```

Out[102]: (33017, 52)

In [103]:
```
# Let's look at the relation between UNIT_PRICE and UNIT_SALES_PRFT
with a scatterplot.
# It looks like is outliers free, and the relation between the two
Vs is positive
data = pd.concat([Pdummies['UNIT_PRICE'], Pdummies['UNIT_SALES_PRF
T']], axis=1)
data.plot.scatter(x='UNIT_SALES_PRFT', y='UNIT_PRICE')
```

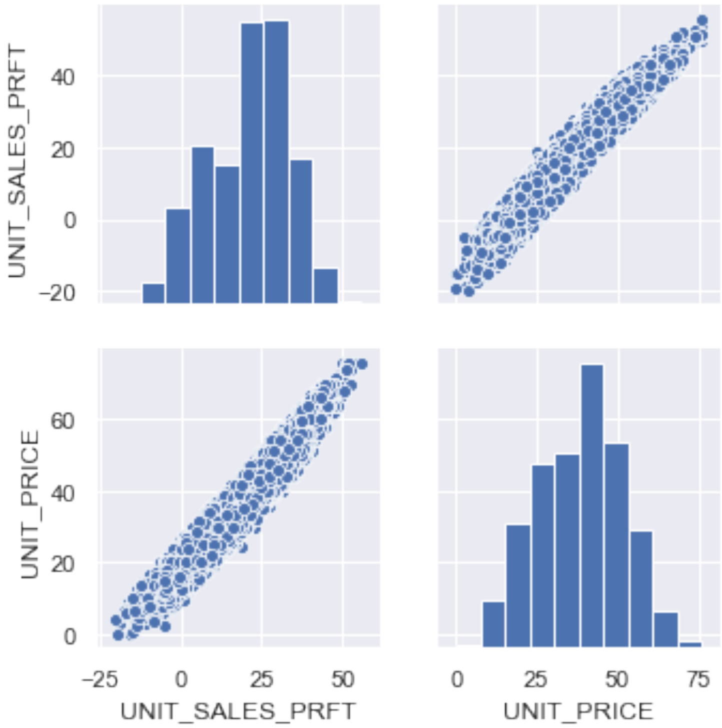
```
'c' argument looks like a single numeric RGB or RGBA sequence, whi
ch should be avoided as value-mapping will have precedence in case
its length matches with 'x' & 'y'.  Please use a 2-D array with a
single row if you really want to specify the same RGB or RGBA valu
e for all points.
```

Out[103]: <matplotlib.axes._subplots.AxesSubplot at 0x68aeefd8c8>

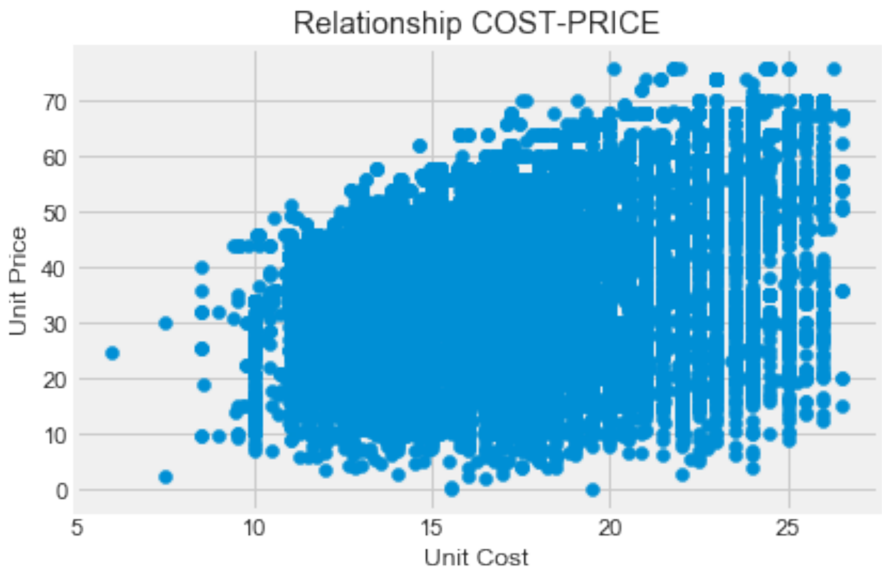


In [104]:
```
# The like-normal distribution of the two related variables here is clear
cols = interesting_variables.index.values.tolist() + ['UNIT_PRICE']
sns.pairplot(Pdummies[cols], size=2.5)
plt.show()
```

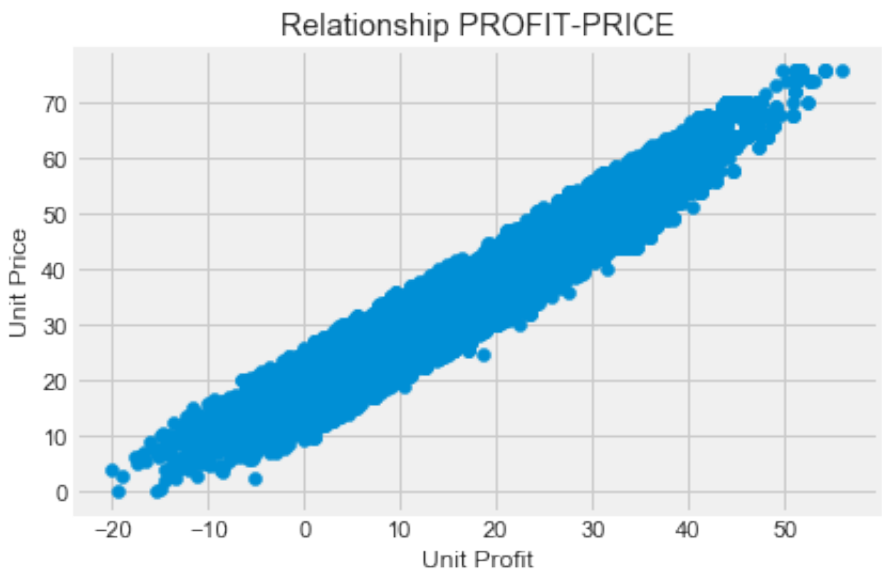
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\axisgrid.py:206
5: UserWarning: The `size` parameter has been renamed to `height`;
pleaes update your code.
  warnings.warn(msg, UserWarning)
```



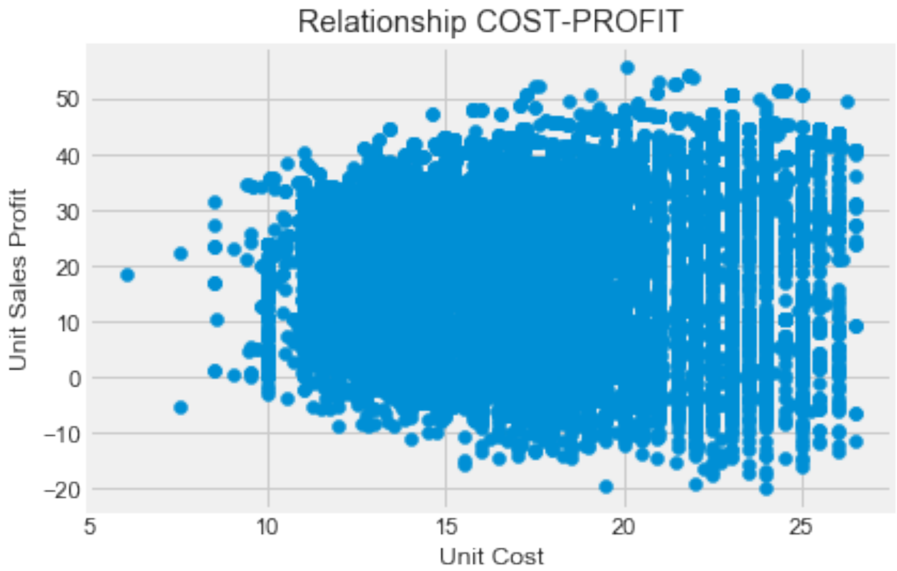
In [191]:
```
plt.scatter(Pdummies['UNIT_COST'],Pdummies['UNIT_PRICE'])
plt.rcParams['axes.facecolor'] = 'black'
plt.title('Relationship COST-PRICE')
plt.xlabel('Unit Cost')
plt.ylabel('Unit Price')
plt.show()
```



In [189]:
```
plt.scatter(Pdummies['UNIT_SALES_PRFT'],Pdummies['UNIT_PRICE'])
plt.title('Relationship PROFIT-PRICE')
plt.xlabel('Unit Profit')
plt.ylabel('Unit Price')
plt.show()
```



In [190]:
```
plt.scatter(Pdummies['UNIT_COST'],Pdummies['UNIT_SALES_PRFT'])
plt.title('Relationship COST-PROFIT')
plt.xlabel('Unit Cost')
plt.ylabel('Unit Sales Profit')
plt.show()
```



In [105]:
```
# Let's define X and Y
# X = features
# y = price
```

In [106]:
```
price = Pdummies['UNIT_PRICE']
features = Pdummies.drop(['UNIT_PRICE','UNIT_SALES_PRFT','STATE','C
ITY'], axis=1)
```

In [107]:
```
price1 = Pdrop['UNIT_PRICE']
features1 = Pdrop.drop(['UNIT_PRICE','UNIT_SALES_PRFT','STATE','CIT
Y'], axis=1)
```

In [108]:
```
# IMPORT R2_SCORE
from sklearn.metrics import r2_score,mean_squared_error

def performance_metric(y_true, y_predict):
    # calculates and returns the performance score between true (y_
true) and predicted (y_predict) values based on the metric chosen
    R2_score = r2_score(y_true, y_predict)
    MSE_score = mean_squared_error(y_true, y_predict)
    return R2_score, MSE_score
```

In [109]:
```
# Import 'train_test_split'
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, pric
e, test_size=0.2, random_state=100)

print("Training and testing split was successsful.")
```

```
Training and testing split was successsful.
```

In [110]:
```
# Import 'train_test_split'
from sklearn.model_selection import train_test_split

X_train2, X_test2, y_train2, y_test2 = train_test_split(features1,
price1, test_size=0.2, random_state=100)

print("Training and testing split was successsful.")
```

```
Training and testing split was successsful.
```

In [111]:
```
# Import 'make_scorer', 'DecisionTreeRegressor', and 'GridSearchCV'
from sklearn.model_selection import ShuffleSplit
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import make_scorer
from sklearn.model_selection import GridSearchCV
```

LINEAR REGRESSION

In [112]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics
```

In [113]:
```
model_lin = linear_model.LinearRegression()
model_lin.fit(X_train,y_train)
```

Out[113]:
```
LinearRegression()
```

In [114]:
```
print('Coefficients: \n', model_lin.coef_)
print('Variance score: {}'.format(model_lin.score(X_test,y_test)))
```

```
Coefficients: 
 [-4.11860958e-04 -1.10549025e-06 -9.86147468e-06 -3.55666163e-04
  3.72375938e-02 -1.28803445e-01 -5.30731888e-02  1.07878582e+00
  3.19679131e+00  3.12045986e+00  4.22372782e-02  3.59386597e+00
 -1.61835672e+00  6.17987060e-01 -8.95298476e+00 -1.50901028e+00
  4.50146272e-01  2.24453871e-01  2.74497304e-01  2.15094108e+00
  1.91824512e-01 -3.84978491e-02  2.72748129e+00  2.20399177e+00
  3.23528916e+00  4.22372782e-02  6.17987060e-01 -1.61835672e+00
 -8.95298476e+00 -2.04293136e+00  2.04293136e+00  3.49217965e-02
 -2.64297131e-01 -2.33878660e-01 -8.64016480e-02  5.18711276e-01
 -1.91890930e+00  6.54270405e-01  1.78707791e-01  1.00507886e-01
  1.89041166e-02 -1.54253303e-01  1.27884882e+00 -1.27132055e-01
  1.91890930e+00 -1.91890930e+00  8.69538969e+00 -8.69538969e+00]
Variance score: 0.471104736860828
```

In [115]:
```
# Plot for residual error
plt.style.use('fivethirtyeight')
# Plot residual errors in training data
plt.scatter(model_lin.predict(X_train), model_lin.predict(X_train)
- y_train, color = "green",
            s = 10, label = 'Train data')
# Plot residual errors in test data
plt.scatter(model_lin.predict(X_test), model_lin.predict(X_test) -
y_test, color = "blue",
            s = 10, label = 'Test data')
## Plotting line for zero residual error
plt.hlines(y = 0, xmin = 0, xmax = 78, linewidth = 2)
## Plotting legend
plt.legend(loc ='upper right')
## Plot title
plt.title("Residual errors")
plt.show()
```

RANDOM FOREST REGRESSOR

In [116]:
```
# Let's try with a RANDOM FOREST regression

from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=100, oob_score=True,n_jo
bs=-1, random_state=42)
```

In [117]:
```
model.fit(X_train,y_train)
```

Out[117]:
```
RandomForestRegressor(n_jobs=-1, oob_score=True, random_state=42)
```

In [118]:
```
model.oob_score_
```

Out[118]:
```
0.7865986265055964
```

In [214]:
```
for w in range(10,100,10):
    model=RandomForestRegressor(n_estimators=w,oob_score=True, rand
om_state=42)
    model.fit(X_train,y_train)
    oob=model.oob_score_
    print('For n_estimators = '+str(w))
    print('OOB score is '+str(oob))
    print('************************')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_fores
t.py:832: UserWarning: Some inputs do not have OOB scores. This pr
obably means too few trees were used to compute any reliable oob e
stimates.
  warn("Some inputs do not have OOB scores. "

For n_estimators = 10
OOB score is 0.6243968618833451
************************

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_fores
t.py:832: UserWarning: Some inputs do not have OOB scores. This pr
obably means too few trees were used to compute any reliable oob e
stimates.
  warn("Some inputs do not have OOB scores. "

For n_estimators = 20
OOB score is 0.7580071088513909
************************
For n_estimators = 30
OOB score is 0.7708245541091368
************************
For n_estimators = 40
OOB score is 0.7775998014757093
************************
For n_estimators = 50
OOB score is 0.7799629347745498
************************
For n_estimators = 60
OOB score is 0.7814109780871583
************************
For n_estimators = 70
OOB score is 0.7834088690419828
************************
For n_estimators = 80
OOB score is 0.7845484304079753
************************
For n_estimators = 90
OOB score is 0.7859157020233106
************************
```

In [215]:
```
# Our best n_estimators = ....

for w in range(90,200,10):
    model=RandomForestRegressor(n_estimators=w,oob_score=True, rand
om_state=42)
    model.fit(X_train,y_train)
    oob=model.oob_score_
    print('For n_estimators = '+str(w))
    print('OOB score is '+str(oob))
    print('************************')
```

```
For n_estimators = 90
OOB score is 0.7859157020233106
************************
For n_estimators = 100
OOB score is 0.7865986265055964
************************
For n_estimators = 110
OOB score is 0.786762490066383
************************
For n_estimators = 120
OOB score is 0.7873163333211528
************************
For n_estimators = 130
OOB score is 0.7879335806533665
************************
For n_estimators = 140
OOB score is 0.7887214979768251
************************
For n_estimators = 150
OOB score is 0.7890399931460832
************************
For n_estimators = 160
OOB score is 0.789464797932944
************************
For n_estimators = 170
OOB score is 0.7897606867879189
************************
For n_estimators = 180
OOB score is 0.7899219935683667
************************
For n_estimators = 190
OOB score is 0.7900537613113483
************************
```

In [217]:
```
# Our best n_estimators = ....

for w in range(190,300,10):
    model=RandomForestRegressor(n_estimators=w,oob_score=True, rand
om_state=42)
    model.fit(X_train,y_train)
    oob=model.oob_score_
    print('For n_estimators = '+str(w))
    print('OOB score is '+str(oob))
    print('************************')
```

```
For n_estimators = 190
OOB score is 0.7900537613113483
************************
For n_estimators = 200
OOB score is 0.7901937291134193
************************
For n_estimators = 210
OOB score is 0.7904510021409402
************************
For n_estimators = 220
OOB score is 0.7905373233763064
************************
For n_estimators = 230
OOB score is 0.7906095828130622
************************
For n_estimators = 240
OOB score is 0.7908806870944751
************************
For n_estimators = 250
OOB score is 0.7911040463503228
************************
For n_estimators = 260
OOB score is 0.791162628278429
************************
For n_estimators = 270
OOB score is 0.7912904447411685
************************
For n_estimators = 280
OOB score is 0.7914195031669815
************************
For n_estimators = 290
OOB score is 0.7916179972285158
************************
```

In [218]:
```
# Our best n_estimators = 330!

for w in range(300,410,10):
    model=RandomForestRegressor(n_estimators=w,oob_score=True, rand
om_state=42)
    model.fit(X_train,y_train)
    oob=model.oob_score_
    print('For n_estimators = '+str(w))
    print('OOB score is '+str(oob))
    print('************************')
```

```
For n_estimators = 300
OOB score is 0.7917359384974916
************************
For n_estimators = 310
OOB score is 0.7917550451612843
************************
For n_estimators = 320
OOB score is 0.7918289354458401
************************
For n_estimators = 330
OOB score is 0.7918513369043578
************************
For n_estimators = 340
OOB score is 0.7917513996718457
************************
For n_estimators = 350
OOB score is 0.7918124240354597
************************

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent c
all last)
<ipython-input-218-09e835f02804> in <module>
      3 for w in range(300,410,10):
      4     model=RandomForestRegressor(n_estimators=w,oob_score=T
rue, random_state=42)
----> 5     model.fit(X_train,y_train)
      6     oob=model.oob_score_
      7     print('For n_estimators = '+str(w))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_fores
t.py in fit(self, X, y, sample_weight)
    390                 verbose=self.verbose, class_weight=sel
f.class_weight,
    391                 n_samples_bootstrap=n_samples_bootstra
p)
--> 392             for i, t in enumerate(trees))
    393 
    394         # Collect newly grown trees

C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in _
```

```
_call__(self, iterable)
   1005                 self._iterating = self._original_iterator
is not None
   1006
-> 1007             while self.dispatch_one_batch(iterator):
   1008                 pass
   1009

C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in d
ispatch_one_batch(self, iterator)
    833                 return False
    834             else:
--> 835                 self._dispatch(tasks)
    836                 return True
    837

C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in _
dispatch(self, batch)
    752         with self._lock:
    753             job_idx = len(self._jobs)
--> 754             job = self._backend.apply_async(batch, callbac
k=cb)
    755             # A job can complete so quickly than its callb
ack is
    756             # called before we get here, causing self._job
s to

C:\ProgramData\Anaconda3\lib\site-packages\joblib\_parallel_backen
ds.py in apply_async(self, func, callback)
    207     def apply_async(self, func, callback=None):
    208         """Schedule a func to be run"""
--> 209         result = ImmediateResult(func)
    210         if callback:
    211             callback(result)

C:\ProgramData\Anaconda3\lib\site-packages\joblib\_parallel_backen
ds.py in __init__(self, batch)
    588         # Don't delay the application, to avoid keeping th
e input
    589         # arguments in memory
--> 590         self.results = batch()
    591
    592     def get(self):

C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in _
_call__(self)
    254         with parallel_backend(self._backend, n_jobs=self._
n_jobs):
    255             return [func(*args, **kwargs)
--> 256                     for func, args, kwargs in self.items]
    257
    258     def __len__(self):
```

```
C:\ProgramData\Anaconda3\lib\site-packages\joblib\parallel.py in <
listcomp>(.0)
    254         with parallel_backend(self._backend, n_jobs=self._
n_jobs):
    255             return [func(*args, **kwargs)
--> 256                     for func, args, kwargs in self.items]
    257
    258     def __len__(self):

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_fores
t.py in _parallel_build_trees(tree, forest, X, y, sample_weight, t
ree_idx, n_trees, verbose, class_weight, n_samples_bootstrap)
    166                                                            in
dices=indices)
    167
--> 168         tree.fit(X, y, sample_weight=curr_sample_weight, c
heck_input=False)
    169     else:
    170         tree.fit(X, y, sample_weight=sample_weight, check_
input=False)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.p
y in fit(self, X, y, sample_weight, check_input, X_idx_sorted)
   1244             sample_weight=sample_weight,
   1245             check_input=check_input,
-> 1246             X_idx_sorted=X_idx_sorted)
   1247         return self
   1248

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.p
y in fit(self, X, y, sample_weight, check_input, X_idx_sorted)
    373                                             min_impurity_sp
lit)
    374
--> 375         builder.build(self.tree_, X, y, sample_weight, X_i
dx_sorted)
    376
    377         if self.n_outputs_ == 1 and is_classifier(self):

KeyboardInterrupt:
```

In [106]:
```
# Finalize 330 trees
model = RandomForestRegressor(n_estimators=330, oob_score=True, ran
dom_state=42)


#HYPERPARAMETERS currently in use
from pprint import pprint
print('Parameters currently in use:\n')
pprint(model.get_params())
```

```
Parameters currently in use:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 330,
 'n_jobs': None,
 'oob_score': True,
 'random_state': 42,
 'verbose': 0,
 'warm_start': False}
```

In [107]:
```
# RANDOM  HYPERPARAMETERS GRID
#To use RandomizedSearchCV, we first need to create a parameter gri
d to sample from during fitting

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10,stop = 360,
num = 10)]
# Number of features to consider at every split
max_features =['auto','sqrt','log2']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10,110,num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split =[2,5,10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1,2,4]
# Method of selecting samples for training each tree
bootstrap = [True,False]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}
pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, Non
e],
 'max_features': ['auto', 'sqrt', 'log2'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [10, 48, 87, 126, 165, 204, 243, 282, 321, 360]}
```

In [110]:
```
from sklearn.model_selection import RandomizedSearchCV
# Use the random grid to search for best hyperparameters
# First create the base model to tune
model = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation, searc
h across 100 different combinations, and use all available cores
model_random = RandomizedSearchCV(estimator = model, param_distribu
tions = random_grid, n_iter=100, cv = 3, verbose=2,
                                random_state=42, n_jobs=-1)
```

In [111]:
```
# Fit the random search model
model_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  4.2min
[Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed: 15.5min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed: 29.5min fini
shed
```

Out[111]:
```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_iter
=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, Fals
e],
                                        'max_depth': [10, 20, 30,
40, 50, 60,
                                                      70, 80, 90,
100, 110,
                                                      None],
                                        'max_features': ['auto', '
sqrt',
                                                         'log2'],
                                        'min_samples_leaf': [1, 2,
4],
                                        'min_samples_split': [2,
5, 10],
                                        'n_estimators': [10, 48, 8
7, 126, 165,
                                                         204, 243,
282, 321,
                                                         360]},
                   random_state=42, verbose=2)
```

In [112]:
```
# Best parameters from fitting the random search
# From these results we should be able to narrow the range of value
s for each hyperparameter
model_random.best_params_
```

Out[112]:
```
{'n_estimators': 243,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 100,
 'bootstrap': True}
```

In [119]:
```
# EVALUATE RANDOM SEARCH
# to determine if random search yelded a better model, we compare t
he base model with the best random search model

def evaluate(model,X_test,y_test):
    predictions = model.predict(X_test)
    errors = abs(predictions - y_test)
    mape = 100*np.mean(errors / y_test)
    accuracy = 100 - mape
    print('Model Performance')
    print('Average Error:{:0.4f} degrees.'.format(np.mean(errors)))
    print('Accuracy = {:0.2f}%.'.format(accuracy))
    return accuracy
```

In [202]:
```
# Base model performances
model_base = RandomForestRegressor(n_estimators = 165, random_state
=42)
model_base.fit(X_train,y_train)
accuracy_base = evaluate(model_base,X_test,y_test)
```

```
Model Performance
Average Error:4.5988 degrees.
Accuracy = 84.15%.
```

In [203]:
```
# Random model performances
best_random = model_random.best_estimator_
accuracy_random = evaluate(best_random,X_test,y_test)
```

```
Model Performance
Average Error:4.5972 degrees.
Accuracy = 84.15%.
```

In [204]:
```
# Improvement from base to best random model
print('Improvement of {:0.2f}%.'.format(100 * (accuracy_random - ac
curacy_base) / accuracy_base))
```

```
Improvement of 0.01%.
```

In [109]:
```
# GRID SEARCH with CROSS VALIDATION
# Random search allowed us to narrow down the range for each hyperp
arameter.
# Now that we know where to concentrate our search, we can explicit
ly specify every combination of settings to try.
# We do this with GridSearchCV, a method that, instead of sampling
randomly from a distribution, evaluates all combinations we define
# To use Grid Search, we make another grid based on the best values
provided by random search:

from sklearn.model_selection import GridSearchCV
#Create the parameter grid based on the results of random search (m
odel_random.best_params_)
param_grid = {'bootstrap': [True],
            'max_depth': [100,110],
            'max_features': ['auto'],
            'min_samples_leaf': [1],
            'min_samples_split': [2,4,5,6,7],
            'n_estimators': [330, 360]}
```

In [206]:
```
# Create a base model
model = RandomForestRegressor()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = model, param_grid = param_gr
id, cv=3, n_jobs=-1,verbose=2)
```

In [207]:
```
# Fit the grid search to the data
grid_search.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent
workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  7.7min
[Parallel(n_jobs=-1)]: Done  60 out of  60 | elapsed: 12.3min fini
shed
```

Out[207]:
```
GridSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
             param_grid={'bootstrap': [True], 'max_depth': [100, 1
10],
                         'max_features': ['auto'], 'min_samples_le
af': [1],
                         'min_samples_split': [2, 4, 5, 6, 7],
                         'n_estimators': [165, 180]},
             verbose=2)
```

In [208]:
```
# Best parameters for grid search
grid_search.best_params_
```

Out[208]:
```
{'bootstrap': True,
 'max_depth': 100,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 4,
 'n_estimators': 180}
```

In [209]:
```
best_grid = grid_search.best_estimator_
grid_accuracy = evaluate(best_grid,X_test,y_test)
```

```
Model Performance
Average Error:4.5870 degrees.
Accuracy = 84.19%.
```

In [210]:
```
# Improvement from base to best grid model
print('Improvement of {:0.2f}%.'.format(100 * (grid_accuracy - accu
racy_base) / accuracy_base))
```

```
Improvement of 0.06%.
```

In [ ]:

In [120]:
```
# Let's feed the final model with all the already tuned parameters
model_final = RandomForestRegressor(n_estimators = 330, min_samples
_split = 4, min_samples_leaf = 1, max_features = 'auto',
                                    max_depth = 100, bootstrap = Tr
ue, n_jobs=-1, verbose=2, random_state=42)
```

In [121]:
```
model_final.fit(X_train,y_train)
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 2 concu
rrent workers.
```

```
building tree 1 of 330
building tree 2 of 330
building tree 3 of 330
building tree 4 of 330
building tree 5 of 330
building tree 6 of 330
building tree 7 of 330
building tree 8 of 330
building tree 9 of 330
building tree 10 of 330
building tree 11 of 330
building tree 12 of 330
building tree 13 of 330
building tree 14 of 330
building tree 15 of 330
building tree 16 of 330
building tree 17 of 330
building tree 18 of 330
building tree 19 of 330
building tree 20 of 330
building tree 21 of 330
building tree 22 of 330
building tree 23 of 330
building tree 24 of 330
building tree 25 of 330
building tree 26 of 330
building tree 27 of 330
building tree 28 of 330
building tree 29 of 330
building tree 30 of 330
building tree 31 of 330
building tree 32 of 330
building tree 33 of 330
building tree 34 of 330
building tree 35 of 330
building tree 36 of 330
building tree 37 of 330
building tree 38 of 330

[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:    4.6s

building tree 39 of 330
building tree 40 of 330
building tree 41 of 330
building tree 42 of 330
building tree 43 of 330
building tree 44 of 330
building tree 45 of 330
building tree 46 of 330
building tree 47 of 330
building tree 48 of 330
building tree 49 of 330
building tree 50 of 330
building tree 51 of 330
```

```
building tree 52 of 330
building tree 53 of 330
building tree 54 of 330
building tree 55 of 330
building tree 56 of 330
building tree 57 of 330
building tree 58 of 330
building tree 59 of 330
building tree 60 of 330
building tree 61 of 330
building tree 62 of 330
building tree 63 of 330
building tree 64 of 330
building tree 65 of 330
building tree 66 of 330building tree 67 of 330

building tree 68 of 330building tree 69 of 330

building tree 70 of 330
building tree 71 of 330
building tree 72 of 330
building tree 73 of 330
building tree 74 of 330
building tree 75 of 330
building tree 76 of 330
building tree 77 of 330
building tree 78 of 330
building tree 79 of 330
building tree 80 of 330
building tree 81 of 330
building tree 82 of 330
building tree 83 of 330
building tree 84 of 330
building tree 85 of 330
building tree 86 of 330
building tree 87 of 330
building tree 88 of 330
building tree 89 of 330
building tree 90 of 330
building tree 91 of 330
building tree 92 of 330
building tree 93 of 330
building tree 94 of 330
building tree 95 of 330
building tree 96 of 330
building tree 97 of 330
building tree 98 of 330
building tree 99 of 330
building tree 100 of 330
building tree 101 of 330
building tree 102 of 330
building tree 103 of 330
building tree 104 of 330
```

```
building tree 105 of 330
building tree 106 of 330
building tree 107 of 330
building tree 108 of 330
building tree 109 of 330
building tree 110 of 330
building tree 111 of 330
building tree 112 of 330
building tree 113 of 330building tree 114 of 330

building tree 115 of 330building tree 116 of 330

building tree 117 of 330
building tree 118 of 330
building tree 119 of 330
building tree 120 of 330
building tree 121 of 330
building tree 122 of 330
building tree 123 of 330
building tree 124 of 330
building tree 125 of 330
building tree 126 of 330
building tree 127 of 330
building tree 128 of 330
building tree 129 of 330
building tree 130 of 330
building tree 131 of 330
building tree 132 of 330
building tree 133 of 330
building tree 134 of 330
building tree 135 of 330
building tree 136 of 330
building tree 137 of 330
building tree 138 of 330
building tree 139 of 330
building tree 140 of 330
building tree 141 of 330
building tree 142 of 330
building tree 143 of 330
building tree 144 of 330
building tree 145 of 330
building tree 146 of 330
building tree 147 of 330building tree 148 of 330

building tree 149 of 330
building tree 150 of 330
building tree 151 of 330
building tree 152 of 330
building tree 153 of 330
building tree 154 of 330
building tree 155 of 330
building tree 156 of 330
building tree 157 of 330
```

```
building tree 158 of 330
building tree 159 of 330
building tree 160 of 330
```

```
[Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed:   19.3s
```

```
building tree 161 of 330
building tree 162 of 330
building tree 163 of 330
building tree 164 of 330
building tree 165 of 330
building tree 166 of 330
building tree 167 of 330
building tree 168 of 330
building tree 169 of 330
building tree 170 of 330
building tree 171 of 330
building tree 172 of 330
building tree 173 of 330
building tree 174 of 330
building tree 175 of 330
building tree 176 of 330
building tree 177 of 330
building tree 178 of 330
building tree 179 of 330
building tree 180 of 330building tree 181 of 330

building tree 182 of 330
building tree 183 of 330
building tree 184 of 330
building tree 185 of 330
building tree 186 of 330
building tree 187 of 330
building tree 188 of 330
building tree 189 of 330
building tree 190 of 330
building tree 191 of 330
building tree 192 of 330
building tree 193 of 330
building tree 194 of 330
building tree 195 of 330
building tree 196 of 330
building tree 197 of 330
building tree 198 of 330
building tree 199 of 330
building tree 200 of 330
building tree 201 of 330
building tree 202 of 330
building tree 203 of 330
building tree 204 of 330
building tree 205 of 330
building tree 206 of 330
building tree 207 of 330
building tree 208 of 330
```

```
building tree 209 of 330
building tree 210 of 330
building tree 211 of 330
building tree 212 of 330
building tree 213 of 330
building tree 214 of 330
building tree 215 of 330
building tree 216 of 330
building tree 217 of 330
building tree 218 of 330
building tree 219 of 330
building tree 220 of 330
building tree 221 of 330
building tree 222 of 330
building tree 223 of 330
building tree 224 of 330
building tree 225 of 330
building tree 226 of 330
building tree 227 of 330
building tree 228 of 330
building tree 229 of 330
building tree 230 of 330
building tree 231 of 330
building tree 232 of 330
building tree 233 of 330
building tree 234 of 330
building tree 235 of 330
building tree 236 of 330
building tree 237 of 330
building tree 238 of 330
building tree 239 of 330
building tree 240 of 330
building tree 241 of 330
building tree 242 of 330
building tree 243 of 330
building tree 244 of 330
building tree 245 of 330
building tree 246 of 330
building tree 247 of 330
building tree 248 of 330
building tree 249 of 330
building tree 250 of 330
building tree 251 of 330
building tree 252 of 330
building tree 253 of 330
building tree 254 of 330
building tree 255 of 330
building tree 256 of 330
building tree 257 of 330
building tree 258 of 330
building tree 259 of 330
building tree 260 of 330
building tree 261 of 330
```

```
building tree 262 of 330
building tree 263 of 330
building tree 264 of 330
building tree 265 of 330
building tree 266 of 330
building tree 267 of 330
building tree 268 of 330
building tree 269 of 330
building tree 270 of 330
building tree 271 of 330
building tree 272 of 330
building tree 273 of 330
building tree 274 of 330
building tree 275 of 330
building tree 276 of 330
building tree 277 of 330
building tree 278 of 330
building tree 279 of 330
building tree 280 of 330
building tree 281 of 330
building tree 282 of 330
building tree 283 of 330
building tree 284 of 330
building tree 285 of 330
building tree 286 of 330
building tree 287 of 330
building tree 288 of 330
building tree 289 of 330
building tree 290 of 330
building tree 291 of 330
building tree 292 of 330
building tree 293 of 330
building tree 294 of 330
building tree 295 of 330
building tree 296 of 330
building tree 297 of 330
building tree 298 of 330
building tree 299 of 330
building tree 300 of 330
building tree 301 of 330
building tree 302 of 330
building tree 303 of 330
building tree 304 of 330
building tree 305 of 330
building tree 306 of 330
building tree 307 of 330
building tree 308 of 330
building tree 309 of 330
building tree 310 of 330
building tree 311 of 330
building tree 312 of 330
building tree 313 of 330
building tree 314 of 330
```

```
building tree 315 of 330
building tree 316 of 330
building tree 317 of 330
building tree 318 of 330
building tree 319 of 330
building tree 320 of 330
building tree 321 of 330
building tree 322 of 330
building tree 323 of 330
building tree 324 of 330
building tree 325 of 330
building tree 326 of 330
building tree 327 of 330
building tree 328 of 330
building tree 329 of 330
building tree 330 of 330

[Parallel(n_jobs=-1)]: Done 330 out of 330 | elapsed:   40.9s fini
shed
```

Out[121]:
```
RandomForestRegressor(max_depth=100, min_samples_split=4, n_estima
tors=330,
                      n_jobs=-1, random_state=42, verbose=2)
```

In [122]:
```python
model_final_accuracy = evaluate(model_final,X_test,y_test)
```

```
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concur
rent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed:    0.2s

Model Performance
Average Error:4.0439 degrees.
Accuracy = 85.46%.

[Parallel(n_jobs=2)]: Done 330 out of 330 | elapsed:    0.5s finis
hed
```

In [123]:
```
model_final.feature_importances_
```

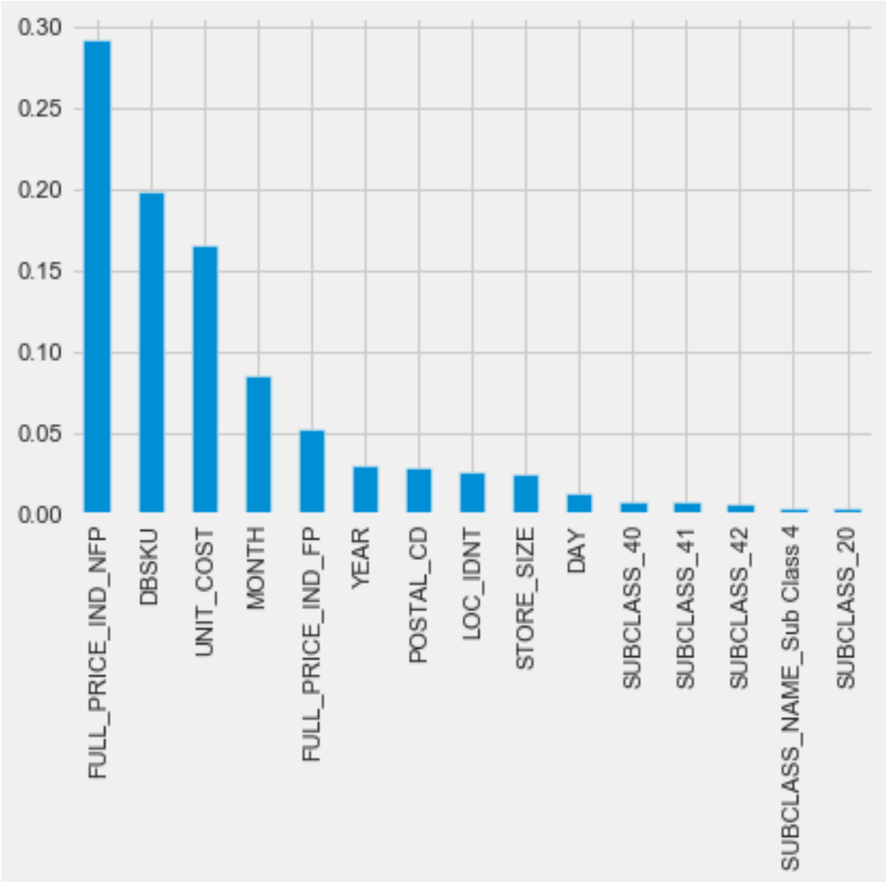
Out[123]:
```
array([2.62510504e-02, 1.98860303e-01, 2.95661217e-02, 2.55289859e
-02,
       1.26014662e-02, 8.54160787e-02, 2.97623123e-02, 1.66016889e
-01,
       3.13878176e-03, 2.30870764e-03, 2.81561374e-03, 3.79160646e
-03,
       2.44188465e-03, 1.20084619e-03, 3.27565931e-05, 2.85618649e
-03,
       1.71068459e-03, 3.69321204e-03, 2.25678085e-03, 2.80873268e
-03,
       2.44697995e-03, 1.94173456e-03, 8.11087917e-03, 7.79789073e
-03,
       6.88905366e-03, 2.73743288e-03, 1.10128332e-03, 2.44700385e
-03,
       2.87547916e-05, 1.59457366e-03, 1.52544772e-03, 1.28620651e
-03,
       1.58242917e-04, 5.40363687e-04, 1.03728977e-03, 3.75462624e
-04,
       2.16708913e-04, 9.84249014e-04, 2.19105921e-03, 2.62790411e
-03,
       1.81317017e-03, 2.70655089e-03, 6.21870971e-05, 8.31608444e
-04,
       2.31075398e-04, 2.03121920e-04, 5.31694205e-02, 2.91885343e
-01])
```

In [124]:
```
imp_feat=pd.Series(model_final.feature_importances_, index=feature
s.columns.tolist())
imp_feat.sort_values(ascending = False)[:15]
```

Out[124]:
```
FULL_PRICE_IND_NFP             0.291885
DBSKU                          0.198860
UNIT_COST                      0.166017
MONTH                          0.085416
FULL_PRICE_IND_FP              0.053169
YEAR                           0.029762
POSTAL_CD                      0.029566
LOC_IDNT                       0.026251
STORE_SIZE                     0.025529
DAY                            0.012601
SUBCLASS_40                    0.008111
SUBCLASS_41                    0.007798
SUBCLASS_42                    0.006889
SUBCLASS_NAME_Sub Class 4      0.003792
SUBCLASS_20                    0.003693
dtype: float64
```

In [125]:
```
imp_feat.sort_values(ascending = False)[:15].plot(kind='bar')
```

Out[125]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x68afc8c388>
```



In [ ]:

In [126]:
```
# Let's PREDICT THE PRICE
PricePred = model_final.predict(X_train)
```

```
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concur
rent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed:    0.9s
[Parallel(n_jobs=2)]: Done 330 out of 330 | elapsed:    1.8s finis
hed
```

In [127]:
```
PricePred_df = pd.DataFrame(PricePred)
PricePred_df.head()
```

Out[127]:

	0
0	36.421319
1	34.712388
2	39.498403
3	38.911634
4	53.602488

In [128]:
```
PricePred_df = PricePred_df.rename(columns={0:'Predicted_Price'})
```

In [129]:
```
PricePred_df.head()
```

Out[129]:

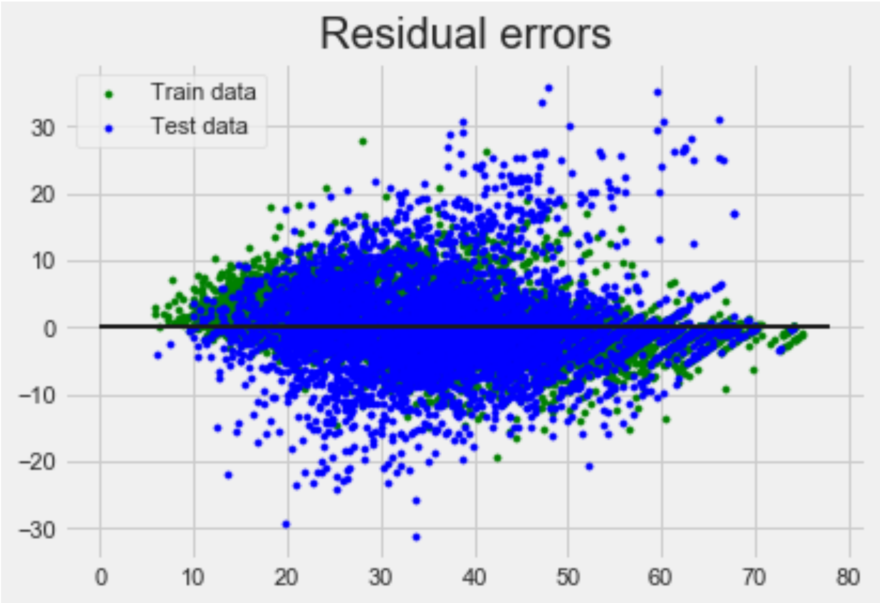
	Predicted_Price
0	36.421319
1	34.712388
2	39.498403
3	38.911634
4	53.602488

In [130]:
```
PricePred_df.shape
```

Out[130]:
```
(26413, 1)
```

In [194]:
```
y_pred = model_final.predict(X_test)
# Plot for residual error for the RANDOM FOREST REGRESSOR Model
plt.style.use('fivethirtyeight')
# Plot residual errors in training data
plt.scatter(model_final.predict(X_train), model_final.predict(X_tra
in) - y_train, color = "green",
            s = 10, label = 'Train data')
# Plot residual errors in test data
plt.scatter(y_pred,y_pred - y_test, color = "blue",
            s = 10, label = 'Test data')
## Plotting line for zero residual error
plt.hlines(y = 0, xmin = 0, xmax = 78, linewidth = 2)
## Plotting legend
plt.legend(loc ='upper left')
## Plot title
plt.title("Residual errors")
plt.show()
```

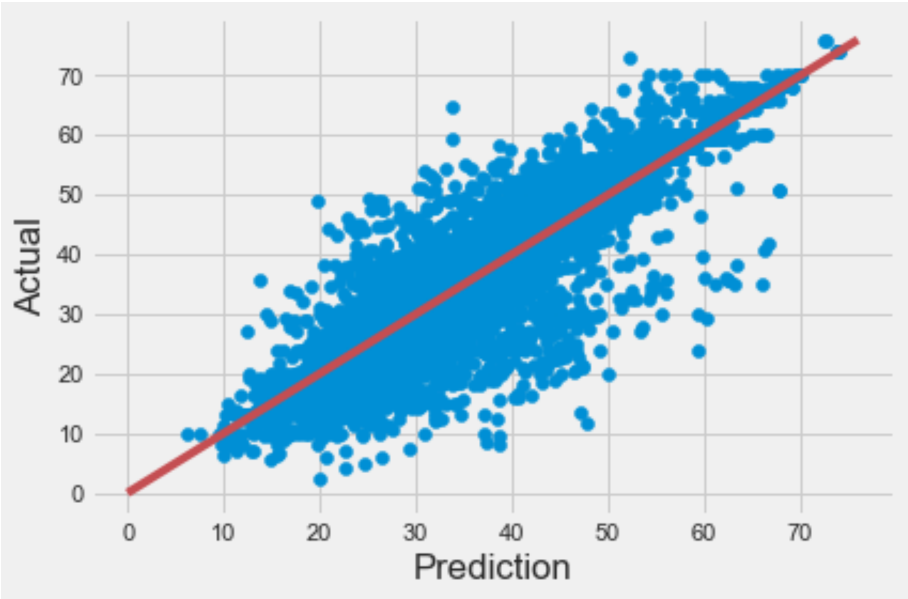
```
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concur
rent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 330 out of 330 | elapsed:    0.4s finis
hed
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concur
rent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed:    0.6s
[Parallel(n_jobs=2)]: Done 330 out of 330 | elapsed:    1.3s finis
hed
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concur
rent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed:    0.6s
[Parallel(n_jobs=2)]: Done 330 out of 330 | elapsed:    1.5s finis
hed
```



In [ ]:

In [131]:
```
y_pred = model_final.predict(X_test)

# Build a plot
plt.scatter(y_pred, y_test)
plt.xlabel('Prediction')
plt.ylabel('Actual')

# Now add the perfect prediction line
diagonal = np.linspace(0, np.max(y_test), 100)
plt.plot(diagonal, diagonal, '-r')
plt.show()
```

```
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concur
rent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed:    0.2s
[Parallel(n_jobs=2)]: Done 330 out of 330 | elapsed:    0.5s finis
hed
```



In [132]:
```
from sklearn.metrics import mean_absolute_error
validation_predictions = model_final.predict(X_test)

validation_prediction_errors = mean_absolute_error(y_test, validati
on_predictions)

validation_prediction_errors
```

```
[Parallel(n_jobs=2)]: Using backend ThreadingBackend with 2 concur
rent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:    0.0s
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed:    0.1s
[Parallel(n_jobs=2)]: Done 330 out of 330 | elapsed:    0.4s finis
hed
```

Out[132]: 4.043857218050528

In [133]:
```
from sklearn.metrics import mean_squared_log_error

print('MAE:\t$%.2f' % mean_absolute_error(y_test, y_pred))
print('MSLE:\t%.5f' % mean_squared_log_error(y_test, y_pred))
```

```
MAE:	$4.04
MSLE:	0.04101
```

In [ ]:

Optimal Price and Confidence Intervals

In [134]:
```
from sklearn.ensemble import GradientBoostingRegressor

# Set lower and upper quantile
LOWER_ALPHA = 0.1
UPPER_ALPHA = 0.9

#Each model has to be separate
lower_model = GradientBoostingRegressor(loss="quantile",alpha=LOWER_ALPHA)

#The mid modelwill use the default loss
mid_model = GradientBoostingRegressor(loss="ls")

upper_model = GradientBoostingRegressor(loss="quantile",alpha=UPPER_ALPHA)
```

In [135]:
```
# Fit models
lower_model.fit(X_train, y_train)
mid_model.fit(X_train, y_train)
upper_model.fit(X_train, y_train)

# Record actual values on test set
predictions = pd.DataFrame(y_test)

#Predict
predictions['lower'] = lower_model.predict(X_test)
predictions['mid'] = mid_model.predict(X_test)
predictions['upper'] = upper_model.predict(X_test)

print(predictions)
```

```
         UNIT_PRICE      lower        mid      upper
8070497       31.36  20.770770  36.915526  43.744457
2338770       18.62  14.822289  27.281679  43.373915
2129208       44.80  23.048162  42.474627  53.860184
4004185       21.00  19.838728  28.602717  39.358592
8182937       54.35  20.863218  38.490079  50.893279
...             ...        ...        ...        ...
7950230       76.00  58.569038  65.700326  70.228634
49377         11.52  16.076383  29.965231  46.597373
2281404       30.80  21.205157  30.207343  37.734566
1067750       21.00  19.609851  28.642351  37.926679
685097        30.00  29.913078  30.767914  34.821279

[6604 rows x 4 columns]
```

In [ ]:

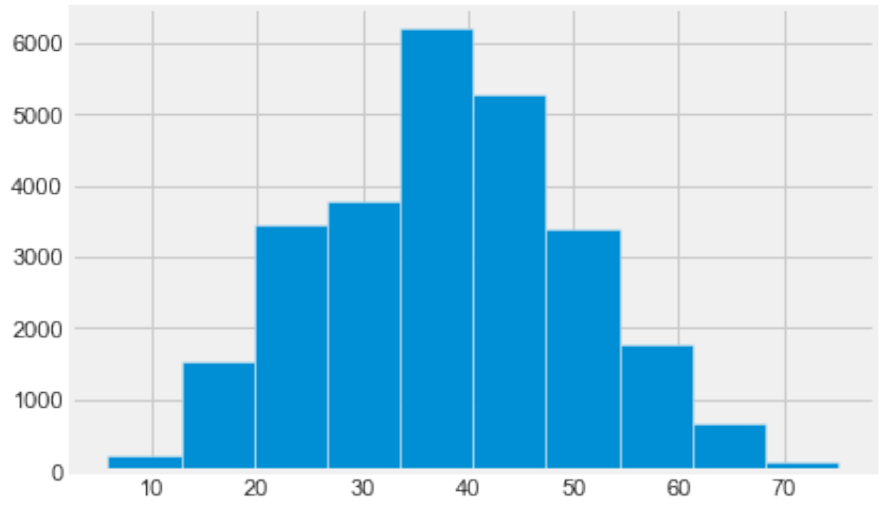
In [136]:
```
from scipy.stats import norm
import numpy as np
```

In [137]:
```
norm.ppf(0.975) # 95% of confidence level
```

Out[137]:
```
1.959963984540054
```

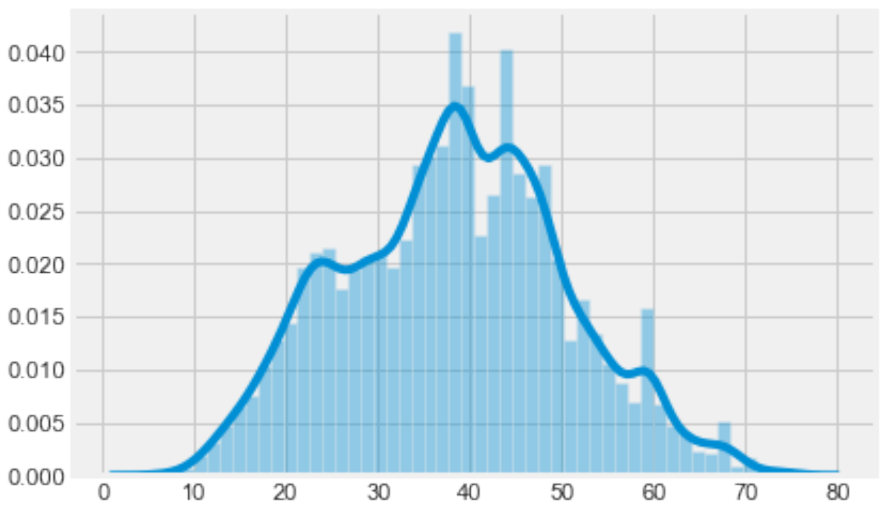
In [138]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import math
```

In [139]:
```
plt.hist(PricePred)
plt.show()
```



In [140]:
```
sns.distplot(PricePred)
```

Out[140]: <matplotlib.axes._subplots.AxesSubplot at 0x68af079908>



In [141]:
```
n = len(PricePred_df)

con_coef = .95

# The alpha level
alpha = 1. - con_coef
```

In [142]:
```
x_bar = PricePred_df.mean()
x_bar
```

Out[142]:
```
Predicted_Price    38.284827
dtype: float64
```

In [143]:
```
sigma = PricePred_df.std()
sigma
```

Out[143]:
```
Predicted_Price    12.194137
dtype: float64
```

In [144]:
```
import scipy.stats as stats

z_critical = stats.norm.ppf(q = 0.975)
z_critical
```

Out[144]:
```
1.959963984540054
```

In [145]:
```
zinterval = stats.norm.interval(alpha=con_coef)
zinterval
```

Out[145]:
```
(-1.959963984540054, 1.959963984540054)
```

In [146]:
```
# Standard Error needed to calculate the bounds
standard_error = sigma / math.sqrt(n)
standard_error
```

Out[146]:
```
Predicted_Price    0.075031
dtype: float64
```

In [147]:
```
CI_lower = x_bar - z_critical * standard_error
CI_upper = x_bar + z_critical * standard_error
```

In [148]:
```
# This would be the the optimal average price lies, feeding our for
mula with the standard_error
CI_lower, CI_upper
```

Out[148]:
```
(Predicted_Price    38.137769
 dtype: float64, Predicted_Price    38.431886
 dtype: float64)
```

In [149]:
```
# TAKING SAMPLE to cross validate our optimal price level of confid
ence
n_sample = 10000
Price_sample = PricePred_df.ix[np.random.choice(PricePred_df.index,
n)]
Price_sample.head()
```

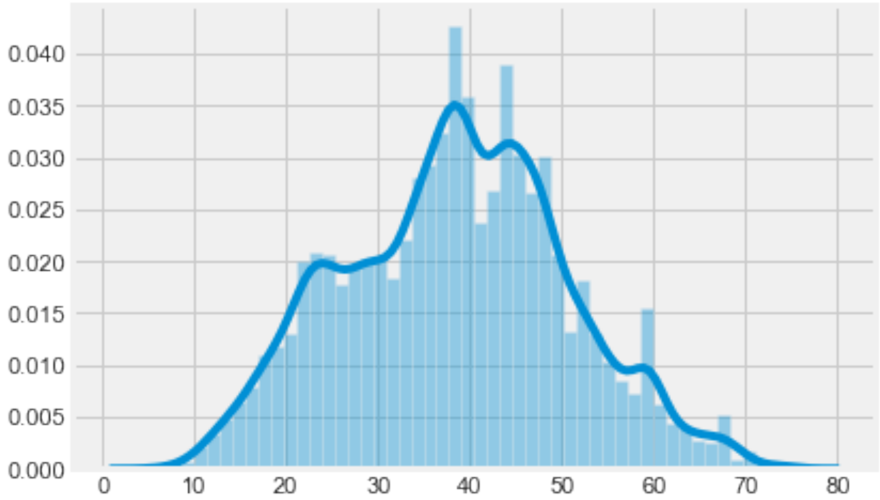
```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:
3: FutureWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.ht
ml#ix-indexer-is-deprecated
  This is separate from the ipykernel package so we can avoid doin
g imports until
```

Out[149]:

	Predicted_Price
2605	40.967729
24133	24.193187
20680	37.329838
595	51.311299
3177	15.155890

In [150]:
```
## We can see the distribution is almost the same as the NOT sample
d PredPrice variable

import seaborn as sns

sns.distplot(Price_sample)
```

Out[150]: <matplotlib.axes._subplots.AxesSubplot at 0x68afd91308>



In [151]:
```
# Let's again calculate what is necessary to obtain our range...
# The range obtained by our sample is incredibly similar to that of
the population.

xbar_sample = Price_sample.mean()
xbar_sample

sigma_sample = Price_sample.std()
sigma_sample

SE_sample = sigma_sample / math.sqrt(n_sample)
SE_sample

CI_lower_sample = xbar_sample - z_critical * SE_sample
CI_upper_sample = xbar_sample + z_critical * SE_sample

CI_lower_sample, CI_upper_sample
```

Out[151]:
```
(Predicted_Price    38.157116
 dtype: float64, Predicted_Price    38.636205
 dtype: float64)
```

In [ ]:

In [152]:
```
# ONCE we made sure that our results are CI are crossvalidated
# we moved on to get the CI at a level of conf of 95% for our predi
cted prices grouped by SUBCLASS
# using the error provided by our RANDOM Forest' model
```

In [153]:
```
# Applying the MAE deriving from our RANDOM FOREST model (instead o
f the standard_error)
model_error = 4.0540
```

In [154]:
```
CI_lower_mod = x_bar - z_critical * model_error
CI_upper_mod = x_bar + z_critical * model_error
```

In [155]:
```
# This is the interval where the optimal average price lies. (with
a level of conf of 95%)
CI_lower_mod, CI_upper_mod
```

Out[155]:
```
(Predicted_Price    30.339133
 dtype: float64, Predicted_Price    46.230521
 dtype: float64)
```

In [156]:
```
# Let's calculate the CI for the whole list of predicted prices gen
erated (level of conf 95%)
CI_lower_mod1 = PricePred - z_critical * model_error
CI_upper_mod1 = PricePred + z_critical * model_error
```

In [157]:
```
# This is the interval where the optimal prices lie. (with a level
of conf of 95%)
CI_lower_mod1, CI_upper_mod1
```

Out[157]:
```
(array([28.4756251 , 26.76669396, 31.55270895, ..., 24.46830051,
        37.92578725, 20.96119468]),
 array([44.36701309, 42.65808194, 47.44409694, ..., 40.3596885 ,
        53.81717523, 36.85258267]))
```

In [158]:
```
# Converting lower and upper CI to a DF will help us building a fin
al chart to represent the optimal PRICES
CI_lower_mod_df = pd.DataFrame(CI_lower_mod1)
CI_lower_mod_df = CI_lower_mod_df.rename(columns={0:'Lower CI'})
CI_lower_mod_df.head()
```

Out[158]:

	Lower CI
0	28.475625
1	26.766694
2	31.552709
3	30.965940
4	45.656794

In [159]:
```
CI_upper_mod_df = pd.DataFrame(CI_upper_mod1)
CI_upper_mod_df = CI_upper_mod_df.rename(columns={0:'Upper CI'})
CI_upper_mod_df.head()
```

Out[159]:

	Upper CI
0	44.367013
1	42.658082
2	47.444097
3	46.857328
4	61.548182

In [160]:
```
Unit_Price_df =pd.DataFrame(y_train)
Unit_Price_df = Unit_Price_df.rename(columns={'UNIT_PRICE':'Actual
Price'})
Unit_Price_df.head()
```

Out[160]:

	Actual Price
616830	36.00
8178533	33.60
466550	42.78
8621450	39.50
5868536	54.00

In [161]:
```
Unit_Price_df.reset_index(inplace=True)
```

In [162]:
```
Unit_Price_df = Unit_Price_df.rename(columns={'index':'Index'})
```

In [163]:
```
Unit_Price_df.head()
```

Out[163]:

	Index	Actual Price
0	616830	36.00
1	8178533	33.60
2	466550	42.78
3	8621450	39.50
4	5868536	54.00

In [164]:
```
# After few DF conversion
## let's create an object that concatenates the random index of spl
it of our dataset,
# the actual price, and the predicted price with its lowwe and uppe
r CI (conf 95%)
Intervals = pd.concat([Unit_Price_df, PricePred_df, CI_lower_mod_d
f, CI_upper_mod_df],axis=1,sort=False)
```

In [165]:
```
Intervals.head()
```

Out[165]:

	Index	Actual Price	Predicted_Price	Lower CI	Upper CI
0	616830	36.00	36.421319	28.475625	44.367013
1	8178533	33.60	34.712388	26.766694	42.658082
2	466550	42.78	39.498403	31.552709	47.444097
3	8621450	39.50	38.911634	30.965940	46.857328
4	5868536	54.00	53.602488	45.656794	61.548182

In [166]:
```
# Let's get back to our train2 dataset where a copy of our split da
taset without dummification is present
# Let's start some DF conversion and 'cleaning'
```

In [167]:
```
# A copy of my original split dataset, only without dummies!
X_train2.head()
```

Out[167]:

	LOC_IDNT	DBSKU	ONLINE_FLAG	FULL_PRICE_IND	DEPARTMENT	SUBCLAS
616830	494	472522.0	0.0	NFP	10	4
8178533	234	600882.0	0.0	NFP	10	2
466550	573	539015.0	0.0	NFP	10	2
8621450	529	533539.0	0.0	NFP	10	3
5868536	1159	2109512.0	0.0	FP	12	4

In [168]:
```
# SUBCLASS
Subclass_df = pd.DataFrame(X_train2['SUBCLASS'])
Subclass_df.shape
Subclass_df.head()
```

Out[168]:

	SUBCLASS
616830	41
8178533	21
466550	20
8621450	31
5868536	40

In [169]:
```
## Re-INDEXING
Subclass_df.reset_index(inplace=True)
Subclass_df = Subclass_df.rename(columns={'index':'Index'})
Subclass_df.head()
```

Out[169]:

	Index	SUBCLASS
0	616830	41
1	8178533	21
2	466550	20
3	8621450	31
4	5868536	40

In [170]:
```
# SUBCLASS_NAME
Subclass_n_df = pd.DataFrame(X_train2['SUBCLASS_NAME'])
Subclass_n_df.shape
Subclass_n_df.head()
```

Out[170]:

	SUBCLASS_NAME
616830	Sub Class 2
8178533	Sub Class 2
466550	Sub Class 4
8621450	Sub Class 2
5868536	Sub Class 4

In [171]:
```
# Re-INDEXING
Subclass_n_df.reset_index(inplace=True)
Subclass_n_df = Subclass_n_df.rename(columns={'index':'Index1'})
Subclass_n_df.head()
```

Out[171]:

	Index1	SUBCLASS_NAME
0	616830	Sub Class 2
1	8178533	Sub Class 2
2	466550	Sub Class 4
3	8621450	Sub Class 2
4	5868536	Sub Class 4

In [172]:
```
# Subclasses = Subclass df concatenated to Subclass_name df
```

In [173]:
```
Subclasses = pd.concat([Subclass_df,Subclass_n_df],axis=1,sort=Fals
e)
```

In [174]:
```
Subclasses.head()
```

Out[174]:

	Index	SUBCLASS	Index1	SUBCLASS_NAME
0	616830	41	616830	Sub Class 2
1	8178533	21	8178533	Sub Class 2
2	466550	20	466550	Sub Class 4
3	8621450	31	8621450	Sub Class 2
4	5868536	40	5868536	Sub Class 4

In [175]:
```
Subclasses = Subclasses.drop(["Index"],axis=1)
```

In [176]:
```
# Our chart showing pred prices with CI and related SUBCLASS
CI_subclass = pd.concat([Intervals,Subclasses],axis=1,sort=False)
```

In [177]:
```
CI_subclass = CI_subclass[["Index", "SUBCLASS", "SUBCLASS_NAME", "A
ctual Price", "Predicted_Price",
                           "Lower CI", "Upper CI"]]
```

In [178]:
```
CI_subclass.head()
```

Out[178]:

	Index	SUBCLASS	SUBCLASS_NAME	Actual Price	Predicted_Price	Lower CI	Upper CI
0	616830	41	Sub Class 2	36.00	36.421319	28.475625	44.367013
1	8178533	21	Sub Class 2	33.60	34.712388	26.766694	42.658082
2	466550	20	Sub Class 4	42.78	39.498403	31.552709	47.444097
3	8621450	31	Sub Class 2	39.50	38.911634	30.965940	46.857328
4	5868536	40	Sub Class 4	54.00	53.602488	45.656794	61.548182

In [179]:
```
# Calculate the optimal average price and its CI (conf=95%) per eac
h subclass
```

In [180]:
```
# Let's use groupby and aggregate function to check the mean Predic
ted Price per SUBCLASS_NAME
Subclass_Predicted = CI_subclass.groupby(['SUBCLASS_NAME']).agg({'P
redicted_Price': ['mean','median','min', 'max','count']})
print(Subclass_Predicted)
```

```
                Predicted_Price                                    
                           mean     median        min        max  co
unt
SUBCLASS_NAME
Sub Class 1           41.256353  41.708693   8.764416  75.119227   3
487
Sub Class 2           37.495549  38.210553   5.909506  73.594056   6
815
Sub Class 3           35.297244  36.309508   8.314859  64.096223   2
051
Sub Class 4           38.938613  38.915250   6.677076  74.630069  12
836
Sub Class 5           28.597024  30.012121  11.094582  42.611436    
415
Sub Class 6           34.350511  35.898228   5.855392  46.061205    
806
Sub Class 7           19.707178  22.924619  11.439001  24.757915    
3
```

In [181]:
```
# Let's use groupby and aggregate function to check the mean Actual Price per SUBCLASS_NAME
Subclass_Actual = CI_subclass.groupby(['SUBCLASS_NAME']).agg({'Actual Price': ['mean','median','min', 'max','count']})
print(Subclass_Actual)
```

```
              Actual Price
                      mean median   min    max  count
SUBCLASS_NAME
Sub Class 1      41.364032  43.20  0.01  76.00   3487
Sub Class 2      37.544437  39.00  0.01  76.00   6815
Sub Class 3      35.181468  37.09  3.84  64.00   2051
Sub Class 4      38.963842  39.50  2.09  76.00  12836
Sub Class 5      28.366458  30.00  6.80  45.00    415
Sub Class 6      34.275403  37.53  2.94  47.92    806
Sub Class 7      14.136667  14.42  4.99  23.00      3
```

In [182]:
```
# Subclass X_bars (MEAN)
xbar_SUB1 = 41.256353
xbar_SUB2 = 37.495549
xbar_SUB3 = 35.297244
xbar_SUB4 = 38.938613
xbar_SUB5 = 28.597024
xbar_SUB6 = 34.350511
xbar_SUB7 = 19.707178
```

In [183]:
```
# Let's calculate the lower and upper level at a Confidence interval of 95% for each SUBCLASS
CI_lower_SUB_1 = xbar_SUB1 - z_critical * model_error
CI_upper_SUB_1 = xbar_SUB1 + z_critical * model_error

CI_lower_SUB_2 = xbar_SUB2 - z_critical * model_error
CI_upper_SUB_2 = xbar_SUB2 + z_critical * model_error

CI_lower_SUB_3 = xbar_SUB3 - z_critical * model_error
CI_upper_SUB_3 = xbar_SUB3 + z_critical * model_error

CI_lower_SUB_4 = xbar_SUB4 - z_critical * model_error
CI_upper_SUB_4 = xbar_SUB4 + z_critical * model_error

CI_lower_SUB_5 = xbar_SUB5 - z_critical * model_error
CI_upper_SUB_5 = xbar_SUB5 + z_critical * model_error

CI_lower_SUB_6 = xbar_SUB6 - z_critical * model_error
CI_upper_SUB_6 = xbar_SUB6 + z_critical * model_error

CI_lower_SUB_7 = xbar_SUB7 - z_critical * model_error
CI_upper_SUB_7 = xbar_SUB7 + z_critical * model_error
```

In [184]:
```
print('CI for Predicted Prices of SUBCLASS 1 ')
print(CI_lower_SUB_1, '|', CI_upper_SUB_1)
print('======================================')
print('CI for Predicted Prices of SUBCLASS 2 ')
print(CI_lower_SUB_2, '|', CI_upper_SUB_2)
print('======================================')
print('CI for Predicted Prices of SUBCLASS 3 ')
print(CI_lower_SUB_3, '|', CI_upper_SUB_3)
print('======================================')
print('CI for Predicted Prices of SUBCLASS 4 ')
print(CI_lower_SUB_4, '|', CI_upper_SUB_4)
print('======================================')
print('CI for Predicted Prices of SUBCLASS 5 ')
print(CI_lower_SUB_5, '|', CI_upper_SUB_5)
print('======================================')
print('CI for Predicted Prices of SUBCLASS 6 ')
print(CI_lower_SUB_6, '|', CI_upper_SUB_6)
print('======================================')
print('CI for Predicted Prices of SUBCLASS 7 ')
print(CI_lower_SUB_7, '|', CI_upper_SUB_7)
```

```
CI for Predicted Prices of SUBCLASS 1
33.31065900667462 | 49.202046993325375
======================================
CI for Predicted Prices of SUBCLASS 2
29.54985500667462 | 45.441242993325375
======================================
CI for Predicted Prices of SUBCLASS 3
27.35155000667462 | 43.24293799332538
======================================
CI for Predicted Prices of SUBCLASS 4
30.99291900667462 | 46.884306993325374
======================================
CI for Predicted Prices of SUBCLASS 5
20.65133000667462 | 36.54271799332538
======================================
CI for Predicted Prices of SUBCLASS 6
26.40481700667462 | 42.296204993325375
======================================
CI for Predicted Prices of SUBCLASS 7
11.76148400667462 | 27.652871993325377
```

In [185]:
```
# Let's create the DF with all the data gathered above to better sh
owcase our results indexed by
# SUBCLASS
```

In [186]:
```
CI_subclass_avg = {'Actual_Price_avg' : [41.364032,37.544437,35.181
468,38.963842,
                                          28.366458,34.275403,14.1366
67],
                   'Predicted_Price_avg' : [41.256353,37.495549,35.
297244,38.938613,
                                             28.597024,34.350511,19.7
07178],
                   'Lower CI': [33.310659,29.549855,27.351550,30.99
2919,
                                20.651330,26.404817,11.761484],
                  'Upper CI' : [49.202046,45.441242,43.242937,46.88
4306,
                                36.542717,42.296204,27.652871]}

CI_subclass_avg_df = pd.DataFrame(CI_subclass_avg, columns = ['Actu
al_Price_avg',
                                                              'Predi
cted_Price_avg', 'Lower CI',
                                                              'Upper
CI'], index=['SUBCLASS_1',
'SUBCLASS_2',
'SUBCLASS_3',
'SUBCLASS_4',
'SUBCLASS_5',
'SUBCLASS_6',
'SUBCLASS_7'])
```

In [187]:
```
CI_subclass_avg_df
```

Out[187]:

	Actual_Price_avg	Predicted_Price_avg	Lower CI	Upper CI
SUBCLASS_1	41.364032	41.256353	33.310659	49.202046
SUBCLASS_2	37.544437	37.495549	29.549855	45.441242
SUBCLASS_3	35.181468	35.297244	27.351550	43.242937
SUBCLASS_4	38.963842	38.938613	30.992919	46.884306
SUBCLASS_5	28.366458	28.597024	20.651330	36.542717
SUBCLASS_6	34.275403	34.350511	26.404817	42.296204
SUBCLASS_7	14.136667	19.707178	11.761484	27.652871

In [ ]: