Task:Excel Functionality and Data Normalization

In [1182]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from pandas.tools.plotting import autocorrelation_plot
from statsmodels.tsa.arima_model import ARIMA
from sklearn import preprocessing
```

Data Loading

Note: some of the eda analysis is performed in notebook for "Python Test"

In [828]:
```
xls = pd.ExcelFile('Excel_Credit_Card_Data.xlsx')
transaction_daily = pd.read_excel(xls, 'Daily Data')
transaction_panel = pd.read_excel(xls, 'Total Panel Information')
```

Data Processing

Filter merchants that we would like to perform analysis on

In [829]:
```
merchant_list = ['Papa Johns','Pizza Hut','Dominos Pizza','McDonalds','Wendys','Burge
data = transaction_daily.loc[transaction_daily['MERCHANT'].isin(merchant_list)]
```

overiew of data

In [830]:
```
for col in data.columns:
    print(data[col].describe())
```

```
count                    5430
unique                    899
top       2014-10-29 00:00:00
freq                       12
first     2012-06-21 00:00:00
last      2014-12-31 00:00:00
Name: TRANSACTION_DATE, dtype: object
count          5430
unique            6
top       YUM-USAA
freq            905
Name: COMPANY, dtype: object
count           5430
unique             6
top        Pizza Hut
freq             905
Name: MERCHANT, dtype: object
count       5430.000000
mean       26484.863536
```

In [831]:
```
data.TRANSACTION_DATE = pd.to_datetime(data['TRANSACTION_DATE'])
```

In [832]:
```
data.set_index(['TRANSACTION_DATE','MERCHANT'],inplace=True)
```

We found 905 date values but only 899 of them are unique, need to drop the duplicated index

In [833]:
```
data[data.groupby(data.index).count()['COMPANY'] > 1]
```

```
/Users/charles-18/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  """Entry point for launching an IPython kernel.
```

We found two entires for all merchants for trasaction_date from 2014-10-26 to 2014-10-31.Surprisingly, october 2014 has a large portion of its date index missing

In [834]:
```
data = data.loc[~data.index.duplicated(keep='first')]
data = data.reset_index().set_index('TRANSACTION_DATE')
```

For each merchant, aggregate sales by month

Use transaction value to represent sales number instead of transaction count (which is a volume measure)

In [837]:
```
df = pd.DataFrame([])
for merchant in merchant_list:
    df[merchant] = data[data['MERCHANT'] == merchant].groupby(pd.Grouper(freq='M'))['
```

In [839]:
```
df.head()
```

Out[839]:

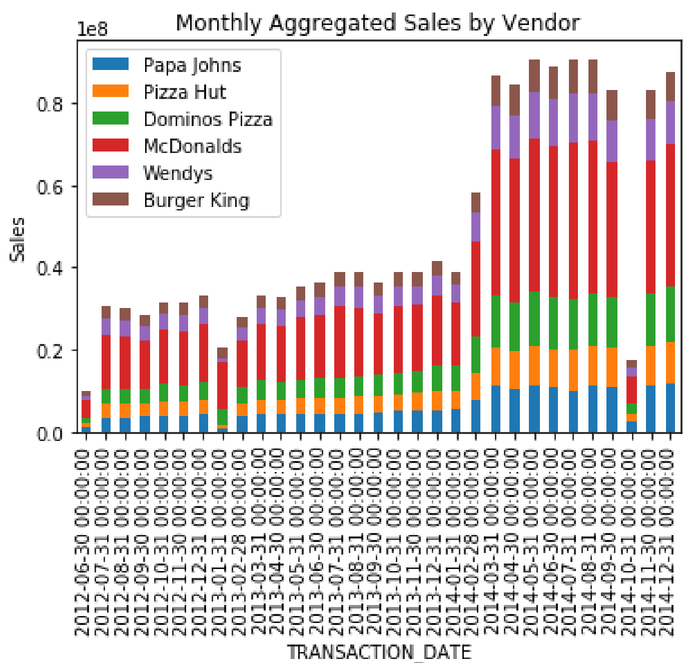
	Papa Johns	Pizza Hut	Dominos Pizza	McDonalds	Wendys	Burger King
TRANSACTION_DATE						
2012-06-30	1.105055e+06	1.000628e+06	1.195164e+06	4.344916e+06	1.206916e+06	1.017541e+06
2012-07-31	3.396087e+06	3.340302e+06	3.589351e+06	1.341434e+07	3.719245e+06	3.123232e+06
2012-08-31	3.404822e+06	3.302680e+06	3.655413e+06	1.290179e+07	3.717556e+06	3.009567e+06
2012-09-30	3.688791e+06	3.212009e+06	3.683851e+06	1.169774e+07	3.289801e+06	2.843822e+06
2012-10-31	3.885754e+06	3.520777e+06	4.235643e+06	1.321084e+07	3.754474e+06	2.954558e+06

The following grap provides a visual of aggregated monthly transaction value data by merchant (month end)

1. we observe a sharp transaction value drop during 2014 october, this is caused by missing dates from october (justified later)
2. Pizza hut/Papa Johns/Wendys had large sale drop during 2013 Janurary whereas burger chains appear to be less affected
3. Mcdonald takes the largest sale share of the entire market

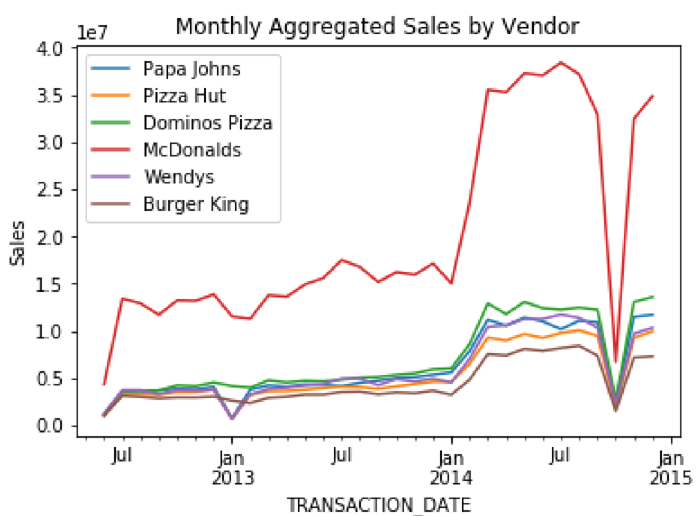
In [840]:
```
df.plot(kind='bar',stacked = True)
plt.title('Monthly Aggregated Sales by Vendor')
plt.ylabel('Sales')
```

Out[840]: Text(0, 0.5, 'Sales')



In [841]:
```
df.plot()
plt.title('Monthly Aggregated Sales by Vendor')
plt.ylabel('Sales')
```

Out[841]: Text(0, 0.5, 'Sales')



SQL: aggregate sales by month (assume TRANSACTION_DATE could be either string or date)

SELECT FORMAT(CONVERT(date, TRANSCATION_DATE),'yyyy-MM') Date,
MERCHANT,
sum(TRANSACTED_VALUE) TRANSACTED_VALUE_MONTHLY
FROM Daily_Data
WHERE MERCHANT in ('Papa Johns','Pizza Hut','Dominos Pizza','McDonalds','Wendys','Burger King')
GROUP BY DATEPART(YEAR, TRANSCATION_DATE), DATEPART(MONTH, TRANSCATION_DATE), MERCHANT

Compute Monthly Sales adjusting by panel size

First we need to adjust daily transaction data by panel size

In [842]:
```
data_adj_daily = pd.merge(data,transaction_panel,left_on=data.index,right_on=transact
```

In [844]:
```
data_adj_daily.head()
```

Out[844]:

	MERCHANT	COMPANY	TRANSACTION_COUNT	TRANSACTED_VALUE	NORMALIZATION_FACTOR
Date					
2012-06-21	Dominos Pizza	DPZ-USAA	4555	110335.863309	3728765
2012-06-21	McDonalds	MCD-USAA	56662	413671.839212	3728765
2012-06-21	Papa Johns	PZZA-USAA	3896	89310.510018	3728765
2012-06-21	Burger King	QSR-USAA	11266	95312.430822	3728765
2012-06-21	Wendys	WEN-USAA	13271	118047.214765	3728765

Assume Transacted_Value = transacted_Value * trasaction_count / normalization_factor, we are unable normalize the data (mcdoanld dominates others in terms of sales) and to get rid of the sharpe increase cross sectionally, which looks to be systematic. Therefore we use
Transacted_Value = Transacted_Value / normalizationz_factor

In [845]:
```
data_adj_daily['TRANSACTED_VALUE']  = data_adj_daily['TRANSACTED_VALUE'] / data_adj_c
```

In [846]:
```
df_adj = pd.DataFrame([])
for merchant in merchant_list:
    df_adj[merchant] = data_adj_daily[data_adj_daily['MERCHANT'] == merchant].groupby
```

In [847]:
```
df_adj.head()
```

Out[847]:

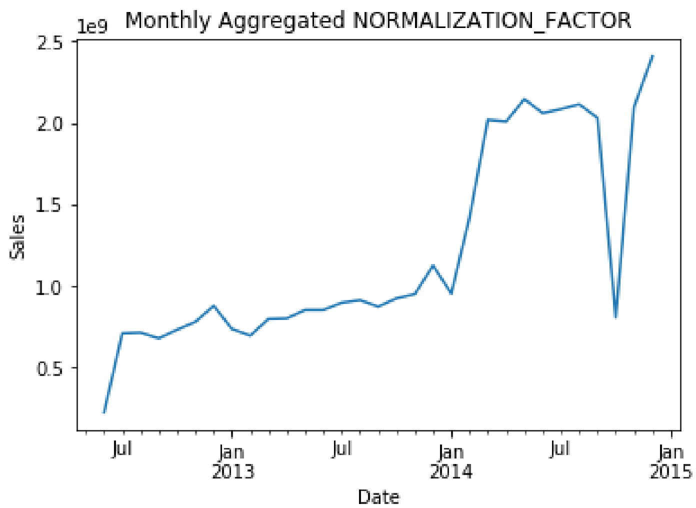
Date	Papa Johns	Pizza Hut	Dominos Pizza	McDonalds	Wendys	Burger King
2012-06-30	0.293800	0.267053	0.318262	1.156992	0.321056	0.270923
2012-07-31	0.889014	0.876775	0.940688	3.518445	0.973574	0.818745
2012-08-31	0.884303	0.861276	0.951367	3.366148	0.970339	0.785340
2012-09-30	0.975574	0.850902	0.974215	3.098485	0.870795	0.753888
2012-10-31	0.984422	0.891924	1.072802	3.352601	0.953917	0.750261

We still observe a drop for several merchants during 2013 Jan, this is not justified by normalization factor. (More analysis below)

Plot the monthly aggregated normalziation factor as a reference

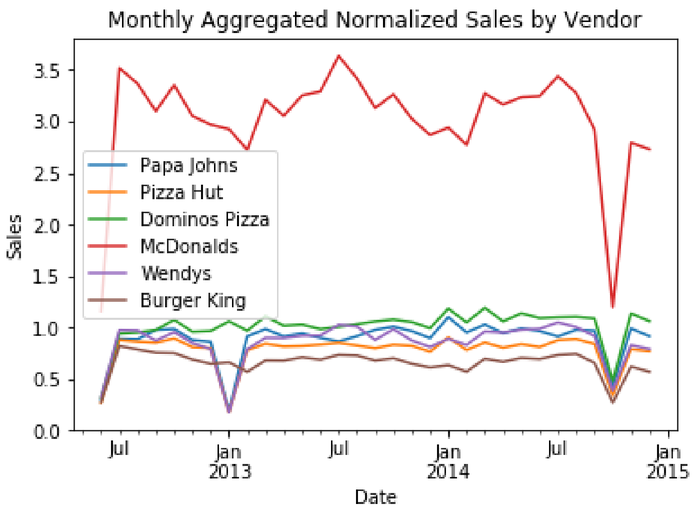
In [848]:
```
data_adj_daily['NORMALIZATION_FACTOR'].groupby(pd.Grouper(freq='M')).sum().plot()
plt.title('Monthly Aggregated NORMALIZATION_FACTOR')
plt.ylabel('Sales')
```

Out[848]:
```
Text(0, 0.5, 'Sales')
```



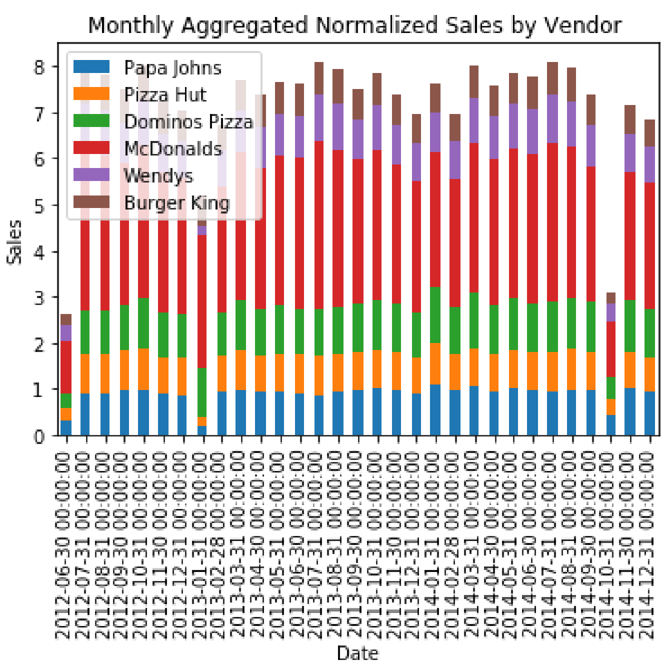
In [851]:
```
df_adj.plot()
plt.title('Monthly Aggregated Normalized Sales by Vendor')
plt.ylabel('Sales')
```

Out[851]: Text(0, 0.5, 'Sales')



In [852]:
```
df_adj.plot(kind='bar',stacked = True)
plt.title('Monthly Aggregated Normalized Sales by Vendor ')
plt.ylabel('Sales')
```

Out[852]: Text(0, 0.5, 'Sales')



SQL: Compute Monthly Sales adjusting by panel size

SELECT
FORMAT(CONVERT(date, a.TRANSCATION_DATE),'yyyy-MM') Date,
a.MERCHANT MERCHANT,
sum(a.TRANSACTED_VALUE/b.NORMALIZATION_FACTOR) TRANSACTED_VALUE_MONTHLY_ADJ
FROM Daily_Data a,Total_Panel_Information b

WHERE a.MERCHANT in ('Papa Johns','Pizza Hut','Dominos Pizza','McDonalds','Wendys','Burger King')
AND a.TRANSCATION_DATE = b.TRANSCATION_DATE
GROUP BY DATEPART(YEAR, a.TRANSCATION_DATE), DATEPART(MONTH, a.TRANSCATION_DATE), MERCHANT

Compare Monthly YOY Data

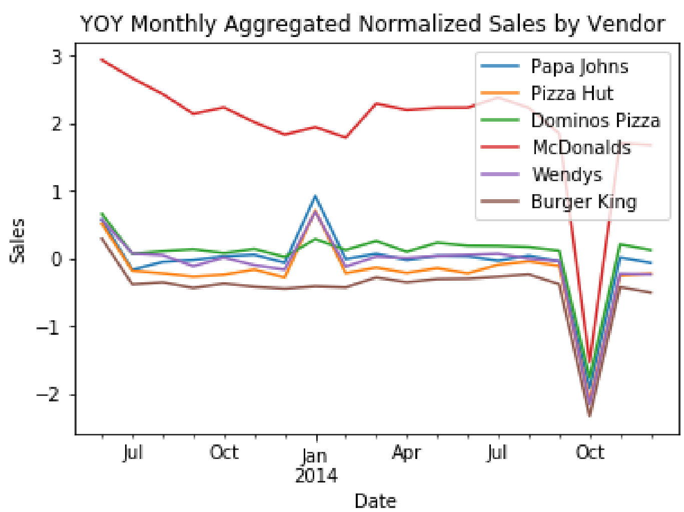
In [853]:
```
df_monthly_yoy = (df_adj - df_adj.shift(12)/ df_adj).dropna()
df_monthly_yoy.head()
```

Out[853]:

Date	Papa Johns	Pizza Hut	Dominos Pizza	McDonalds	Wendys	Burger King
2013-06-30	0.571679	0.512026	0.660127	2.939248	0.573515	0.291918
2013-07-31	-0.167870	-0.182385	0.068157	2.668350	0.074150	-0.383899
2013-08-31	-0.051363	-0.220705	0.107889	2.432894	0.047889	-0.353579
2013-09-30	-0.023573	-0.269150	0.136023	2.139727	-0.115466	-0.435178
2013-10-31	0.030389	-0.240426	0.078323	2.235734	0.011656	-0.371329

In [854]:
```
df_monthly_yoy.plot()
plt.title('YOY Monthly Aggregated Normalized Sales by Vendor ')
plt.ylabel('Sales')
```

Out[854]: `Text(0, 0.5, 'Sales')`



SQL: Compare Monthly YOY Data

SELECT * INTO #TMP FROM
(SELECT a.TRANSCATION_DATE,
a.MERCHANT MERCHANT,
DATEPART(YEAR, a.TRANSCATION_DATE) YEAR,
DATEPART(MONTH, a.TRANSCATION_DATE) MONTH,
sum(a.TRANSACTED_VALUE/b.NORMALIZATION_FACTOR) TRANSACTED_VALUE_MONTHLY_ADJ
FROM Daily_Data a,Total_Panel_Information b
WHERE a.MERCHANT in ('Papa Johns','Pizza Hut','Dominos Pizza','McDonalds','Wendys','Burger King')
AND a.TRANSCATION_DATE = b.TRANSCATION_DATE

GROUP BY DATEPART(YEAR, a.TRANSCATION_DATE), DATEPART(MONTH, a.TRANSCATION_DATE), MERCHANT))

SELECT FORMAT(CONVERT(date, a.TRANSCATION_DATE),'yyyy-MM') Date,
a.MERCHANT,
(a.TRANSACTED_VALUE_MONTHLY_ADJ - b.TRANSACTED_VALUE_MONTHLY_ADJ) / b.TRANSACTED_VALUE_MONTHLY_ADJ MONTHLY_YOY_GROWTH
from #TMP a,#TMP b
WHERE a.MERCHANT = b.MERCHANT
and a.YEAR = b.YEAR + 1
and a.MONTH = b.MONTH

SQL: WEEKLY VIEW OF DAILY DATA TABLE

Compute Aggregated sales data for a week.Use week start day as index

SELECT FORMAT(DATEADD(day, DATEDIFF(day, 0, CONVERT(date,TRANSCATION_DATE)) /7*7, 0),'yyyy-MM-dd') as Date,
MERCHANT,
sum(TRANSACTED_VALUE) TRANSACTED_VALUE,
sum(TRANSACTION_COUNT) TRANSACTION_COUNT,
FROM Daily_Data
GROUP BY datepart(year, TRANSCATION_DATE), datepart(week, TRANSCATION_DATE),MERCHANT

Data Modeling

Pull together the credit card, survey, and URL tracking data Construct a hierachiecal dataframe with Columns being "nQYY" format with super index being MERCHANTS/CROSS-SECTIONAL SURVEY and subindex being specific measures such as Average Minutes per Visit, Sales Value (modeled by transaction data) and YOY sales change etc. The data is stored in "Composite" worksheet of URL Data.xlsx

Transaction Data

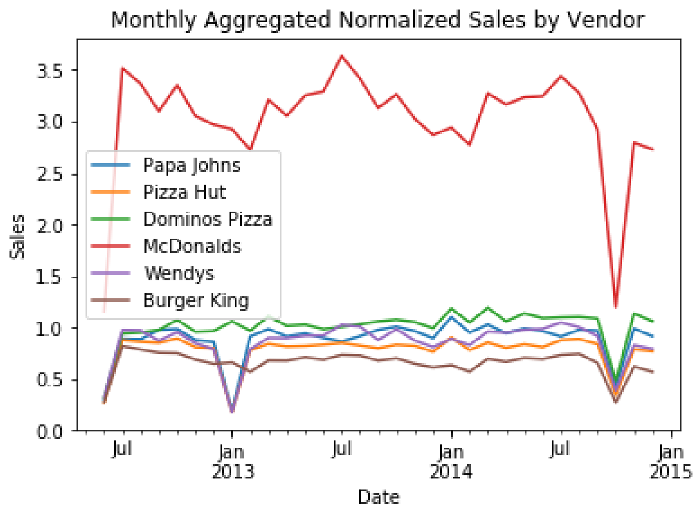
EDA and Convert Transaction Data into quaterly format

Rerring to the graph "Monthly Aggregated Normalized Sales by Vendor", we found:

1. June 2012 sales value is significantly lower, this is likely due to the fact the start date of data is mid month. We can drop this month's data and start with July 2012(3Q12)
2. October 2014 sales value look suspicioiusly low across the board. This is because we have missing data for October 2014.
3. Pizza hut/Papa Johns/Wendys had a drop in sales for Januaray 2013. We need to verfiy if this is related to data error or outliers.
4. Sales data tends to have negative spikes in Janurary and December, these are likely related to thanksgiving and new year

In [855]:
```
df_adj.plot()
plt.title('Monthly Aggregated Normalized Sales by Vendor')
plt.ylabel('Sales')
```

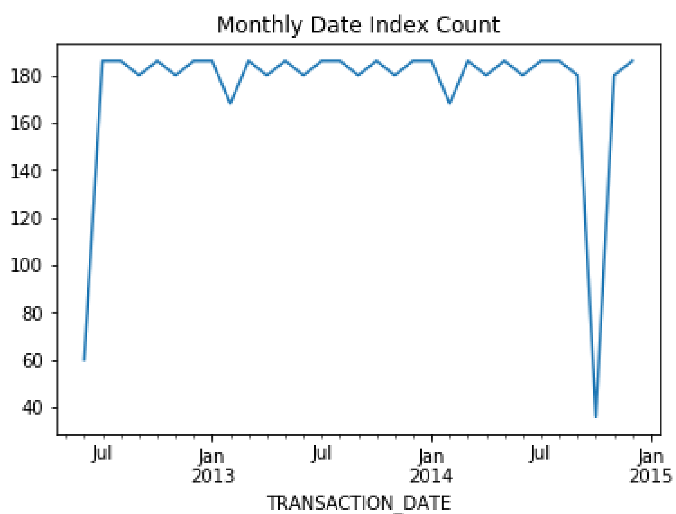
Out[855]: Text(0, 0.5, 'Sales')



Verfiy Missing data count for Octoer 2014

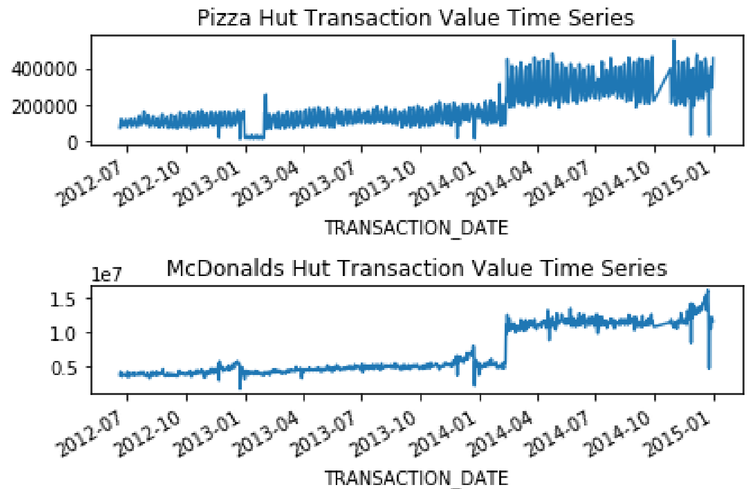
In [856]:
```
data.groupby(pd.Grouper(freq='M'))['TRANSACTED_VALUE'].count().plot()
plt.title('Monthly Date Index Count')
```

Out[856]: Text(0.5, 1.0, 'Monthly Date Index Count')



Verfiy data error for Pizza shops during Janurary 2013, the drop in value is not continous

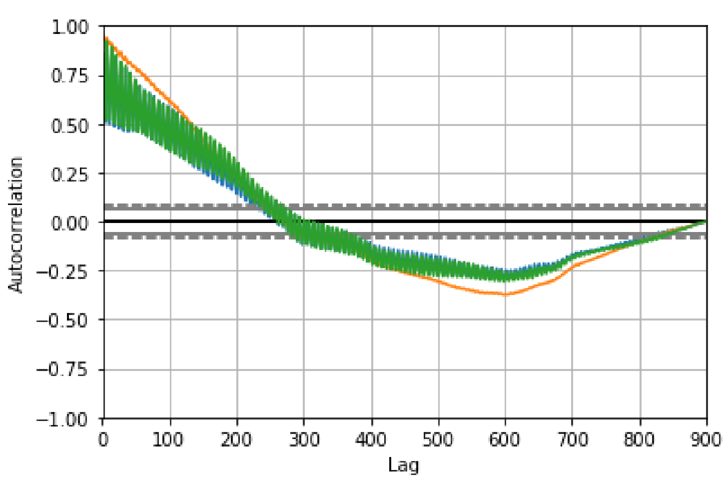
In [860]:
```
fig = plt.figure()
plt.subplot(2, 1, 1)
data[data.MERCHANT =='Pizza Hut']['TRANSACTED_VALUE'].plot()
plt.title('Pizza Hut Transaction Value Time Series')
plt.subplot(2, 1, 2)
transaction_panel.set_index('TRANSACTION_DATE')['NORMALIZATION_FACTOR'].plot()
plt.title('McDonalds Hut Transaction Value Time Series')
fig.tight_layout()
```



Use arima to deal with the missing/erroneous data,for simplicity only use first order diffrerencing and with 10-day lag

In [863]:
```
for merchant in ['Pizza Hut','Wendys','Papa Johns']:
    autocorrelation_plot(data[data.MERCHANT == merchant]['TRANSACTED_VALUE'])
```

```
/Users/charles-18/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: Fu
tureWarning: 'pandas.tools.plotting.autocorrelation_plot' is deprecated, import 'pa
ndas.plotting.autocorrelation_plot' instead.
```



First six time lags appear to be significant, to better fit the data (less concerned about overfitting here since we asssume seasonality and are only interseted in imputing with previous values), we use 10 day lag and 1st order differencing

In [872]:
```
train_index = (data.index < '2013-01-01')
test_index = (data.index >= '2013-01-01') & (data.index < '2013-02-01')
model = ARIMA(data[(train_index) & (data.MERCHANT =='Pizza Hut')]['TRANSACTED_VALUE']
model_fit = model.fit(disp=0)
model_fit.summary()
```

Out[872]:

ARIMA Model Results

Dep. Variable:	D.y	**No. Observations:**	193
Model:	ARIMA(10, 1, 0)	**Log Likelihood**	-2144.650
Method:	css-mle	**S.D. of innovations**	15864.017
Date:	Sun, 28 Jul 2019	**AIC**	4313.300
Time:	14:40:47	**BIC**	4352.452
Sample:	1	**HQIC**	4329.155

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	162.5615	196.972	0.825	0.410	-223.497	548.620
ar.L1.D.y	-0.8221	0.075	-10.958	0.000	-0.969	-0.675
ar.L2.D.y	-0.9433	0.099	-9.513	0.000	-1.138	-0.749
ar.L3.D.y	-0.8943	0.123	-7.293	0.000	-1.135	-0.654
ar.L4.D.y	-0.8880	0.140	-6.351	0.000	-1.162	-0.614
ar.L5.D.y	-0.8004	0.145	-5.509	0.000	-1.085	-0.516
ar.L6.D.y	-0.7548	0.146	-5.181	0.000	-1.040	-0.469
ar.L7.D.y	0.0308	0.141	0.218	0.828	-0.246	0.308
ar.L8.D.y	0.0134	0.124	0.108	0.914	-0.230	0.257
ar.L9.D.y	0.0999	0.100	0.994	0.321	-0.097	0.297
ar.L10.D.y	0.0644	0.078	0.829	0.408	-0.088	0.217

Roots

	Real	**Imaginary**	**Modulus**	**Frequency**
AR.1	0.6294	-0.7885j	1.0089	-0.1428
AR.2	0.6294	+0.7885j	1.0089	0.1428
AR.3	-0.9356	-0.4811j	1.0520	-0.4244
AR.4	-0.9356	+0.4811j	1.0520	0.4244
AR.5	-0.2318	-1.0472j	1.0726	-0.2847
AR.6	-0.2318	+1.0472j	1.0726	0.2847
AR.7	1.9122	-0.0000j	1.9122	-0.0000
AR.8	-0.0634	-1.6639j	1.6651	-0.2561
AR.9	-0.0634	+1.6639j	1.6651	0.2561
AR.10	-2.2608	-0.0000j	2.2608	-0.5000

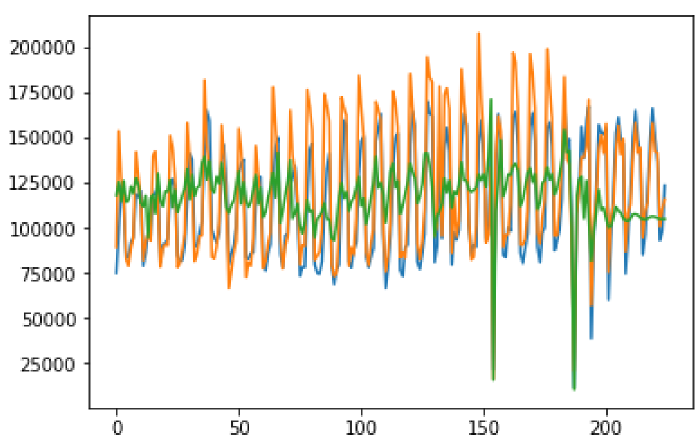
In [873]:
```python
train = {}
for merchant in ['Pizza Hut','Wendys','Papa Johns']:
    train[merchant] = data[(train_index) & (data.MERCHANT ==merchant)]['TRANSACTED_VA
    for idx in data[(data.MERCHANT ==merchant) & test_index].index:
        model = ARIMA(train[merchant], order=(10,1,0))
        model_fit = model.fit(disp=0)
        output = model_fit.forecast()
        train[merchant] = np.append(train[merchant],output[0])
        print(output[0])
```

```
[38985.45583749]
[94323.81733868]
[123174.71650481]
[157018.51740939]
[152816.7766431]
[151643.9293559]
[155053.22608898]
[60219.12207462]
[93927.20000351]
[121756.42514758]
[152670.25113898]
[160998.64588151]
[149612.99717617]
[146208.83953536]
[74720.80861081]
[94692.75754648]
[121861.87062073]
[150049.4033678]
[164837.3303121]
```

In [874]:
```python
for merchant in ['Pizza Hut','Wendys','Papa Johns']:
      test_len =  len(data[(data.MERCHANT ==merchant) & test_index].index)
      data.loc[(data.MERCHANT ==merchant) & test_index,'TRANSACTED_VALUE'] = train[me
```

In [875]:
```python
plt.plot(train['Pizza Hut'])
plt.plot(train['Papa Johns'])
plt.plot(train['Wendys'])
```

Out[875]: [<matplotlib.lines.Line2D at 0x1c4501e668>]



In [883]:
```
# train_2 = {}
# train_index = (data.index >= '2013-01-01') & (data.index < '2014-10-01')
# for merchant in merchant_list:
#     train_2[merchant] = data[(train_index) & (data.MERCHANT ==merchant)]['TRANSACTI
#     for idx in range(25):
#         model = ARIMA(train[merchant], order=(10,1,0))
#         model_fit = model.fit(disp=0)
#         output = model_fit.forecast()
#         train_2[merchant] = np.append(train_2[merchant],output[0])
#         print(output[0])
```

Recompute monthtly sales using imputed sereis

In [926]:
```
data_adj_daily = pd.merge(data,transaction_panel,left_on=data.index,right_on=transact
data_adj_daily['TRANSACTED_VALUE']  = data_adj_daily['TRANSACTED_VALUE'] / data_adj_d
f_adj = pd.DataFrame([])
for merchant in merchant_list:
    df_adj[merchant] = data_adj_daily[data_adj_daily['MERCHANT'] == merchant].groupby
```

We are also interested in imputing the missing 2014 October data for all the merchants. For simplicity, we proportionally scale up the sales figure we have using dates we have and put a discount factor of 0.5, taking seasonablity into accout.

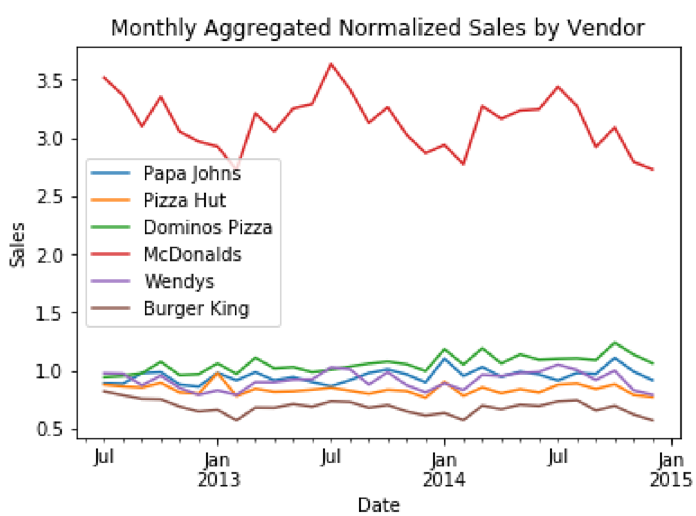
In [927]:
```
month_dates = data.groupby([pd.Grouper(freq='M'),'MERCHANT'])['TRANSACTED_VALUE'].cou
scale_factor = ((31/ month_dates[month_dates.index.get_level_values(0) == '2014-10-31
df_adj[df_adj.index=='2014-10-31'] = df_adj[df_adj.index=='2014-10-31']  * scale_fact
```

Remove 2012 July Data, observing strong seasonality

In [928]:
```
df_adj = df_adj[df_adj.index > '2012-06-30']
```

In [929]:
```
df_adj.plot()
plt.title('Monthly Aggregated Normalized Sales by Vendor')
plt.ylabel('Sales')
```

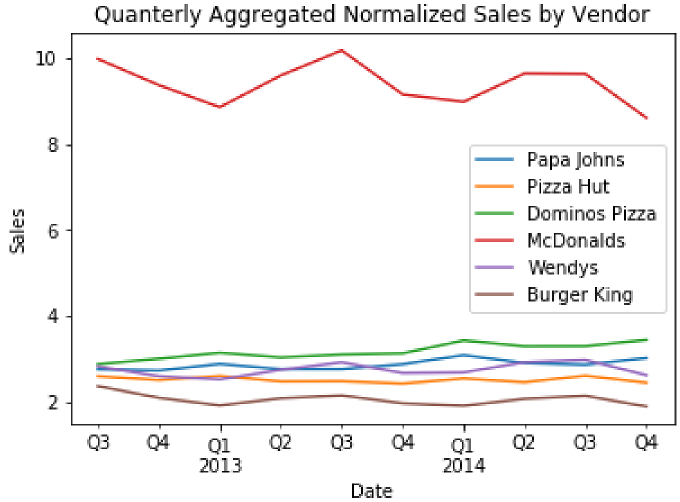
Out[929]:
```
Text(0, 0.5, 'Sales')
```



Convert to Quaterly

In [943]:
```
df_adj_quaterly = df_adj.groupby(pd.PeriodIndex(df_adj.index, freq='Q')).sum()
df_adj_quaterly.plot()
plt.title('Quanterly Aggregated Normalized Sales by Vendor')
plt.ylabel('Sales')
```

Out[943]:
```
Text(0, 0.5, 'Sales')
```



Web Traffic and Survey Data

1. Web Traffic and Survey Data only continas data from 2013Q1 to 2014Q4.
2. Survey Data is agnostic of merchants/brands
3. Web Traffic Data contains 7 distinct vendors, 4 of them have sales data from credit transaction dataset.

In [1080]:
```
xls = pd.ExcelFile('URL Data.xlsx')
composite = pd.read_excel(xls, 'Composite',index_col=[0,1]).T
composite.index = pd.PeriodIndex(composite.index, freq='Q')
```

In [1081]:
```
set(composite.columns.get_level_values(0))
```

Out[1081]:
```
{'Dominos Pizza',
 'McDonalds',
 'Papa Johns',
 'Popeyes',
 'Sonic',
 'Survey',
 'Taco Bell',
 'Wendys'}
```

Combine with Sales data

In [1082]:
```
data_dict['Dominos Pizza']
```

Out[1082]:

	Visits	Dollars Spent	Promotions	Use_App	Total Unique Viewers	Reach	Total Visits	Average Visits per Unique Visitor/Viewer	Total Minutes	Average Minutes per Visit	
2013Q1	7.0	25.0	3.0	0.12	8211.0	0.008211	73899.0	9.0	413834.4	5.6	3.1
2013Q2	5.0	25.0	4.0	0.14	7348.0	0.007348	58784.0	8.0	329190.4	5.6	3.(
2013Q3	3.0	25.0	2.0	0.23	8430.0	0.008430	59010.0	7.0	330456.0	5.6	3.(
2013Q4	4.0	15.0	6.0	0.32	8512.0	0.008512	68096.0	8.0	381337.6	5.6	3.1
2014Q1	7.0	15.0	4.0	0.54	8874.0	0.008874	79866.0	9.0	399330.0	5.0	3.4
2014Q2	8.0	25.0	7.0	0.66	9354.0	0.009354	84186.0	9.0	420930.0	5.0	3.2
2014Q3	2.0	20.0	2.0	0.62	9355.0	0.009355	93550.0	10.0	374200.0	4.0	3.2
2014Q4	3.0	30.0	3.0	0.71	9745.0	0.009745	97450.0	10.0	389800.0	4.0	3.4

In [1083]:
```
data_dict = {}
for col in composite.columns:
    merchant = col[0]
    if merchant in df_adj_quaterly.columns:
        data_dict[merchant] = pd.concat([composite['Survey'],composite[merchant],pd.I
```

In [1084]:
```
data_dict['Dominos Pizza']
```

Out[1084]:

	Visits	Dollars Spent	Promotions	Use_App	Total Unique Viewers	Reach	Total Visits	Average Visits per Unique Visitor/Viewer	Total Minutes	Average Minutes per Visit	
2013Q1	7.0	25.0	3.0	0.12	8211.0	0.008211	73899.0	9.0	413834.4	5.6	3.1
2013Q2	5.0	25.0	4.0	0.14	7348.0	0.007348	58784.0	8.0	329190.4	5.6	3.(
2013Q3	3.0	25.0	2.0	0.23	8430.0	0.008430	59010.0	7.0	330456.0	5.6	3.(
2013Q4	4.0	15.0	6.0	0.32	8512.0	0.008512	68096.0	8.0	381337.6	5.6	3.1
2014Q1	7.0	15.0	4.0	0.54	8874.0	0.008874	79866.0	9.0	399330.0	5.0	3.4
2014Q2	8.0	25.0	7.0	0.66	9354.0	0.009354	84186.0	9.0	420930.0	5.0	3.2
2014Q3	2.0	20.0	2.0	0.62	9355.0	0.009355	93550.0	10.0	374200.0	4.0	3.2
2014Q4	3.0	30.0	3.0	0.71	9745.0	0.009745	97450.0	10.0	389800.0	4.0	3.4

We would like to construct a panel with no differentiation between merchants and see if bivariate relationship can be identified

In [1085]:
```
panel = pd.DataFrame([])
for item in data_dict:
    data_dict[item]['Company'] = item
    panel = pd.concat([panel,data_dict[item]],axis=0)
panel.columns = [x.strip() for x in panel.columns]
```

From the panel description, total visits of 0 look to be an outlier, however, this could also be caused by website being down.Need to verfiy if this is legitimate or a data error

In [1086]:
```
panel.describe()
```

Out[1086]:

	Visits	Dollars Spent	Promotions	Use_App	Total Unique Viewers	Reach	Total Visits	Average Visits per Unique Visitor/Viewer	Total I
count	32.00000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32
mean	4.87500	22.500000	3.875000	0.417500	5845.870625	0.005846	38504.797500	5.500000	191087
std	2.12132	5.080005	1.718026	0.230161	2315.209033	0.002315	33801.115018	3.388786	173255
min	2.00000	15.000000	2.000000	0.120000	2542.000000	0.002542	0.000000	1.000000	7966
25%	3.00000	18.750000	2.750000	0.207500	3965.000000	0.003965	7930.000000	2.000000	21981
50%	4.50000	25.000000	3.500000	0.430000	5343.500000	0.005344	29393.000000	5.500000	158521
75%	7.00000	25.000000	4.500000	0.630000	7698.875000	0.007699	67119.665000	9.000000	354802
max	8.00000	30.000000	7.000000	0.710000	9745.000000	0.009745	97450.000000	11.000000	420930

This is indeed a data error other columns returns non-zero data, we can approximate total visits using Total minutes / average minutes per visit

In [1087]:
```
panel_error = panel[panel['Total Visits'] == 0]
panel_error
```

Out[1087]:

	Visits	Dollars Spent	Promotions	Use_App	Total Unique Viewers	Reach	Total Visits	Average Visits per Unique Visitor/Viewer	Total Minutes	Average Minutes per Visit	
2013Q2	5.0	25.0	4.0	0.14	4122.67	0.004123	0.0	3.0	25972.821	2.1	9.5
2013Q3	3.0	25.0	2.0	0.23	4043.56	0.004044	0.0	3.0	24261.360	2.0	10.1

In [1088]:
```
panel.loc[panel['Total Visits'] == 0,'Total Visits']  = panel_error['Total Minutes']
```

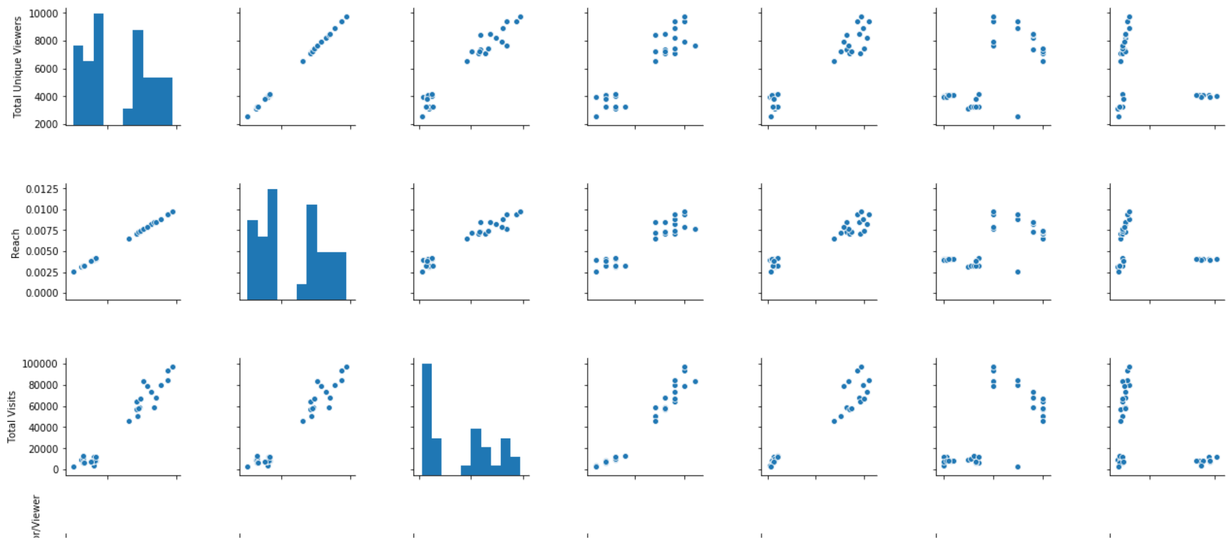
Visually compute correlations and pairplot before further investigation, drop Survey data for now

In [1089]:
```
panel_no_survey = panel.drop(columns = composite['Survey'].columns)
```

In [1090]:
```
corrmat = panel_no_survey.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True,annot=True);
plt.show()
```

In [1091]:
```
sns.pairplot(panel_no_survey)
```

Out[1091]: <seaborn.axisgrid.PairGrid at 0x1c4f0b7a58>



Observation and Hyptohesis

Reach, Total Minutes, Total Visits, Total Unique Viewers

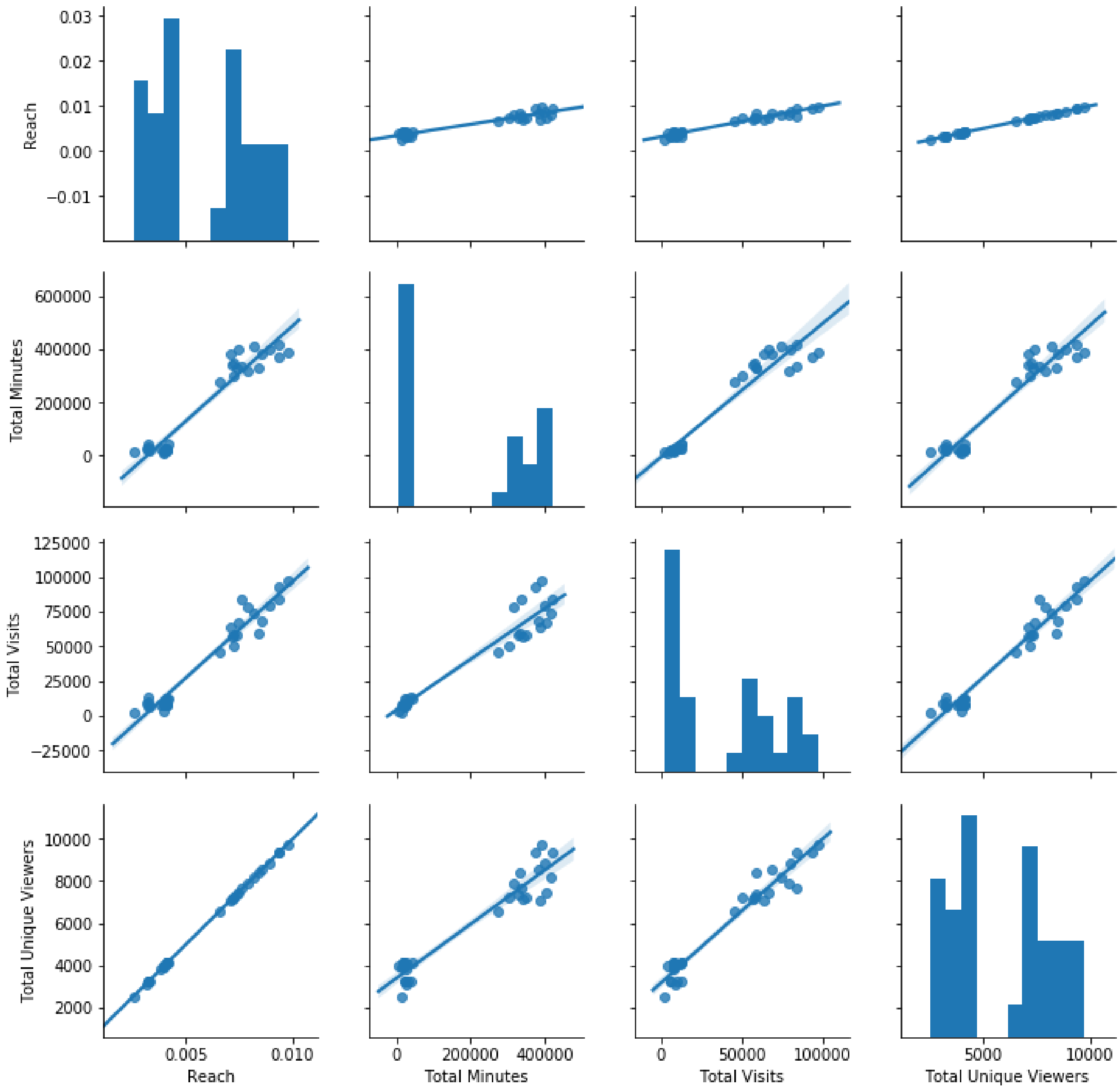
All four of these are popularity measures

1. There are strong positive correlations among reach, total unique viewers,total visits ,total minutes. Reach appears to be a redundant measure of total unique viewers with a correlation of 1. As number of total unique viewers go up, the number of total visits go up given each unique viewer visits the site at least once and total number of viewing minutes go up. We might only find value in one of the four variables to avoid redundant information.
2. It is worth noticing that data are sperated to two groups. One group contain merchants with low user exposure.

In [1112]:
```
sns.pairplot(panel_no_survey[['Reach','Total Minutes','Total Visits','Total Unique Vi
```

```
/Users/charles-18/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecat
ed; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpr
eted as an array index, `arr[np.array(seq)]`, which will result either in an error
or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

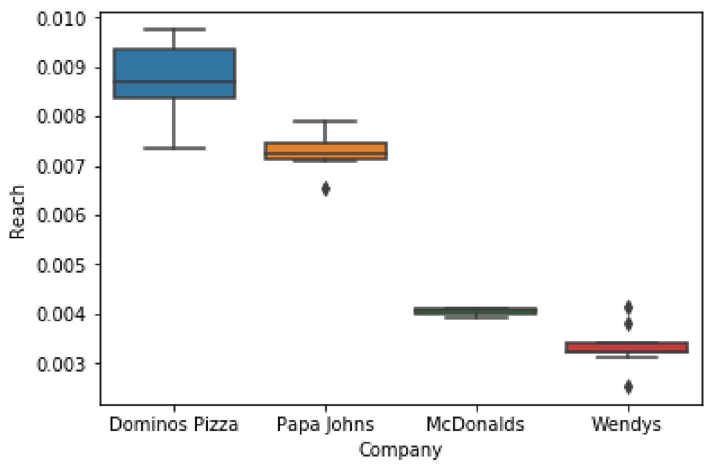
Out[1112]: <seaborn.axisgrid.PairGrid at 0x1c59fccd68>



We have observed that number of the reach of macdonalds and wendys are signficantly lower than that of Dominos and Papa Johns. The hypothesis is that Mcdonalds and Wendys either have poor website desgin or the primary reason of people's visits are due to the need to order food whereas Mcdonalds and Wendy either did not have food delivery option on their website or they were hard to use

In [1110]:
```
sns.boxplot(x='Company',y='Reach',data=panel_no_survey)
```

Out[1110]: <matplotlib.axes._subplots.AxesSubplot at 0x1c59eb03c8>



Avergae Minutes Per Visit, Average visit per user

The two variables are simple computed using other variables.

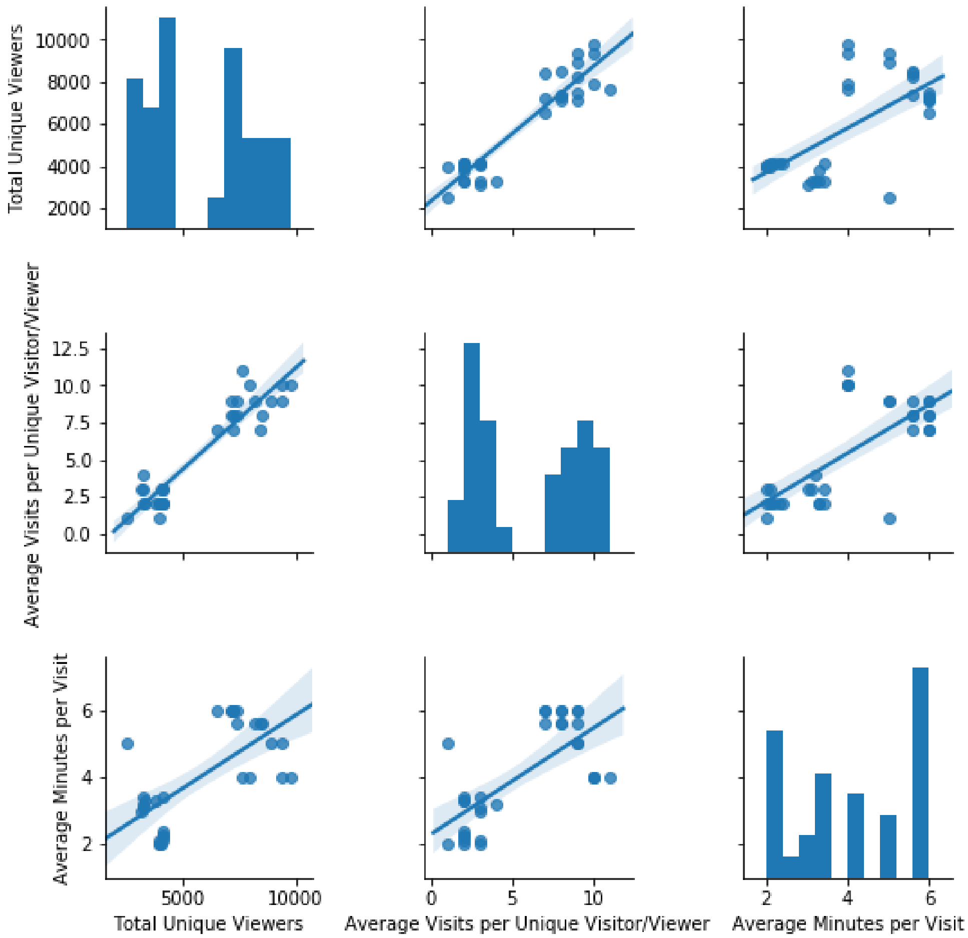
Average Minutes per visit measures user's dwell time on the page, this could be either good or bad. A longer dwell time could imply that the page is hard to use and an overly short dwell time might imply a high bounce rate, meaning the user arrived at the wrong site or hated the site because of its poor design

Average Visit per user implies the loyalty of a user, a higher rate implies better user retention

1. Average minutes per vist = Total Minutes / Total visits
2. Average Visit per unique user = Total Visits / Total Unique Viewers

In [1118]:
```
sns.pairplot(panel_no_survey.drop(columns = ['Reach','Total Minutes','Total Visits',
```

Out[1118]: <seaborn.axisgrid.PairGrid at 0x1c5b0353c8>



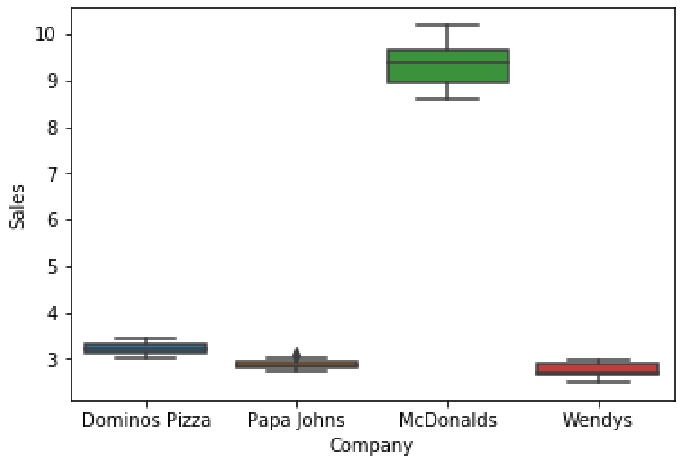
Sales

From the correlation graph ealier, we observed negative correaltion between sales number and other variables. This is misleading because we can visually observe 3 distinct groups in the scatter plot.

Maconalds dominates the market with at least 3 times the sales of other companies, we are interseted in learning how much of this is driven by online sales

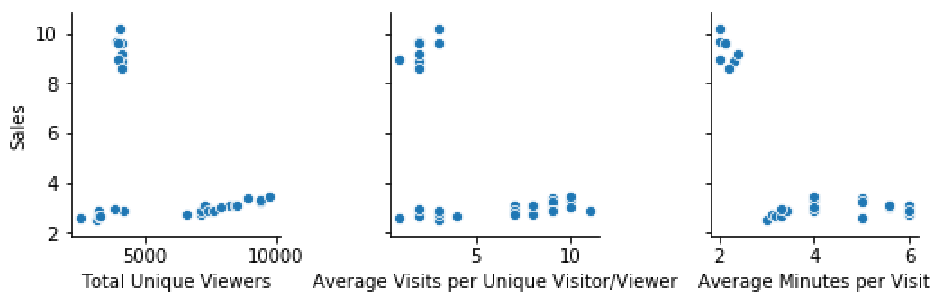
In [1147]:
```
sns.boxplot(x='Company',y='Sales',data=panel_no_survey)
```

Out[1147]: <matplotlib.axes._subplots.AxesSubplot at 0x1c63813080>



Data seems to be seprated to different groups. The top group corresponds to mcdonalds based on high sale number

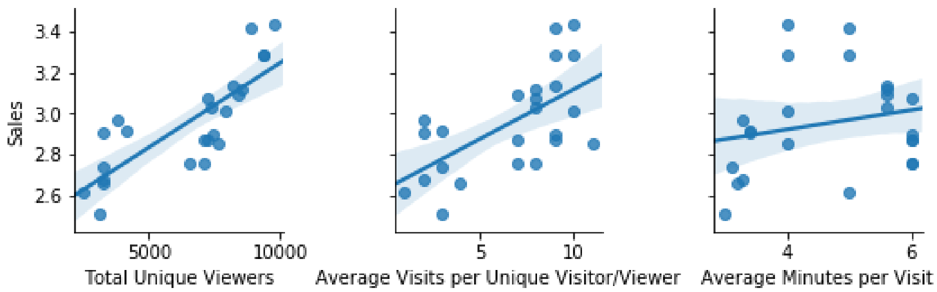
In [1148]:
```
pp = sns.pairplot(data= panel_no_survey,
                  y_vars=['Sales'],
                  x_vars=panel_no_survey.drop(columns=['Company','Sales','Reach','Tot
```



Excluding Mcdonalds, we find positive relationship between total unique veiwers/ average visits per unique viewers and sales. We dont find such relationship between average minutes per visit and sales. Our hypthesis is that, given a constant probability for a unique user that visits the website to make a purchase during a single visit, more people visits the websits indicates more people making a purchase. The more a unique user visits the website, the more purchanse she/he is going to make.

In [1156]:
```
pp = sns.pairplot(data= panel_no_survey[panel_no_survey['Company']!='McDonalds'],
                  y_vars=['Sales'],
                  x_vars=panel_no_survey.drop(columns=['Company','Sales','Reach','Tot
```

```
/Users/charles-18/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecat
ed; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpr
eted as an array index, `arr[np.array(seq)]`, which will result either in an error
or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

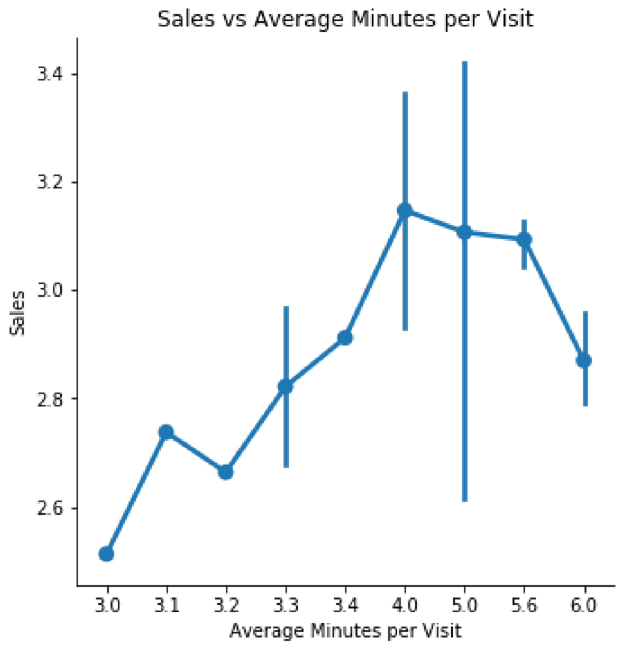


In addition to this, we are interestd to see if there exists a non-linear relationship between Sales and Averge Minutes Per Visit. Based on the factor plot, we develop the following hypothesis: There is a "sweet spot" for time spent on the the website for a consumer to make a purchase. If a consuemr spends less time or more time, it relates to a lower probability for them to make an order. Assuming there is real causation behind this reasoning, it is reasonable for companies to desgin user friendly delivery pages that is not overly complicated to use.

In [1168]:
```
def bivariate(df,target):
    target_type = df[target].dtype
    for col in df.columns:
        if col != target:
            col_type = df[col].dtype
            sns.factorplot(y=target,x=col,data=df)
            plt.title(target+' vs '+col)
            plt.show()
```

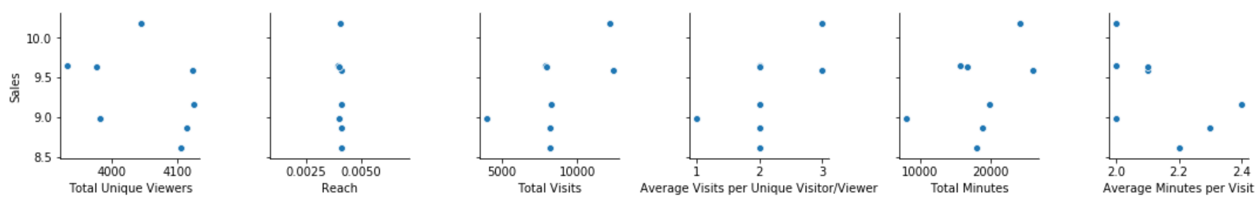
In [1173]:
```
bivariate(panel_no_survey[panel_no_survey.Company != 'McDonalds'][['Average Minutes p
```

```
/Users/charles-18/anaconda3/lib/python3.7/site-packages/seaborn/categorical.py:366
6: UserWarning: The `factorplot` function has been renamed to `catplot`. The origin
al name will be removed in a future release. Please update your code. Note that the
default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```



For Mcdonalds, we find weak evidence suggesting linear relationship between sales and other variables. One hypthesis is that: Mcdonalds generates much higher sales number than other merchants. The majority of which is directly from its large network of retail stores. Comparing to this, its website revenue are relatively small. This is also a potential indicator that McDonalds' online sales channel is underdeveloped

In [1174]:
```
pp = sns.pairplot(data= panel_no_survey[panel_no_survey['Company']=='McDonalds'],
                  y_vars=['Sales'],
                  x_vars=panel_no_survey.drop(columns=['Company','Sales']).columns)
```



Survey Data

In [1252]:
```
panel_quaterly = panel.groupby(panel.index).mean()
```

In [1253]:
```
panel_quanterly.head()
```

Out[1253]:

	Visits	Dollars Spent	Promotions	Use_App	Total Unique Viewers	Reach	Total Visits	Average Visits per Unique Visitor/Viewer	Total Minutes	Average Minutes per Visit
0	0.833333	0.666667	0.2	0.000000	0.390088	0.390088	0.230514	0.2	0.472086	0.921053
1	0.500000	0.666667	0.4	0.033898	0.000000	0.000000	0.000000	0.2	0.000000	0.894737
2	0.166667	0.666667	0.0	0.186441	0.441705	0.441705	0.119832	0.0	0.267268	0.947368
3	0.333333	0.000000	0.8	0.338983	0.483150	0.483150	0.401573	0.6	0.758656	1.000000
4	0.833333	0.000000	0.4	0.711864	0.595934	0.595934	0.328384	0.0	0.533769	0.763158

From the below graph, we have the following observations and hypothesis

Store Visits / Promotion

Store visits are positively correlated with promotion, indicating the effectiveness of promotion

Store visits are negatively correlated with both total visits and average visits per unique user, indicating the supplemental relationship between actual store visist and online purchase. However, it is positiviely correlated with total minutes and average minutes per visit. Our hypothesis is that, they share a confunding variable - Promotions. Promotion is both positiely correlated with total store visits and total minutes / average minutes per online visit. The more promotions user receives, the more likely they are paying store visits and spend more time placing online order to input promotion code.

Store visists are also negatively related to sales, potentially because promotions serve as a confunding variable here as well. The more promotion users receive, the more likely they are paying store visit, but sales are discounted due to promotion.

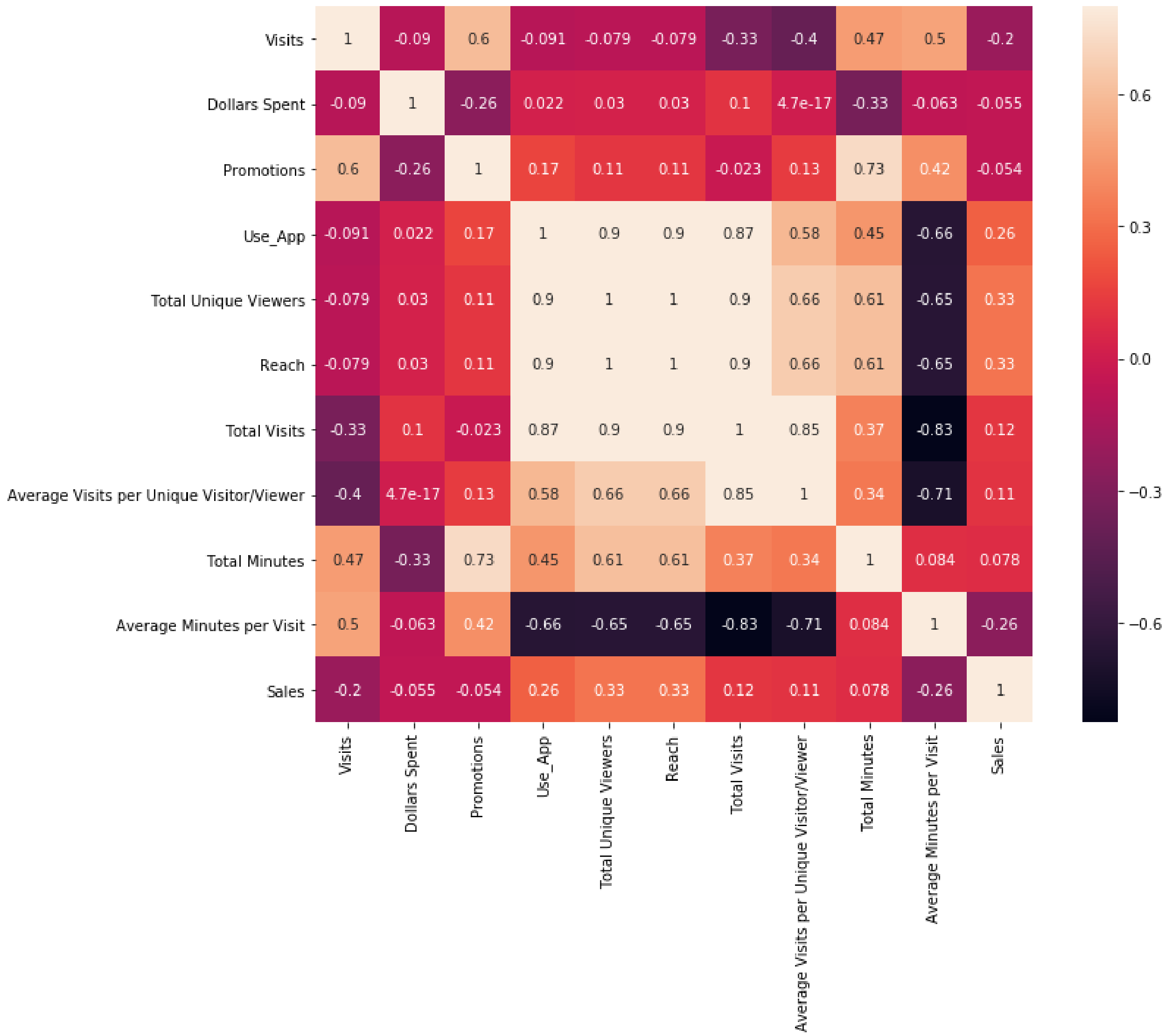
Dollar Spent

Negatively correlated with promotion due to discount recieved

Use_App

Strong positive correlation with website traffic measures as reach/unique viewer/total minutes etc. Interesting it is negatively correlated to average minutues per vist. This is likely due to the fact that phone apps provide an easier and faster way for food purchases. It also has a positive correlation with Sales, the hypothesis is that phone app adoption increases reach of the company and boosted sales directly

In [1249]:
```
corrmat = panel_quanterly.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True,annot=True);
plt.show()
```



Trend Analysis

In [1256]:
```
panel_quaterly = panel.groupby(panel.index).mean()
```

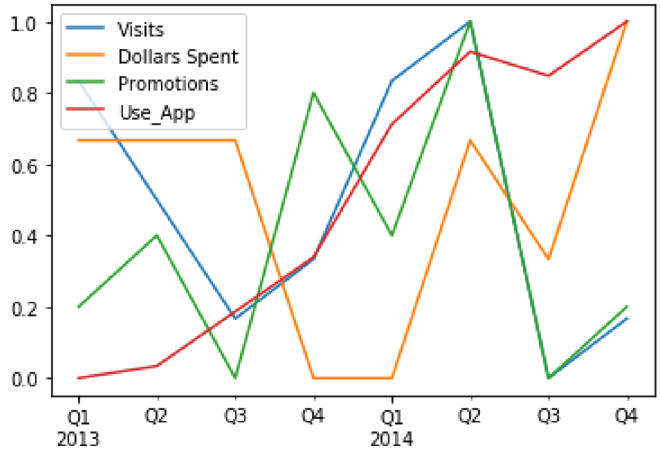
Normalize data

In [1266]:
```
x = panel_quaterly.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
panel_quaterly = pd.DataFrame(x_scaled,index = panel_quaterly.index,columns = panel_q
```

We identify clear trend of increasing adoption of mobile apps and seasonality of store visits

In [1267]:
```
panel_quaterly[['Visits','Dollars Spent','Promotions','Use_App']].plot()
```

Out[1267]: <matplotlib.axes._subplots.AxesSubplot at 0x1c60e562e8>



Investigate Contemporaneous Relationship between Sales and survey Data

In [1317]:
```
cont_table = []
for col in ['Visits','Dollars Spent','Promotions','Use_App']:
    for company in set(panel_no_survey.Company):
        cont_table.append((company,col,panel_quaterly[col].corr(panel_no_survey[panel
cont_df = pd.DataFrame(cont_table)
cont_df.columns= ['Company','Metrics','Correlation']
```

We find strong positive contemporaneous correlation between Papa Johns and Dominos Pizza and adoption of app uses in the industry, combing with the fact they are the only two companies with sales growth for 2018Q4, and high reach/total visits/total online visit time etc of its web traffic. We have a hypothesis that online sale has become a more important driver of revenue growth and Papa Johns and Dominos are leaders in the industry making such adoption to fule its sale growth.

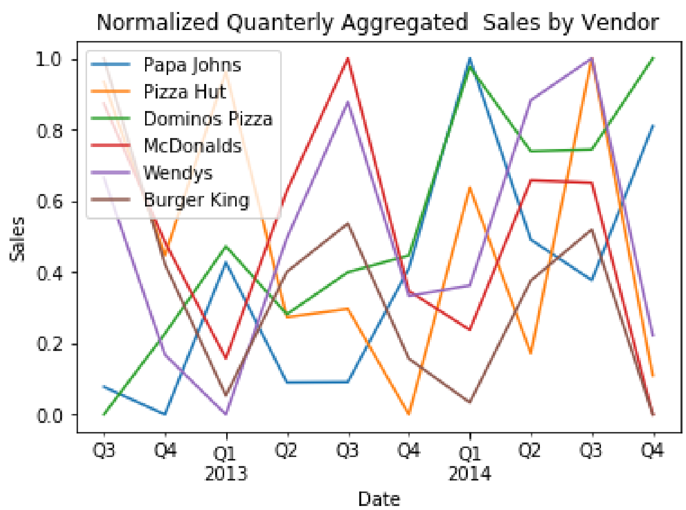
In [1330]:
```
cont_df.sort_values('Correlation')[-3:]
```

Out[1330]:

	Company	Metrics	Correlation
12	Wendys	Use_App	0.313628
15	Papa Johns	Use_App	0.645490
13	Dominos Pizza	Use_App	0.880831

In [1337]:
```
x = df_adj_quaterly.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
df_adj_quaterly = pd.DataFrame(x_scaled,index = df_adj_quaterly.index,columns = df_ad
df_adj_quaterly.plot()
plt.title('Normalized Quanterly Aggregated  Sales by Vendor')
plt.ylabel('Sales')
```

Out[1337]: Text(0, 0.5, 'Sales')



Additonal Survey Questions

Q1. What is your favourite fast food restaurant?

With this data we can investigate the relationship between the reputation of a restaurant and its sales growth. Also investigate market shares in relative to its reputation comparing with peers

Q2. How many times did you order fast food through an app over the past quater?

With this data we can model the portion of app sales over total sales and verfy if the ratio is increasing.

Additional Metrics

It will be helpful if can know the type of device which the visitor traffic is from. We can investigate the adoption of mobile apps for different merchants and whether higher mobile app adoption rate results in more online visits, shorter visiting time and higher sales due to its convenience.

Pros / Cons data sets

Transaction Data

Pros:

1. Uncover sales insight before quaterly release.

Cons:

1. Adjusting for changing panel size might not include all market participants

2. Does not cover other transaction types such as cash and transactions going through thrid parties such as Paypal etc

Survey Data

Pros

1. Quick to design, implement and inexpensive to collect
2. Very flexible in terms of information colleted

Cons

1. Samping bias, a more tech-savy group will be more likely to respond with a higher rate of app adoption
2. Response bias
3. Limited volume of data

Web Traffic Data

Pros

1. Direct measure of user activity/loyalty, brand popularity

Cons

1. There are potential noise especially when the websites contain other information other than delivery pages (like company intro/ investor relationship / custoer service etc
2. Measures are exposed to systematic risks such as infrastruture breaks

Additional Data Sources

Geolocation Data

It will be interesting to collect check-in data from yelp/foursquare for fast food restaruant as an alternative measure of sales. The data will capture information regarding traditional in-store dining activities, including both credit and non-credit transactions.

Food Delivery Data

This can serve as a direct measure of online sales for different merchants, getting rid of noise of other web traffic.

Social Media Follow/Review

Another potential contributor to future sales growth inclues social media food review. We can performan sentiment anlysis on user comments and understand market perception of a certain brand. A positive change in reviews indicates higher liklihood to increased future sales.