Participants: Mariuxi Leon, Cameron Milligan, Jorge Gonzalez

Data Analysis

The data consist of the number of people present at different times in gym and also includes additional factor. Using this dataset interesting insights into attendace of the gym shall be derived. Also a predictive model shall ve developed which will, predict the number of people attending the gym given the values of other factors.

Basic to do list

Specify the type of data analytic question (e.g. exploration, association causality) before touching the data

1. How attendace is affected by time of the week

2. How attendace is affected by time of the temperature

3. How attendace is affected by the start of the semester

4. Create a model predicting the number of people attending the gym

Define the metric for success before beginning

1. Logical reasoning shall match

2. Logical reasoning shall match

3. Logical reasoning shall match

4. 90% accuracy

Business Application

This data from university gym. This data consists of information about attendace of the gym and other factors taken at the same timestamp. Using this data a pedictive model can be developed which will predict the attendace of the gym which might be useful for the gym owner.

In [1]:
```
import datetime
import numpy as np
import pandas as pd
import seaborn as sb

# We will use matplotlib to plot figures
import matplotlib.pyplot as plt
%matplotlib inline

# For regression analysis we will use the statsmodels package
import statsmodels.api as sm
import statsmodels.formula.api as smf

# For visual inspection of the regression models
from statsmodels.graphics.regressionplots import plot_regress_exog, plot
_fit, plot_leverage_resid2, influence_plot

# This function will help us to create ordinal variables
from pandas.api.types import CategoricalDtype
```

```
/Users/cammilligan/anaconda/lib/python3.6/site-packages/statsmodels/com
pat/pandas.py:56: FutureWarning: The pandas.core.datetools module is de
precated and will be removed in a future version. Please use the panda
s.tseries module instead.
  from pandas.core import datetools
```

In [2]:
```
df = pd.read_csv('https://gist.githubusercontent.com/cameronmilligan/d4b
77b01e5fd03fd1d1b26f10f9d0c9c/raw/c807fd88a46698170151e283c749de44fde5e7
e5/data.csv',)
df = df[["number_people", "timestamp", "day_of_week", "is_weekend", "is_
holiday", "temperature", "is_start_of_semester", "is_during_semester",
"date", "month", "hour"]]
#converting to celcius
df.temperature = (df.temperature-32)/1.8
df_copy = df.copy()
```

In [3]:
```
df.temperature.describe()
```

Out[3]:
```
count    62184.000000
mean        14.753949
std          3.509109
min          3.411111
25%         12.777778
50%         14.633333
75%         16.822222
max         30.650000
Name: temperature, dtype: float64
```

In [4]:
```
df.head(1)
```

Out[4]:

	number_people	timestamp	day_of_week	is_weekend	is_holiday	temperature	is_sta
0	37	61211	4	0	0	22.088889	0

In [5]:
```
df.shape
```

Out[5]: (62184, 11)

In [6]:
```
df.describe()
```

Out[6]:

	number_people	timestamp	day_of_week	is_weekend	is_holiday	tempe
count	62184.000000	62184.000000	62184.000000	62184.000000	62184.000000	62184.(
mean	29.072543	45799.437958	2.982504	0.282870	0.002573	14.753
std	22.689026	24211.275891	1.996825	0.450398	0.050660	3.5091
min	0.000000	0.000000	0.000000	0.000000	0.000000	3.4111
25%	9.000000	26624.000000	1.000000	0.000000	0.000000	12.777
50%	28.000000	46522.500000	3.000000	0.000000	0.000000	14.633
75%	43.000000	66612.000000	5.000000	1.000000	0.000000	16.822
max	145.000000	86399.000000	6.000000	1.000000	1.000000	30.650

The dataset seems to be clean because:

Number of rows for all the data columns are same

Maximum number of people = 145 and min = 0 , looks reasonable

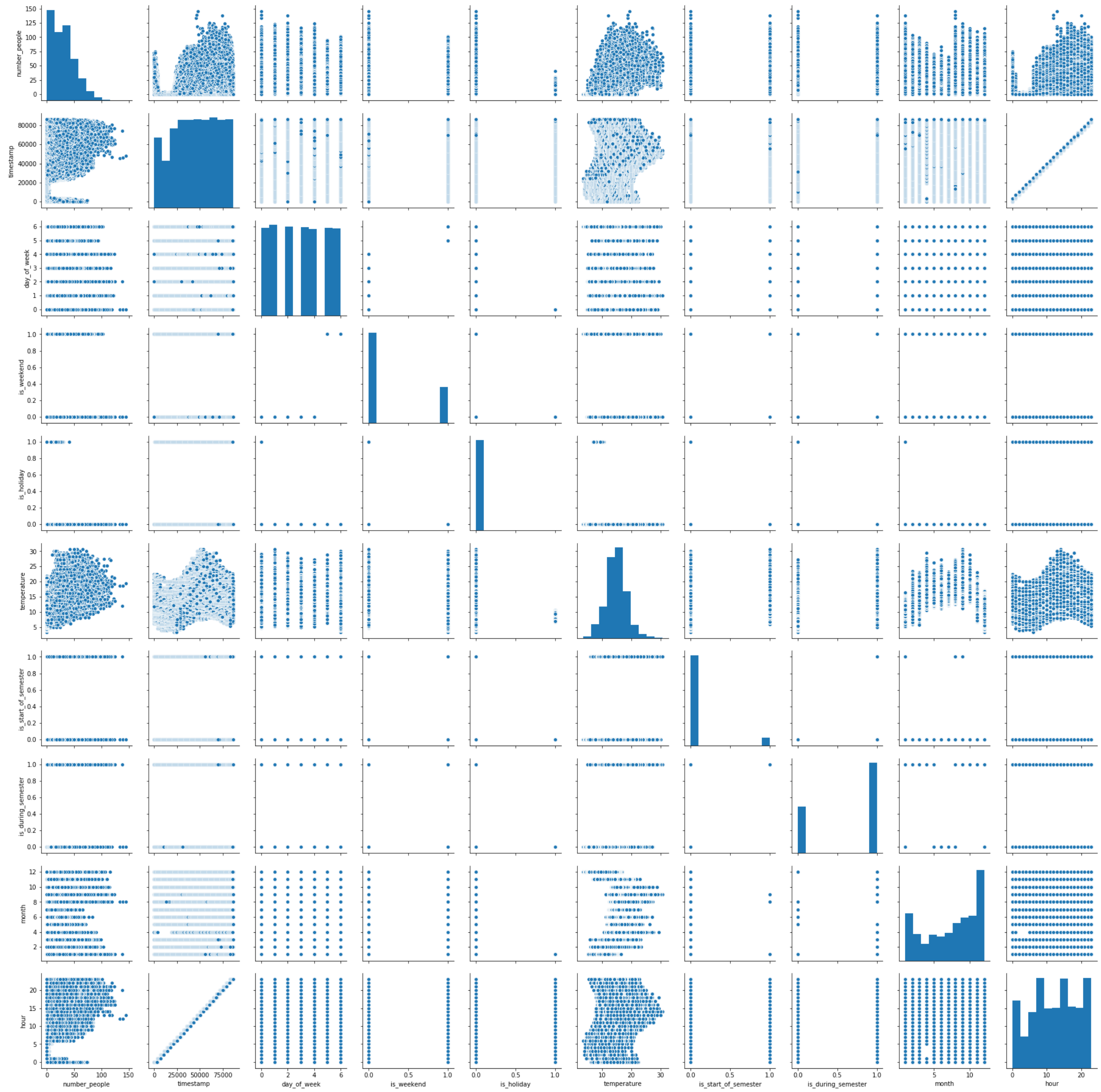
Maximum timestampvalue = 86399 which is close to 24 hrs which is reasonable

Max day of the week = 6 and min = 0 , seems reasonable

Below we pair plot, to get pictorial overview of the complete data. Pairplot shows histograms of the columns in diagonal of the matrix and pairwise scatter plot of the data.

In [7]:
```
sb.pairplot(df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x119874d30>

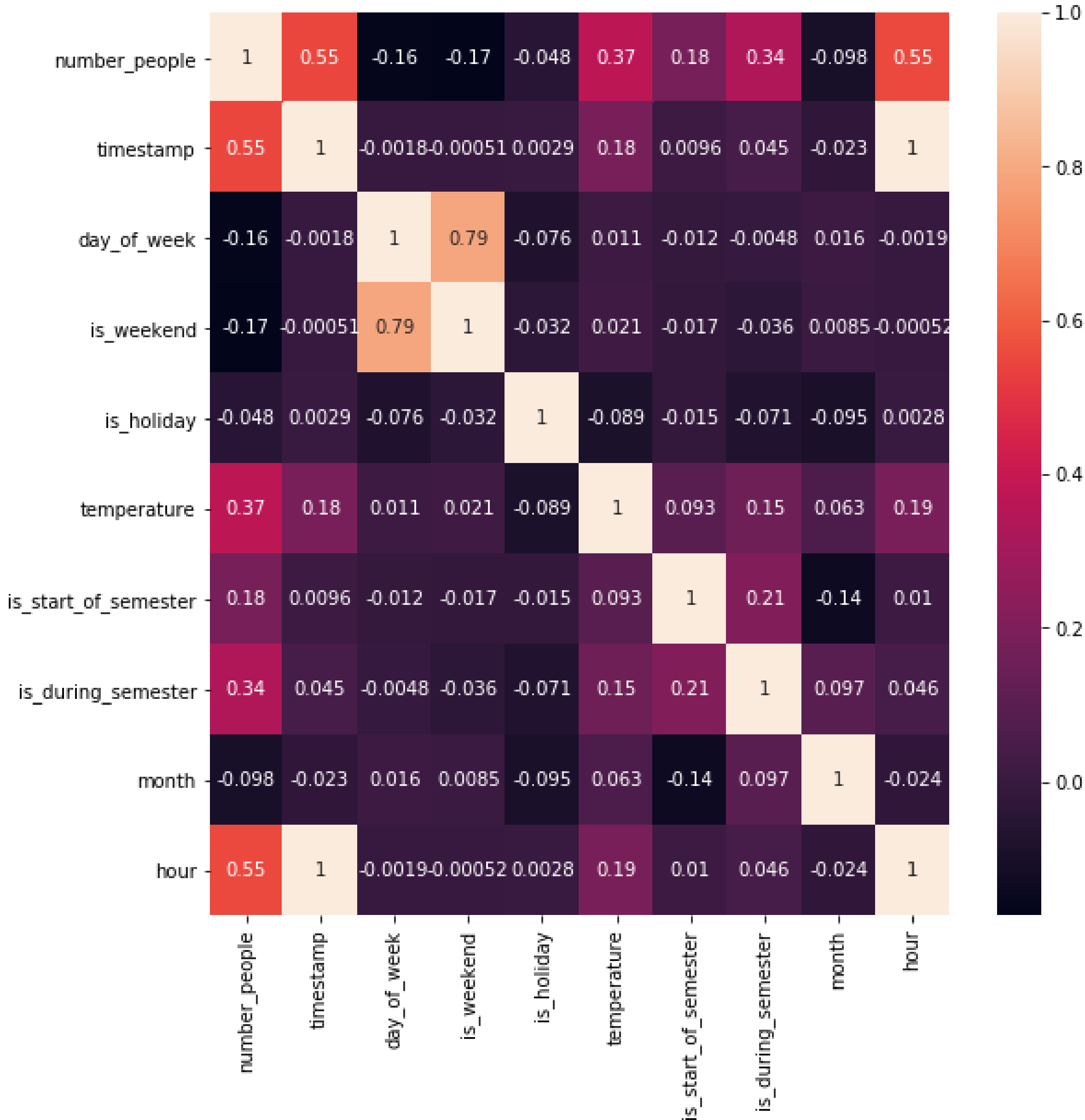


From above pairplot few things can be implied

1. Number of people attending gym and temperatures are showing gaussian distribution which is good thing

2. There is some kind of relationship exiting between temperatures and number of people attending gym

In [8]:
```
corrmat = df.corr()
f, ax = plt.subplots(figsize=(9, 9))
# Draw the heatmap using seaborn
sb.heatmap(corrmat, square=False, annot=True)
plt.show()
```



Some of the correlations above can be ignored such as day_of_week and is_weekend being correlated.

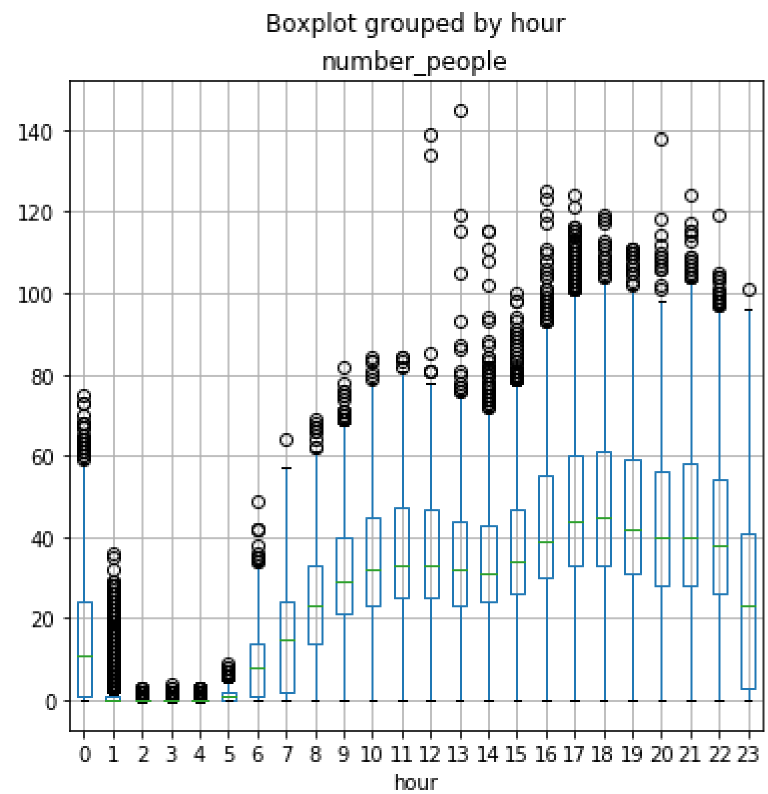
Some of the correlations that stand out are the hour, day_of_week/is_weekend, temperature and whether or not it is the start of the semester

Analysis: Hour vs Number of Gym Attendees

Hypothesis: A greater number of people are attending the gym later in the day.

In [9]:
```
df.boxplot(column="number_people", by= "hour", figsize= (6,6))
```

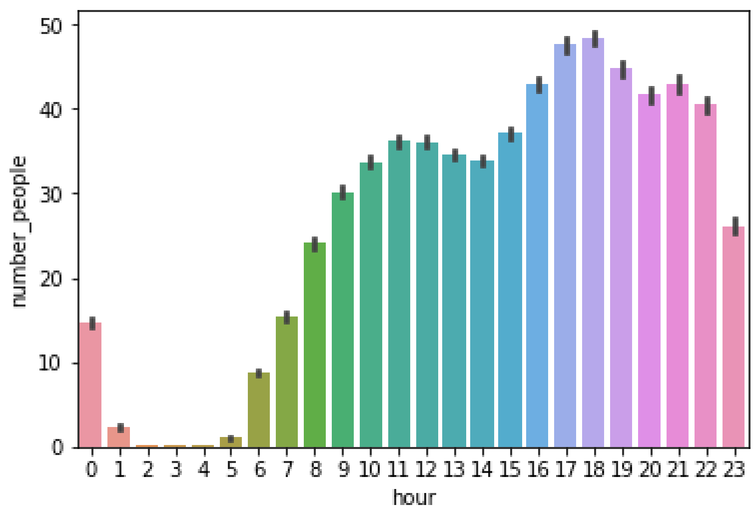
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x127c92cc0>



In [10]:
```
sb.barplot(x='hour',y='number_people',data=df)
```

```
/Users/cammilligan/anaconda/lib/python3.6/site-packages/scipy/stats/sta
ts.py:1626: FutureWarning: Using a non-tuple sequence for multidimensio
nal indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]
`. In the future this will be interpreted as an array index, `arr[np.ar
ray(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1294ce518>
```



Statement: "higher number of people are attending the gym at later hours" appears true from above graph

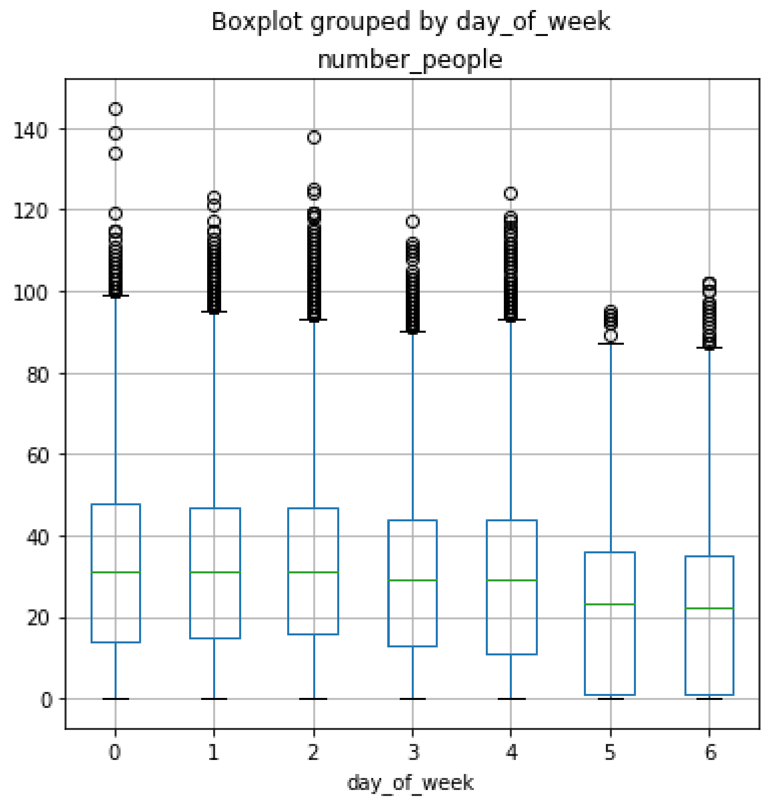
Analysis: Day of the Week vs Number of Gym Attendees

From pairplot there is negative correlation between day_of_week and number of people going to gym.

Hypothesis: A greater number of people attending the gym on day = 0 i.e monday compared day = 6 i.e sunday.

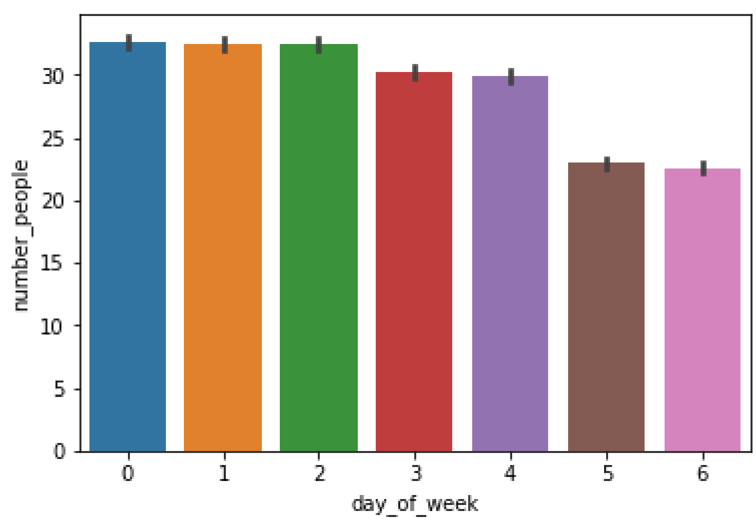
In [11]:
```
df.boxplot(column="number_people", by= "day_of_week", figsize= (6,6))
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x129704d30>



In [12]:
```
sb.barplot(x='day_of_week',y='number_people',data=df)
```

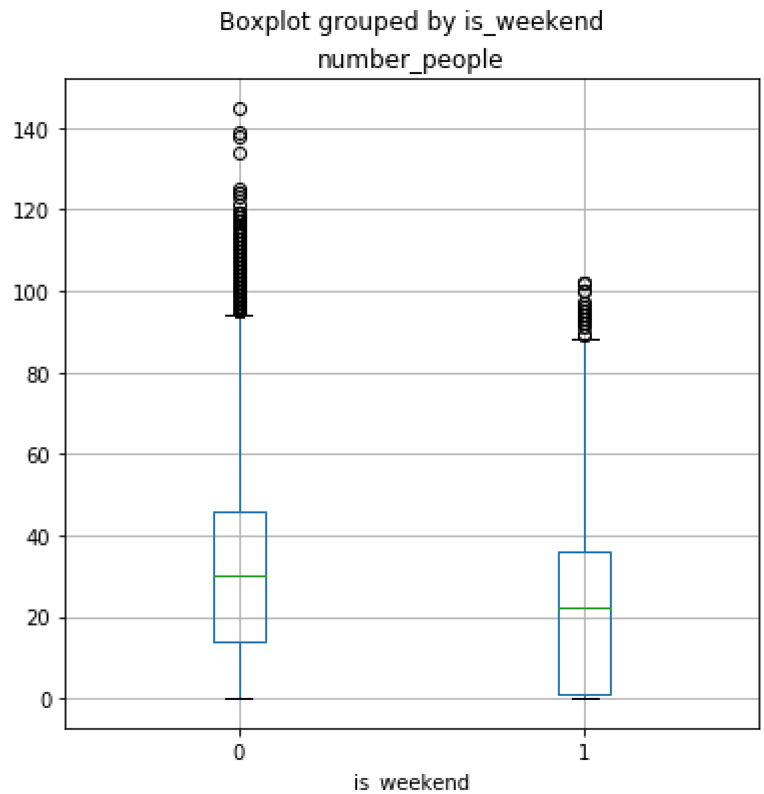
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1299f61d0>



Statement: Gym attendance steadily declines from Monday through Sunday.

In [13]:
```
df.boxplot(column="number_people", by= "is_weekend", figsize= (6,6))
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x129c26be0>



The above statement can analyzed in another angle by bucketing the data further and comparing weekdays verse weekend days. It is clear more people attend the gym during the week than on weekends.

Analysis: Temperature vs Number of Gym Attendees

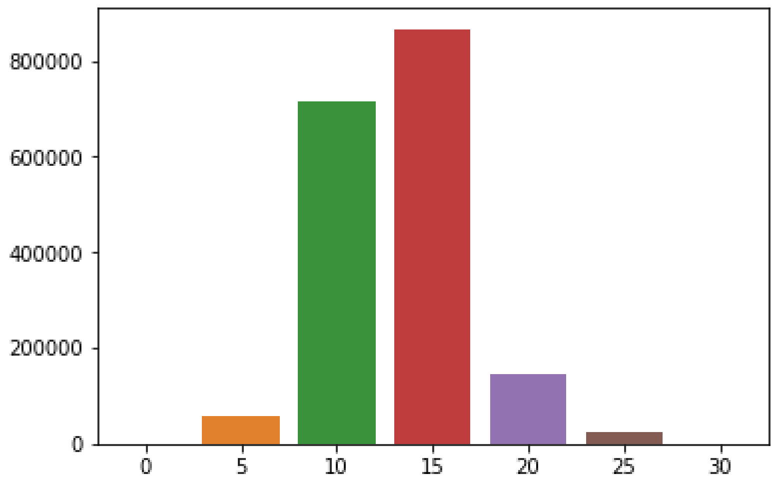
From correlation plot, there exists the positive correlation between number of people going to gym and temperature.

Hypothesis: The higher the temperature the lower the gym attendance.

Temperature data is continuous in nature, so using a histogram is a better way to visualize it compared to boxplots or barcharts

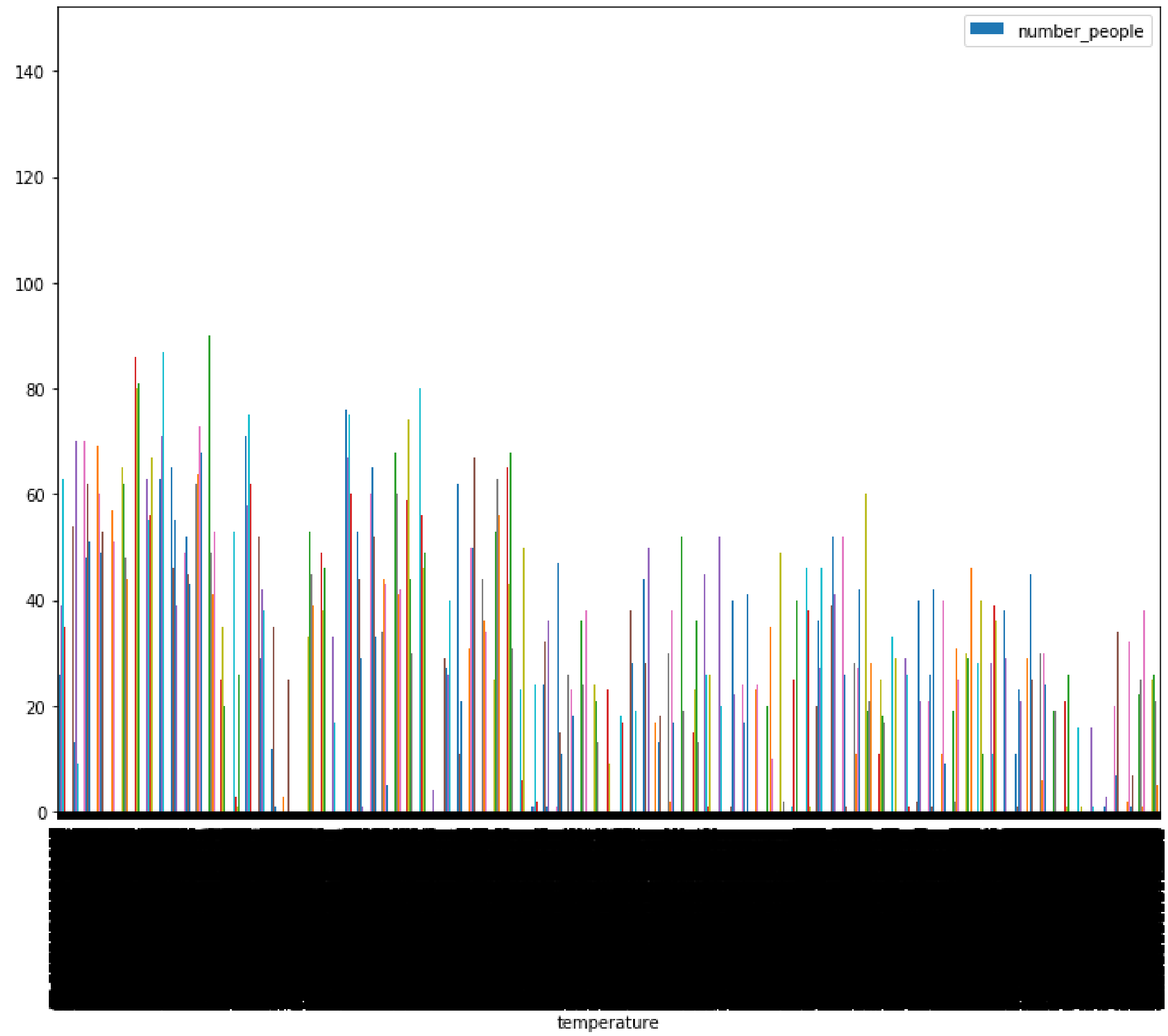
In [14]:
```
Bins = []
for i in range(0,35,5):
    people_count = 0
    for index, row in df.iterrows():
        if(row['temperature'] >= i and row['temperature'] < i+5):
            people_count = people_count + row['number_people']
    Bins.append((people_count))

sb.barplot(list(range(0,35,5)),Bins)
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x129fa9198>



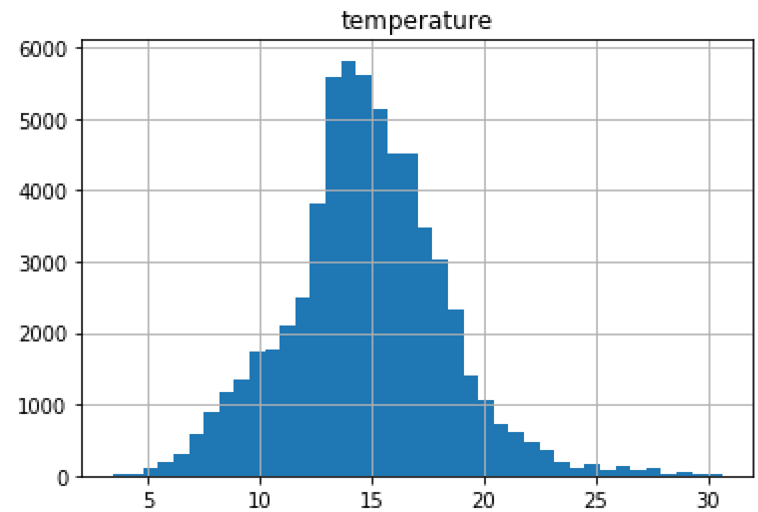
In [15]:
```
df.plot(kind='bar',x='temperature',y='number_people', figsize=(12,9))
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x12a7de470>



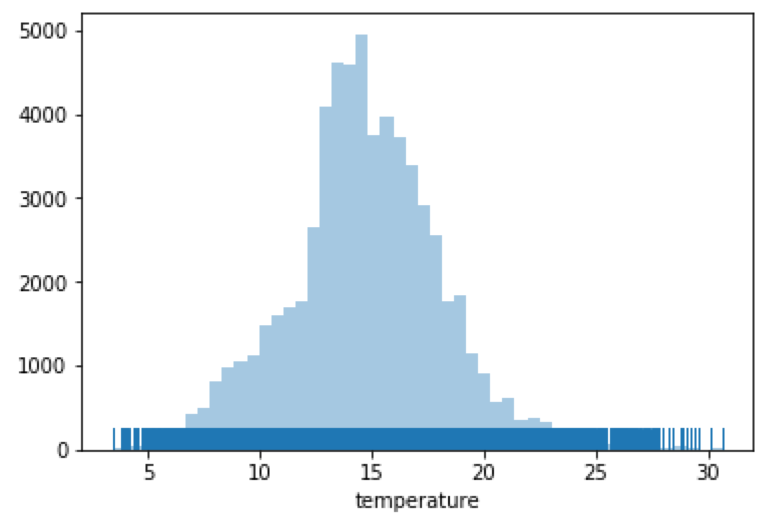
In [16]:
```
df.hist(column='temperature',bins=40)
```

Out[16]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x12a67e3c8
>]],
      dtype=object)
```



In [17]:
```
sb.distplot(df['temperature'], kde=False, rug=True)
```

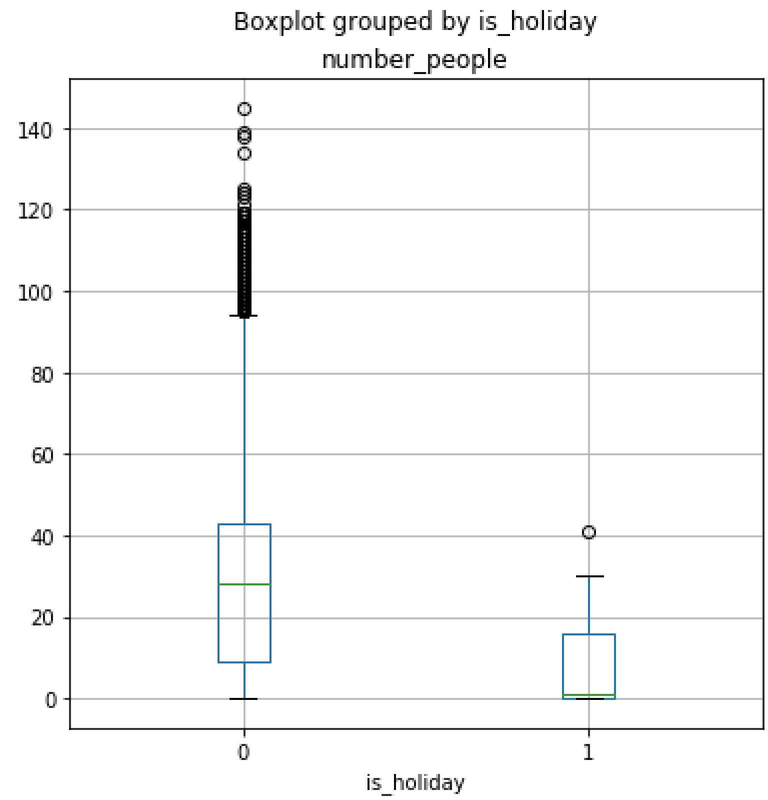
Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12bae64e0>
```



Even though from the distribution it seems like more people attend gym in warm weather its not true as most number of days weather remains in range 50 to 70. So the analysis that more people attend gym during 50 to 70 temperature range is not entirely correct.

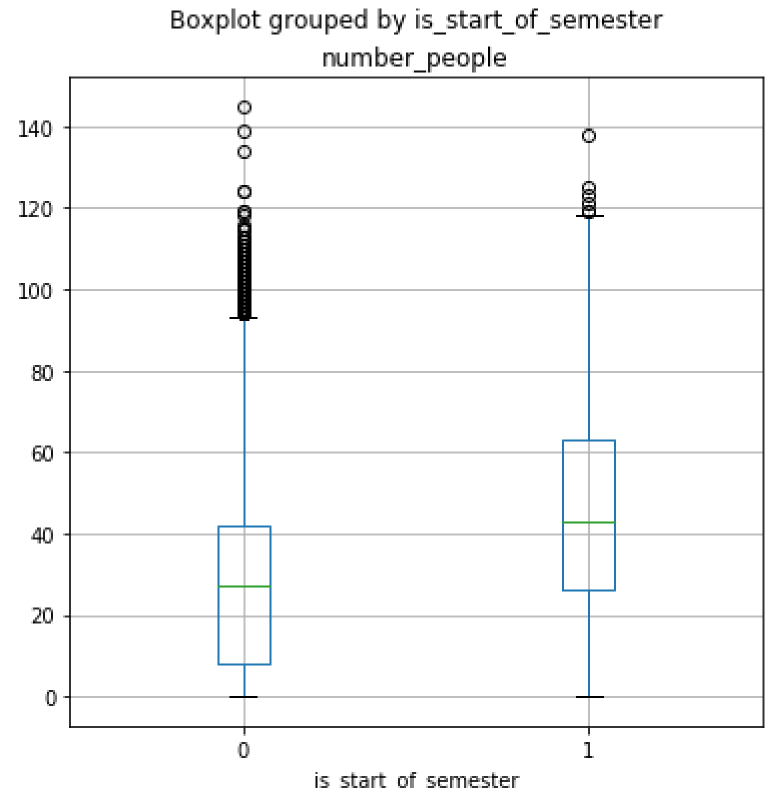
In [18]:
```
df.boxplot(column="number_people", by= "is_holiday", figsize= (6,6))
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x14469b6a0>



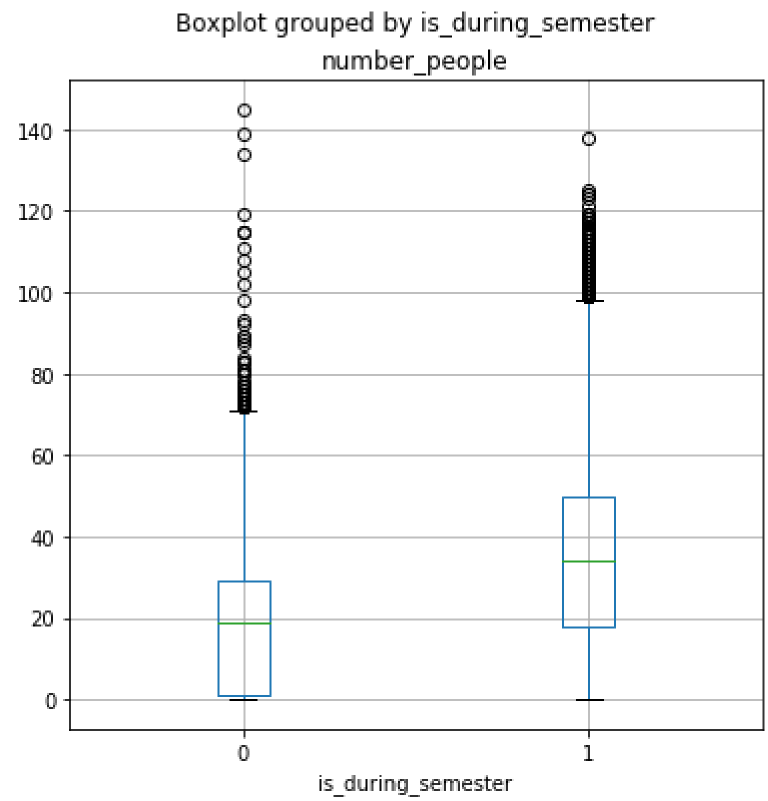
In [19]:
```
df.boxplot(column="number_people", by= "is_start_of_semester", figsize=
(6,6))
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x14541d240>



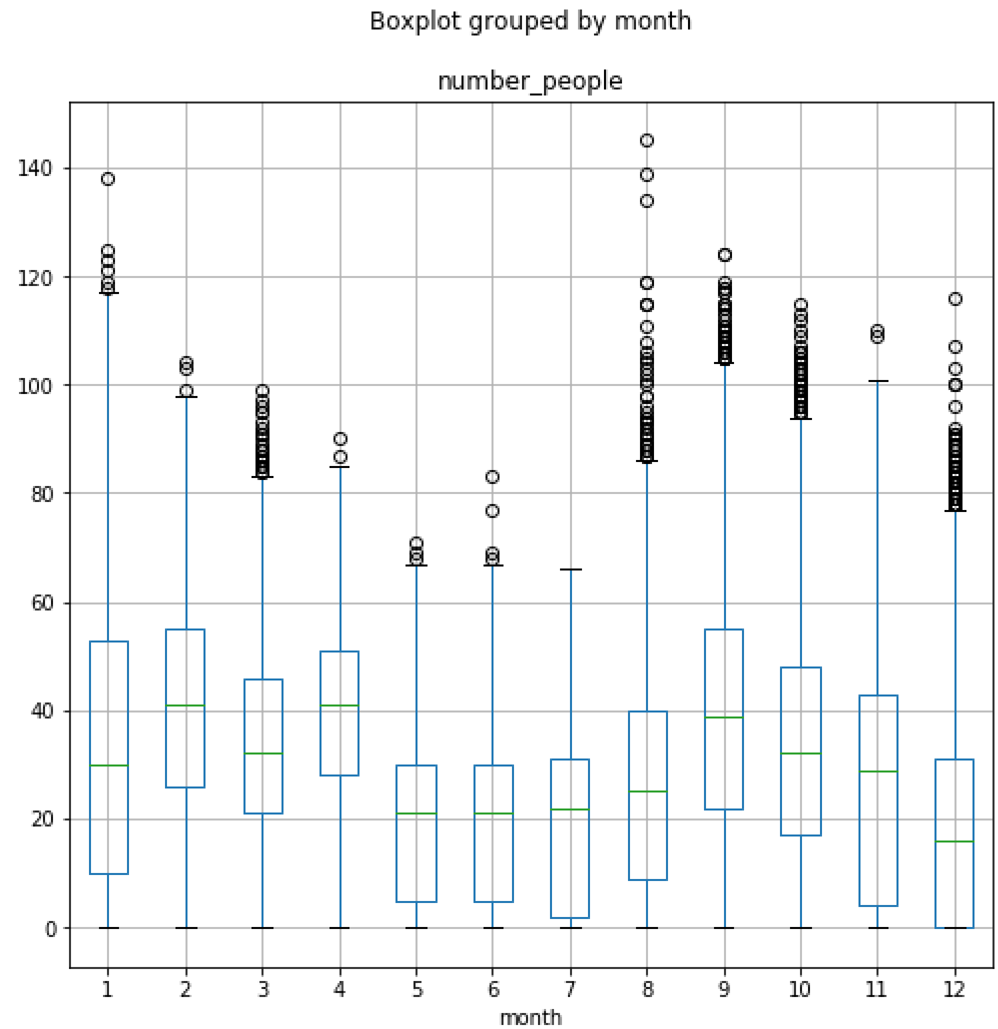
In [20]:
```
df.boxplot(column="number_people", by= "is_during_semester", figsize= (6
,6))
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x14559f7b8>



In [21]:
```
df.boxplot(column="number_people", by= "month", figsize= (8,8))
#could this basically be the same information as is_during_semester and
 is_start_of_semester? concerns about overfitting maybe? Worth at least
 discussing in the assignment
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x12bc1f710>



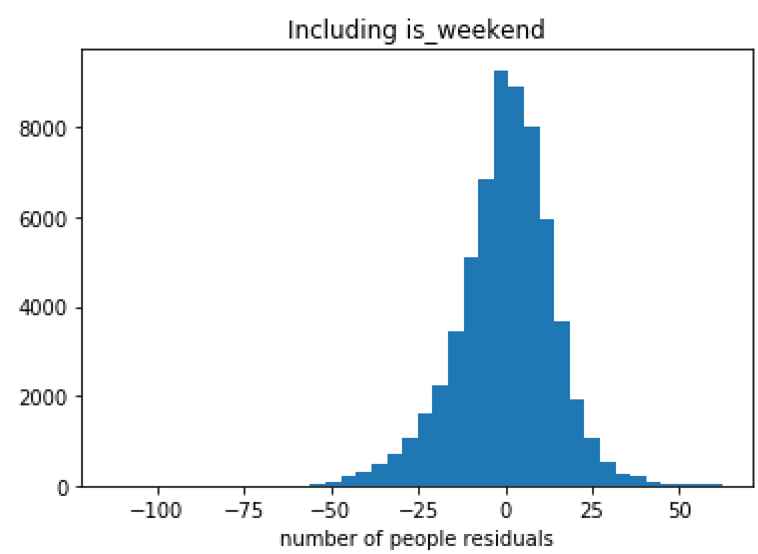
In [22]:
```
df.dtypes
```

Out[22]:
```
number_people            int64
timestamp                int64
day_of_week              int64
is_weekend               int64
is_holiday               int64
temperature            float64
is_start_of_semester     int64
is_during_semester       int64
date                    object
month                    int64
hour                     int64
dtype: object
```

In [23]:
```
#A lot of the features are categorical, so they habe to be changed to th
e category datatype
df["day_of_week"] = df["day_of_week"].astype('category')
df["is_weekend"] = df["is_weekend"].astype('category')
df["is_holiday"] = df["is_holiday"].astype('category')
df["is_start_of_semester"] = df["is_start_of_semester"].astype('categor
y')
df["is_during_semester"] = df["is_during_semester"].astype('category')
df["month"] = df["month"].astype('category')
df["hour"] = df["hour"].astype('category')
```

In [24]:
```
# Initialize and fit the model
m1 = smf.ols(formula='number_people ~ day_of_week + is_weekend + is_holi
day + temperature + is_start_of_semester + is_during_semester + month +
 hour', data=df)
m1 = m1.fit()
df['number_people_predicted'] = m1.predict(df)
df['number_people_redisuals'] = df['number_people_predicted'] - df['numb
er_people']
# Residuals mean
print("residuals mean: ",df.number_people_redisuals.mean())
print("residuals mean: ",m1.rsquared)
# Plot histogram of the residuals
plt.hist(df.number_people_redisuals,bins=40)
plt.xlabel('number of people residuals')
plt.title('Including is_weekend')
plt.show()
```

```
residuals mean:  -1.0022087366870980e-13
residuals mean:  0.6265773532323791
```



In [25]:
```
print(m1.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:           number_people   R-squared:
0.627
Model:                             OLS   Adj. R-squared:
0.626
Method:                  Least Squares   F-statistic:
2370.
Date:                 Mon, 16 Sep 2019   Prob (F-statistic):
0.00
Time:                         18:25:32   Log-Likelihood:                -2.5
174e+05
No. Observations:                62184   AIC:                             5.
036e+05
Df Residuals:                    62139   BIC:                             5.
040e+05
Df Model:                           44
Covariance Type:             nonrobust
==============================================================================
======================
                                  coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
----------------------
Intercept                       3.7228      0.451      8.258      0.000
2.839       4.606
day_of_week[T.1]               -0.5321      0.208     -2.556      0.011
-0.940      -0.124
day_of_week[T.2]               -1.0535      0.209     -5.036      0.000
-1.464      -0.643
day_of_week[T.3]               -2.8412      0.210    -13.561      0.000
-3.252      -2.431
day_of_week[T.4]               -3.5789      0.211    -16.993      0.000
-3.992      -3.166
day_of_week[T.5]               -2.9220      0.121    -24.162      0.000
-3.159      -2.685
day_of_week[T.6]               -3.7137      0.121    -30.666      0.000
-3.951      -3.476
is_weekend[T.1]                -6.6357      0.121    -54.649      0.000
-6.874      -6.398
is_holiday[T.1]               -17.3138      1.135    -15.249      0.000
-19.539     -15.088
is_start_of_semester[T.1]       3.1802      0.279     11.385      0.000
2.633       3.728
is_during_semester[T.1]        14.8879      0.230     64.765      0.000
14.437      15.339
month[T.2]                     -4.6966      0.388    -12.111      0.000
-5.457      -3.937
month[T.3]                    -11.6965      0.377    -30.989      0.000
-12.436     -10.957
month[T.4]                     -8.0203      0.416    -19.287      0.000
-8.835      -7.205
month[T.5]                    -12.5490      0.342    -36.659      0.000
-13.220     -11.878
month[T.6]                     -7.2086      0.357    -20.181      0.000
-7.909      -6.508
```

```
month[T.7]                      -5.5862      0.346    -16.166      0.000
-6.264      -4.909
month[T.8]                      -8.0697      0.319    -25.333      0.000
-8.694      -7.445
month[T.9]                      -8.5284      0.352    -24.195      0.000
-9.219      -7.838
month[T.10]                    -12.0302      0.377    -31.928      0.000
-12.769     -11.292
month[T.11]                    -13.8252      0.355    -38.935      0.000
-14.521     -13.129
month[T.12]                    -12.7204      0.306    -41.602      0.000
-13.320     -12.121
hour[T.1]                      -10.3511      0.416    -24.888      0.000
-11.166      -9.536
hour[T.2]                      -11.5346      0.438    -26.305      0.000
-12.394     -10.675
hour[T.3]                      -11.3915      0.439    -25.950      0.000
-12.252     -10.531
hour[T.4]                      -10.5569      0.447    -23.610      0.000
-11.433      -9.681
hour[T.5]                      -12.3209      0.379    -32.489      0.000
-13.064     -11.578
hour[T.6]                       -5.1348      0.367    -13.988      0.000
-5.854      -4.415
hour[T.7]                        1.1104      0.366      3.037      0.002
0.394       1.827
hour[T.8]                        9.2578      0.365     25.350      0.000
8.542       9.974
hour[T.9]                       14.7418      0.365     40.335      0.000
14.025      15.458
hour[T.10]                      17.5772      0.367     47.946      0.000
16.859      18.296
hour[T.11]                      19.4239      0.369     52.689      0.000
18.701      20.146
hour[T.12]                      18.6265      0.370     50.276      0.000
17.900      19.353
hour[T.13]                      16.5558      0.373     44.413      0.000
15.825      17.286
hour[T.14]                      15.7203      0.375     41.962      0.000
14.986      16.455
hour[T.15]                      19.0739      0.374     50.964      0.000
18.340      19.808
hour[T.16]                      25.1287      0.372     67.577      0.000
24.400      25.858
hour[T.17]                      30.4123      0.367     82.938      0.000
29.694      31.131
hour[T.18]                      31.5467      0.367     85.999      0.000
30.828      32.266
hour[T.19]                      28.5130      0.366     77.897      0.000
27.796      29.230
hour[T.20]                      25.8457      0.366     70.589      0.000
25.128      26.563
hour[T.21]                      27.4416      0.366     74.920      0.000
26.724      28.159
hour[T.22]                      25.3289      0.366     69.177      0.000
24.611      26.047
hour[T.23]                      11.2920      0.363     31.117      0.000
```

```
10.581      12.003
temperature                  1.0314      0.025     41.704      0.000
0.983       1.080
========================================================================
=======
Omnibus:                   3845.022   Durbin-Watson:
0.234
Prob(Omnibus):                0.000   Jarque-Bera (JB):                7
366.758
Skew:                         0.451   Prob(JB):
0.00
Kurtosis:                     4.425   Cond. No.
2.25e+15
========================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The smallest eigenvalue is 2.84e-24. This might indicate that there
are
strong multicollinearity problems or that the design matrix is singula
r.
```

In [26]:

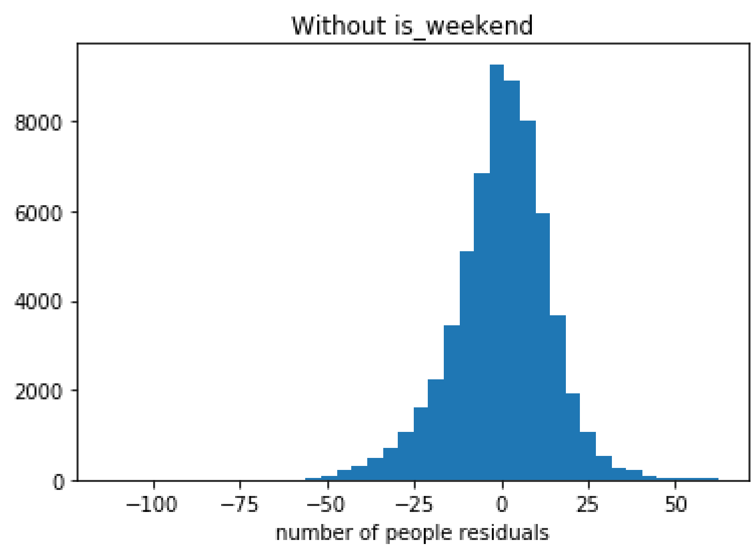
```
df.head(1)
```

Out[26]:

	number_people	timestamp	day_of_week	is_weekend	is_holiday	temperature	is_sta
0	37	61211	4	0	0	22.088889	0

In [27]:
```
# Initialize and fit the model
params = ['day_of_week' , 'is_holiday' , 'temperature' , 'is_start_of_semester' , 'is_during_semester' , 'month' , 'hour']
m2 = smf.ols(formula='number_people ~ day_of_week + is_holiday + temperature + is_start_of_semester + is_during_semester + month + hour', data=df)
m2 = m2.fit()

df['number_people_predicted'] = m2.predict(df)
df['number_people_redisuals'] = df['number_people_predicted'] - df['number_people']
# Residuals mean
print("residuals mean: ",df.number_people_redisuals.mean())
print("residuals mean: ",m2.rsquared)
# Plot histogram of the residuals
plt.hist(df.number_people_redisuals,bins=40)
plt.xlabel('number of people residuals')
plt.title('Without is_weekend')
plt.show()
```

```
residuals mean:  -2.516511235711655e-13
residuals mean:  0.626577353232379
```



In [28]:
```
print(m2.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:          number_people   R-squared:
0.627
Model:                            OLS   Adj. R-squared:
0.626
Method:                 Least Squares   F-statistic:
2370.
Date:                Mon, 16 Sep 2019   Prob (F-statistic):
0.00
Time:                        18:25:34   Log-Likelihood:              -2.5
174e+05
No. Observations:               62184   AIC:                           5.
036e+05
Df Residuals:                   62139   BIC:                           5.
040e+05
Df Model:                          44
Covariance Type:            nonrobust
==============================================================================
======================
                                coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
----------------------
Intercept                     3.7228      0.451      8.258      0.000
2.839       4.606
day_of_week[T.1]             -0.5321      0.208     -2.556      0.011
-0.940      -0.124
day_of_week[T.2]             -1.0535      0.209     -5.036      0.000
-1.464      -0.643
day_of_week[T.3]             -2.8412      0.210    -13.561      0.000
-3.252      -2.431
day_of_week[T.4]             -3.5789      0.211    -16.993      0.000
-3.992      -3.166
day_of_week[T.5]             -9.5577      0.210    -45.528      0.000
-9.969      -9.146
day_of_week[T.6]            -10.3494      0.210    -49.230      0.000
-10.761      -9.937
is_holiday[T.1]             -17.3138      1.135    -15.249      0.000
-19.539     -15.088
is_start_of_semester[T.1]     3.1802      0.279     11.385      0.000
2.633       3.728
is_during_semester[T.1]      14.8879      0.230     64.765      0.000
14.437      15.339
month[T.2]                   -4.6966      0.388    -12.111      0.000
-5.457      -3.937
month[T.3]                  -11.6965      0.377    -30.989      0.000
-12.436     -10.957
month[T.4]                   -8.0203      0.416    -19.287      0.000
-8.835      -7.205
month[T.5]                  -12.5490      0.342    -36.659      0.000
-13.220     -11.878
month[T.6]                   -7.2086      0.357    -20.181      0.000
-7.909      -6.508
month[T.7]                   -5.5862      0.346    -16.166      0.000
-6.264      -4.909
```

```
month[T.8]                     -8.0697      0.319    -25.333      0.000
-8.694      -7.445
month[T.9]                     -8.5284      0.352    -24.195      0.000
-9.219      -7.838
month[T.10]                   -12.0302      0.377    -31.928      0.000
-12.769     -11.292
month[T.11]                   -13.8252      0.355    -38.935      0.000
-14.521     -13.129
month[T.12]                   -12.7204      0.306    -41.602      0.000
-13.320     -12.121
hour[T.1]                     -10.3511      0.416    -24.888      0.000
-11.166      -9.536
hour[T.2]                     -11.5346      0.438    -26.305      0.000
-12.394     -10.675
hour[T.3]                     -11.3915      0.439    -25.950      0.000
-12.252     -10.531
hour[T.4]                     -10.5569      0.447    -23.610      0.000
-11.433      -9.681
hour[T.5]                     -12.3209      0.379    -32.489      0.000
-13.064     -11.578
hour[T.6]                      -5.1348      0.367    -13.988      0.000
-5.854      -4.415
hour[T.7]                       1.1104      0.366      3.037      0.002
0.394       1.827
hour[T.8]                       9.2578      0.365     25.350      0.000
8.542       9.974
hour[T.9]                      14.7418      0.365     40.335      0.000
14.025      15.458
hour[T.10]                     17.5772      0.367     47.946      0.000
16.859      18.296
hour[T.11]                     19.4239      0.369     52.689      0.000
18.701      20.146
hour[T.12]                     18.6265      0.370     50.276      0.000
17.900      19.353
hour[T.13]                     16.5558      0.373     44.413      0.000
15.825      17.286
hour[T.14]                     15.7203      0.375     41.962      0.000
14.986      16.455
hour[T.15]                     19.0739      0.374     50.964      0.000
18.340      19.808
hour[T.16]                     25.1287      0.372     67.577      0.000
24.400      25.858
hour[T.17]                     30.4123      0.367     82.938      0.000
29.694      31.131
hour[T.18]                     31.5467      0.367     85.999      0.000
30.828      32.266
hour[T.19]                     28.5130      0.366     77.897      0.000
27.796      29.230
hour[T.20]                     25.8457      0.366     70.589      0.000
25.128      26.563
hour[T.21]                     27.4416      0.366     74.920      0.000
26.724      28.159
hour[T.22]                     25.3289      0.366     69.177      0.000
24.611      26.047
hour[T.23]                     11.2920      0.363     31.117      0.000
10.581      12.003
temperature                     1.0314      0.025     41.704      0.000
```

```
0.983       1.080
==============================================================================
=======
Omnibus:                     3845.022   Durbin-Watson:
0.234
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7
366.758
Skew:                           0.451   Prob(JB):
0.00
Kurtosis:                       4.425   Cond. No.
350.
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

Analysis

At start of semester people tend to go to gym more

In [29]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import SGDClassifier,SGDRegressor
```

```
/Users/cammilligan/anaconda/lib/python3.6/site-packages/sklearn/ensembl
e/weight_boosting.py:29: DeprecationWarning: numpy.core.umath_tests is
an internal NumPy module and should not be imported. It will be removed
in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [30]:
```
df2 = df_copy.copy()
y = df2['number_people'].values
df2_X = df2.drop(['number_people','date','timestamp','is_weekend'],axis=
1)
X = df2_X.values
df2.head(2)
```

Out[30]:

	number_people	timestamp	day_of_week	is_weekend	is_holiday	temperature	is_sta
0	37	61211	4	0	0	22.088889	0
1	45	62414	4	0	0	22.088889	0

In [31]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.40)
```

In [32]:
```
# Scale the data to be between -1 and 1
scaler = StandardScaler()
scaler.fit(Xtrain)
Xtrain = scaler.transform(Xtrain)
Xtest = scaler.transform(Xtest)
```

In [33]:
```
sdg = SGDRegressor()
sdg.fit(Xtrain, ytrain)
y_val_l = sdg.predict(Xtest)
print(sdg.score(Xtest, ytest))
```

```
0.505364392240245
```

In [34]:
```
radm = RandomForestRegressor(n_estimators=100)
radm.fit(Xtrain, ytrain)

print(radm.score(Xtest, ytest))
```

```
0.912177349613403
```

Analysis

The predictive model can predict number of people attending the gym with 91% accuracy

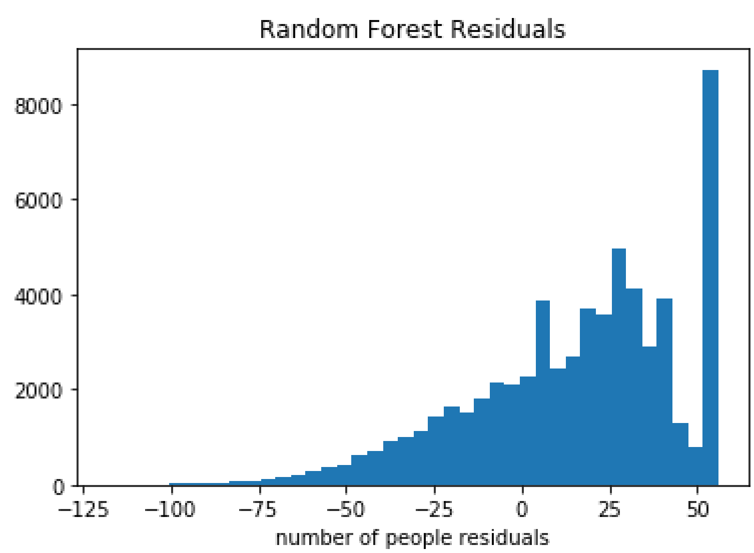
In [35]:
```
indices = np.argsort(radm.feature_importances_)[::-1]

# Print the feature ranking
print('Feature ranking:')

for f in range(df2_X.shape[1]):
    print('%d. feature %d %s (%f)' % (f+1 , indices[f], df2_X.columns[in
dices[f]],
                                                        radm.feature_importances_[indices[
f]]))
```

```
Feature ranking:
1. feature 6 hour (0.519484)
2. feature 2 temperature (0.179122)
3. feature 4 is_during_semester (0.110511)
4. feature 0 day_of_week (0.093581)
5. feature 5 month (0.084394)
6. feature 3 is_start_of_semester (0.012813)
7. feature 1 is_holiday (0.000095)
```

In [36]:
```
df2['number_people_predicted'] = radm.predict(X)
```

In [37]:
```
df2['number_people_redisuals'] = df2['number_people_predicted'] - df2['n
umber_people']
# Residuals mean
print("residuals mean: ",df2.number_people_redisuals.mean())
# Plot histogram of the residuals
plt.hist(df2.number_people_redisuals,bins=40)
plt.xlabel('number of people residuals')
plt.title('Random Forest Residuals')
plt.show()
```

```
residuals mean:  15.401467539222255
```



Analysis

Below is the list of the factors affecting the number of people attending gym ranked from top to bottom in decreasing imprtance manner:

1. Hour
2. Temperature
3. Is during semester
4. Day of week
5. Month
6. Is start of semester
7. Is holiday