PROJECT 1 - MILESTONE REPORT

1. Project Statement

For retail stores, if you want to keep a customer coming back with your business, you need to have enough product to satisfy customer needs. The problem is how many products do you need to order to provide to customers but not keep the product in the inventory so long. This project will predict the total sales of each product each month so the business can maximum the sales of each month and reduce the money for inventory. So below are the questions I want to answer along with my project:

- What is the total sales for each item (was provided in a test file) next month?
- What are the top 10 products sold in each month of each store?
- What range of prices is the customer likely to buy?

2. Data Wrangling

a. Gather and Loading the data

- Data was provided from the kaggle competior

b. Clearning the data

- Our data was cleaned and has no missing value, our goal is to predict the total sale of each item for next month. Column item_cnt_day is used for our training model.
- There are some outliers for some items, it may affect our prediction. So all that item was removed.
- Item_cnt_day was have some negative data, then I just removed so it not make any noise later

3. Story telling

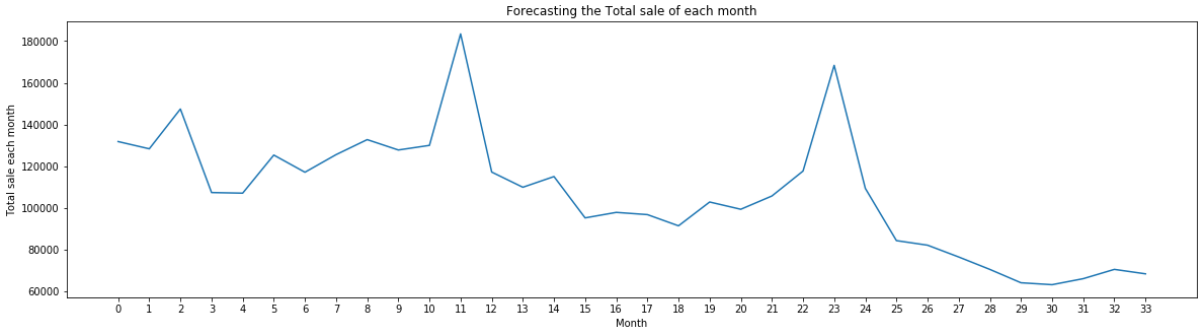
a. Visulize the total sale for all shop

- The plot shows that we have more items sold in november than another month, and there are less items sold in January.

In [150]:
```
all_shop_month_sale()
```

Out[150]:

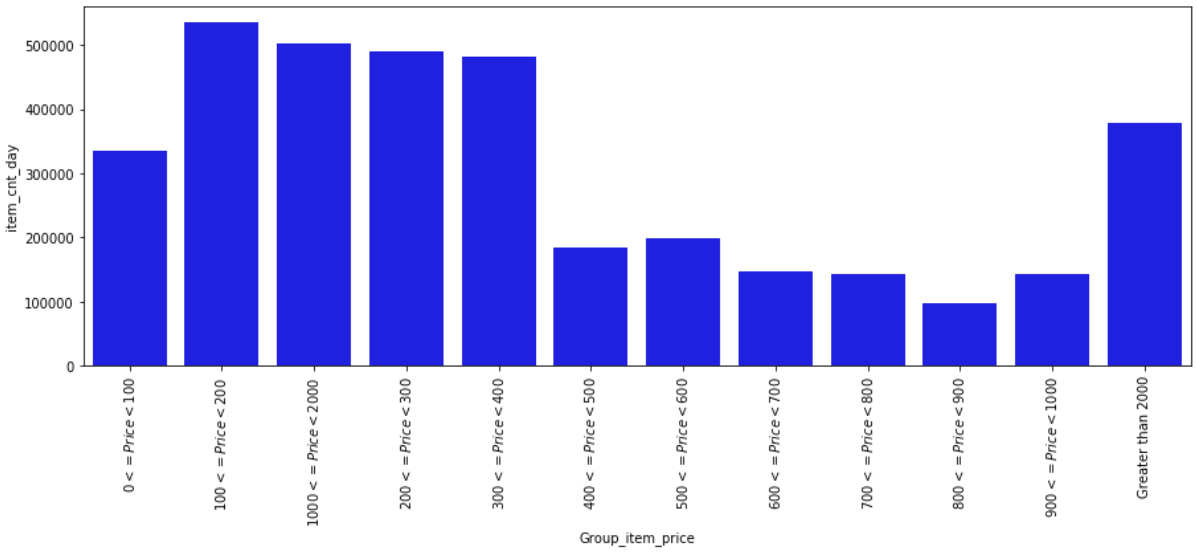
	min	25% quantile	50% quantile	mean	75% quantile	max
0	63316.0	86190.5	107293.0	107023.5	125535.25	183415.0



b. See the total sale for the range of price

- We have more item sale with price less than 400

In [151]:
```
#Plot group the price range and see how much selling that range price
group_item_price_plot()
```

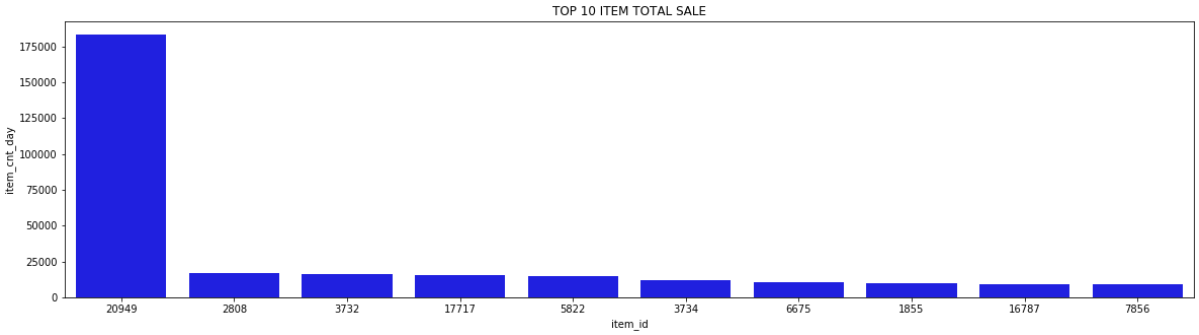


c. Top 10 item sold

In [162]:
```
top10_item_sale_all_shop()
```

Out[162]:

item_id	item_name	item_cnt_day	total_price
20949	Фирменный пакет майка 1С Интерес белый (34*42) 45 мкм	183208.0	9.087687e+05
2808	Diablo III [PC, Jewel, русская версия]	17055.0	1.666347e+07
3732	Grand Theft Auto V [PS3, русские субтитры]	15907.0	4.175331e+07
17717	Прием денежных средств для 1С-Онлайн	15830.0	1.720083e+07
5822	Playstation Store пополнение бумажника: Карта оплаты 1000 руб.	14522.0	1.541673e+07
3734	Grand Theft Auto V [Xbox 360, русские субтитры]	11733.0	3.118972e+07
6675	Sony PlayStation 4 (500 Gb) Black (CUH-1008A/1108A/B01)	10315.0	2.199861e+08
1855	Battlefield 4 [PC, русская версия]	10041.0	9.365399e+06
16787	Одни из нас [PS3, русская версия]	9255.0	2.197261e+07
7856	World of Warcraft. Карта оплаты игрового времени (online) (рус.в.) (60 дней) (Jewel)	9016.0	7.263399e+06



d. Conclusion

- The peak sell for the year is November.
- The total sold are decreasing.
- About 50% of shops sell equal or above the average of all shops sold.
- The most frequency for selling 1 item per day is a lot.
- The less the price, tend to have more sales.
- The most item sold was under 400
- We need to consider item 20949, the plastic bag, have the most sale of - the item. So it brings our mean and total sell way high.
- For now, we are just all assuming the item 20949 was good, so everything we will include them.

4. Apply Statistics

A. Null Hypothesis Test

a. Define hypothesis

- Ho(null hypothesis) : is greater than the mean of total_price sale of the data
- Ha(alternative hypothesis) : is less than the mean of total_price sale of data of data
- Ho > 1954.15
- Ha <= 1954.15

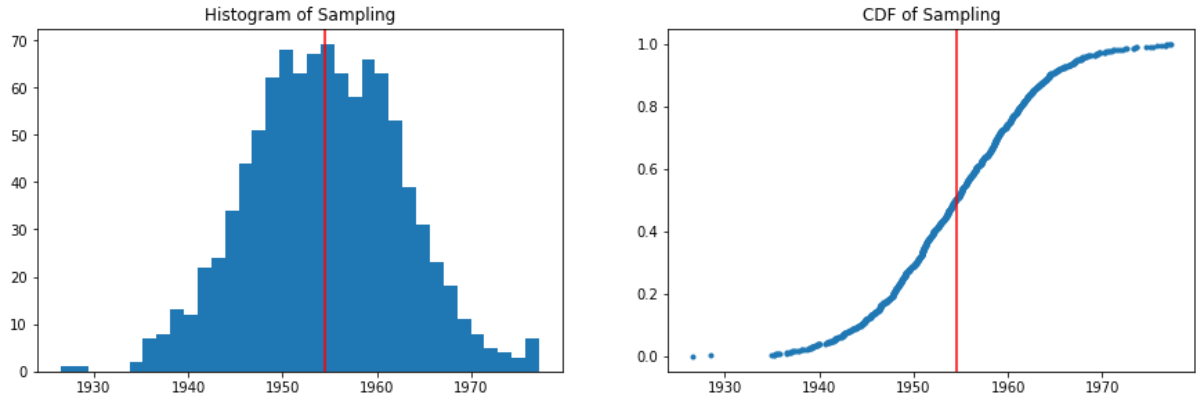
b. Choice 95 % interval

- when we choose a 95% confidence interval. So we will have our significance levels is 5%
- Based on I define the null hypothesis, so we will do one-tail testing.
- So alpha = 5% = 0.05 (we will compare this number with p-value to determine reject the null hypothesis or not)

c. Hypothesis testing

- we set them random seed to 47, so we can get same result all the time
- We simulate the same length as the total_price, and random choice data point point from total_price, then we calculate the mean of new sampling. we repeated it 1000 times. It means we will have 1000 new mean total_price it was generated based on total_price data.

```
In [170]: null_hypothesis_plot()
```



d. Conclusion

- We can not reject Null hypothesis.
- p-value is 0.514
- the standard deviation is so large

Inference Frequency statistic

- To compare the average sold for each shop

a. Simulation T-value table

- After we simulation and this is the table show T-value compare of each shop. Column name and row index are the shop_id

In [174]:
```
T_value_compare.head(10)
```

Out[174]:

	0	1	2	3	4	5	6	7	8	
0	-1.10562	4.99668	-11.6513	-5.47773	-4.06244	-2.78588	-6.21231	-6.64816	7.08934	-44
1	-7.15009	-1.10562	-15.8976	-12.9559	-10.6043	-9.11434	-10.2764	-11.8368	0.854497	-49
2	9.52814	13.9081	-1.10562	6.9419	7.53103	8.39491	3.53833	4.42639	15.2085	-32
3	3.26948	10.7789	-9.03828	-1.10562	0.134377	1.48564	-3.38171	-3.30245	13.9552	-43
4	1.85118	8.45218	-9.65754	-2.34194	-1.10562	0.170185	-4.15636	-4.20363	10.9094	-43
5	0.574637	6.96185	-10.5202	-3.69346	-2.38143	-1.10562	-5.04635	-5.26024	9.26414	-43
6	4.08193	8.28173	-5.74931	1.278	2.02267	2.91392	-1.10562	-0.682711	9.45087	-36
7	4.46657	9.77079	-6.61825	1.14051	2.02005	3.0774	-1.51355	-1.10562	11.4495	-39
8	-9.02076	-2.92061	-17.0102	-15.9247	-12.8425	-11.1963	-11.2496	-13.3014	-1.10562	-5
9	42.7424	47.1215	30.0476	41.1046	41.1536	41.8395	34.3971	37.0069	48.5878	-1.1

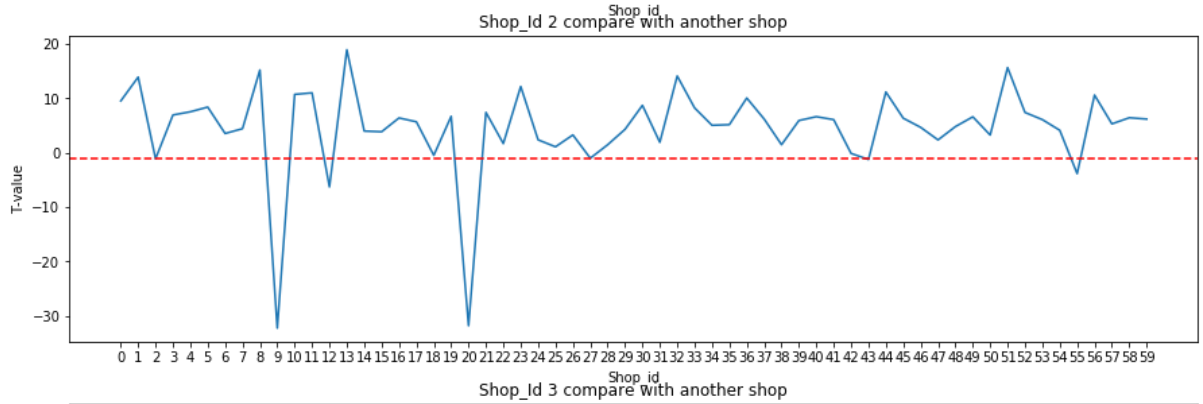
10 rows × 60 columns

b. Plot the t-value

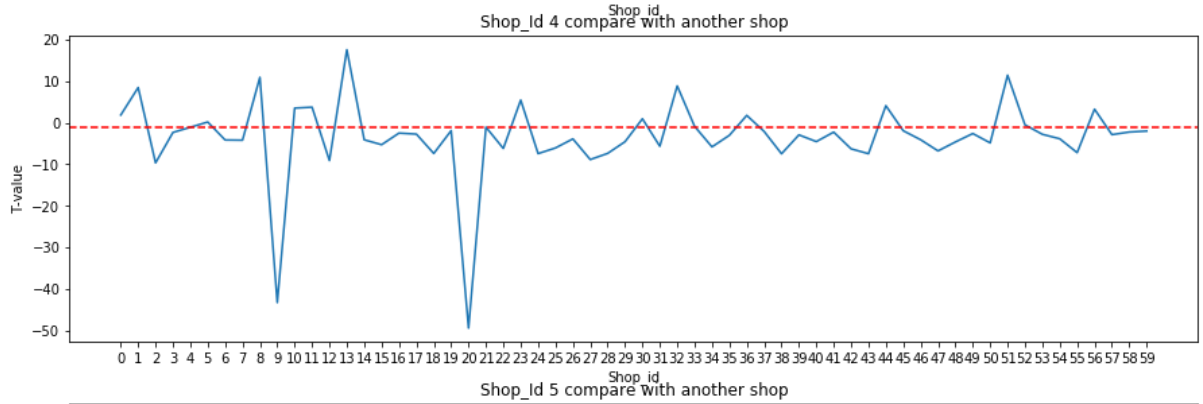
In [175]:
```
shop_range = np.arange(0,60,1)
plot_shop_compare(shop_range)
```


Shop_Id 0 compare with another shop
T-value
Shop_id

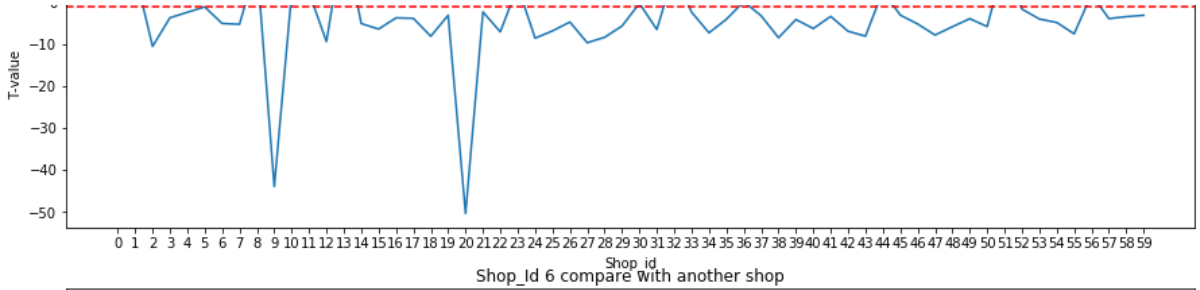

Shop_Id 1 compare with another shop
T-value
Shop_id

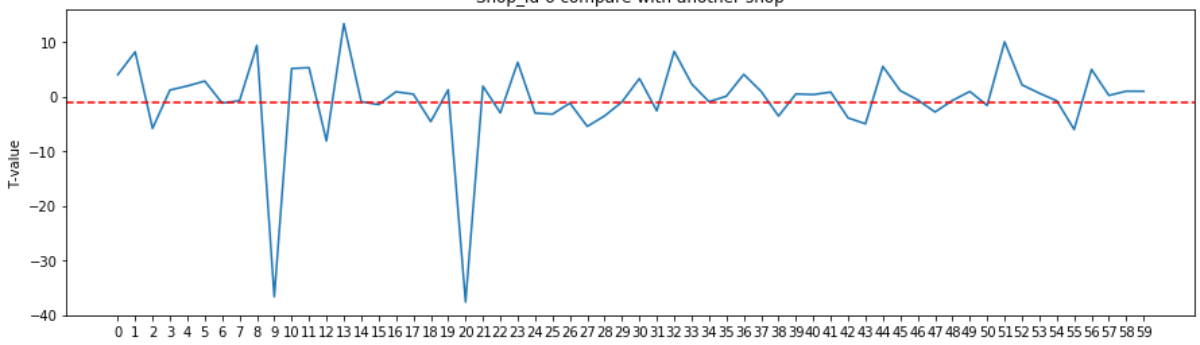

Shop_Id 2 compare with another shop
T-value
Shop_id

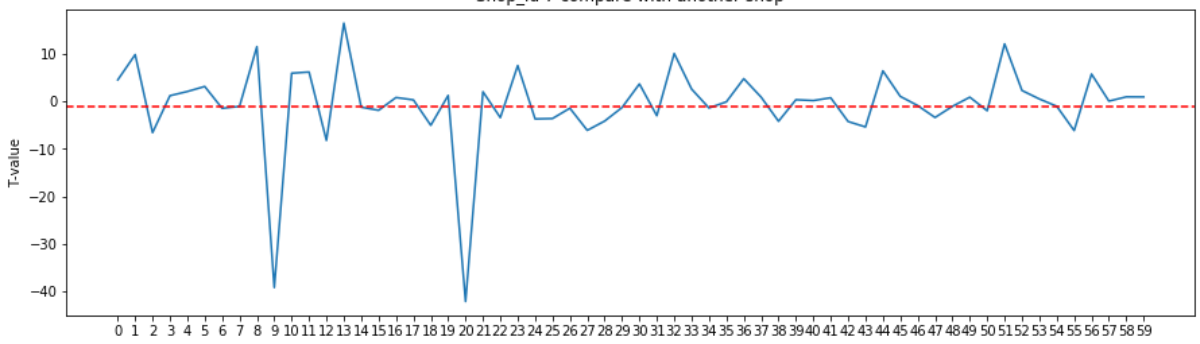

Shop_Id 3 compare with another shop
T-value
Shop_id


Shop_Id 4 compare with another shop
T-value
Shop_id

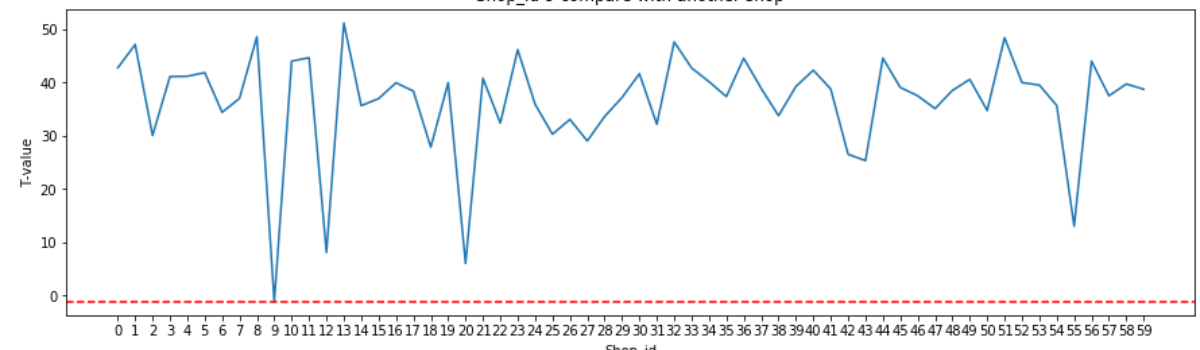

Shop_Id 5 compare with another shop

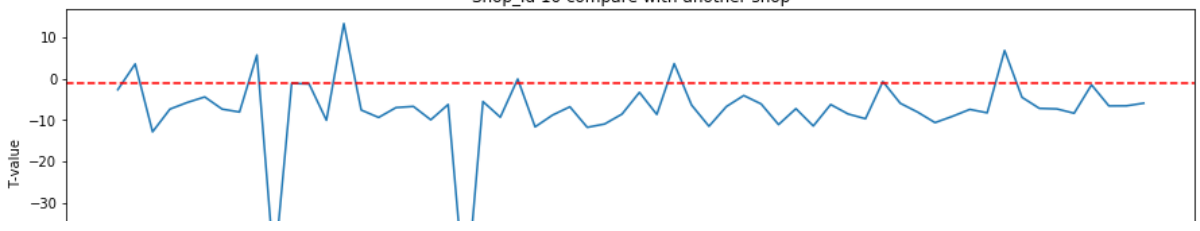

T-value
Shop_id

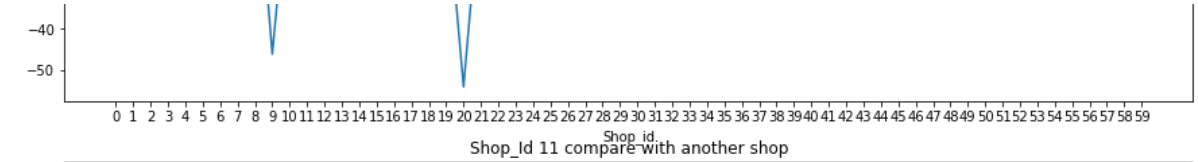

Shop_Id 6 compare with another shop
T-value
Shop_id

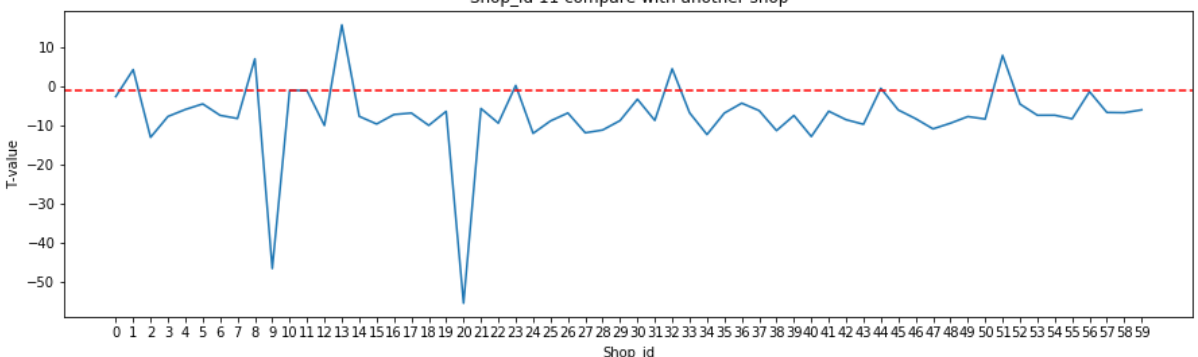

Shop_Id 7 compare with another shop
T-value
Shop_id

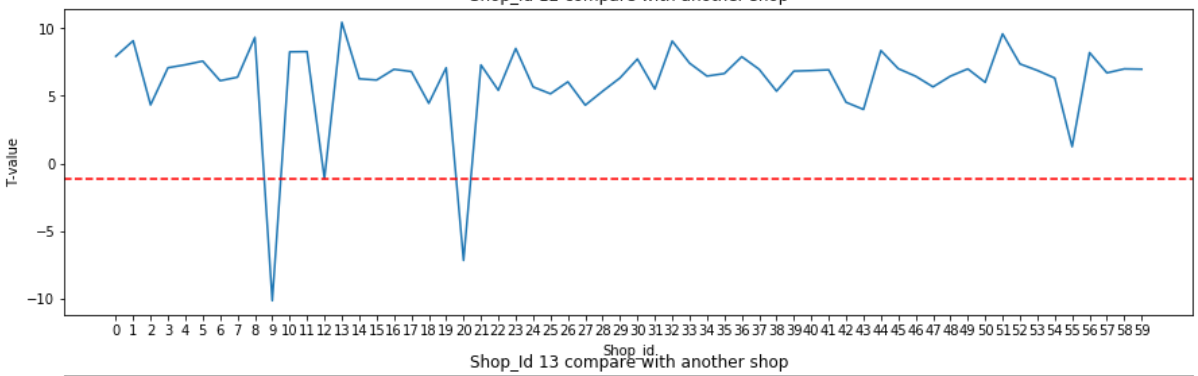

Shop_Id 8 compare with another shop
T-value
Shop_id

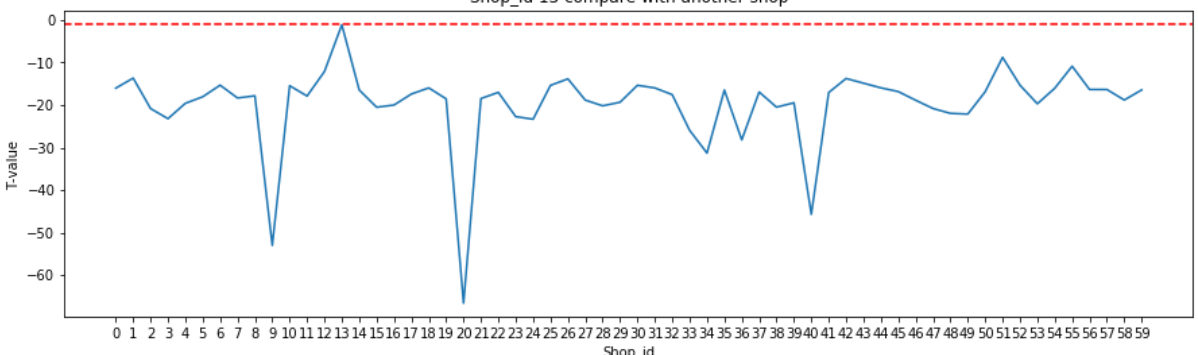

Shop_Id 9 compare with another shop
T-value
Shop_id


Shop_Id 10 compare with another shop
T-value

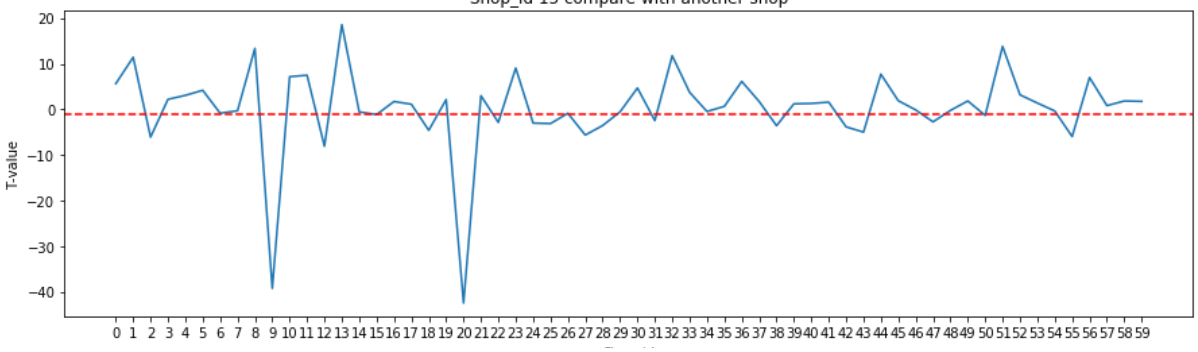

Shop_id

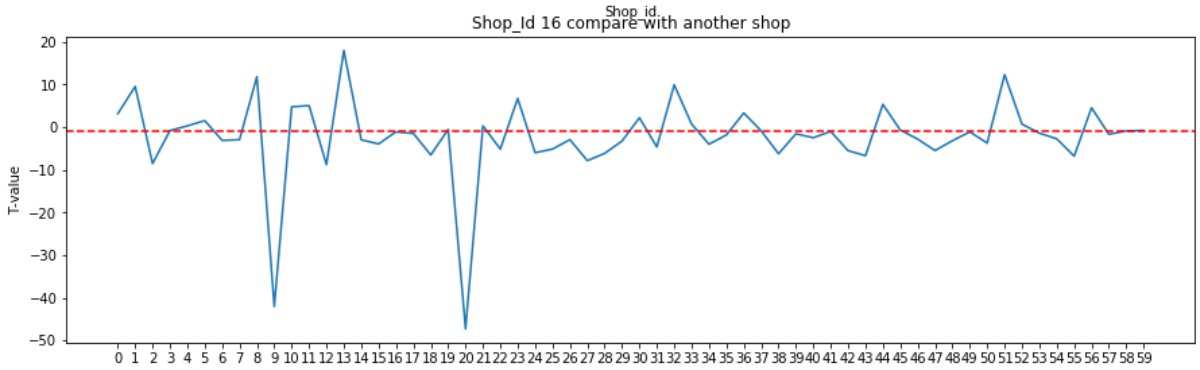

Shop_Id 11 compare with another shop
T-value
Shop_id

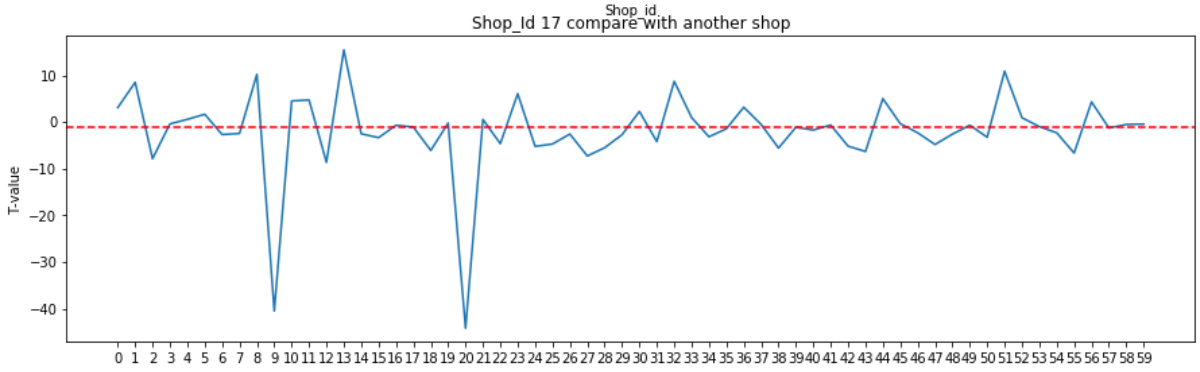

Shop_Id 12 compare with another shop
T-value
Shop_id

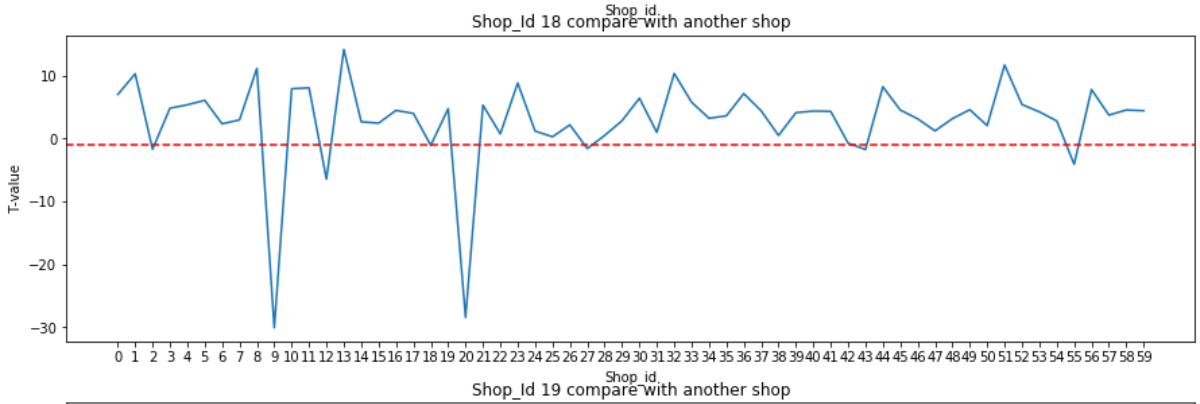

Shop_Id 13 compare with another shop
T-value
Shop_id

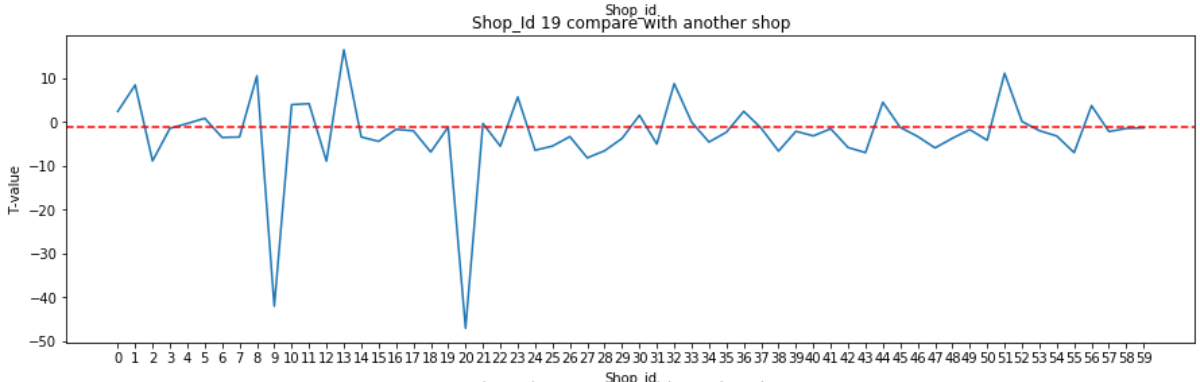

Shop_Id 14 compare with another shop
T-value
Shop_id


Shop_Id 15 compare with another shop
T-value
Shop_id


Shop_Id 16 compare with another shop
T-value
Shop_id

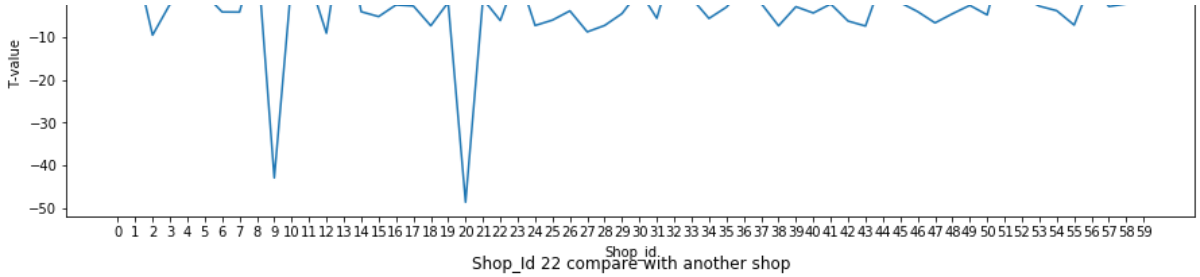

Shop_Id 17 compare with another shop
T-value
Shop_id

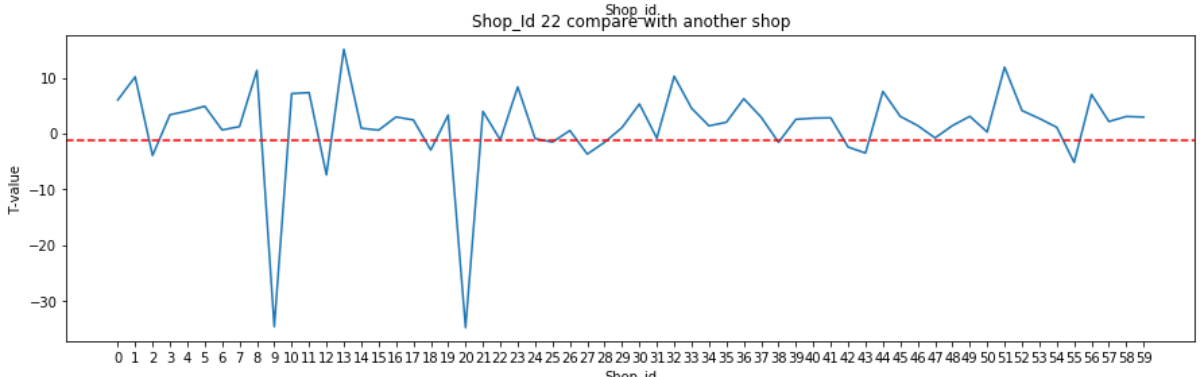

Shop_Id 18 compare with another shop
T-value
Shop_id


Shop_Id 19 compare with another shop
T-value
Shop_id

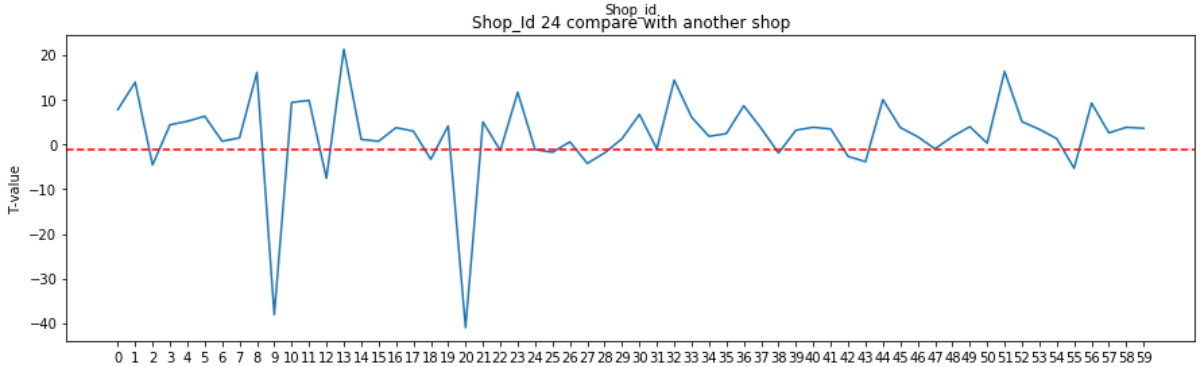

Shop_Id 20 compare with another shop
T-value
Shop_id

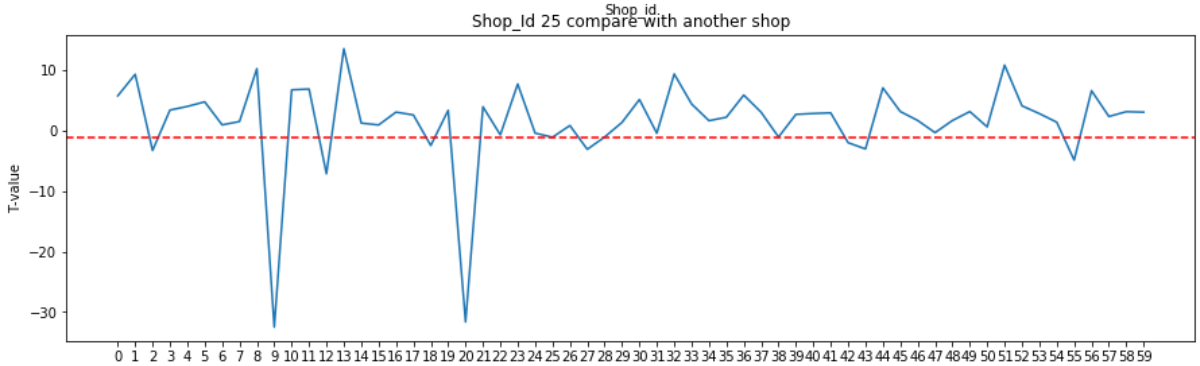

Shop_Id 21 compare with another shop

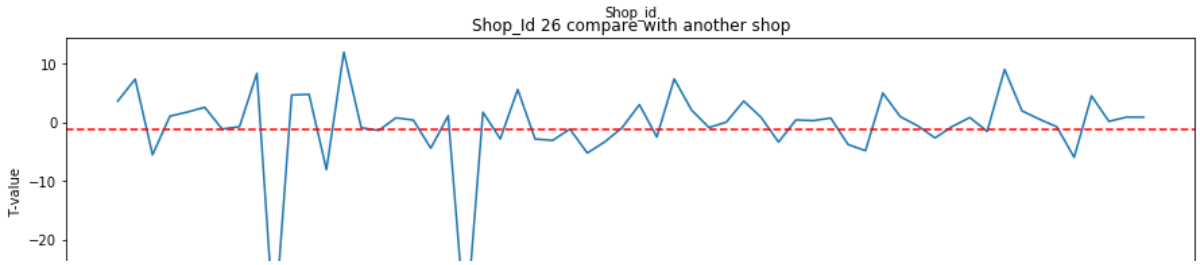

T-value
Shop_id

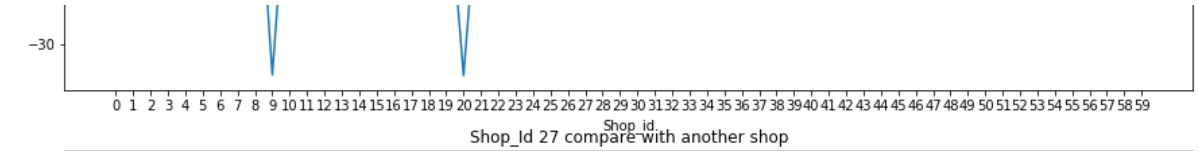

Shop_Id 22 compare with another shop
T-value
Shop_id

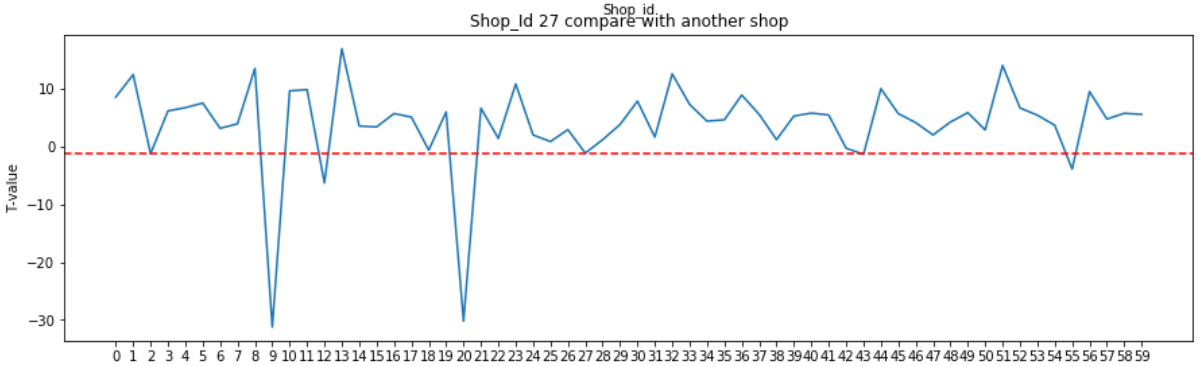

Shop_Id 23 compare with another shop
T-value
Shop_id


Shop_Id 24 compare with another shop
T-value
Shop_id

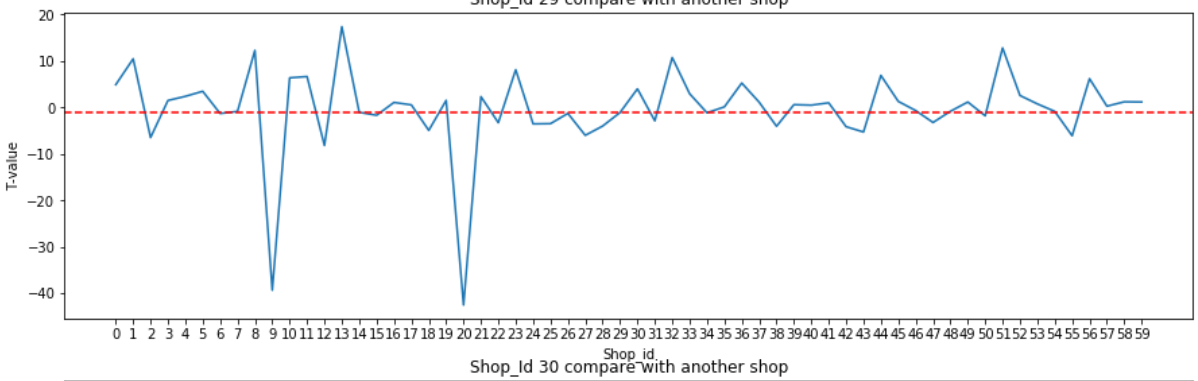

Shop_Id 25 compare with another shop
T-value
Shop_id

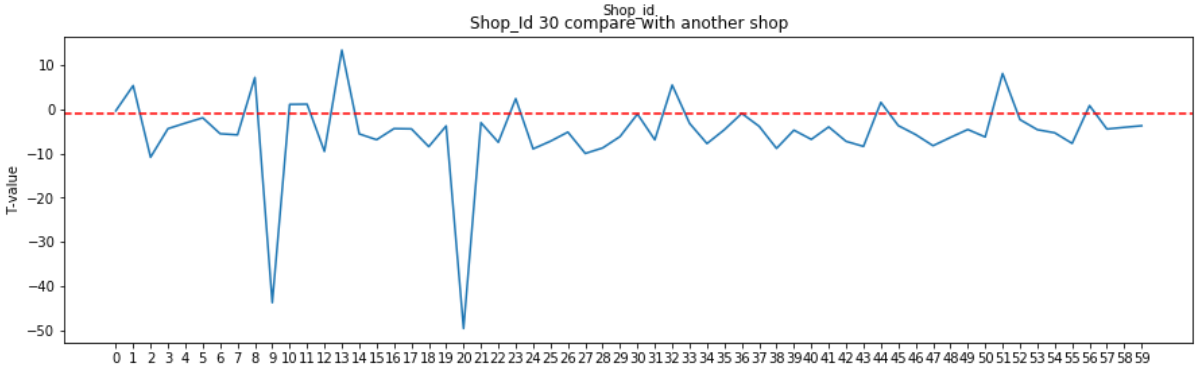

Shop_Id 26 compare with another shop
T-value

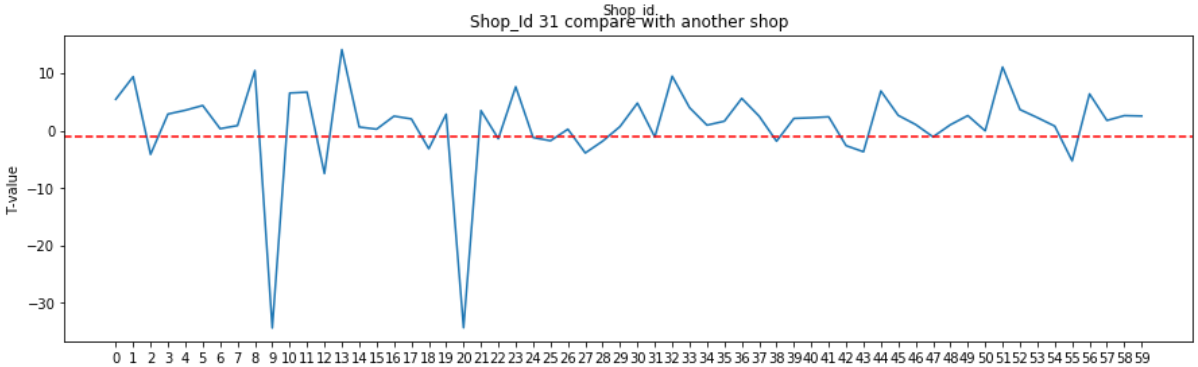

Shop_id

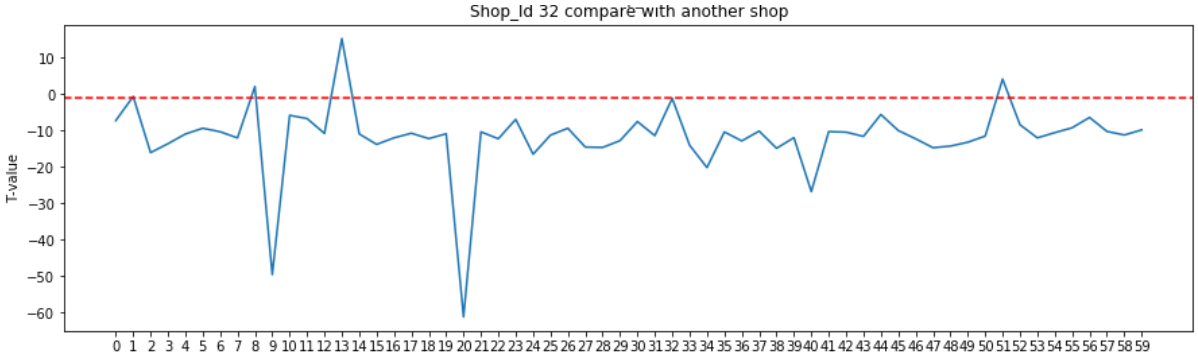

Shop_Id 27 compare with another shop
T-value
Shop_id


Shop_Id 28 compare with another shop
T-value
Shop_id

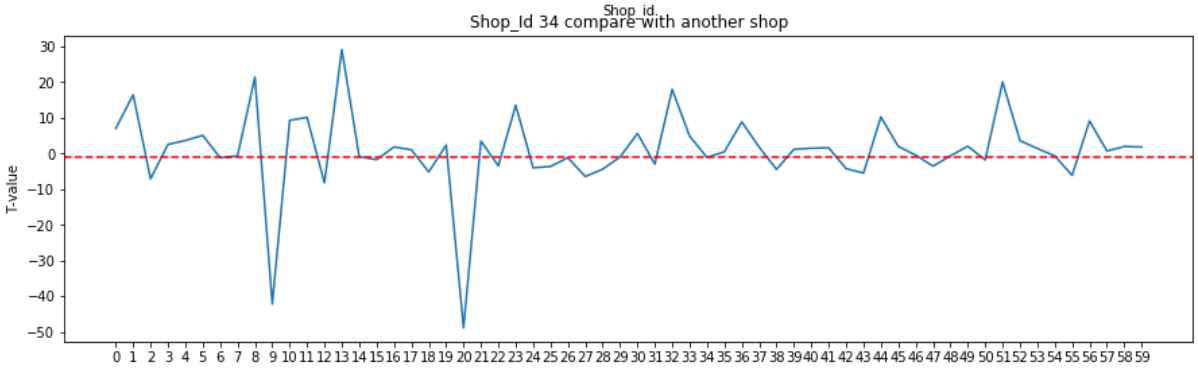

Shop_Id 29 compare with another shop
T-value
Shop_id

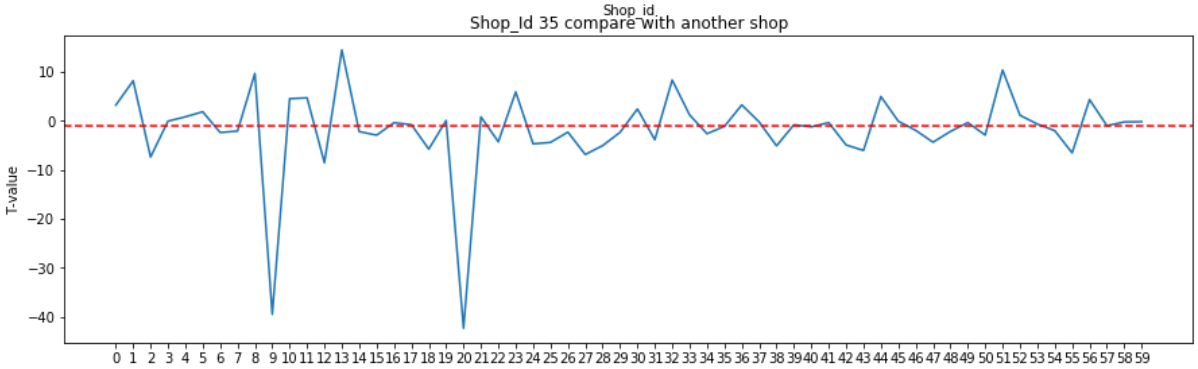

Shop_Id 30 compare with another shop
T-value
Shop_id

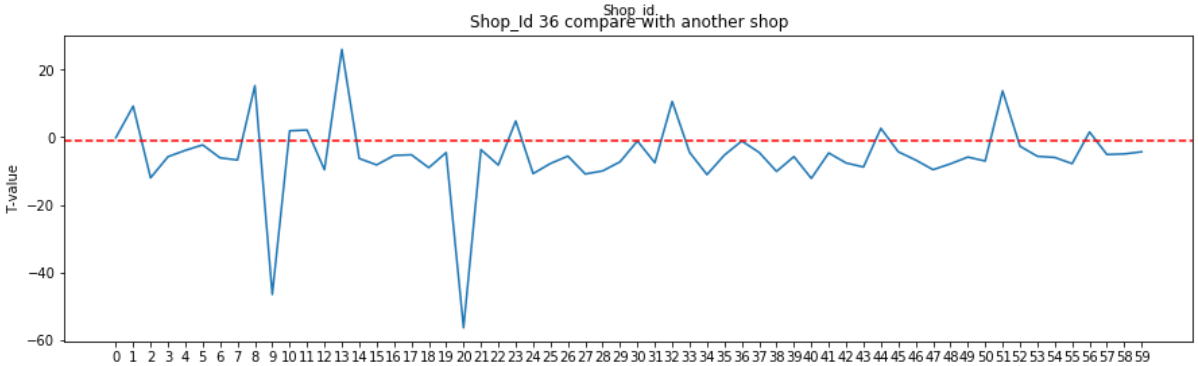

Shop_Id 31 compare with another shop
T-value
Shop_id


Shop_Id 32 compare with another shop
T-value
Shop_id.

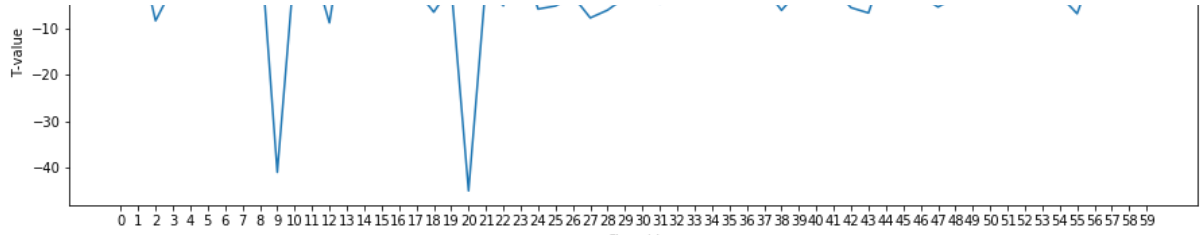

Shop_Id 33 compare with another shop
T-value
Shop_id.

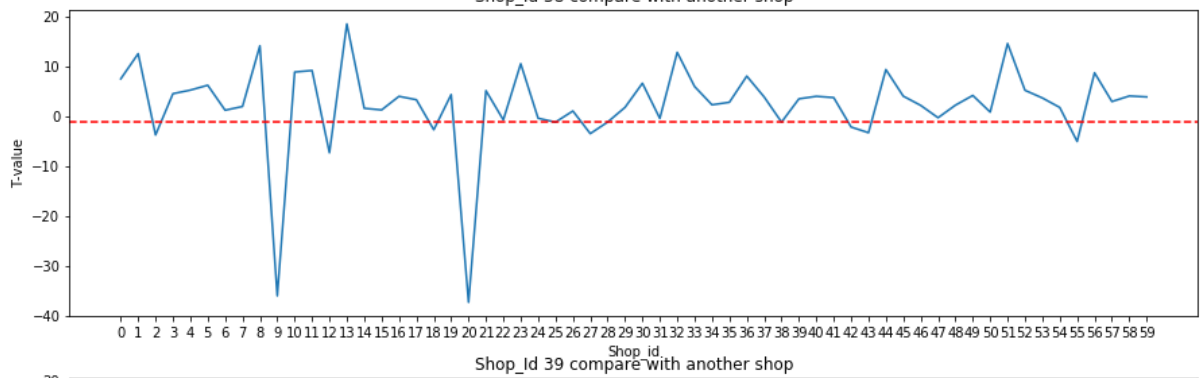

Shop_Id 34 compare with another shop
T-value
Shop_id.

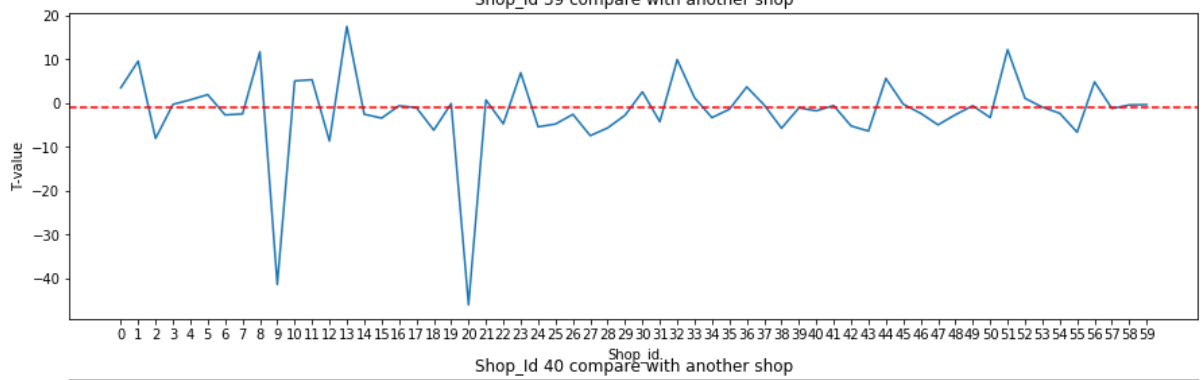

Shop_Id 35 compare with another shop
T-value
Shop_id.

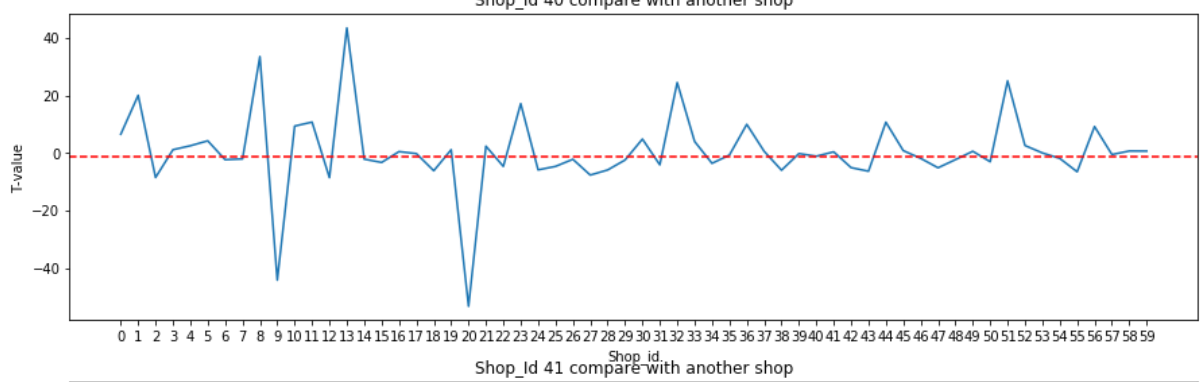

Shop_Id 36 compare with another shop
T-value
Shop_id.


Shop_Id 37 compare with another shop

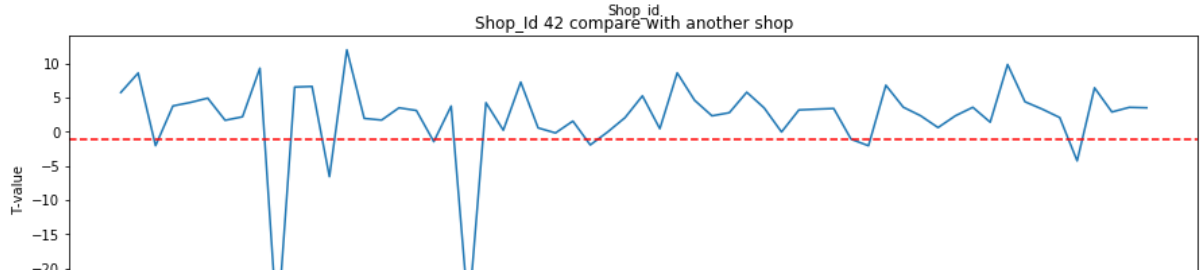

T-value
Shop_id

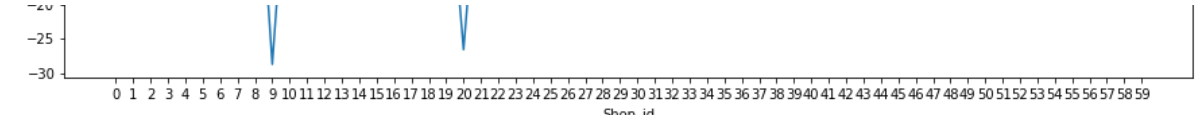

Shop_Id 38 compare with another shop
T-value
Shop_id

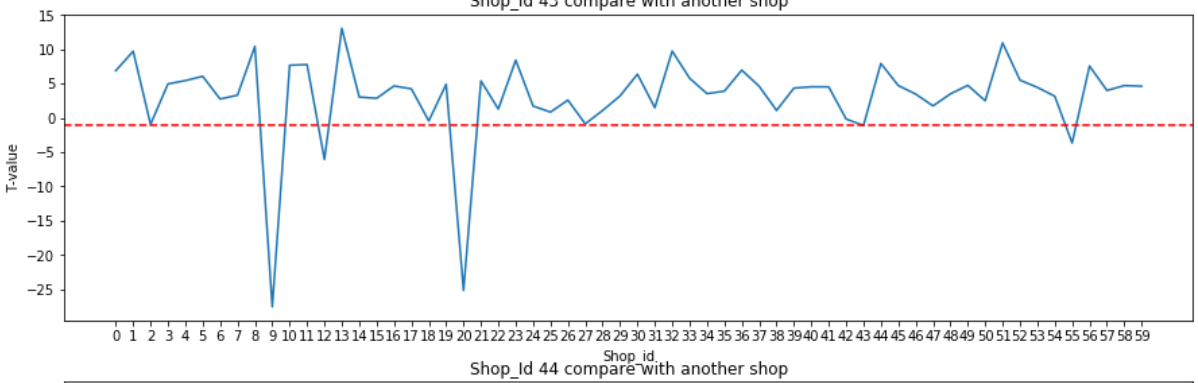

Shop_Id 39 compare with another shop
T-value
Shop_id


Shop_Id 40 compare with another shop
T-value
Shop_id

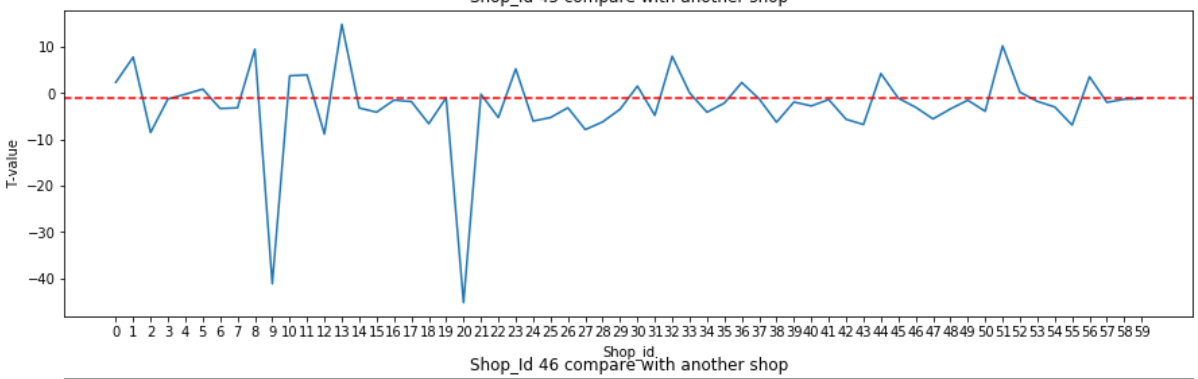

Shop_Id 41 compare with another shop
T-value
Shop_id

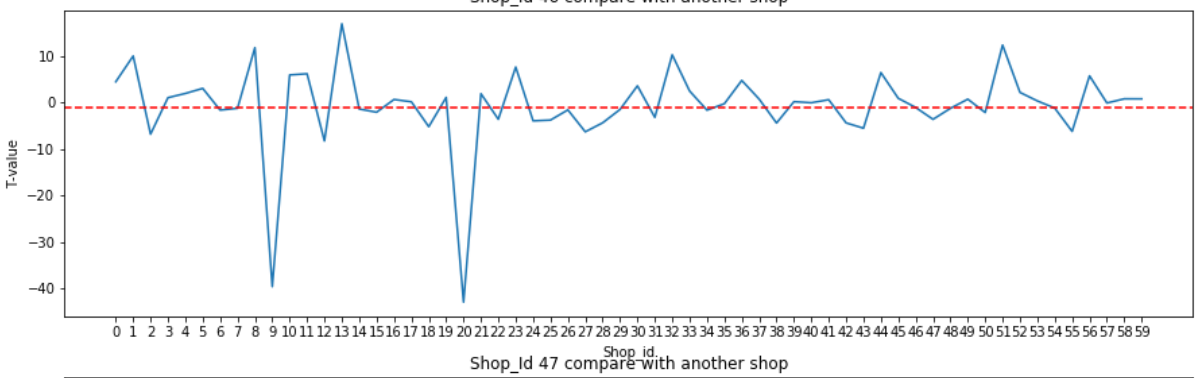

Shop_Id 42 compare with another shop
T-value

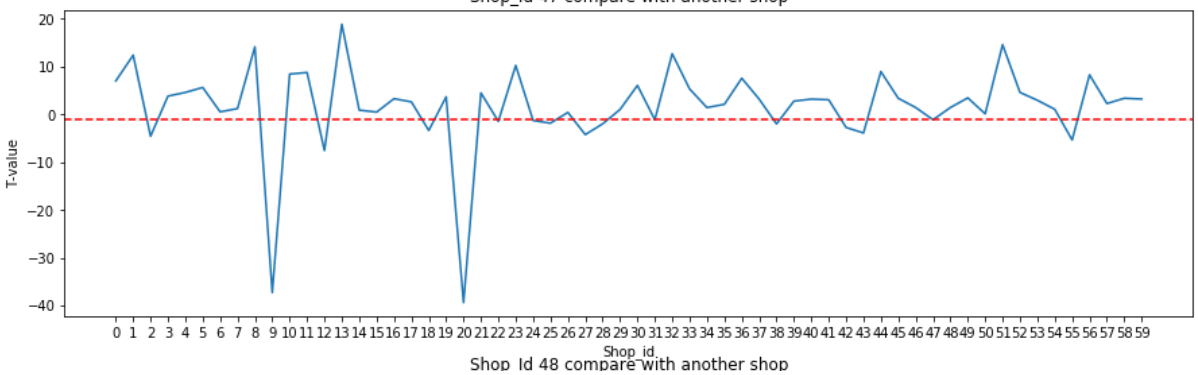

Shop_id

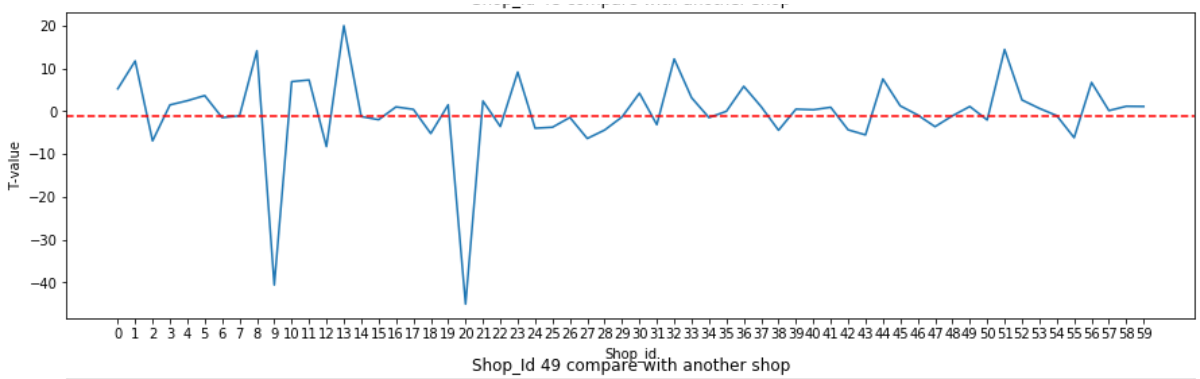

Shop_Id 43 compare with another shop
T-value
Shop_id

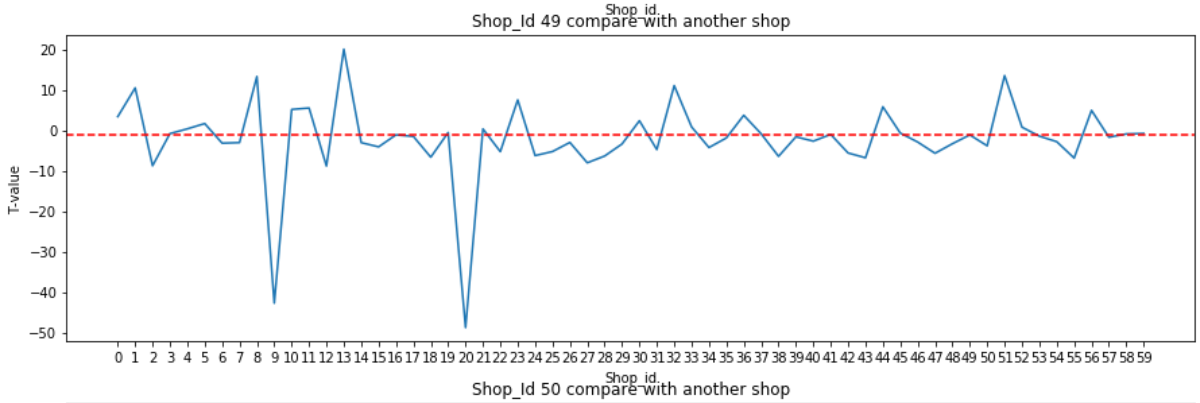

Shop_Id 44 compare with another shop
T-value
Shop_id

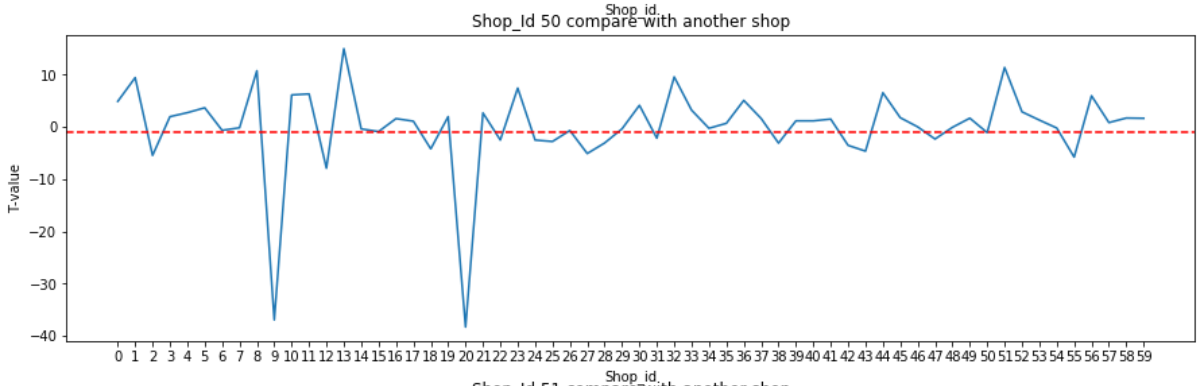

Shop_Id 45 compare with another shop
T-value
Shop_id


Shop_Id 46 compare with another shop
T-value
Shop_id

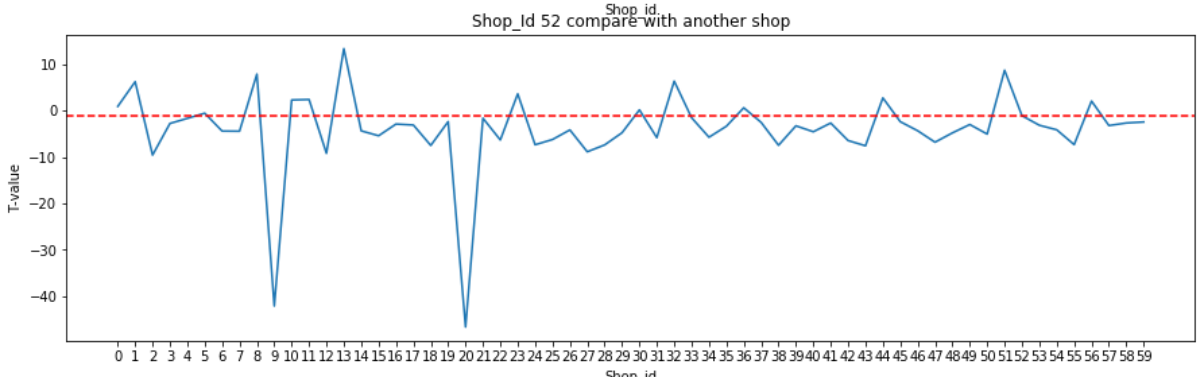

Shop_Id 47 compare with another shop
T-value
Shop_id


T-value
Shop_id

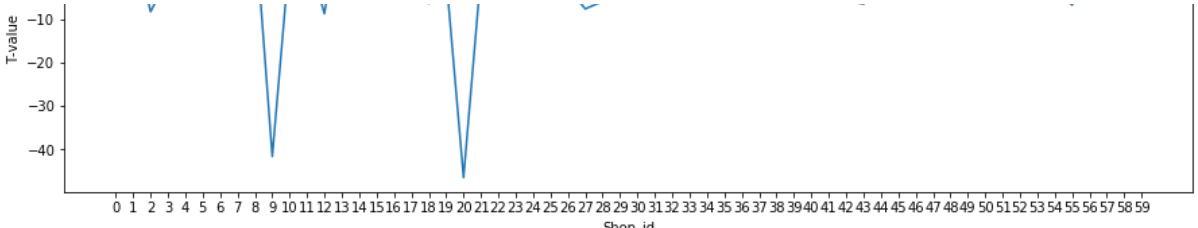

Shop_Id 49 compare with another shop
T-value
Shop_id

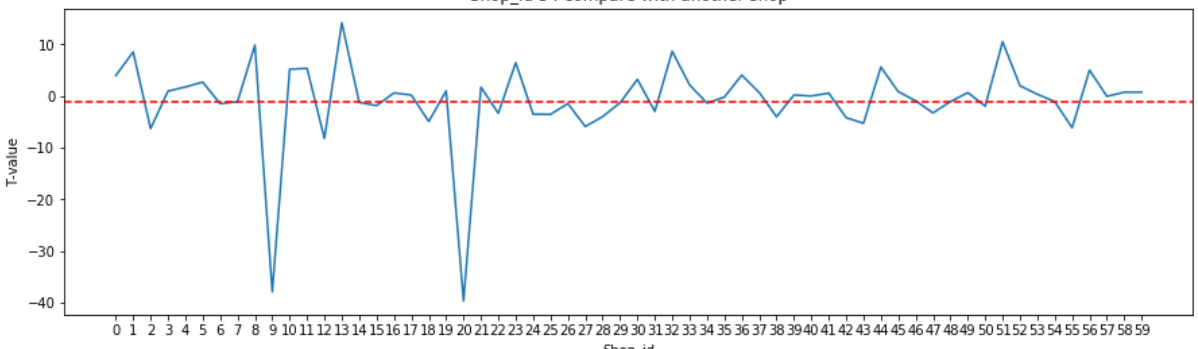

Shop_Id 50 compare with another shop
T-value
Shop_id

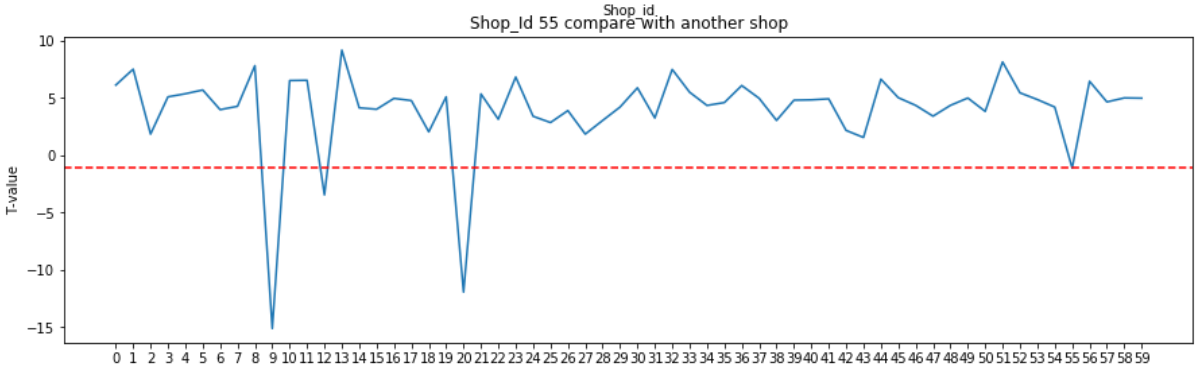

Shop_Id 51 compare with another shop
T-value
Shop_id

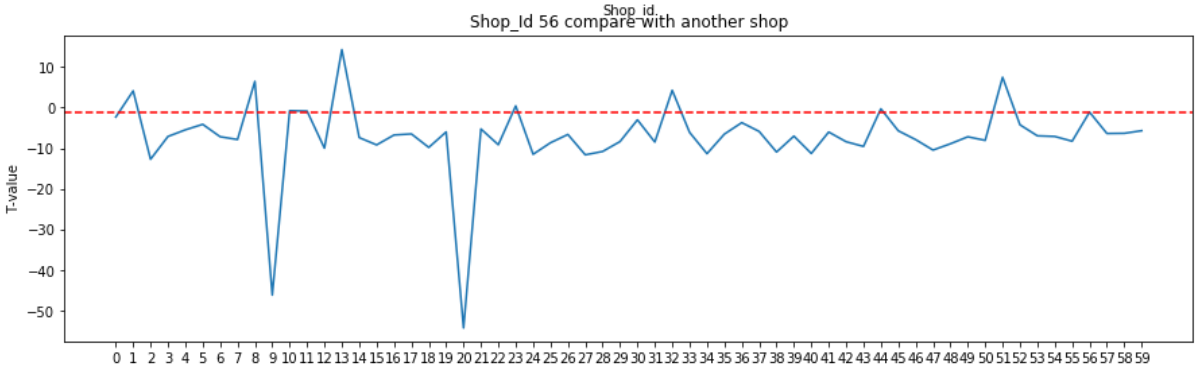

Shop_Id 52 compare with another shop
T-value
Shop_id

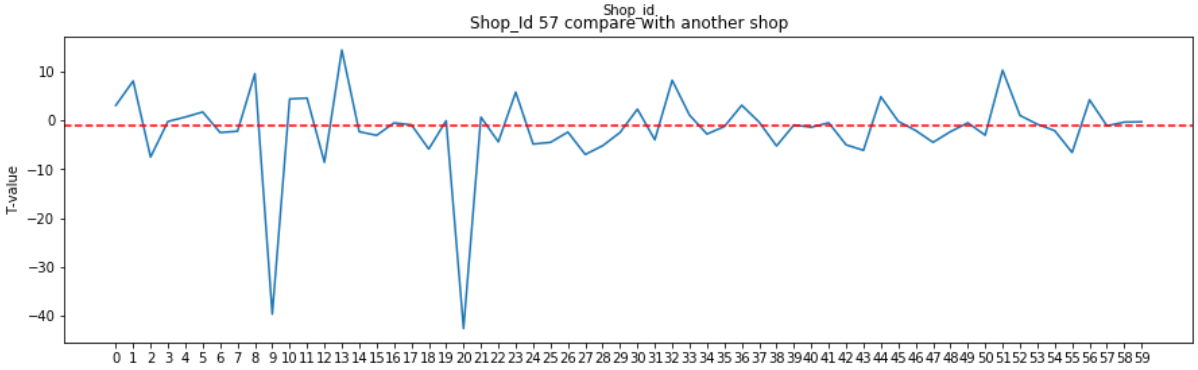

Shop_Id 53 compare with another shop

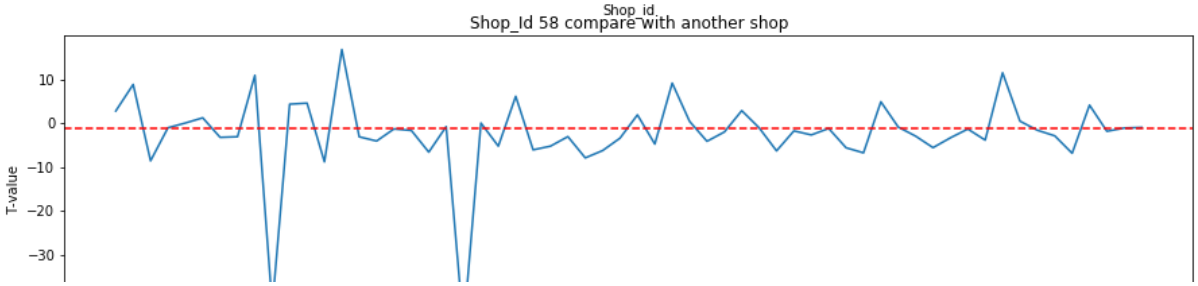

T-value
Shop_id

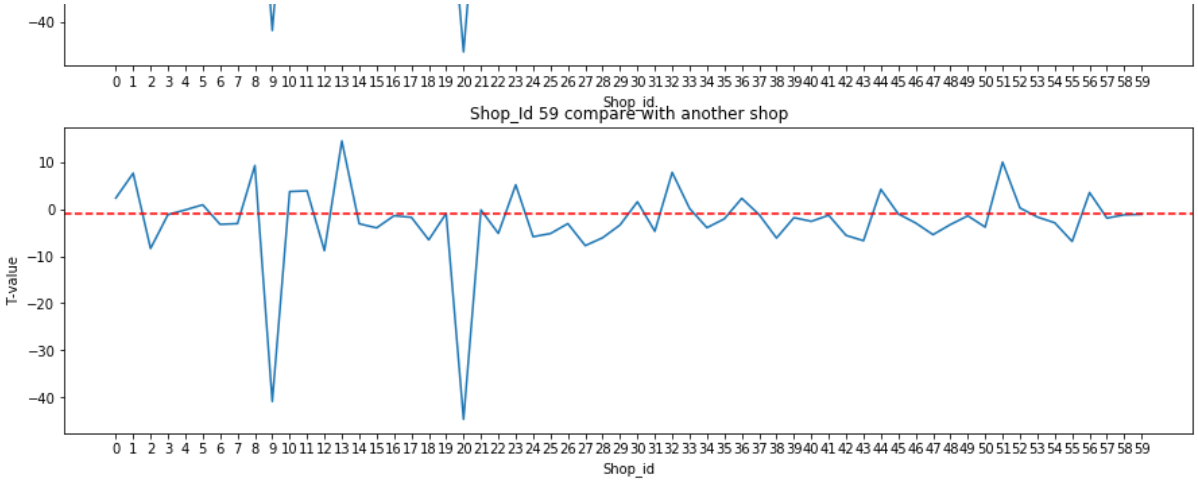

Shop_Id 54 compare with another shop
T-value
Shop_id


Shop_Id 55 compare with another shop
T-value
Shop_id


Shop_Id 56 compare with another shop
T-value
Shop_id


Shop_Id 57 compare with another shop
T-value
Shop_id


Shop_Id 58 compare with another shop
T-value



c. Conclusion

- Looking at the graph you can see the red lines indicate where the 2 shops have the same average.
- And X-axis indicates the shop_id, and each point is the T-value of the shop_id compared to others.
- Shop_id 9, 12,20 is the best three shop that have the average of the sale was higher than another shop

In [ ]:

In [ ]:

CODE

Story telling

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

from ipywidgets import interact, widgets
```

In [137]:
```
#Loading data
data1 = pd.read_csv('data/question1_data.csv').drop('Unnamed: 0', axis =
1)
data2 = pd.read_csv('data/question2_data.csv')
data3 = pd.read_csv('data/question3_data.csv')
data_item_name = pd.read_csv('data/items.csv')
test = pd.read_csv('data/test.csv', index_col= 'ID')
```

In [156]:
```
data1['total_price'] = data1.item_price * data1.item_cnt_day
data1 = pd.merge(data1, data_item_name, how='left')
```

In [139]:
```
#Plot the total month sale for all shop
def plot(shop_id_sub):
    for i in shop_id_sub:
        data = data1_new[data1_new.shop_id == i]
        x = np.arange(0,len(data.date_block_num))
        y = data.item_cnt_day
        _ = plt.plot(x,y)
        _ = plt.xlabel('month')
        _ = plt.ylabel('item sale')
        _ = plt.title('Total Sale by Month of Shop_ID: ' + str(i-9) + '
 to ' +str(i))
        _ = plt.legend(shop_id_sub)
        _ = plt.xticks(np.arange(0, 34, step=1))
```

In [140]:
```
#Empirical cumulative distribution function
def ecdf(data):
    x = sorted(data)
    y = np.arange(1,len(data)+1/len(data))
    return x,y
```

In [149]:
```
#Plot total sale for each month for all shop
def all_shop_month_sale():
    question1_view_month = data1[['date_block_num','item_cnt_day']]
    x1 = question1_view_month.groupby('date_block_num').sum()
    x = x1.item_cnt_day
    x_1,y_1 = ecdf(x)
    quantile = np.percentile(x1, [25,50,75])
    temp_dict = {}
    temp_dict['min'] = [np.min(x1.item_cnt_day)]
    temp_dict['25% quantile'] = [quantile[0]]
    temp_dict['50% quantile'] = [quantile[1]]
    temp_dict['mean'] = [np.mean(x1.item_cnt_day)]
    temp_dict['75% quantile'] = [quantile[2]]
    temp_dict['max'] = [np.max(x1.item_cnt_day)]
    #temp_dict['Mode'] = [stats.mode(data1.item_cnt_day)]
    new_df = pd.DataFrame.from_dict(temp_dict)
    _ = plt.figure(figsize = (20,5))
    _ = plt.plot(x1)
    _ = plt.xticks(np.arange(0,34, step = 1))
    _ = plt.xlabel('Month')
    _ = plt.ylabel('Total sale each month')
    _ = plt.title('Forecasting the Total sale of each month')

    return new_df
```

In [142]:
```
#Plot total sale for each shop
def total_sale_shop():
    question1_view_shop = data1[['shop_id','item_cnt_day']]
    x2 = question1_view_shop.groupby('shop_id').sum()
    _ = plt.figure(figsize = (20,5))
    _ = plt.plot(x2, marker = 'v')
    _ = plt.xticks(np.arange(0,60, step = 1))
    _ = plt.axhline(np.mean(x2.item_cnt_day), color = 'r')
    _ = plt.xlabel('Shop_id')
    _ = plt.ylabel('Total sale each shop')
    _ = plt.title('Forecasting the Total sale of each shop')
    _ = plt.legend(['total sale', 'Mean of all shop'])
```

In [157]:
```
#Plot total sale for each item
def total_sale_item(default = 1000000):
    question1_view_item = data1[['item_id','item_cnt_day']]
    x3 = question1_view_item.groupby('item_id').sum()
    x3 = x3[x3.item_cnt_day < default] # have 1 item them was way large
r, so we just remove it to get better looking graph
    _ = plt.figure(figsize = (20,5))
    _ = plt.plot(x3, marker = 'v')
    #_ = plt.xticks(np.arange(0,60, step = 1))
    _ = plt.axhline(np.mean(x3.item_cnt_day), color = 'r')
    _ = plt.xlabel('Item_id')
    _ = plt.ylabel('Total sale each item')
    _ = plt.title('Forecasting the Total sale of each item')
    _ = plt.legend(['total sale'])
    return 'Mean: ' + str(np.mean(x3.item_cnt_day))

#Plot top 10 items were sold of store
def top10_item_sale_all_shop(default = 10000000):
        question1_view_item = data1[['item_id','item_cnt_day', 'item_nam
e','total_price']]
        x3 = question1_view_item.groupby(['item_id','item_name']).sum()
        x3 = x3[x3.item_cnt_day < default] # have 1 item them was way la
rger, so we just remove it to get better looking graph
        x4 = x3.sort_values(by ='item_cnt_day', ascending = False )
        x4_1 = x4[0:10]
        _ = plt.figure(figsize = (20,5))
        _ = sns.barplot(x = x4_1.index.get_level_values(0), y = x4_1.ite
m_cnt_day, color = 'Blue', order= x4_1.index.get_level_values(0))
        _ = plt.title('TOP 10 ITEM TOTAL SALE')
        return x4_1
```

In [158]:
```
#Plot how much store are sold each month
def Total_money_sale_for_each_month():
    question1_view_sale = data1[['date_block_num', 'total_price']]
    x5 = question1_view_sale.groupby('date_block_num').sum()
    x6 = x5.sort_values(by ='total_price', ascending = False)
    _ = plt.figure(figsize = (20,5))
    _ = sns.barplot(x = x6.index, y = x6.total_price, order= x6.index, c
olor = 'Blue')
    _ = plt.xlabel('Month')
    _ = plt.ylabel('Total sale each month')
    _ = plt.title('TOP 10 Total $$$ sale of each month')
```

In [159]:
```
#Plot total each item sold, for each shop... we use iteractive to create
bar slide and show graph
def plot_shop_item_sale(shop_id):
    data = data1[data1.shop_id == shop_id]
    group_item = data[['item_id','item_cnt_day']].groupby('item_id').sum
()
    #max_item.append(max(group_item.item_cnt_day))
    _ = plt.figure(figsize = (20,5))
    _ = plt.plot(group_item, marker = '.', linestyle = '')
    return group_item[group_item.item_cnt_day == max(group_item.item_cnt
_day)]
```

In [160]:
```
#Get top 5 items were each of each shop. Also use iteractive to create b
ar slide
def top5_item_sale(shop_id):
    data = data1[data1.shop_id == shop_id]
    group_item = data[['item_id','item_cnt_day','item_name','item_price'
]].groupby(['item_name','item_price','item_id'], as_index =False).sum()
    sort_group_item = group_item.sort_values(by = 'item_cnt_day',ascendi
ng = False)
    y = sort_group_item[0:5]
    _ = plt.figure(figsize = (15,5))
    ax = sns.barplot(x ='item_name', y = 'item_cnt_day', data = y, color
='Blue')
    ax.set_xticklabels(y.item_id)
    _ = plt.title('TOP 5 ITEM SALE FOR SHOP_ID: ' + str(shop_id))
   # _ = plt.xticks(sort_group_item.index['item_id'])
    return y
```

In [161]:
```
#Plot group of range price and total sale
item_price_and_item_cnt = data1.loc[:,['item_cnt_day','item_price']]
def group_price_together(dl):
    if dl < 100:
        return '$0 <= Price < $100'
    elif dl< 200:
        return '$100 <= Price < $200 '
    elif dl < 300:
        return '$200 <= Price < $300'
    elif dl< 400:
        return '$300 <= Price < $400'
    elif dl < 500:
        return '$400 <= Price < $500'
    elif dl < 600:
        return '$500 <= Price < $600'
    elif dl< 700:
        return '$600 <= Price < $700'
    elif dl < 800:
        return '$700 <= Price < $800'
    elif dl < 900:
        return '$800 <= Price < $900'
    elif dl< 1000:
        return '$900 <= Price < $1000'
    elif dl< 2000:
        return '$1000 <= Price < $2000'
    else:
        return 'Greater than 2000'
def group_item_price_plot():
    item_price_and_item_cnt['Group_item_price']= item_price_and_item_cnt
.item_price.map(group_price_together)
    item_group_month = item_price_and_item_cnt.groupby(['Group_item_pric
e'], as_index = True).sum().drop(['item_price'],axis = 1)
    _ = plt.figure(figsize = (15,5))
    _ = sns.barplot(x = item_group_month.index, y = item_group_month.ite
m_cnt_day, color = 'Blue')
    _ = plt.xticks(rotation=90)
```

Apply Statistics

In [163]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter

from scipy.stats import ttest_ind_from_stats
from ipywidgets import interact
```

In [164]:
```
data = pd.read_csv('data/question1_data.csv').drop('Unnamed: 0', axis =
1)
data['Total_price'] = data.item_price * data.item_cnt_day
test = pd.read_csv('data/question1_data.csv')
```

In [165]:
```
#Emprical Cumulative density function
def ecdf(data):
    x = sorted(data)
    y = np.arange(1, len(x)+ 1)/len(x)
    return x,y
```

In [166]:
```
#Function for bootstrap replication
def bs_replicate_1d(data,func):
    return func(np.random.choice(data, size=len(data)))
def bs_draw_replicate(data, func, size = 1):
    #initialize array of replicate
    bs_rep = np.empty(size)
    #Genrate replicate
    for i in range(size):
        bs_rep[i]= bs_replicate_1d(data,func)
    return bs_rep
```

In [169]:
```
def null_hypothesis_plot():
    origin = data.Total_price
    np.random.seed(47)
    sampling = bs_draw_replicate(origin, np.mean, 1000)
    mean_sampling = np.mean(sampling)
    x,y = ecdf(sampling)
    _ = plt.figure(figsize = (15,10))
    _ = plt.subplot(2,2,1)
    _ = plt.hist(sampling, bins = 35)
    _ = plt.title('Histogram of Sampling')
    _ = plt.axvline(mean_sampling, c = 'red')
    _ = plt.subplot(2,2,2)
    _ = plt.plot(x,y, marker = '.', linestyle = 'none')
    _ = plt.title('CDF of Sampling')
    _ = plt.axvline(mean_sampling, c = 'red')
```

Interface

In [171]:
```
def Frequentist_test(shop1_id, shop2_id):

    #Extract the 2 vector total_sale for 2 shop
    shop1_data = data.loc[data.shop_id == shop1_id, 'Total_price']
    shop2_data = data.loc[data.shop_id == shop2_id, 'Total_price']

    #Calculation the mean and std of 2 shop
    mean_shop1 = np.mean(shop1_data)
    std_shop1 = np.std(shop1_data)
    mean_shop2 = np.mean(shop2_data)
    std_shop2 = np.std(shop2_data)

    #Do Normal sample over mean and std of 2 shop. with 10000 sample
    np.random.seed(42)
    sample_shop1 = np.random.normal(mean_shop1, std_shop1, 10000)
    sample_shop2 = np.random.normal(mean_shop2, std_shop2, 10000)

    #Get the mean , std and len for t-test
    sample_mean_shop1 = np.mean(sample_shop1)
    sample_std_shop1 = np.std(sample_shop1)
    sample_len_shop1 = len(sample_shop1)
    sample_mean_shop2 = np.mean(sample_shop2)
    sample_std_shop2 = np.std(sample_shop2)
    sample_len_shop2 = len(sample_shop2)

    #Frequentist test from scipy.stats
    sample_test = ttest_ind_from_stats(sample_mean_shop1, sample_std_sho
p1, sample_len_shop1, sample_mean_shop2, sample_std_shop2, sample_len_sh
op2, equal_var= True)

    #Plot the histogram data before and after sampling
    _ = plt.figure(figsize = (15,10))
    _ = plt.subplot(2,2,1)
    _ = plt.hist(np.log(shop1_data), bins = 30)
    _ = plt.title ('shop1_data_hist')
    _ = plt.xlabel('Total sale')
    _ = plt.ylabel('Frequency')
   # _ = plt.xlim(0,20000)

    _ = plt.subplot(2,2,2)
    _ = plt.hist(np.log(shop2_data), bins = 30)
    _ = plt.title ('shop2_data_hist')
    _ = plt.xlabel('Total sale')
    _ = plt.ylabel('Frequency')
   # _ = plt.xlim(0,20000)

    _ = plt.subplot(2,2,3)
    _ = plt.hist(sample_shop1, bins = 30)
    _ = plt.axvline(sample_mean_shop1, c = 'r')
    _ = plt.title ('shop1_sample_hist')
    _ = plt.xlabel('Random Normal Mean')
    _ = plt.ylabel('Frequency')

    _ = plt.subplot(2,2,4)
    _ = plt.hist(sample_shop2, bins = 30)
    _ = plt.axvline(sample_mean_shop2, c = 'r')
```

```
    _ = plt.title ('shop2_sample_hist')
    _ = plt.xlabel('Random Normal Mean')
    _ = plt.ylabel('Frequency')


    print('The mean of shop_id ' + str(shop1_id) +' after sampling is: '
+ str(np.round(sample_mean_shop1,3)))
    print('The std of shop_id  ' + str(shop1_id) +' after sampling is: '
+ str(np.round(sample_std_shop1,3)))
    print('The mean of shop_id ' + str(shop2_id) +' after sampling is: '
+ str(np.round(sample_mean_shop2,3)))
    print('The std of shop_id ' + str(shop2_id)  +' after sampling is: '
+ str(np.round(sample_std_shop2,3)))
    print('T_Value =  ',(np.round(sample_test[0],4)), ' \nIt tell the sh
op_id', shop1_id,'is different than shop_id', shop2_id ,'by', sample_tes
t[0], 'standard deviation of the mean')
    print('P_Value: ' + str(sample_test[1]))

    #print('T_value is telling us is how much is different between of 2
 data shop')

    return sample_test[0]
```

In [172]:
```
def Frequentist_test_tvalue(shop1_id, shop2_id):

    #Extract the 2 vector total_sale for 2 shop
    shop1_data = data.loc[data.shop_id == shop1_id, 'Total_price']
    shop2_data = data.loc[data.shop_id == shop2_id, 'Total_price']

    #Calculation the mean and std of 2 shop
    mean_shop1 = np.mean(shop1_data)
    std_shop1 = np.std(shop1_data)
    mean_shop2 = np.mean(shop2_data)
    std_shop2 = np.std(shop2_data)

    #Do Normal sample over mean and std of 2 shop. with 10000 sample
    np.random.seed(42)
    sample_shop1 = np.random.normal(mean_shop1, std_shop1, 10000)
    sample_shop2 = np.random.normal(mean_shop2, std_shop2, 10000)

    #Get the mean , std and len for t-test
    sample_mean_shop1 = np.mean(sample_shop1)
    sample_std_shop1 = np.std(sample_shop1)
    sample_len_shop1 = len(sample_shop1)
    sample_mean_shop2 = np.mean(sample_shop2)
    sample_std_shop2 = np.std(sample_shop2)
    sample_len_shop2 = len(sample_shop2)

    #Frequentist test from scipy.stats
    sample_test = ttest_ind_from_stats(sample_mean_shop1, sample_std_sho
p1, sample_len_shop1, sample_mean_shop2, sample_std_shop2, sample_len_sh
op2, equal_var= True)

    return sample_test[0]
```

In [173]:
```
#Creating the empty data frame for T_value
T_value_compare = pd.DataFrame(index = np.arange(0,60,1), columns= np.ar
ange(0,60,1))
#Append T-value to empty dataframe
for i in range(0,60):
    for j in range(0,60):
        x = Frequentist_test_tvalue(i,j) #Function return the T-value
        T_value_compare.iloc[i,j] = x #Append to the postion in the data
frame

#Define function plot all the data
def plot_shop_compare(shop_range):
    _ = plt.figure(figsize = (15,300))
    for i in shop_range:
        _ = plt.subplot(len(shop_range),1,i+1)
        _ = plt.plot(T_value_compare.iloc[i,])
        _ = plt.axhline(-1.10562, c= 'r', linestyle = '--')
        _ = plt.title('Shop_Id ' + str(i) + ' compare with another shop'
)
        _ = plt.xlabel('Shop_id')
        _ = plt.ylabel('T-value')
        _ = plt.xticks(np.arange(0,60,1))
```

In [ ]: