Первичный анализ данных

In [ ]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from matplotlib import pyplot, dates
from matplotlib.ticker import FuncFormatter
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn import preprocessing
from sklearn.cluster import KMeans
```

In [ ]:
```python
# загруждаем датасеты
data_march = pd.read_csv("C:\\Users\\marty\\Desktop\\vkus\\1_march.csv", sep = ';')
data_april = pd.read_csv("C:\\Users\\marty\\Desktop\\vkus\\1_april.csv", sep = ';')
data_may = pd.read_csv("C:\\Users\\marty\\Desktop\\vkus\\1_may.csv", sep = ';')
```

In [ ]:
```python
display(data_march.head(3))
display(data_april.head(3))
display(data_may.head(3))
```

In [ ]:
```python
# добавим в таблицы колонку, которая отвечает за размер скидки в рублях для каждого чека
data_march['price_down'] = data_march['Price_retail'] * data_march['Quantity'] - data_march['BasePrice'] * data_march['Quantity']
data_april['price_down'] = data_april['Price_retail'] * data_april['Quantity'] - data_april['BasePrice'] * data_april['Quantity']
data_may['price_down'] = data_may['Price_retail'] * data_may['Quantity'] - data_may['BasePrice'] * data_may['Quantity']
```

In [ ]:
```python
# сгруппируем данные по торговым точкам и чекам так, чтобы какждая строка чека содержала в себе
# количество купленных товаров, сумму, дату, время, тип заказа

# март
data_march_pivot = data_march.groupby(['id_tt_cl', 'CheckUID', 'order_type', 'date_ch', 'time_ch']).agg({'id_tov_cl':'count',
                                           'BaseSum':'sum',
                                           'price_down':'sum'}).sort_values(by=['date_ch', 'time_ch'],
                                                                           ascending=True).reset_index()

# переименуем столбцы
data_march_pivot = data_march_pivot.rename(columns={'id_tov_cl':'count_tov_sales'})

# апрель
data_april_pivot = data_april.groupby(['id_tt_cl', 'CheckUID', 'order_type', 'date_ch', 'time_ch']).agg({'id_tov_cl':'count',
                                           'BaseSum':'sum',
                                           'price_down':'sum'}).sort_values(by=['date_ch', 'time_ch'],
                                                                           ascending=True).reset_index()
# переименуем столбцы
data_april_pivot = data_april_pivot.rename(columns={'id_tov_cl':'count_tov_sales'})


# май
data_may_pivot = data_may.groupby(['id_tt_cl', 'CheckUID', 'order_type', 'date_ch', 'time_ch']).agg({'id_tov_cl':'count',
                                           'BaseSum':'sum',
                                           'price_down':'sum'}).sort_values(by=['date_ch', 'time_ch'],
                                                                           ascending=True).reset_index()
# переименуем столбцы
data_may_pivot = data_may_pivot.rename(columns={'id_tov_cl':'count_tov_sales'})
```

In [ ]:
```python
# проверка
display(data_march_pivot.head(3))
display(data_april_pivot.head(3))
display(data_may_pivot.head(3))
```

In [ ]:
```python
# проверка
display(data_march_pivot.head(3))
display(data_april_pivot.head(3))
display(data_may_pivot.head(3))
```

In [ ]:
```python
# чтобы посчитать процент онлайн и оффлайн продаж, запишем оффлайн продажи как 0, а онлайн продажи как 1
data_march_pivot.loc[data_march_pivot['order_type'] == 'offline', 'order_type'] = 0
data_march_pivot.loc[data_march_pivot['order_type'] == 'online', 'order_type'] = 1

data_april_pivot.loc[data_april_pivot['order_type'] == 'offline', 'order_type'] = 0
data_april_pivot.loc[data_april_pivot['order_type'] == 'online', 'order_type'] = 1

data_may_pivot.loc[data_may_pivot['order_type'] == 'offline', 'order_type'] = 0
data_may_pivot.loc[data_may_pivot['order_type'] == 'online', 'order_type'] = 1
```

In [ ]:
```python
# теперь проанализируем полученные продажи по чекам за март
data_march_tt = data_march_pivot.groupby("id_tt_cl").agg({"CheckUID": 'nunique', 'count_tov_sales':["mean", 'sum'],
                                       'BaseSum':["sum", 'mean'], 'order_type' : "mean",
                                       'price_down': ['sum', 'mean']})

data_march_tt.columns = data_march_tt.columns.droplevel(0)
data_march_tt = data_march_tt.reset_index()

# переименуем колонки
data_march_tt.columns = ["id_tt_cl", 'CheckUID_count', 'count_tov_sales_mean', 'count_tov_sales_sum',
                         "BaseSum_sum", "BaseSum_mean", "percent_of_online_orders",  "price_down_sum",
                         "price_down_mean"]

data_april_tt = data_april_pivot.groupby("id_tt_cl").agg({"CheckUID": 'nunique', 'count_tov_sales':["mean", 'sum'],
                                       'BaseSum':["sum", 'mean'], 'order_type' : "mean",
                                       'price_down': ['sum', 'mean']})
data_april_tt.columns = data_april_tt.columns.droplevel(0)
data_april_tt = data_april_tt.reset_index()

data_april_tt.columns = ["id_tt_cl", 'CheckUID_count', 'count_tov_sales_mean', 'count_tov_sales_sum',
                          "BaseSum_sum", "BaseSum_mean", "percent_of_online_orders",  "price_down_sum",
                         "price_down_mean"]

data_may_tt = data_may_pivot.groupby("id_tt_cl").agg({"CheckUID": 'nunique', 'count_tov_sales':["mean", 'sum'],
                                       'BaseSum':["sum", 'mean'], 'order_type' : "mean",
                                       'price_down': ['sum', 'mean']})

data_may_tt.columns = data_may_tt.columns.droplevel(0)
data_may_tt = data_may_tt.reset_index()

data_may_tt.columns = ["id_tt_cl", 'CheckUID_count', 'count_tov_sales_mean', 'count_tov_sales_sum',
                          "BaseSum_sum", "BaseSum_mean", "percent_of_online_orders",  "price_down_sum",
                         "price_down_mean"]
```

In [ ]:
```python
# проверка
display(data_march_tt.sample(5))
display(data_april_tt.sample(5))
display(data_may_tt.sample(5))
```

создадим графики распределния по полученным показателям

In [ ]:
```python
#общая таблица выручки по месяцам для каждой отдельной торговой точки
data_revenue = data_march_tt[["id_tt_cl", 'BaseSum_sum']].merge(
    right = data_april_tt[["id_tt_cl", 'BaseSum_sum']], on = 'id_tt_cl', how= 'left').merge(
    right = data_may_tt[["id_tt_cl", 'BaseSum_sum']], on = 'id_tt_cl', how= 'left')
data_revenue.columns = ['id_tt_cl', 'march_total_revenue', 'april_total_revenue', 'may_total_revenue']
```

In [ ]:
```python
#Распределение выручки по месяцам
fig = plt.gcf()
data_revenue['march_total_revenue'].hist(bins=15, legend=True)
data_revenue['april_total_revenue'].hist(bins=15, alpha=0.5, legend=True)
data_revenue['may_total_revenue'].hist(bins=15, alpha=0.3, legend=True)
plt.title('Распределение выручки по месяцам')
plt.xlabel('Сумма, руб.')
plt.ylabel('Количество ТТ')
plt.show()
```

Создадим графики распределения по среднему чеку

In [ ]:
```python
#общая таблица среднего чека по месяцам для каждой отдельной торговой точки
data_mean_check = data_march_tt[["id_tt_cl", 'BaseSum_mean']].merge(
    right = data_april_tt[["id_tt_cl", 'BaseSum_mean']], on = 'id_tt_cl', how= 'left').merge(
    right = data_may_tt[["id_tt_cl", 'BaseSum_mean']], on = 'id_tt_cl', how= 'left')
data_mean_check.columns = ['id_tt_cl', 'march_mean_check', 'april_mean_check', 'may_mean_check']
```

In [ ]:
```python
#Распределение по среднему чеку
fig = plt.gcf()
data_mean_check['march_mean_check'].hist(bins=15, legend=True)
data_mean_check['april_mean_check'].hist(bins=15, alpha=0.5, legend=True)
data_mean_check['may_mean_check'].hist(bins=15, alpha=0.3, legend=True)
plt.title('Распределение стоимости среднего чека по месяцам')
plt.xlabel('Сумма, руб.')
plt.ylabel('Количество ТТ')
plt.show()
```

Графики распределения по количеству чеков в ТТ

In [ ]:
```python
#общая таблица количества чеков по месяцам для каждой отдельной торговой точки
data_check_count = data_march_tt[["id_tt_cl", 'CheckUID_count']].merge(
    right = data_april_tt[["id_tt_cl", 'CheckUID_count']], on = 'id_tt_cl', how= 'left').merge(
    right = data_may_tt[["id_tt_cl", 'CheckUID_count']], on = 'id_tt_cl', how= 'left')
data_check_count.columns = ['id_tt_cl', 'march_check_count', 'april_check_count', 'may_check_count']
```

In [ ]:
```python
#Распределение по среднему чеку
fig = plt.gcf()
data_check_count['march_check_count'].hist(bins=15, legend=True)
data_check_count['april_check_count'].hist(bins=15, alpha=0.5, legend=True)
data_check_count['may_check_count'].hist(bins=15, alpha=0.3, legend=True)
plt.title('Распределение по количеству чеков по месяцам')
plt.xlabel('Количество чеков, шт.')
plt.ylabel('Количество ТТ')
plt.show()
```

Среднее количество товаров в чеке

In [ ]:
```python
data_mean_tov_count = data_march_tt[["id_tt_cl", 'count_tov_sales_mean']].merge(
    right = data_april_tt[["id_tt_cl", 'count_tov_sales_mean']], on = 'id_tt_cl', how= 'left').merge(
    right = data_may_tt[["id_tt_cl", 'count_tov_sales_mean']], on = 'id_tt_cl', how= 'left')
data_mean_tov_count.columns = ['id_tt_cl', 'march_check_count', 'april_check_count', 'may_check_count']
```

In [ ]:
```python
#Распределение по среднему количеству товаров в чеке
fig = plt.gcf()
data_mean_tov_count['march_check_count'].hist(bins=15, legend=True)
data_mean_tov_count['april_check_count'].hist(bins=15, alpha=0.5, legend=True)
data_mean_tov_count['may_check_count'].hist(bins=15, alpha=0.3, legend=True)
plt.title('Распределение по среднему количеству товаров в чеке по месяцам')
plt.xlabel('Количество проданных товаров в чеке, шт.')
plt.ylabel('Количество ТТ')
plt.show()
```

Общее количество проданных товаров

In [ ]:
```python
data_count_tov_sales_sum = data_march_tt[["id_tt_cl", 'count_tov_sales_sum']].merge(
    right = data_april_tt[["id_tt_cl", 'count_tov_sales_sum']], on = 'id_tt_cl', how= 'left').merge(
    right = data_may_tt[["id_tt_cl", 'count_tov_sales_sum']], on = 'id_tt_cl', how= 'left')
data_count_tov_sales_sum.columns = ['id_tt_cl', 'march_check_count', 'april_check_count', 'may_check_count']
```

In [ ]:
```python
#Распределение по количеству проданных товаров
fig = plt.gcf()
data_count_tov_sales_sum['march_check_count'].hist(bins=15, legend=True)
data_count_tov_sales_sum['april_check_count'].hist(bins=15, alpha=0.5, legend=True)
data_count_tov_sales_sum['may_check_count'].hist(bins=15, alpha=0.3, legend=True)
plt.title('Общее количество проданных товаров')
plt.xlabel('Количество проданных товаров в ТТ, шт.')
plt.ylabel('Количество ТТ')
plt.show()
```

Количество онлайн заказов по месяцам

In [ ]:
```python
data_online = data_march_tt[["id_tt_cl", 'percent_of_online_orders']].merge(
    right = data_april_tt[["id_tt_cl", 'percent_of_online_orders']], on = 'id_tt_cl', how= 'left').merge(
    right = data_may_tt[["id_tt_cl", 'percent_of_online_orders']], on = 'id_tt_cl', how= 'left')
data_online.columns = ['id_tt_cl', 'march_check_count', 'april_check_count', 'may_check_count']
```

In [ ]:
```python
# Процент онлайн заказов по месяцам
fig = plt.gcf()
data_online['march_check_count'].hist(bins=15, legend=True)
data_online['april_check_count'].hist(bins=15, alpha=0.5, legend=True)
data_online['may_check_count'].hist(bins=15, alpha=0.3, legend=True)
plt.title('Процент онлайн заказов по месяцам')
plt.xlabel('Процент онлайн заказов по месяцам, %%')
plt.ylabel('Количество ТТ')
plt.show()
```

Общая сумма скидок в ТТ

In [ ]:
```python
price_down_sum = data_march_tt[["id_tt_cl", "price_down_sum"]].merge(
    right = data_april_tt[["id_tt_cl", 'price_down_sum']], on = 'id_tt_cl', how= 'left').merge(
    right = data_may_tt[["id_tt_cl", 'price_down_sum']], on = 'id_tt_cl', how= 'left')
price_down_sum.columns = ['id_tt_cl', 'march_check_count', 'april_check_count', 'may_check_count']
```

In [ ]:
```python
# Общая сумма скидок в ТТ
fig = plt.gcf()
price_down_sum['march_check_count'].hist(bins=15, legend=True)
price_down_sum['april_check_count'].hist(bins=15, alpha=0.5, legend=True)
price_down_sum['may_check_count'].hist(bins=15, alpha=0.3, legend=True)
plt.title('Общая сумма скидок в ТТ')
plt.xlabel('Общая сумма скидок в ТТ, руб')
plt.ylabel('Количество ТТ')
plt.show()
```

Средняя скидка в ТТ

In [ ]:
```python
price_down_mean = data_march_tt[["id_tt_cl", "price_down_mean"]].merge(
    right = data_april_tt[["id_tt_cl", 'price_down_mean']], on = 'id_tt_cl', how= 'left').merge(
    right = data_may_tt[["id_tt_cl", 'price_down_mean']], on = 'id_tt_cl', how= 'left')
price_down_mean.columns = ['id_tt_cl', 'march_check_count', 'april_check_count', 'may_check_count']
```

In [ ]:
```python
# Средняя скидка в ТТ
fig = plt.gcf()
price_down_mean['march_check_count'].hist(bins=15, legend=True)
price_down_mean['april_check_count'].hist(bins=15, alpha=0.5, legend=True)
price_down_mean['may_check_count'].hist(bins=15, alpha=0.3, legend=True)
plt.title('Средняя скидка в ТТ')
plt.xlabel('Средняя скидка в ТТ, руб')
plt.ylabel('Количество ТТ')
plt.show()
```

In [ ]:
```python
#вычисляем матрицу корреляций за март
march_cm = data_march_tt.drop('id_tt_cl', inplace= False, axis=1 )
cm = march_cm.corr()
plt.figure(figsize = (9,9))
#нарисуем тепловую карту с подписями для матрицы корреляций
sns.heatmap(cm, annot=True, square=True)
plt.show()
```

In [ ]:
```python
#вычисляем матрицу корреляций за апрель
april_cm = data_april_tt.drop('id_tt_cl', inplace= False, axis=1 )
cm = april_cm.corr()
plt.figure(figsize = (9,9))
#нарисуем тепловую карту с подписями для матрицы корреляций
sns.heatmap(cm, annot=True, square=True)
plt.show()
```

In [ ]:
```python
#вычисляем матрицу корреляций за май
may_cm = data_may_tt.drop('id_tt_cl', inplace= False, axis=1 )
cm = may_cm.corr()
plt.figure(figsize = (9,9))
#нарисуем тепловую карту с подписями для матрицы корреляций
sns.heatmap(cm, annot=True, square=True)
plt.show()
```

Информация по матрице корреляций общей выручки за март с параметрами выручки:

Согласно матрице корреляций на сегодняшний день сильно с выручкой коррелирует:

- Общее количество чеков
- Общее количество проданных товаров
- Общая сумма скидки

Обратно коррелирует с выручкой:

- Средняя сумма скидки в магазине

Не коррелирует или слабо коррелирует с выручкой:

- Среднее количество товаров в чеке
- Средний чек в руб.
- Количество онлайн продаж

Данные корреляции сохраняются стабильными для всех исследуемых месяцев.

In [ ]:
```python
#Сводная таблица с выручкой, расчет изменений

data_revenue['change_from_march_to_april'] = ((data_revenue['april_total_revenue'] -
                                               data_revenue['march_total_revenue']) /
                                               data_revenue['march_total_revenue'] * 100)
data_revenue['change_from_april_to_may'] = ((data_revenue['may_total_revenue'] - data_revenue['april_total_revenue']) / data_revenue['may_total_r
data_revenue['change_from_march_to_may'] = ((data_revenue['may_total_revenue'] - data_revenue['march_total_revenue']) / data_revenue['march_total
```

In [ ]:
```python
#Распределение изменений выручки - динамика выручки
fig = plt.gcf()
fig.set_size_inches(12, 4)
plt.subplot(1, 2, 1)
data_revenue['change_from_march_to_april'].hist(bins=84, legend=True)
data_revenue['change_from_april_to_may'].hist(bins=84, alpha=0.5, legend=True)
plt.title('Распределение изменения выручки март-апрель и апрель май')
plt.xlabel('Изменение, %')
plt.ylabel('Количество')
plt.subplot(1, 2, 2)
data_revenue['change_from_march_to_may'].hist(bins=84, legend=True)
plt.title('Распределение изменения выручки за три месяца')
plt.xlabel('Изменение, %')
plt.ylabel('Количество')
plt.show()
```

In [ ]:
```python
data_revenue['change_from_march_to_may'].quantile([0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95])
```

In [ ]:
```python
data_revenue['change_from_april_to_may'].quantile([0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95])
```

In [ ]:
```python
data_revenue['change_from_march_to_april'].quantile([0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95])
```

Создадим матрицу корреляций динамики и параметров выручки

Матрица динамики март-апрель

In [ ]:
```python
# пришьем к таблице по апрелю динамику выручки в процентах
data_change_from_march_to_april = data_april_tt.merge(data_revenue[['id_tt_cl', 'change_from_march_to_april']],
                                                       on = 'id_tt_cl')
# вычтем из месяца апрель месяц март
data_march_tt["change_from_march_to_april"] = 0
data_change_from_march_to_april = data_change_from_march_to_april.subtract(data_march_tt, fill_value=None)
```

In [ ]:
```python
#вычисляем матрицу корреляций с динамикой за март-апрель
data_change_from_march_to_april_cm = data_change_from_march_to_april.drop('id_tt_cl', inplace= False, axis=1 )
cm = data_change_from_march_to_april_cm.corr()
plt.figure(figsize = (9,9))
#нарисуем тепловую карту с подписями для матрицы корреляций
sns.heatmap(cm, annot=True, square=True)
plt.show()
```

In [ ]:
```python
# уберем лишний столбец из нужной нам таблицы
data_march_tt = data_march_tt.drop("change_from_march_to_april", inplace= False, axis=1 )
# data_march_tt
```

Матрица динамиики апрель-май

In [ ]:
```python
# пришьем к таблице по апрелю динамику выручки в процентах
data_change_from_april_to_may = data_may_tt.merge(data_revenue[['id_tt_cl', 'change_from_april_to_may']],
                                                   on = 'id_tt_cl')
# вычтем из месяца апрель месяц март
data_april_tt["change_from_april_to_may"] = 0
data_change_april_to_may = data_change_from_april_to_may.subtract(data_april_tt, fill_value=None)
```

In [ ]:
```python
#вычисляем матрицу корреляций с динамикой за апрель-май
data_change_april_to_may_cm = data_change_april_to_may.drop('id_tt_cl', inplace= False, axis=1 )
cm = data_change_april_to_may_cm.corr()
plt.figure(figsize = (9,9))
#нарисуем тепловую карту с подписями для матрицы корреляций
sns.heatmap(cm, annot=True, square=True)
plt.show()
```

Информация по динамике выручки: Графики динамики выручки представляют собой нормальное распределение. Мы можем заметить, что с марта по апрель у нас большая часть магазинов (на медианной величине) показала отрицательную динамику выручки, но уже с апреля по мая вернулась к положительной динамике (на медианной величине). Такое распределение может свидетельствовать о значительном факторе нормального (случайного) колебания динамики выручки для многих ТТ. В первой матрице динамика март-апрель не показывает наличие корреляций. Динамика выручки апрель-май отражает общие корреляции, уже выделенные для выручки в общем.

In [ ]:
```python
# удалим лишние столбцы, созданные для того, чтобы рассчитать матрицу корреляций
data_april_tt = data_april_tt.drop('change_from_april_to_may', inplace= False, axis=1)
```

Разделим торговые точки на магазины похожего формата

In [ ]:
```python
data_revenue.columns = ['id_tt', 'march_total_revenue', 'april_total_revenue',
                        'may_total_revenue', 'change_from_march_to_april', 'change_from_april_to_may',
                        'change_from_march_to_may']
```

In [ ]:
```python
data_zao = pd.read_csv("C:\\Users\\marty\\Desktop\\vkus\\zao_info.csv", sep = ';')
```

In [ ]:
```python
data_zao = data_zao.merge(data_revenue, on = 'id_tt')
```

In [ ]:
```python
#data_zao_standard = data_zao.drop(['id_tt', 'adress', 'Shirota', 'city_tt', 'Dolgota'], inplace= False, axis=1 )
```

In [ ]:
```python
#le = preprocessing.LabelEncoder()
#data_zao_standard['Район'] = le.fit_transform(data_zao_standard['Район'])
#data_zao_standard['Hours'] = le.fit_transform(data_zao_standard['Hours'])
#data_zao_standard['format'] = le.fit_transform(data_zao_standard['format'])
```

In [ ]:
```python
# стандартизируем данные
#sc = StandardScaler()
#X_sc = sc.fit_transform(data_zao_standard)
#linked = linkage(X_sc, method = 'ward')
#plt.figure(figsize=(15, 10))
#dendrogram(linked, orientation='top')
#plt.title('Hierarchial clustering for GYM')
#plt.show()
```

In [ ]:
```python
def show_clusters_on_plot(df, x_name, y_name, cluster_name):
    plt.figure(figsize=(5, 5))
    sns.scatterplot(
        df[x_name], df[y_name], hue=df[cluster_name], palette='deep'
    )
    plt.title('{} vs {}'.format(x_name, y_name))
    plt.show()
```

In [ ]:
```python
#km = KMeans(n_clusters = 3, random_state=0)
#labels = km.fit_predict(X_sc)
#data_zao_standard['cluster_km'] = labels
#data_zao_standard.groupby(['cluster_km']).mean()
```

Провести класстеризацию не удалось. И с выручкой, и без выручки класстеризация показывает странное разделение магазинов. Необходимо провести разделение магазинов на типы/форматы вручную. Сделать это лучше исходя из площади.

In [ ]:
```python
data_zao
```

In [ ]:
```python
data_zao['ploshad'].hist(bins = 84)
```

In [ ]:
```python
data_zao['ploshad'].quantile([0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95])
```

In [ ]:
```python
data_zao['ploshad'].min()
```

In [ ]:
```python
data_zao['ploshad'].max()
```

In [ ]:
```python
data_zao.loc[data_zao['ploshad'] > 119.275, 'ploshad_type'] = 'middle'
data_zao.loc[data_zao['ploshad'] <= 119.275, 'ploshad_type'] = 'small'
data_zao.loc[data_zao['ploshad'] >= 200, 'ploshad_type'] = 'big'
```

Найдем магазины в одном районе с одним форматом, рассмотрим их выручку.

In [ ]:
```python
def searching_shop (data, region, ploshad_type):
    result = []
    for index, row in data.iterrows():
        if row['Район'] == region and row['ploshad_type'] == ploshad_type:
            result.append(row['id_tt'])
    if len(set(result)) > 1:
        print('ID магазинов размера', ploshad_type, 'в регионе', region, ':', result)
        display(data_zao.query('id_tt in @result'))
```

In [ ]:
```python
for ploshad_type in data_zao['ploshad_type'].unique() :
    for region in data_zao['Район'].unique():
        searching_shop(data_zao, region, ploshad_type)
```

плохой: 11309, 13342, 11662, 12326, 12481, 11980, 11888, 12004, 12834, 13429, 15970, 12299

хороший: 10778, 11634, 11458, 11673, 14030, 11958, 12084, 12192, 12192, 12938, 10786, 16130

In [ ]:
```python
data_ost = pd.read_csv("C:\\Users\\marty\\Desktop\\vkus\\1_ost.csv", sep = ';')
```

In [ ]:
```python
data_ost
```

Проверка гипотез

Обратите внимание, что первым должно даваться в функцию число, которое отражает id "плохого" магазина с низкой выручкой, второе число отражает id "хорошего" магазина с высокой выручки. Сравнение проводиться по магазинам одинакового формата. Так мы найдем ассортимент, который есть в хороших магазинах, но нет в плохих. Если же сначала в функцию запишем хороший магазин, а затем плохой, найдем, какие товары есть в плохих магазинах, каких нет в хороших.

In [ ]:
```python
data_march.columns = ['CheckUID', 'id_tt', 'id_tov', 'Price_retail', 'Quantity', 'BasePrice', 'BaseSum', 'date_ch', 'time_ch', 'order_type',
                      'price_down']

data_april.columns = ['CheckUID', 'id_tt', 'id_tov', 'Price_retail', 'Quantity', 'BasePrice', 'BaseSum', 'date_ch', 'time_ch', 'order_type',
                      'price_down']

data_may.columns = ['CheckUID', 'id_tt', 'id_tov', 'Price_retail', 'Quantity', 'BasePrice', 'BaseSum', 'date_ch', 'time_ch', 'order_type',
                      'price_down']
```

In [ ]:
```python
def find_difference(id_first_bad, id_second_good):
    bad_one = data_ost[data_ost['id_tt'] == id_first_bad][['id_tt','id_tov','name_tov']]
    good_one = data_ost[data_ost['id_tt'] == id_second_good][['id_tt','id_tov','name_tov']]
    difference = good_one[~good_one['name_tov'].isin(bad_one['name_tov'])]
    difference = difference.drop_duplicates(subset=["name_tov"])
    checks = data_april[data_april['id_tt'] == id_second_good][['id_tov', 'Quantity']]
    checks = checks.merge(difference, on = ['id_tov'], how = 'right')
    ckeck_new = checks.groupby('name_tov').agg({'Quantity':'sum'}).reset_index().sort_values(by = 'Quantity', ascending = False).head(20)
    ckeck_new.plot(x = 'name_tov', y = 'Quantity', kind = 'bar', figsize = (12,6))
```

In [ ]:
```python
#find_difference(10778, 11309)
```

In [ ]:
```python
#find_difference(13342, 11634)
```

In [ ]:
```python
#find_difference(12338, 11777)
```

In [ ]:
```python
#find_difference(11420, 11103)
```

In [ ]:
```python
#find_difference(11662, 11458)
```

In [ ]:
```python
#find_difference(12481, 11783)
```

In [ ]:
```python
#find_difference(12518, 12025)
```

In [ ]:
```python
#find_difference(13316, 11901)
```

In [ ]:
```python
#find_difference(12481, 14030)
```

плохой: 11309, 13342, 11662, 12326, 12481, 11980, 11888, 12004, 12834, 13429, 15970, 12299

хороший: 10778, 11634, 11458, 11673, 14030, 11958, 12084, 12192, 12192, 12938, 10786, 16130

In [ ]:
```python
bad_types = [11309, 13342, 11662, 12326, 12481, 11980, 11888, 12004, 12834, 13429, 15970, 12299]
```

In [ ]:
```python
good_types = [10778, 11634, 11458, 11673, 14030, 11958, 12084, 12192, 12192, 12938, 10786, 16130]
```

Март

In [ ]:
```python
bad_types_stores = data_ost[data_ost['id_tt'].isin(bad_types)][['id_tt','id_tov','name_tov']]
good_types_stores = data_ost[data_ost['id_tt'].isin(good_types)][['id_tt','id_tov','name_tov']]
difference = good_types_stores[~good_types_stores['name_tov'].isin(bad_types_stores['name_tov'])]


checks = data_march[data_march['id_tt'].isin(good_types)][['id_tov', 'Quantity']]
checks = checks.merge(difference, on = ['id_tov'], how = 'right')
check_new = checks.groupby('name_tov').agg({'Quantity':'sum'}).reset_index().sort_values(
    by = 'Quantity', ascending = False)
check_new.head(30).plot(x = 'name_tov', y = 'Quantity', kind = 'bar', figsize = (10,5))
```

Апрель

In [ ]:
```python
bad_types_stores = data_ost[data_ost['id_tt'].isin(bad_types)][['id_tt','id_tov','name_tov']]
good_types_stores = data_ost[data_ost['id_tt'].isin(good_types)][['id_tt','id_tov','name_tov']]
difference = good_types_stores[~good_types_stores['name_tov'].isin(bad_types_stores['name_tov'])]


checks = data_april[data_april['id_tt'].isin(good_types)][['id_tov', 'Quantity']]
checks = checks.merge(difference, on = ['id_tov'], how = 'right')
check_new = checks.groupby('name_tov').agg({'Quantity':'sum'}).reset_index().sort_values(
    by = 'Quantity', ascending = False)
check_new.head(30).plot(x = 'name_tov', y = 'Quantity', kind = 'bar', figsize = (10,5))
```

Май

In [ ]:
```python
bad_types_stores = data_ost[data_ost['id_tt'].isin(bad_types)][['id_tt','id_tov','name_tov']]
good_types_stores = data_ost[data_ost['id_tt'].isin(good_types)][['id_tt','id_tov','name_tov']]
difference = good_types_stores[~good_types_stores['name_tov'].isin(bad_types_stores['name_tov'])]


checks = data_may[data_may['id_tt'].isin(good_types)][['id_tov', 'Quantity']]
checks = checks.merge(difference, on = ['id_tov'], how = 'right')
check_new = checks.groupby('name_tov').agg({'Quantity':'sum'}).reset_index().sort_values(
    by = 'Quantity', ascending = False)
check_new.head(30).plot(x = 'name_tov', y = 'Quantity', kind = 'bar', figsize = (10,5))
```

In [ ]:
```python
difference.head(3)
```

In [ ]:
```python

```

In [ ]:
```python
data_march = pd.read_csv("C:\\Users\\marty\\Desktop\\vkus\\1_march.csv", sep = ';')
```

In [ ]:
```python
projects_for_TT = pd.read_csv("C:\\Users\\marty\\Desktop\\vkus\\projects_for_TT.csv", sep = ';')
```

In [ ]:
```python
bad_stores_projects = projects_for_TT[projects_for_TT['id_TT'].isin(bad_types_stores['id_tt'])]
```

In [ ]:
```python
good_stores_projects = projects_for_TT[projects_for_TT['id_TT'].isin(good_types_stores['id_tt'])]
```

In [ ]:
```python
projects_for_TT.columns = ['id_tt', 'project_name']
projects_for_TT
```

In [ ]:
```python
march_corr_projects = data_zao[['id_tt', 'ploshad', 'march_total_revenue']].merge(projects_for_TT, on = 'id_tt')
```

In [ ]:
```python
march_corr_projects['rub_on_meter'] = march_corr_projects['march_total_revenue']/march_corr_projects['ploshad']
```

In [ ]:
```python
#вычисляем матрицу корреляций с динамикой за март-апрель
#cm = march_corr_projects.corr()
#plt.figure(figsize = (9,9))
#нарисуем тепловую карту с подписями для матрицы корреляций
#sns.heatmap(cm, annot=True, square=True)
#plt.show()
```

In [ ]:
```python
march_corr_projects['coffe'] = np.where(march_corr_projects['project_name'] == 'Кофе с собой', True, False)
march_corr_projects['sokomat'] = np.where(march_corr_projects['project_name'] == 'Сокомат', True, False)
march_corr_projects['bez_upakovky'] = np.where(march_corr_projects['project_name'] == 'Отдел без упаковки', True, False)
march_corr_projects['rybnaya_vitrina'] = np.where(march_corr_projects['project_name'] == 'Рыбная витрина', True, False)
march_corr_projects['kafe-pekarnya'] = np.where(march_corr_projects['project_name'] == 'Кафе-Пекарня', True, False)
march_corr_projects['myasnaya_vitrina'] = np.where(march_corr_projects['project_name'] == 'Мясная витрина', True, False)
```

In [ ]:
```python
march_corr_projects
```

In [ ]:
```python
#вычисляем матрицу корреляций с динамикой за март-апрель
cm = march_corr_projects.corr()
plt.figure(figsize = (9,9))
#нарисуем тепловую карту с подписями для матрицы корреляций
sns.heatmap(cm, annot=True, square=True)
plt.show()
```

In [ ]:
```python
data_march_tt.head(3)
```

In [ ]:
```python
good_types_stores = data_march_tt[data_march_tt['id_tt_cl'].isin(good_types)]
```

In [ ]:
```python
good_types_stores['price_down_mean'].describe()
```

In [ ]:
```python
bad_types_stores = data_march_tt[data_march_tt['id_tt_cl'].isin(bad_types)]
```

In [ ]:
```python
bad_types_stores['price_down_mean'].describe()
```

In [ ]:
```python
good_types_stores['price_down_sum'].describe()
```

In [ ]:
```python
bad_types_stores
```

In [ ]:
```python
good_types_stores
```

In [ ]:
```python
bad_types_stores['price_down_sum'].describe()
```

In [ ]:
```python
good_types_stores_score = good_types_stores[['id_tt_cl', 'BaseSum_sum']]
bad_types_stores_score = bad_types_stores[['id_tt_cl', 'BaseSum_sum']]
```

In [ ]:
```python
pd.set_option("display.max_colwidth", -1)
data_zao[data_zao['id_tt'] == 16130]
```

In [ ]:
```python
good_types_stores_score['score'] = 0
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 10778, 'score'] = 4.5
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 10786, 'score'] = 4.4
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 11458, 'score'] = 4.5
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 11634, 'score'] = np.nan
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 11673, 'score'] = 4.4
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 11958, 'score'] = 4.7
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 12084, 'score'] = 4.8
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 12192, 'score'] = 4.6
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 12938, 'score'] = 4.7
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 14030, 'score'] = np.nan
good_types_stores_score.loc[good_types_stores_score['id_tt_cl'] == 16130, 'score'] = np.nan
```

In [ ]:
```python
pd.set_option("display.max_colwidth", -1)
data_zao[data_zao['id_tt'] == 15970]
```

In [ ]:
```python
bad_types_stores_score['score'] = 0
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 11309, 'score'] = 4.6
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 11662, 'score'] = 4.4
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 11888, 'score'] = 4.5
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 11980, 'score'] = 4.4
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 12004, 'score'] = 4.6
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 12299, 'score'] = 4.5
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 12326, 'score'] = 4.5
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 12481, 'score'] = 4.5
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 12834, 'score'] = 4.7
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 13342, 'score'] = 5
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 13429, 'score'] = 4.6
bad_types_stores_score.loc[bad_types_stores_score['id_tt_cl'] == 15970, 'score'] = 5
```

In [ ]:
```python
bad_types_stores_score
```

In [ ]:
```python
bad_types_stores_score['score'].describe()
```

In [ ]:
```python
good_types_stores_score['score'].describe()
```

In [ ]:
```python
good_types_stores_score
```

In [ ]:
```python
difference.head(3)
```

In [ ]:
```python
good_types_stores_score.head(3)
```

In [ ]:
```python
good_shops_data_may = data_may.loc[data_may['id_tt'].isin(good_types_stores_score['id_tt_cl'])]
good_shops_data_may["part_of_good_tov"] = False
good_shops_data_may.loc[good_shops_data_may['id_tov'].isin(difference['id_tov']), 'part_of_good_tov'] = True
good_shops_data_may.head(3)
```

In [ ]:
```python
good_shops_data_may['BaseSum'].sum()
```

In [ ]:
```python
good_shops_data_may.loc[good_shops_data_may['part_of_good_tov'] == True]['BaseSum'].sum()/good_shops_data_may['BaseSum'].sum()*100
```

In [ ]:
```python
good_shops_data_may.loc[good_shops_data_may['part_of_good_tov'] == True]['BaseSum'].sum()
```

In [ ]:
```python
good_shops_data_may['Quantity'].sum()
```

In [ ]:
```python
good_shops_data_may.loc[good_shops_data_may['part_of_good_tov'] == True]['Quantity'].sum()
```

In [ ]:
```python
good_shops_data_may.loc[good_shops_data_may['part_of_good_tov'] == True]['Quantity'].sum()/good_shops_data_may['Quantity'].sum()*100
```

In [ ]:
```python
good_shops_data_may.loc[good_shops_data_may['part_of_good_tov'] == True].count()
```

In [ ]:
```python
percent_of_check = good_shops_data_may.loc[good_shops_data_may['part_of_good_tov'] == True]
data_may_cheks = data_may.loc[data_may['CheckUID'].isin(percent_of_check['CheckUID'])]
```

In [ ]:
```python
data_may['CheckUID'].count()
```

In [ ]:
```python
data_may_cheks['CheckUID'].count()
```

In [ ]:
```python
data_may_cheks['CheckUID'].count()/data_may['CheckUID'].count()*100
```

In [ ]:
```python
data_may_cheks['BaseSum'].sum()
```

In [ ]:
```python
data_may_cheks['BaseSum'].sum()/data_may['BaseSum'].sum()*100
```