Департамент образования и науки города Москвы Государственное автономное образовательное учреждение высшего образования города Москвы «Московский городской педагогический университет» Институт цифрового образования Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Инструменты для хранения и обработки больших данных

Практическая работа №2

Тема:

Продуктовая аналитика

Выполнила: Шепелева Е. В., группа: АДЭУ-201

Преподаватель: Т. М. Босенко

Москва

2022

Этап №1. Объединение датасетов

[2] 1 сек.

```
import pandas as pd


def convert_csv_to_df(csv_name, source_type):
    """ Converts an NPS CSV into a DataFrame with a column for the source.

    Args:
        csv_name (str): The name of the NPS CSV file.
        source_type (str): The source of the NPS responses.

    Returns:
        A DataFrame with the CSV data and a column, source.
    """
    df = pd.read_csv(csv_name)
    df['source'] = source_type
    return df
```

13 сек.

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Выбрать файлы 2020Q4_...mobile.csv

- **2020Q4_nps_mobile.csv**(text/csv) - 34419 bytes, last modified: 11.02.2023 - 100% done

```
Saving 2020Q4_nps_mobile.csv to 2020Q4_nps_mobile.csv
User uploaded file "2020Q4_nps_mobile.csv" with length 34419 bytes
```

✓ 0 сек.

```
# Test the function on the mobile data:
convert_csv_to_df("2020Q4_nps_mobile.csv", "mobile")
```

	response_date	user_id	nps_rating	source
0	2020-12-29	14178	3	mobile
1	2020-10-29	33221	1	mobile
2	2020-11-01	21127	10	mobile
3	2020-12-07	42894	3	mobile
4	2020-11-26	30501	5	mobile
...	...	...	...	...
1796	2020-12-29	49529	3	mobile
1797	2020-12-24	23671	7	mobile
1798	2020-11-28	39954	7	mobile
1799	2020-12-19	21098	7	mobile
1800	2020-12-23	14919	7	mobile

1801 rows × 4 columns

[5]
```python
def check_csv(csv_name):
    """ Checks if a CSV has three columns: response_date, user_id, nps_rating

    Args:
        csv_name (str): The name of the CSV file.

    Returns:
        Boolean: True if the CSV is valid, False otherwise
    """
    with open(csv_name) as f:
        first_line = f.readline()
        # Return true if the CSV has the three specified columns:
        if first_line == "response_date,user_id,nps_rating\n":
            return True

        return False
```

```python
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Выбрать файлы corrupted.csv

- **corrupted.csv**(text/csv) - 45787 bytes, last modified: 11.02.2023 - 100% done

```
Saving corrupted.csv to corrupted.csv
User uploaded file "corrupted.csv" with length 45787 bytes
```

[7]
```python
# Test the function on a corrupted NPS file:
print(check_csv('corrupted.csv'))
```

```
False
```

✓ 0 сек.

[8]
```
def combine_nps_csvs(csvs_dict):
    # Define combine as an empty DataFrame:
    combined = pd.DataFrame()

    # Iterate over csvs_dict to get the name and source of the CSVs:
    for name, source in csvs_dict.items():
        # Check if the csv is valid:
        if check_csv(name):
            # Convert the CSV using convert_csv_to_df():
            temp = convert_csv_to_df(name, source)
            # Concatenate combined and temp:
            combined = pd.concat([combined, temp])

        # If the file is not valid, print a message with the CSV's name:
        else:
            print(name + " is not a valid file and will not be added.")

    # Return the combined DataFrame
    return combined
```

✓ 15 сек.

[9]
```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Выбрать файлы Число файлов: 4

- **2020Q4_nps_email.csv**(text/csv) - 37531 bytes, last modified: 11.02.2023 - 100% done
- **2020Q4_nps_mobile.csv**(text/csv) - 34419 bytes, last modified: 11.02.2023 - 100% done
- **2020Q4_nps_web.csv**(text/csv) - 44042 bytes, last modified: 11.02.2023 - 100% done
- **corrupted.csv**(text/csv) - 45787 bytes, last modified: 11.02.2023 - 100% done

```
Saving 2020Q4_nps_email.csv to 2020Q4_nps_email.csv
Saving 2020Q4_nps_mobile.csv to 2020Q4_nps_mobile (1).csv
Saving 2020Q4_nps_web.csv to 2020Q4_nps_web.csv
Saving corrupted.csv to corrupted (1).csv
User uploaded file "2020Q4_nps_email.csv" with length 37531 bytes
User uploaded file "2020Q4_nps_mobile.csv" with length 34419 bytes
User uploaded file "2020Q4_nps_web.csv" with length 44042 bytes
User uploaded file "corrupted.csv" with length 45787 bytes
```

0 сек.

```
my_files = {
  "2020Q4_nps_email.csv": "email",
  "2020Q4_nps_mobile.csv": "mobile",
  "2020Q4_nps_web.csv": "web",
  "corrupted.csv": "social_media"
}

# Test the function on the my_files dictionary:
combine_nps_csvs(my_files)
```

```
corrupted.csv is not a valid file and will not be added.
```

	response_date	user_id	nps_rating	source
0	2020-11-06	11037	7	email
1	2020-12-24	34434	9	email
2	2020-12-03	49547	8	email
3	2020-10-04	13821	7	email
4	2020-10-23	29407	9	email
...	...	...	...	...
2285	2020-12-25	10656	8	web
2286	2020-11-07	32918	10	web
2287	2020-10-16	15667	10	web
2288	2020-11-20	47153	7	web
2289	2020-10-17	47071	5	web

6043 rows × 4 columns

✓ 0 сек.

[11]
```
def categorize_nps(x):
    """
    Takes a NPS rating and outputs whether it is a "promoter",
    "passive", "detractor", or "invalid" rating. "invalid" is
    returned when the rating is not between 0-10.

    Args:
        x: The NPS rating

    Returns:
        String: the NPS category or "invalid".
    """
    if (x >= 0 and x <= 6):
        return 'detractor'
    elif (x >= 7 and x <= 8):
        return 'passive'
    elif (x >= 9 and x <= 10):
        return 'promoter'

    return 'invalid'


# Test our function:
categorize_nps(8)
```

```
'passive'
```

```python
def convert_csv_to_df(csv_name, source_type):
    """
    Convert an NPS CSV into a DataFrame with columns for the source and NPS group.

    Args:
        csv_name (str): The name of the NPS CSV file.
        source_type (str): The source of the NPS responses.

    Returns:
         A DataFrame with the CSV data and columns: source and nps_group.
    """
    df = pd.read_csv(csv_name)
    df['source'] = source_type
    # New column nps_group which applies categorize_nps to nps_rating:
    df['nps_group'] = df['nps_rating'].apply(categorize_nps)
    return df


# Test the updated function with mobile data:
convert_csv_to_df("2020Q4_nps_mobile.csv", "mobile")
```

	response_date	user_id	nps_rating	source	nps_group
0	2020-12-29	14178	3	mobile	detractor
1	2020-10-29	33221	1	mobile	detractor
2	2020-11-01	21127	10	mobile	promoter
3	2020-12-07	42894	3	mobile	detractor
4	2020-11-26	30501	5	mobile	detractor

✓ 0 сек.

```
[13] def calculate_nps(dataframe):
         # Calculate the NPS score using the nps_group column
         counts = dataframe['nps_group'].value_counts()
         detract = counts['detractor']
         promo = counts['promoter']

         # Return the NPS Score:
         return ((promo-detract)/ counts.sum()) * 100


     my_files = {
       "2020Q4_nps_email.csv": "email",
       "2020Q4_nps_web.csv": "web",
       "2020Q4_nps_mobile.csv": "mobile",
     }


     # Test the function on the my_files dictionary:
     q4_nps = combine_nps_csvs(my_files)
     calculate_nps(q4_nps)
```

```
9.995035578355122
```

```
def calculate_nps_by_source(dataframe):
    # Group the DataFrame by source and apply calculate_nps():
    return dataframe.groupby(['source']).apply(calculate_nps)


my_files = {
  "2020Q4_nps_email.csv": "email",
  "2020Q4_nps_web.csv": "web",
  "2020Q4_nps_mobile.csv": "mobile",
}


# Test the function on the my_files dictionary:
q4_nps = combine_nps_csvs(my_files)
calculate_nps_by_source(q4_nps)
```

```
source
email     18.596311
mobile   -14.714048
web       22.096070
dtype: float64
```

Ссылка: https://colab.research.google.com/drive/1AbGWGYRw1Wrx6j7X9W-80tTxQXG1WIec?usp=sharing

Этап №2. Графическое оформление полученных результатов NPS методами Matplotlib

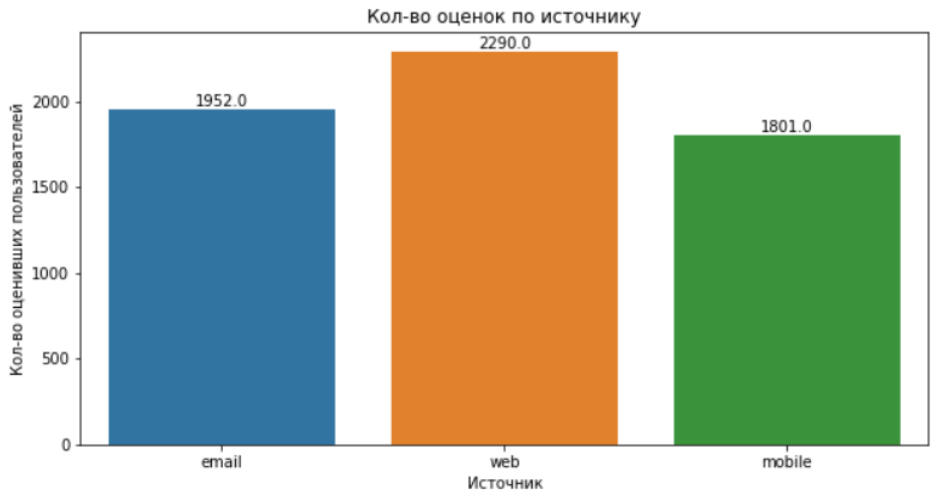
```
[15] import requests
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     import numpy as np
```

```
[16] unique_source = set(q4_nps['source'])
     email = len(q4_nps[q4_nps['source']=='email'])
     web = len(q4_nps[q4_nps['source']=='web'])
     mobile = len(q4_nps[q4_nps['source']=='mobile'])
```

```
df1 = pd.DataFrame({'source': ['email', 'web', 'mobile'], 'count': [email, web, mobile]})
df1
```

	source	count
0	email	1952
1	web	2290
2	mobile	1801

```
plt.figure(figsize=(10,5))
ax=sns.barplot(x='source',y='count', data=df1.head())
ax.set_xticklabels(df1['source'])
ax.set_title("Кол-во оценок по источнику")
ax.set_ylabel("Кол-во оценивших пользователей")
ax.set_xlabel("Источник")
for p in ax.patches:
  ax.annotate(p.get_height(),(p.get_x()+p.get_width()/2, p.get_height()),ha='center',va='bottom')
```




```
[20] df2 = q4_nps.drop(columns=['response_date','user_id','nps_group'])
     df2
```

	nps_rating	source
0	7	email
1	9	email
2	8	email
3	7	email
4	9	email
...	...	...
1796	3	mobile
1797	7	mobile
1798	7	mobile
1799	7	mobile
1800	7	mobile

6043 rows × 2 columns

```
email_0 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 0))])
email_1 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 1))])
email_2 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 2))])
email_3 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 3))])
email_4 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 4))])
email_5 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 5))])
email_6 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 6))])
email_7 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 7))])
email_8 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 8))])
email_9 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 9))])
email_10 = len(df2.loc[((df2['source'] == 'email') & (df2['nps_rating' ] == 10))])
web_0 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 0))])
web_1 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 1))])
web_2 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 2))])
web_3 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 3))])
web_4 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 4))])
web_5 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 5))])
web_6 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 6))])
web_7 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 7))])
web_8 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 8))])
web_9 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 9))])
web_10 = len(df2.loc[((df2['source'] == 'web') & (df2['nps_rating' ] == 10))])
mobile_0 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 0))])
mobile_1 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 1))])
mobile_2 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 2))])
mobile_3 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 3))])
mobile_4 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 4))])
mobile_5 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 5))])
mobile_6 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 6))])
mobile_7 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 7))])
mobile_8 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 8))])
mobile_9 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 9))])
mobile_10 = len(df2.loc[((df2['source'] == 'mobile') & (df2['nps_rating' ] == 10))])
```

```
df3 = pd.DataFrame({
    'nps_rating': ['0','1','2','3','4','5','6','7','8','9','10'],
    'count_email': [email_0, email_1, email_2, email_3, email_4, email_5, email_6, email_7, email_8, email_9, email_10, ],
    'count_web': [web_0, web_1, web_2, web_3, web_4, web_5, web_6, web_7, web_8, web_9, web_10],
    'count_mobile': [mobile_0, mobile_1, mobile_2, mobile_3, mobile_4, mobile_5, mobile_6, mobile_7, mobile_8, mobile_9, mobile_10],
                    })
df3
```

	nps_rating	count_email	count_web	count_mobile
0	0	16	18	18
1	1	36	44	83
2	2	46	57	52
3	3	46	45	87
4	4	63	64	84
5	5	97	85	177
6	6	152	181	141
7	7	581	686	534
8	8	96	110	248
9	9	409	501	210
10	10	410	499	167

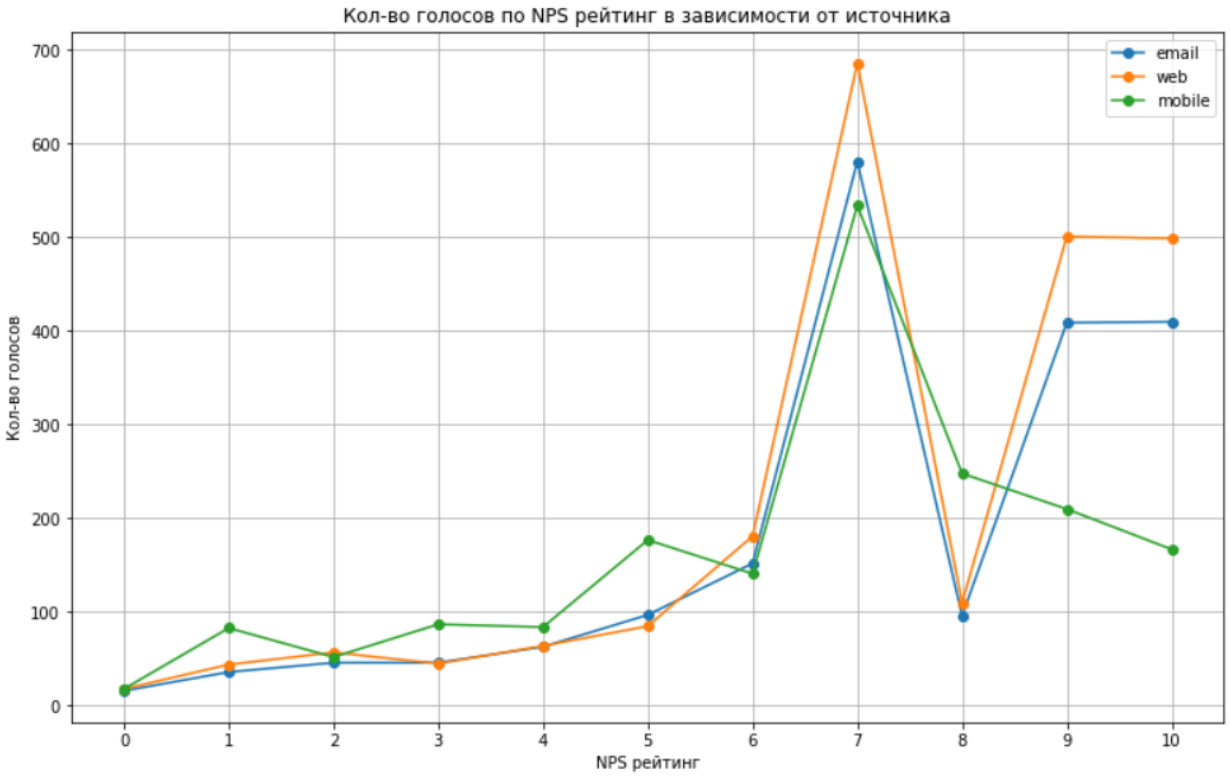
```
[23] x = df3.nps_rating
     y1 = df3.count_email
     y2 = df3.count_web
     y3 = df3.count_mobile
```

0 сек.

```
plt.figure(figsize=(13,8))
plt.title("Кол-во голосов по NPS рейтинг в зависимости от источника")
plt.xlabel("NPS рейтинг")
plt.ylabel("Кол-во голосов")
plt.grid()
plt.plot(x, y1, 'o-', label = 'email')
plt.plot(x, y2, 'o-', label = 'web')
plt.plot(x, y3, 'o-', label = 'mobile')
plt.legend()
```

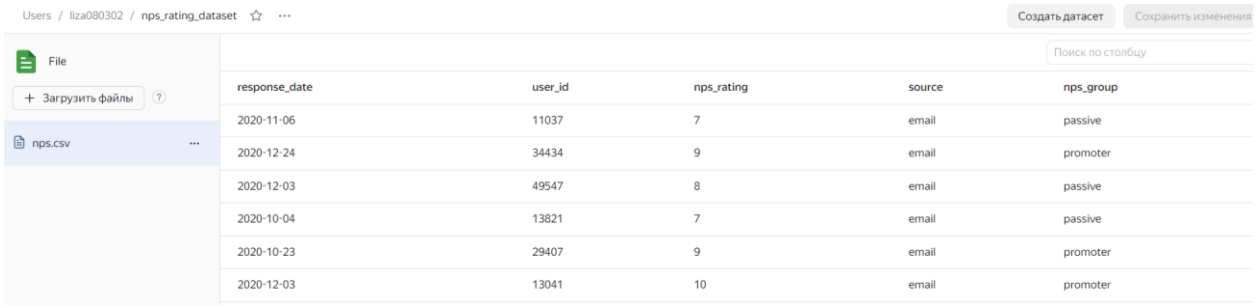
0 сек.

```
<matplotlib.legend.Legend at 0x7f78d158e640>
```

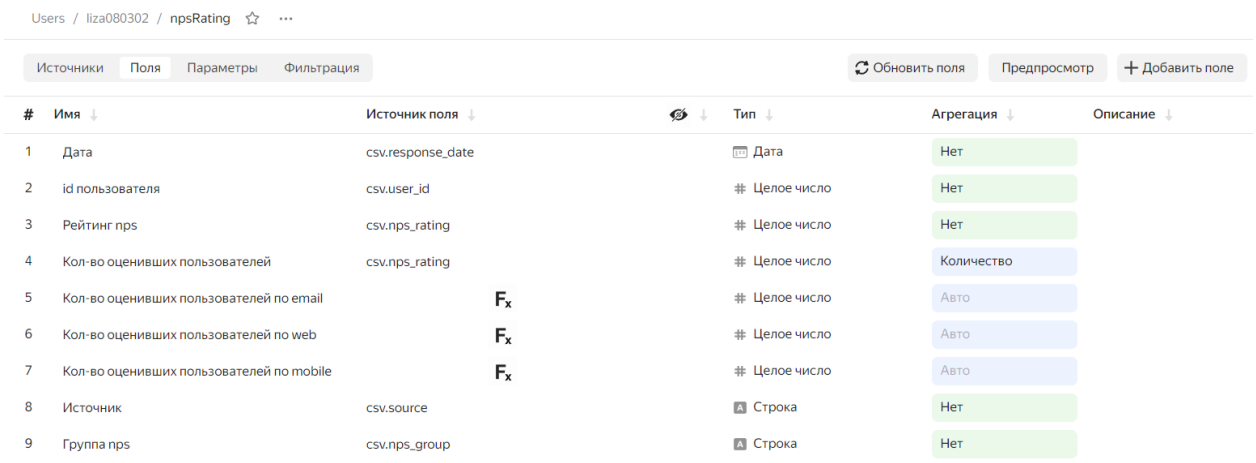


Этап №3. Реализовать выгрузку полученных результатов в csv или xlsx форматы.

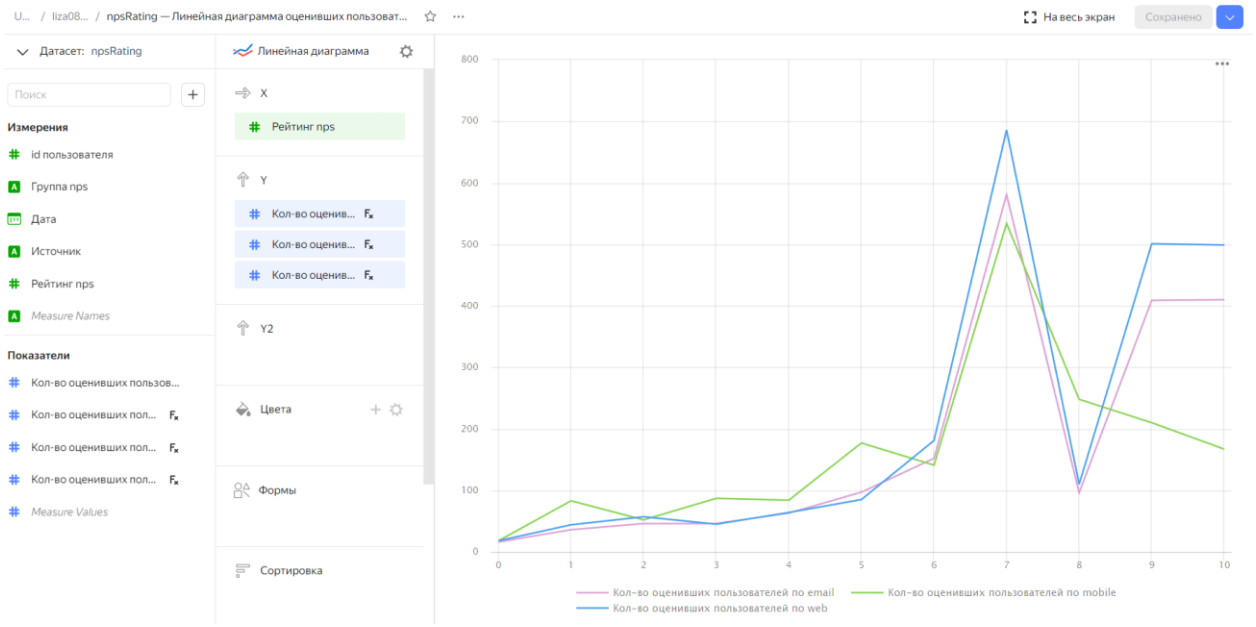
Этап №4. Визуализировать входные и выходные результаты в Yandex DataLens и Google dashboard.

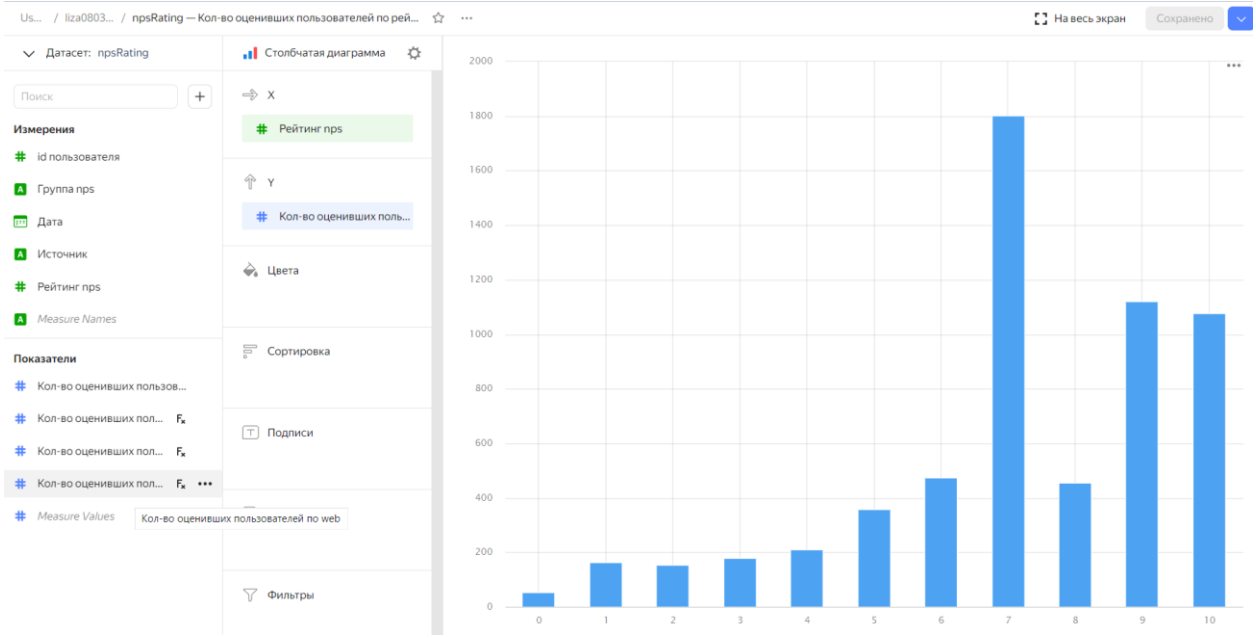


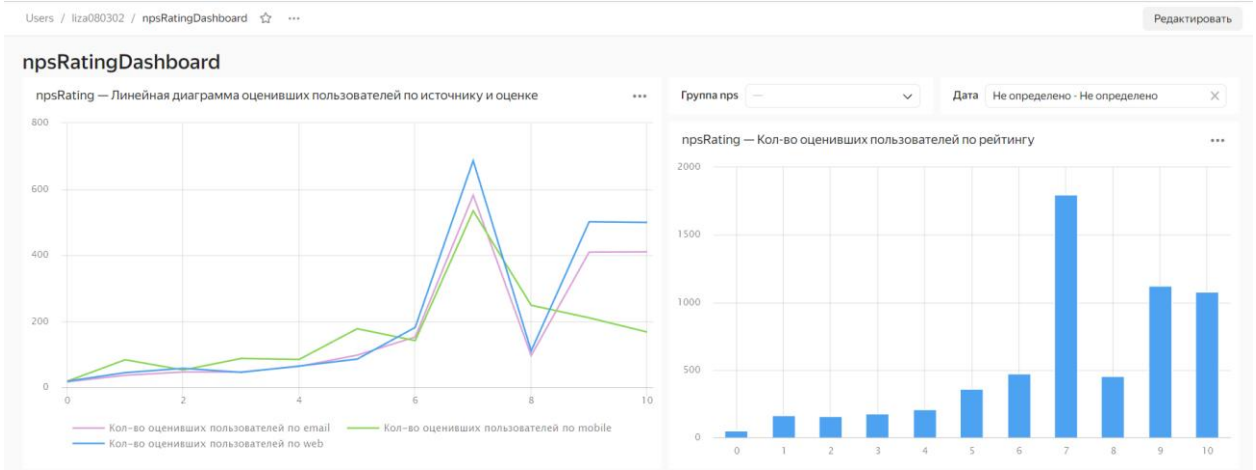
response_date	user_id	nps_rating	source	nps_group
2020-11-06	11037	7	email	passive
2020-12-24	34434	9	email	promoter
2020-12-03	49547	8	email	passive
2020-10-04	13821	7	email	passive
2020-10-23	29407	9	email	promoter
2020-12-03	13041	10	email	promoter

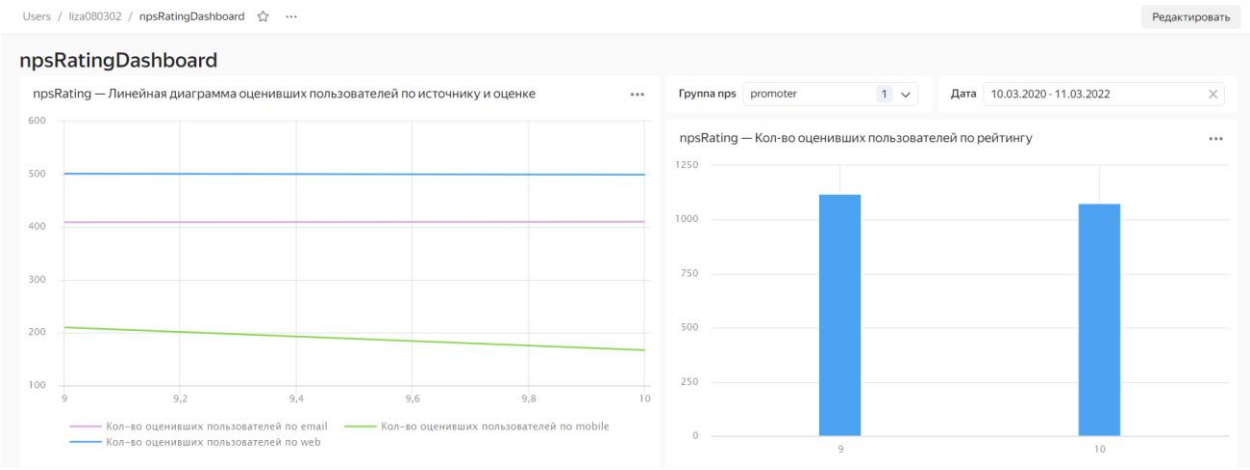


#	Имя	Источник поля	Тип	Агрегация	Описание
1	Дата	csv.response_date	Дата	Нет	
2	id пользователя	csv.user_id	Целое число	Нет	
3	Рейтинг nps	csv.nps_rating	Целое число	Нет	
4	Кол-во оценивших пользователей	csv.nps_rating	Целое число	Количество	
5	Кол-во оценивших пользователей по email	Fx	Целое число	Авто	
6	Кол-во оценивших пользователей по web	Fx	Целое число	Авто	
7	Кол-во оценивших пользователей по mobile	Fx	Целое число	Авто	
8	Источник	csv.source	Строка	Нет	
9	Группа nps	csv.nps_group	Строка	Нет	


U... / liza08... / npsRating — Линейная диаграмма оценивших пользоват...
На весь экран
Сохранено
Датасет: npsRating
Линейная диаграмма
Поиск
Измерения
id пользователя
Группа nps
Дата
Источник
Рейтинг nps
Measure Names
Показатели
Кол-во оценивших пользов...
Кол-во оценивших пол...
Кол-во оценивших пол...
Кол-во оценивших пол...
Measure Values
X
Рейтинг nps
Y
Кол-во оценив...
Кол-во оценив...
Кол-во оценив...
Y2
Цвета
Формы
Сортировка
800
700
600
500
400
300
200
100
0
0 1 2 3 4 5 6 7 8 9 10
Кол-во оценивших пользователей по email
Кол-во оценивших пользователей по mobile
Кол-во оценивших пользователей по web


Us... / liza0803... / npsRating — Кол-во оценивших пользователей по рей...
На весь экран
Сохранено
Датасет: npsRating
Столбчатая диаграмма
Поиск
Измерения
id пользователя
Группа nps
Дата
Источник
Рейтинг nps
Measure Names
Показатели
Кол-во оценивших пользов...
Кол-во оценивших пол...
Кол-во оценивших пол...
Кол-во оценивших пол...
Measure Values
Кол-во оценивших пользователей по web
X
Рейтинг nps
Y
Кол-во оценивших поль...
Цвета
Сортировка
Подписи
Фильтры
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0 1 2 3 4 5 6 7 8 9 10


Users / liza080302 / npsRatingDashboard
Редактировать
npsRatingDashboard
npsRating — Линейная диаграмма оценивших пользователей по источнику и оценке
Группа nps
Дата
Не определено - Не определено
npsRating — Кол-во оценивших пользователей по рейтингу
800
600
400
200
0
0 2 4 6 8 10
Кол-во оценивших пользователей по email
Кол-во оценивших пользователей по mobile
Кол-во оценивших пользователей по web
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10

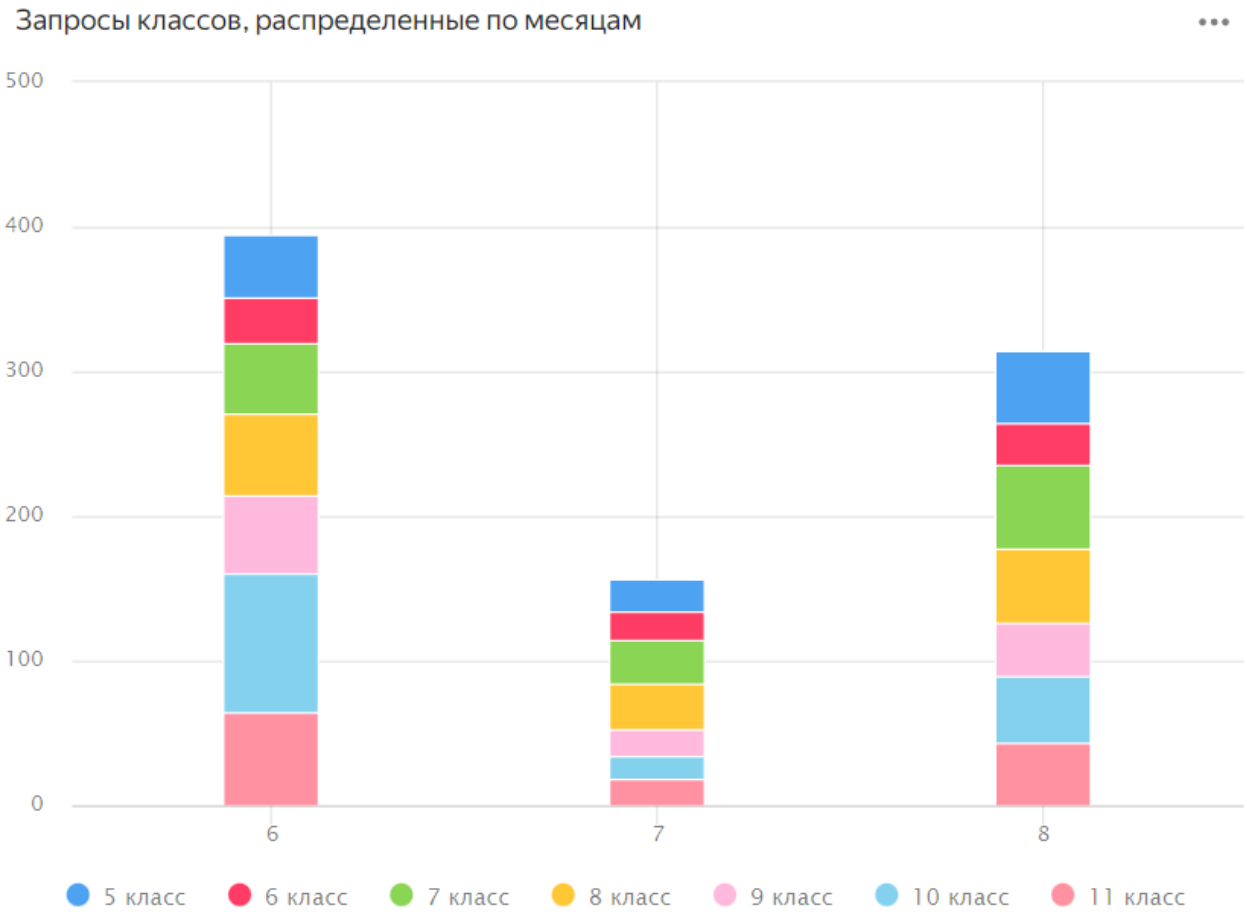


Ссылка: https://datalens.yandex/49dd4r281ld8u

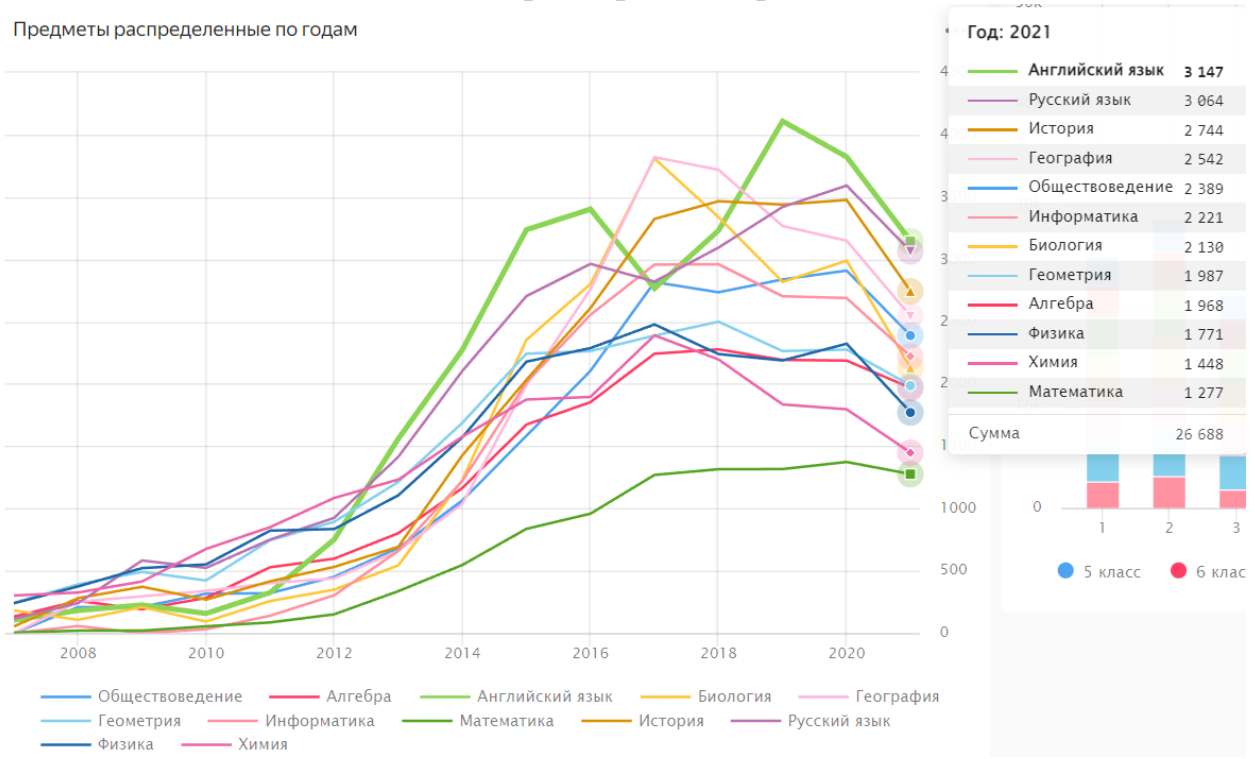
Этап №5. Выводы

Подведу несколько выводов, которые можно сделать по диаграммам:

- Самое наименьше количество запросов поступает в следующие месяцы: июнь, июль и август;



- Английский язык является лидером среди запросов с 2019-2021 года;



- Наибольшее кол-во запросов с 2007-2021 год поступило в 2019 году;
- Три лидирующих класса по кол-ву запросов (от наибольшего к наименьшему): 10, 8 и 7.

Ссылка на дашборд

Ссылка на дашборд: https://datalens.yandex/ff0fmfrorrki5