

ГУАП

КАФЕДРА № 41

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

Старший преподаватель  
\_\_\_\_\_  
должность, уч. степень, звание

\_\_\_\_\_  
подпись, дата

В.В. Боженко  
\_\_\_\_\_  
инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ 2024

по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4217

\_\_\_\_\_  
подпись, дата

Д.М. Никитин  
\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург 2024

1. Цель работы: осуществить предварительную обработку данных csv-файла, выявить и устранить проблемы в этих данных.

2. Вариант №9:

Данные пользовательских сессии магазина:

1. уникальный идентификатор пользователя
2. страна пользователя
3. устройство пользователя
4. идентификатор рекламного источника, из которого пришел пользователь
5. дата и время начала сессии
6. дата и время окончания сессии

Задание 1 : Группировка - region и количество устройств (device ).

Результат должен иметь примерно

следующий вид:

![alt text](Лаб1.jpg "Так")

Задание 2 : Группировка - region и количество устройств (device ).

Создать датафрейм. Переименовать

столбец с количеством в “count”. Отсортировать по убыванию столбца “count”.

Задание 3: Сводная таблица (pivot\_table) - количество пользователей для каждого канала (device).

Отсортировать по убыванию количества.

Задание 4: Сводная таблица (pivot\_table) - уникальное количество пользователей для каждого

устройства (device) - столбцы и региона- строки. Отсортировать по возрастанию столбца region.

3. Ход работы:

Был создан Jupiter-Notebook файл. Далее были записано задание в Markdown поле, его можно увидеть на рисунке 1.

Никтин Дмитрий Вариант задания 9

Данные пользовательских сессии магазина:

1. уникальный идентификатор пользователя
2. страна пользователя
3. устройство пользователя
4. идентификатор рекламного источника, из которого пришел пользователь
5. дата и время начала сессии
6. дата и время окончания сессии

Задание 1 : Группировка - region и количество устройств (device). Результат должен иметь примерно следующий вид:

region	device	
United States	Android	185
	Mac	242
	PC	103
	iPhone	422

Name: index, dtype: int64

Задание 2 : Группировка - region и количество устройств (device). Создать датафрейм. Переименовать столбец с количеством в "count". Отсортировать по убыванию столбца "count".

Задание 3: Сводная таблица (pivot\_table) - количество пользователей для каждого канала (device). Отсортировать по убыванию количества.

Задание 4: Сводная таблица (pivot\_table) - уникальное количество пользователей для каждого устройства (device) - столбцы и региона-строки. Отсортировать по возрастанию столбца region.

Рисунок 1 – Markdown поле с заданием

Далее была открыта база данных и выведены на экран первые 20 строк всё это видно на рисунке 2.

```

1 import pandas as pd
2
3 visits: any = pd.read_csv("visits.csv", sep=";")
4 visits.head(20) # Вывод первых 20 строк в таблице
✓ [2] 21ms

```

	User_Id	Region	Device	Channel	Session
0	9,81E+11	United States	iPhone	organic	01.05.2019
1	2,79E+11	United States	iPhone	organic	01.05.2019
2	5,91E+11	United States	Mac	organic	01.05.2019
3	3,26E+11	United States	Android	TipTop	01.05.2019
4	3,50E+11	United States	Mac	organic	01.05.2019
5	43958116050	United States	Android	organic	01.05.2019
6	1,85E+11	United States	iPhone	organic	01.05.2019
7	1,01E+11	United States	Mac	TipTop	01.05.2019
8	3,70E+11	United States	iPhone	organic	01.05.2019
9	1,42E+11	United States	Mac	FaceBoom	01.05.2019
10	9,24E+11	United States	iPhone	organic	01.05.2019
11	7,75E+11	United States	iPhone	MediaTornado	01.05.2019
12	2,45E+11	United States	iPhone	MediaTornado	01.05.2019
13	1,58E+11	United States	Mac	organic	01.05.2019
14	5,25E+11	United States	Mac	organic	01.05.2019
15	1,36E+11	United States	PC	organic	01.05.2019
16	6,85E+11	United States	iPhone	TipTop	01.05.2019

Рисунок 2 – Вывод двадцати строк (видно только 16)

Далее проведём описание данных и предметной области.

В таблице содержатся столбцы User\_Id – уникальный идентификатор пользователя; Region – страна пользователя; Device – устройство пользователя; Channel – идентификатор рекламного источника, из которого пришел пользователь; Session\_Start – дата и время начала сессии; SESSION\_End – дата и время окончания сессии.

Далее были проведено получение информации из базы данных, она представлена на рисунке 3.

```
> 1 visits.info()      visits
✓ [12] 14ms

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 957 entries, 0 to 956
Data columns (total 6 columns):
#   Column          Non-Null Count  Dtype
---  -
0   User_Id         957 non-null   object
1   Region          956 non-null   object
2   Device          955 non-null   object
3   Channel         957 non-null   object
4   Session_Start   955 non-null   object
5   SESSION_End     955 non-null   object
dtypes: object(6)
memory usage: 45.0+ KB
```

Рисунок 3 – Информация о базе данных

В данных содержатся пропуски от 1 до 2 строк в столбцах Region, Device, Session\_Start и SESSION\_End. Все данные отображаются текстовыми.

Далее была использована команда describe(), это можно увидеть на рисунке 4.

visits.describe()  
✓ [13] 33ms

	User_Id	Region	Device	Channel	Session_Start	SESSION_End
count	957	956	955	957	955	955
unique	608	3	6	4	820	836
top	1,73E+11	United States	iPhone	organic	02.05.2019 20:16	02.05.2019 22:48
freq	6	949	421	612	4	3

Рисунок 4 – Использование команды describe()

Опять же видно пустые ячейки, количество уникальных значений в столбцах, практически все приходы были из США, с iPhone и из источника organic.

Посмотрим названия столбцов отдельно.

```
1 visits.columns
✓ [16] < 10 ms

Index(['User_Id', 'Region', 'Device', 'Channel', 'Session_Start',
      'SESSION_End'],
      dtype='object')
```

Рисунок 5 – Проверка названий столбцов

По моему мнению все названия соответствуют параметрам, кроме SESSION\_End - данный текст не соответствует моим эстетическим соображениям, его нужно переименовать. Этот процесс можно лицезреть на рисунке 6.

```
1 visits = visits.rename
  (columns={"SESSION_End": "Session_End"})
2 visits.columns
✓ [15] < 10 ms

Index(['User_Id', 'Region', 'Device',
      'Channel', 'Session_Start',
      'Session_End'],
      dtype='object')
```

Рисунок 6 – Переименование

Далее было найдено количество пропусков данных в столбцах, это показано на рисунке 7.

```
print(visits.isna().sum())  
✓ [16] < 10 ms  
  
User_Id      0  
Region       1  
Device       2  
Channel      0  
Session_Start 2  
Session_End  2  
dtype: int64
```

Рисунок 7 – Поиск пропусков

Был выбран вариант заполнения пропуска «Region» на «United States», заполнение на рисунке 8.

```
1 visits["Region"] = visits['Region'].fillna  
  ('United States')  
2 print(visits.isna().sum())  
✓ [25] < 10 ms  
  
User_Id      0  
Region       0  
Device       2  
Channel      0  
Session_Start 2  
Session_End  2  
dtype: int64
```

Рисунок 8 – Заполнение пропуска

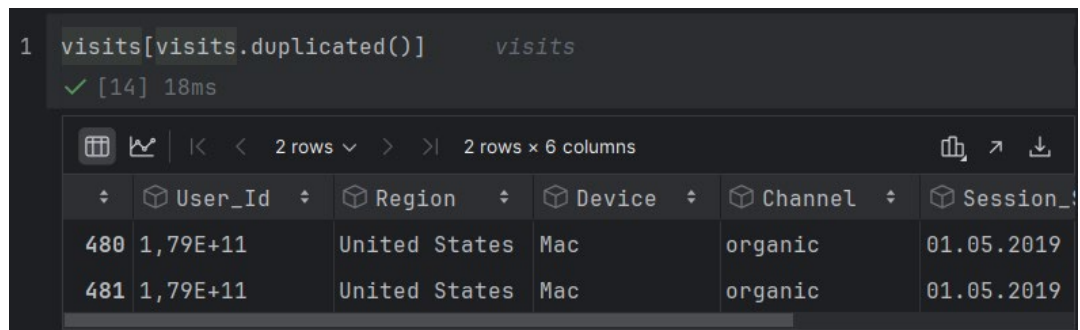
Было принято решение удалить остальные записи, так как точность заполнения была бы значительно меньше. Удаление строк на рисунке 9.

```
1 visits = visits.dropna()      visits  
2 print(visits.isna().sum())  
3 visits.describe()  
✓ [29] 21ms  
  
User_Id      0  
Region       0  
Device       0  
Channel      0  
Session_Start 0  
Session_End  0  
dtype: int64
```

	User_Id	Region	Device	Channel	Session_Start
count	954	954	954	954	954
unique	607	3	6	4	819
top	1,73E+11	United States	iPhone	organic	02.05.2017
freq	6	947	420	610	4

Рисунок 9 – Удаление и результат удаления пропусков

Теперь данные целостные, нужно только проверить на наличие явных и неявных дубликатов. Удалось обнаружить 2 явных дубликата, они показаны на рисунке 10. Данные строки полностью повторяют друг друга.



```
1 visits[visits.duplicated()]    visits
✓ [14] 18ms
```

	User_Id	Region	Device	Channel	Session_S
480	1,79E+11	United States	Mac	organic	01.05.2019
481	1,79E+11	United States	Mac	organic	01.05.2019

Рисунок 10 – Явные дубликаты

Все явные дубликаты были удалены. Стоит заметить, что были удалены обе строки, а не одна из них, так как количество строк уменьшилось на 2. Удаление можно увидеть на рисунке 11.



```
1 visits = visits.drop_duplicates().reset_index(drop=True)    visits
2 print(visits.shape[0])
3 visits[visits.duplicated()]
✓ [22] 13ms
```

952

User_Id	Region	Device	Channel	Session_S
---------	--------	--------	---------	-----------

Рисунок 11 – Удаление и результат удаления явных дубликатов

Приступим к удалению неявных дубликатов. Их оказалось довольно много. Результат на картинке 12.

```
1 print(visits['Region'].value_counts(), end='\n\n') visits
2 print(visits['Device'].value_counts(), end='\n\n')
3 print(visits['Channel'].value_counts(), end='\n\n')
4 print(visits["User_Id"].nunique())
✓ [32] < 10 ms

Region
United States    945
Russia           6
USA              1
Name: count, dtype: int64

Device
iPhone    420
Mac       241
Android   185
PC        103
IPHONE     2
MAC        1
Name: count, dtype: int64

Channel
organic    608
FaceBoom   149
TipTop     142
MediaTornado  53
Name: count, dtype: int64

607
```

Рисунок 12 – Уникальные значения и количество User\_Id

Приступим к удалению неявных дубликатов с помощью переименования. А после проведём проверку, не появились ли явные дубликаты после переименования. Явные и неявные дубликаты удалены.

```
1 visits["Region"] = visits['Region'].replace("USA", "United States")
2 print(visits['Region'].unique())
3
4
5 visits['Device'] = visits['Device'].replace("IPHONE", "iPhone")
6 visits['Device'] = visits['Device'].replace("MAC", "Mac")
7 print(visits['Device'].unique())
8
9 visits[visits.duplicated()]
✓ [37] 12ms

['United States' 'Russia']
['iPhone' 'Mac' 'Android' 'PC']
```

User_Id	Region	Device	Channel	Session_!
---------	--------	--------	---------	-----------

Рисунок 13 – Удаление неявных дубликатов



Далее проверим типы данных. Первым делом переведём User\_Id в Int64. Процесс на рисунке 14.

```
1 if visits['User_Id'].dtype == 'object': visits
2     visits['User_Id'] = visits['User_Id'].str.replace(',', '.')
3
4 visits['User_Id'] = pd.to_numeric(visits['User_Id'], errors='raise')
5 visits['User_Id'] = visits['User_Id'].astype('int64')
6 print(visits['User_Id'].dtype)
7
✓ [59] < 10 ms
int64
```

Рисунок 14 – Перевод значений в Int64

Далее Session\_Start и Session\_End были переведены в datetime64 формат. На этом изменение типов было завершено. Результат на рисунке 15.

```
1 columns = ['Session_Start', 'Session_End']
2 for column in columns:
3     visits[column] = pd.to_datetime(visits[column], format='%d.%m.%Y
4     %H:%M')
5     print(visits[column].dtype)
6 visits.dtypes
✓ [18] 14ms
datetime64[ns]
datetime64[ns]
```

	<unnamed>
User_Id	int64
Region	object
Device	object
Channel	object
Session_Start	datetime64[ns]
Session_End	datetime64[ns]

Рисунок 15 – Форматы данных после изменения и изменения данных на формат datetime64

Приступим далее к созданию сводных таблиц и группировок.

Сначала проведём группировку - Region и количество устройств (Device). Группировка показана на рисунке 16.

```
1 visits.groupby(['Region', 'Device'])['Device'].count()
✓ [32] < 10 ms
```

Region	Device	
Russia	Android	1
	Mac	2
	iPhone	3
United States	Android	184
	Mac	240
	PC	103
	iPhone	419

Name: Device, dtype: int64

Рисунок 16 – Группировка из задания 1

Далее создаётся сгруппированный датафрейм, где добавляется отдельный столбец Count, в котором пишется количество устройств из страны, отсортированный по столбцу Count по убыванию. Результат можно увидеть на рисунке 17.

```
1 group_1: pd.DataFrame = visits.groupby(['Region', 'Device'])['Device'].count().reset_index(name='Count')
2 group_1.rename(columns={'Count': 'Count'})
3 group_1.sort_values(by='Count', ascending=False)
```

✓ [52] 12ms

	Region	Device	Count
6	United States	iPhone	419
4	United States	Mac	240
3	United States	Android	184
5	United States	PC	103
2	Russia	iPhone	3
1	Russia	Mac	2
0	Russia	Android	1

Рисунок 17 – Результат выполнения задания 2

Далее с помощью нехитрого запроса была создана сводная таблица, в которой содержится информация о том, сколько пользователей пришло из источников, отсортированная по убыванию. Результат на рисунке 18.

```
1 pivot_table_1: pd.DataFrame = visits.pivot_table(index=['Channel'],
    values='User_Id',aggfunc='count').reset_index().rename
    (columns={'User_Id': 'Users'}).sort_values(by='Users', ascending=False)
2 print(pivot_table_1)
```

✓ [66] 10ms

	Channel	Users
3	organic	608
0	FaceBoom	149
2	TipTop	142
1	MediaTornado	53

Рисунок 18 – Результат выполнения задания 3

Далее приступим к 4 заданию. Было выполнено 4 задание.

```
1 visits.pivot_table(index=['Region', 'Device'], values='User_Id',
    aggfunc='count').reset_index().sort_values(by='Region', ascending=True),
    .rename(columns={'User_Id': 'Users'})
2 #print(pivot_table_2)
```

✓ [80] 12ms

	Region	Device	Users
0	Russia	Android	1
1	Russia	Mac	2
2	Russia	iPhone	3
3	United States	Android	184
4	United States	Mac	240
5	United States	PC	103
6	United States	iPhone	419

Рисунок 19 – Результат выполнения задания 4

4. Ссылка на Colab:

<https://colab.research.google.com/drive/1uHK6Sc57YdVmksraK3HLJOeQWbDCCoYM?usp=sharing>

5. Вывод: была осуществлена предварительная обработка данных csv-файла, выявлены и устранены проблемы в этих данных. Получены навыки использования библиотеки Pandas для Python и Gogle Collab.

- Всего было обработано 952 сессии.
- Абсолютное большинство данных из США (946).
- Большинство устройств абаентов - iPhone (331) и Mac (201).
- Самый популярный источник - organic (608), второй по популярности Face Boom (149).

- Средняя продолжительность сессии 29 минут.
- Количество сессий с продолжительностью выше среднего - 358.
- Почти сессии являются короткими по классификации, предложенной во время работы (41 значение в средней категории и 1 значение в длинной).

6. Дополнительное задание:

- Сделать отдельный столбец с длительностью сессии в минутах.
- Присвоить длительности сессий 3 категории: короткая, средняя, длинная.
- Выбрать те записи, которые больше среднего количества длительности.
- Создать сводную таблицу категория длительности + устройство + уникальное количество пользователей.
- Описать разницу между функциями count, nunic и другими функциями.

Начнём с создания дополнительного столбца с длительностью сессий в минутах. Процесс показан на рисунке 20.

Code

Markdown

```

1 visits["Duration_Minutes"] = (visits['Session_End'] -
2   visits['Session_Start']).dt.total_seconds() / 60
3 visits
4 [236]

```

<

>

11 rows

<

>

952 rows x 7 columns

Static Output

Session_Start	Session_End	Duration_Minutes
2019-05-01 02:36:00	2019-05-01 02:45:00	9.0
2019-05-01 04:46:00	2019-05-01 04:47:00	1.0
2019-05-01 14:09:00	2019-05-01 15:32:00	83.0
2019-05-01 00:29:00	2019-05-01 00:54:00	25.0
2019-05-01 03:33:00	2019-05-01 03:57:00	24.0
...	...	...
2019-05-04 23:32:00	2019-05-04 23:38:00	6.0
2019-05-04 19:32:00	2019-05-04 19:48:00	16.0
2019-05-04 00:13:00	2019-05-04 00:26:00	13.0
2019-05-04 03:33:00	2019-05-04 03:46:00	13.0
2019-05-04 03:01:00	2019-05-04 03:15:00	14.0

Рисунок 20 – Создание дополнительного столбца с длительностью сессий

В таблице появился столбец `Duration_Minutes`, в котором записана длительность каждой сессии в минутах. Используется тип данных `float64`, так как длительность сессии может быть дробным числом в минутах.

Для определения категорий длительности сессии узнаем максимальное и минимальное время сессии. Далее поделим на 3 равных интервала. Процесс распределения интервалов показан на рисунке 21.

```

1 min_val = visits['Duration_Minutes'].min()
2 max_val = visits['Duration_Minutes'].max()
3
4 interval_1 = [int(min_val), round(float((max_val - min_val) / 3))]
5 interval_2 = [int(interval_1[1]), round(float(((max_val - min_val) / 3)
6   * 2))]
7 interval_3 = [int(interval_2[1]), int(max_val)]
8
9 print(f"Минимальное: {min_val}")
10 print(f"Максимальное: {max_val}\n")
11 print(f"Интервал 1: [{interval_1[0]}, {interval_1[1]}]")
12 print(f"Интервал 2: [{interval_2[0]}, {interval_2[1]}]")
13 print(f"Интервал 3: [{interval_3[0]}, {interval_3[1]}]")
14 [237]

```

Минимальное: 0.0

Максимальное: 262.0

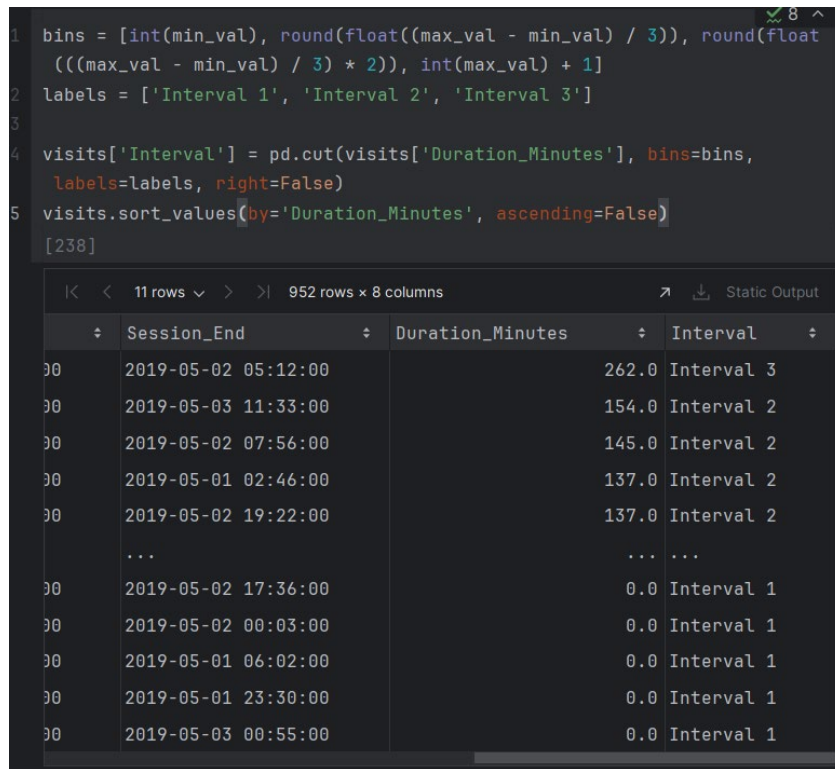
Интервал 1: [0, 87)

Интервал 2: [87, 175)

Интервал 3: [175, 262]

Рисунок 21 – Процесс распределения интервалов

Интервал 1 - короткая сессия, интервал 2 - средняя сессия и интервал 3 - длинная сессия. Создадим новый столбец, в котором присвоим категории длинам интервалов. Для этого воспользуемся функцией `cut`. Процесс показан на рисунке 22.



```
1 bins = [int(min_val), round(float((max_val - min_val) / 3)), round(float  
    (((max_val - min_val) / 3) * 2)), int(max_val) + 1]  
2 labels = ['Interval 1', 'Interval 2', 'Interval 3']  
3  
4 visits['Interval'] = pd.cut(visits['Duration_Minutes'], bins=bins,  
    labels=labels, right=False)  
5 visits.sort_values(by='Duration_Minutes', ascending=False)  
[238]
```

	Session_End	Duration_Minutes	Interval
00	2019-05-02 05:12:00	262.0	Interval 3
00	2019-05-03 11:33:00	154.0	Interval 2
00	2019-05-02 07:56:00	145.0	Interval 2
00	2019-05-01 02:46:00	137.0	Interval 2
00	2019-05-02 19:22:00	137.0	Interval 2
	...	...	...
00	2019-05-02 17:36:00	0.0	Interval 1
00	2019-05-02 00:03:00	0.0	Interval 1
00	2019-05-01 06:02:00	0.0	Interval 1
00	2019-05-01 23:30:00	0.0	Interval 1
00	2019-05-03 00:55:00	0.0	Interval 1

Рисунок 22 – Присвоение категории

Сессии были разделены виды:

- Interval 1 - Короткая
- Interval 2 - Средняя
- Interval 3 - Длинная

Теперь выберем записи, которые больше среднего количества длительности. Процесс и результат показаны на рисунке 23.



```
1 medium_value = visits['Duration_Minutes'].mean() # Средняя
   продолжительность сессии
2 print("Среднее значение:", medium_value)
3
4 visits_above_mean = visits[visits['Duration_Minutes'] > medium_value]
   .reset_index(drop=True)
5 visits_above_mean
[239]
```

Среднее значение: 29.077731092436974

11 rows ▾ > 358 rows x 8 columns						Static Output			
↕	User_Id	↕	Region	↕	Device	↕	Channel	↕	Sessi
0	591000000000		United States		Mac		organic		2019-
1	43958116050		United States		Android		organic		2019-
2	370000000000		United States		iPhone		organic		2019-
3	142000000000		United States		Mac		FaceBoom		2019-
4	525000000000		United States		Mac		organic		2019-
...	...		...		...		...		...
353	652000000000		United States		Mac		organic		2019-
354	301000000000		United States		iPhone		FaceBoom		2019-
355	914000000000		United States		PC		organic		2019-
356	64474110919		United States		iPhone		organic		2019-
357	411000000000		United States		Android		TipTop		2019-

Рисунок 23 – Значения выше среднего

В таблице `visits_above_mean` содержатся строки, в которых продолжительность сессии выше среднего. Создадим сводную таблицу по заданию "длительности + устройство + уникальное количество пользователей". Процесс создания и таблица показаны на рисунке 24.

```
1 additional_pivot_table: pd.DataFrame = visits.pivot_table
  (index=['Interval', columns=['Device', values='User_Id',
  aggfunc=['nunique'], observed=False).reset_index()
2 print(additional_pivot_table)
[241]
```

	Interval	nunique			
Device		Android	Mac	PC	iPhone
0	Interval 1	152	194	83	315
1	Interval 2	9	9	5	18
2	Interval 3	0	0	0	1

## Рисунок 24 – Сводная таблица по дополнительному заданию

По текущей группировке большинство сессий является короткими, средние сессии в меньшинстве, а длинные и вовсе имеются только в одном экземпляре, принадлежит она пользователю iPhone.

Далее будет выполнено задание "Описать разницу между функциями count, nunique и другими функциями."

- count() считает количество непустых (ненулевых и не NaN) значений в столбце или строке.
- nunique() считает количество уникальных значений в столбце (считая только ненулевые и не NaN значения).
- unique() возвращает массив всех уникальных значений в столбце в том порядке, в котором они встречаются. В отличие от nunique(), которая возвращает количество уникальных значений, unique() возвращает сами уникальные элементы.
- mean() вычисляет среднее арифметическое для числовых значений.
- min() и max() определяют минимальное и максимальное значение в столбце.
- std() вычисляет стандартное отклонение значений.
- median() возвращает медиану, то есть центральное значение отсортированного ряда.
- mode() возвращает самое часто встречающееся значение в столбце (моду).
- describe() Возвращает общие статистические данные по столбцу или DataFrame (включает count, mean, std, min, 25%, 50%, 75%, и max).
- quantile() возвращает значение на указанном квантиле (процентиле).
- sum() суммирует все значения в столбце. Для строк или логических значений может работать иначе (например, считать количество True значений).