

ГУАП

КАФЕДРА № 41

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

специалист

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

\_\_\_\_\_  
подпись, дата

В. В. Боженко

\_\_\_\_\_  
инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ

по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4117

\_\_\_\_\_  
подпись, дата

Д. С. Николаев

\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург 2023

## Цель работы

Осуществить предварительную обработку данных csv-файла, выявить и устранить проблемы в этих данных.

## Индивидуальное задание

Вариант 2.

Задание 1 : Группировка - CATEGORY и количество поездок каждого типа (по цели маршрута).

Задание 2 : Группировка - CATEGORY и количество поездок каждого типа (по цели маршрута). Создать датафрейм. Переименовать столбец с количеством в "count". Отсортировать по убыванию столбца "count".

Задание 3: Сводная таблица (pivot\_table) - средняя количество пройденных миль по каждой цели поездки (PURPOSEroute). Отсортировать по убыванию столбца MILES. Округлить значение до двух знаков.

Задание 4: Сводная таблица (pivot\_table) - средняя количество пройденных миль по каждой цели поездки (PURPOSEroute) - столбцы и каждой категории - строки. Отсортировать по убыванию столбца CATEGORY.

## Ход работы

Для начала был загружен датасет через библиотеку Python - pandas. Используем ';' для разделение данных. Результат установки указан на Рисунке 1.

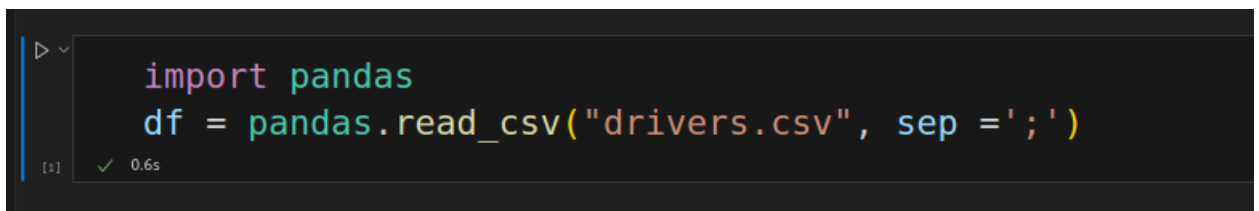
A screenshot of a Jupyter Notebook interface. On the left, there is a vertical toolbar with a play button icon and a dropdown arrow. Below it, a status bar shows '[1]' and a green checkmark followed by '0.6s'. The main area is a code editor with a dark background and light-colored text. It contains two lines of Python code: 'import pandas' and 'df = pandas.read\_csv("drivers.csv", sep = ';')'. The code is syntax-highlighted, with 'import' in blue, 'pandas' in green, 'df' in blue, 'pandas.read\_csv' in green, and the string and separator in orange. The code is enclosed in triple backticks.

Рисунок 1 - Загрузка датасета

Далее были выведены первые 20 строк через метод head() (Рисунок 2).

```
df.head(20)
```

	START_DATE	END_DATE	CATEGORY*	START	STOP	MILES	PURPOSEroute
0	01.10.2016 19:12	01.10.2016 19:32	Business	Midtown	East Harlem	44963.0	MEETING
1	01.11.2016 13:32	01.11.2016 13:46	Business	Midtown	Midtown East	45108.0	Meal/Entertain
2	01.12.2016 12:33	01.12.2016 12:49	Business	Midtown	Hudson Square	45170.0	Meal/Entertain
3	1.13.2016 15:00	1.13.2016 15:28	Business	Gulfton	Downtown	45149.0	Meeting
4	1.29.2016 21:21	1.29.2016 21:40	Business	Apex	Cary	45051.0	Meal/Entertain
5	1.30.2016 18:09	1.30.2016 18:24	Business	Apex	Cary	45112.0	Customer Visit
6	02.01.2016 12:10	02.01.2016 12:43	Business	Chapel Hill	Cary	45008.0	Customer Visit
7	02.04.2016 9:37	02.04.2016 10:09	Business	Morrisville	Cary	45116.0	Meal/Entertain
8	02.07.2016 18:03	02.07.2016 18:17	Business	Apex	Cary	45112.0	Customer Visit
9	02.07.2016 20:22	02.07.2016 20:40	Business	Morrisville	Cary	44932.0	Meeting
10	02.09.2016 20:24	02.09.2016 20:40	Business	Morrisville	Cary	44932.0	Meal/Entertain
11	02.11.2016 20:36	02.11.2016 20:51	Business	Morrisville	Cary	44932.0	Temporary Site
12	02.12.2016 11:14	02.12.2016 11:35	Business	Morrisville	Raleigh	17.0	Customer Visit
13	02.12.2016 15:33	02.12.2016 16:06	Business	Morrisville	Cary	45057.0	Customer Visit
14	2.14.2016 14:46	2.14.2016 15:03	Business	Midtown	Midtown West	2.0	Meeting
15	2.16.2016 10:31	2.16.2016 10:41	BUSINESS	Colombo	Colombo	45079.0	NaN
16	2.16.2016 11:32	2.16.2016 12:02	Business	Colombo	Colombo	45050.0	NaN
17	2.16.2016 12:39	2.16.2016 12:42	Business	Colombo	Colombo	45108.0	NaN
18	2.16.2016 13:43	2.16.2016 13:55	BUSINESS	Colombo	Colombo	45139.0	Temporary Site
19	2.16.2016 16:34	2.16.2016 17:10	Business	Colombo	Colombo	6.0	NaN

Рисунок 2 - Вывод первых 20 элементов

Столбцы:

1. **START\_DATE** и **END\_DATE**: Эти столбцы содержат дату и время начала и завершения поездки. Они указывают, когда началась и завершилась каждая поездка.
2. **CATEGORY**: Этот столбец обозначает категорию поездки. В данном случае, больше поездок "Business", что указывает на то, что она связана с деловой деятельностью.
3. **START** и **STOP**: Эти столбцы указывают начальное и конечное местоположение поездки. Они указывают, откуда и куда направлялась поездка.
4. **MILES**: Этот столбец содержит информацию о количестве миль, пройденных во время поездки. Это может быть полезным для расчета расстояний и затрат на топливо.
5. **PURPOSE**: Этот столбец описывает цель поездки. Он указывает, для чего была совершена поездка, например, "MEETING" (встреча), "Meal/Entertain" (питание/развлечение), "Customer Visit" (визит к клиенту) и т. д.

Предметная область этой таблицы связана с учетом и анализом служебных поездок. Возможно, она используется для отслеживания затрат на бизнес-поездки, анализа расходов на топливо, определения целей и местоположение поездок, а также для управления служебными маршрутами и встречами.

Далее с помощью команды `df.info()` была выведена информацию о DataFrame (Рисунок 3).

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 161 entries, 0 to 160
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype  
---  -
0   START_DATE      161 non-null   object 
1   END_DATE        161 non-null   object 
2   CATEGORY*       161 non-null   object 
3   START           161 non-null   object 
4   STOP            161 non-null   object 
5   MILES           161 non-null   float64 
6   PURPOSEroute    84 non-null    object 
dtypes: float64(1), object(6)
memory usage: 8.9+ KB
```

Рисунок 3 - Информация о DataFrame

Результат выполнения метода `info()` показывает следующую информацию о DataFrame:

1. Всего строк: 161.
2. Количество непустых (non-null) значений для каждого столбца.
3. Типы данных для каждого столбца.

Из этой информации видно следующее:

- Столбцы "START\_DATE", "END\_DATE", "CATEGORY\*", "START", "STOP" и "PURPOSEroute" содержат объекты (строки).
- Столбец "MILES" содержит числовые значения с типом данных float64.
- Столбец "PURPOSEroute" имеет некоторые пропущенные значения, так как количество непустых значений меньше общего числа строк.

С помощью команды `describe` были оценены числовые столбцы (Рисунок 4).

```
df.describe()

MILES
count    161.000000
mean    37766.519255
std     16614.925558
min       0.800000
25%    44931.000000
50%    45008.000000
75%    45081.000000
max     45177.000000
```

Рисунок 4 - Числовые столбцы

Результат выполнения метода describe() для числового столбца "MILES" выглядит следующим образом:

- count: Количество непустых (non-null) значений в столбце - 161.
- mean: Среднее значение - 37766.519255.
- std: Стандартное отклонение - 16614.925558.
- min: Минимальное значение - 0.800000.
- 25%: 25-й процентиль - 44931.000000.
- 50%: Медианное значение (50-й процентиль) - 45008.000000.
- 75%: 75-й процентиль - 45081.000000.
- max: Максимальное значение - 45177.000000.

Эти статистические показатели позволяют оценить основные характеристики числового столбца "MILES", такие как среднее значение, разброс данных (стандартное отклонение), минимальное и максимальное значения, а также квантили, что может быть полезно при анализе данных и поиске выбросов.

Далее были выведены на экран названия столбцов с помощью df.columns (Рисунок 5).

```
df.columns
[140]
... Index(['START_DATE', 'END_DATE', 'CATEGORY*', 'START', 'STOP', 'MILES',
        'PURPOSEroute'],
        dtype='object')
```

Рисунок 5 - Вывод названия столбцов

Проблема заключается в том, что названия не имеют единого формата. Для этого были переименованы столбцы (Рисунок 6).

```
df.columns = ['START_DATE', 'END_DATE', 'CATEGORY', 'START', 'STOP', 'MILES', 'PURPOSE_ROUTE']
df.columns
[141]
... Index(['START_DATE', 'END_DATE', 'CATEGORY', 'START', 'STOP', 'MILES',
        'PURPOSE_ROUTE'],
        dtype='object')
```

Рисунок 6 - Изменения названия столбцов

С помощью метода fillna() были заменены пропуски на 'Unknown' (Рисунок 7).

```
df['PURPOSE_ROUTE'].fillna('Unknown', inplace=True)
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 161 entries, 0 to 160
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   START_DATE      161 non-null   object
 1   END_DATE        161 non-null   object
 2   CATEGORY        161 non-null   object
 3   START           161 non-null   object
 4   STOP            161 non-null   object
 5   MILES           161 non-null   float64
 6   PURPOSE_ROUTE   161 non-null   object
dtypes: float64(1), object(6)
memory usage: 8.9+ KB
```

Рисунок 7 - Замена пропусков

Далее была проведена проверка на явные дубликаты (Рисунок 8).

```
df[df.duplicated()]
```

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE_ROUTE
159	7.26.2016 22:31	7.26.2016 22:39	Business	Morrisville	Cary	45048.0	Meal/Entertain
160	7.26.2016 22:31	7.26.2016 22:39	Business	Morrisville	Cary	45048.0	Meal/Entertain

Рисунок 8 - Проверка на дубликаты

На рисунке 8 видно, что две строки идентичны, избавимся от них с помощью `drop_duplicates()` (Рисунок 9).

```
df.drop_duplicates(inplace=True)
df[df.duplicated()]

START_DATE END_DATE CATEGORY START STOP MILES PURPOSE_ROUTE
```

Рисунок 9 - Удаление дубликатов

Проверка на неявные дубликаты (различные написания одного и того же) при помощи функции `'unique()'` для столбцов с текстовым типом данных (Рисунок 10).

Для начала проверен столбец CATEGORY.

```

df['CATEGORY'].unique()
... array(['Business', 'BUSINESS', 'Personal'], dtype=object)

```

Рисунок 10 - Просмотр на наличие уникальных значение

На рисунке 10 видим, что Business пишется в двух вариациях. С помощью replace() значения были изменены на общий вид (Рисунок 11).

```

df['CATEGORY'] = df['CATEGORY'].replace('BUSINESS', 'Business')
df['CATEGORY'].unique()
... array(['Business', 'Personal'], dtype=object)

```

Рисунок 11 - Замена неявных дубликатов

Тоже самое было проделано и для столбца PURPOSE\_ROUTE, другие не рассматривались, так как это или дата, или название (Рисунок 12).

```

df['PURPOSE_ROUTE'].unique()
... array(['MEETING', 'Meal/Entertain', 'Meeting', 'Customer Visit',
          'Temporary Site', 'Unknown', 'Moving'], dtype=object)

df['PURPOSE_ROUTE'] = df['PURPOSE_ROUTE'].replace('MEETING', 'Meeting')
df['PURPOSE_ROUTE'].unique()
... array(['Meeting', 'Meal/Entertain', 'Customer Visit', 'Temporary Site',
          'Unknown', 'Moving'], dtype=object)

```

Рисунок 12 - Замена неявных дубликатов для другого столбца

У столбцов с датой был изменен тип данных на datetime (Рисунок 13).

```

df['START_DATE'] = pandas.to_datetime(df['START_DATE'], format='%m.%d.%Y %H:%M')
df['END_DATE'] = pandas.to_datetime(df['END_DATE'], format='%m.%d.%Y %H:%M')
df.dtypes
... START_DATE      datetime64[ns]
END_DATE      datetime64[ns]
CATEGORY      object
START      object
STOP      object
MILES      float64
PURPOSE_ROUTE      object
dtype: object

```

Рисунок 13 - Изменение типов данных

Далее были выполнены индивидуальные задания.

#### Задание 1

Группировка - CATEGORY и количество поездок каждого типа (по цели маршрута).

С помощью данной выборки можно определить с какой целью чаще везут и зачем (Рисунок 14).

```
df.groupby(['CATEGORY', 'PURPOSE_ROUTE'])['PURPOSE_ROUTE'].count()

...
CATEGORY  PURPOSE_ROUTE
Business  Customer Visit    30
          Meal/Entertain    34
          Meeting           13
          Temporary Site     4
          Unknown           67
Personal  Moving            1
          Unknown           10
Name: PURPOSE_ROUTE, dtype: int64
```

Рисунок 14 - Код и результат задания 1

## Задание 2

Группировка - CATEGORY и количество поездок каждого типа (по цели маршрута). Создать датафрейм. Переименовать столбец с количеством в “count”. Отсортировать по убыванию столбца “count”.

Создаем группировку как в предыдущем задании, столбец с получаемым значением называем - count и применяем функцию sort\_values для данного столбца по убыванию.

Данный датафрейм позволяет легче обработать информацию и быстрее выявить из нее нужную информацию (Рисунок 15).

```
df1 = df.groupby(['CATEGORY', 'PURPOSE_ROUTE'])['PURPOSE_ROUTE'].count().reset_index(name='count').sort_values('count', ascending=False)
df1

...
CATEGORY  PURPOSE_ROUTE  count
4  Business      Unknown    67
1  Business  Meal/Entertain   34
0  Business  Customer Visit   30
2  Business      Meeting    13
6  Personal      Unknown    10
3  Business  Temporary Site    4
5  Personal      Moving      1
```

Рисунок 15 - Код и результат задания 2

## Задание 3

Сводная таблица (pivot\_table) - средняя количество пройденных миль по каждой цели поездки (PURPOSEroute). Отсортировать по убыванию столбца MILES. Округлить значение до двух знаков.

Для начала создадим сводную таблицу, используя - pivot\_table. Далее округлим столбец MILES до двух знаков после запятой и отсортируем этот же столбец по убыванию.

Данная таблица позволяет посмотреть самые затрачиваемые поездки по количеству миль (Рисунок 16).



```
df_pivot = pandas.pivot_table(df, values='MILES', index='PURPOSE_ROUTE', aggfunc='mean')
df_pivot['MILES'] = df_pivot['MILES'].round(2)
df_pivot.sort_values(by='MILES', ascending=False, inplace=True)
df_pivot
```

PURPOSE_ROUTE	MILES
Temporary Site	45063.50
Moving	44932.00
Meal/Entertain	41054.82
Unknown	36859.20
Customer Visit	36023.20
Meeting	34646.85

Рисунок 16 - Код и результат задания 3

#### Задание 4

Сводная таблица (pivot\_table) - средняя количество пройденных миль по каждой цели поездки (PURPOSE\_ROUTE) - столбцы и каждой категории - строки. Отсортировать по убыванию столбца CATEGORY.

Продолжаем ту же самую работу, что и в прошлом задании, но теперь в pivot\_table передается массив столбцов. Сортировка же для символьных типов данных идет по ASCII таблице (по алфавитному порядку). Результат продемонстрирован на Рисунке 17.

```
df_pivot2 = pandas.pivot_table(df, values='MILES', index=['CATEGORY', 'PURPOSE_ROUTE'], aggfunc='mean')
df_pivot2['MILES'] = df_pivot2['MILES'].round(2)
df_pivot2.sort_values(by='CATEGORY', ascending=False)
df_pivot2
```

CATEGORY	PURPOSE_ROUTE	MILES
Business	Customer Visit	36023.20
	Meal/Entertain	41054.82
	Meeting	34646.85
	Temporary Site	45063.50
	Unknown	36982.22
Personal	Moving	44932.00
	Unknown	36035.00

Рисунок 17 - Код и результат задания 4

## Вывод

В данной лабораторной работе были рассмотрены основные операции с данными с использованием библиотеки pandas в Python. А также использование jupyter для оформления. Вот ключевые моменты и шаги, выполненные в рамках лабораторной работы:

- Загрузка данных: Данные были предоставлены в формате текстового файла или CSV. Мы использовали библиотеку pandas для загрузки данных в DataFrame - удобную структуру данных для анализа и манипуляций с данными.

- Изучение данных: Мы использовали методы .head(), .tail(), .info(), и .describe() для первичного изучения данных. Эти методы позволяют нам получить представление о

данных, включая типы данных, наличие пропусков и основные статистические показатели.

- Обработка данных: Мы рассмотрели различные методы обработки данных, такие как удаление или замена пропусков с использованием `.dropna()` и `.fillna()`, а также удаление дубликатов с использованием `.drop_duplicates()`.

- Создание сводных таблиц: Мы использовали метод `pivot_table` для создания сводных таблиц, позволяющих агрегировать данные и анализировать их в разных срезах. Мы выполнили сортировку сводной таблицы по убыванию.

- Округление значений: Мы использовали метод `.round()` для округления числовых значений в сводной таблице до двух знаков после запятой.

Эти шаги представляют собой основные операции по обработке и анализу данных с использованием библиотеки `pandas`. Все они могут быть адаптированы и расширены для работы с реальными данными и решения конкретных задач анализа данных. Важно иметь понимание о том, как использовать эти инструменты для эффективной обработки и анализа данных в Python.