Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский экономический университет имени Г.В. Плеханова»
МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ

специальность 09.02.07 «Информационные системы и программирование»

Квалификация: Программист

ПРАКТИЧЕСКАЯ РАБОТА №3

ПО МДК 11.01 «Технология разработки и защиты баз данных»

Выполнила студентка
группы П50-4-22
Н.И. Григоренко

Проверил преподаватель
_____ К.А. Перевалов
«___» _____ 2024 года

Москва 2024

«Pandas»

Цель работы: с использованием python-библиотеки pandas посмотреть информацию по датасету, обработать его, вывести первые и последние пять записей, провести срез данных, фильтрацию и сортировку, вывести первичные статистические характеристики датасета, провести группировку по одному столбцу и вызвать одну агрегатную функцию, вывести топ-5 записей, обосновать и описать весь ход работы в отчёте.

Ход работы:

1. **Начало работы и создание файла.**

Сначала через удобный редактор кода или IDE нужно создать папку и файл, в котором будет происходить работа. В данном случае при выполнении заданий будет использоваться **Anaconda вместе с Jupyter Notebook**. Для начала надо перейти на официальный сайт анаконды[1], чтобы её скачать. После того, как переход на сайт по ссылке выполнен – надо нажать на кнопку «Free download» в правом верхнем углу, пропустить регистрацию и скачать файл (см. Рисунок 1 – Скачивание анаконды).

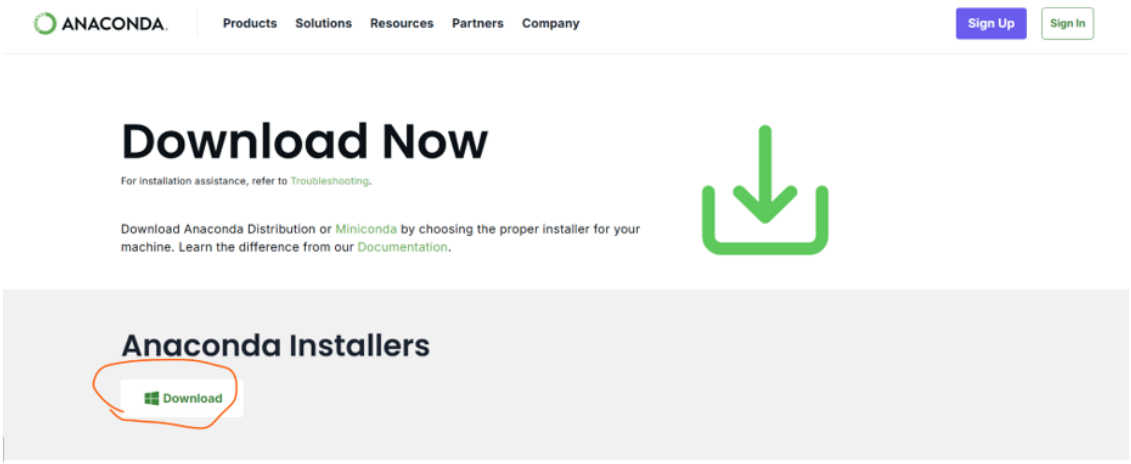


Рисунок 1 - Скачивание анаконды

[1] https://www.anaconda.com/ - ссылка на официальный сайт для скачивания Anaconda.

После недолгой загрузки в «Anaconda Navigator» нужно выбрать «Jupyter Notebook» и запустить приложение, которое открывается в браузере (см. Рисунок 2 – Запуск Jupyter).

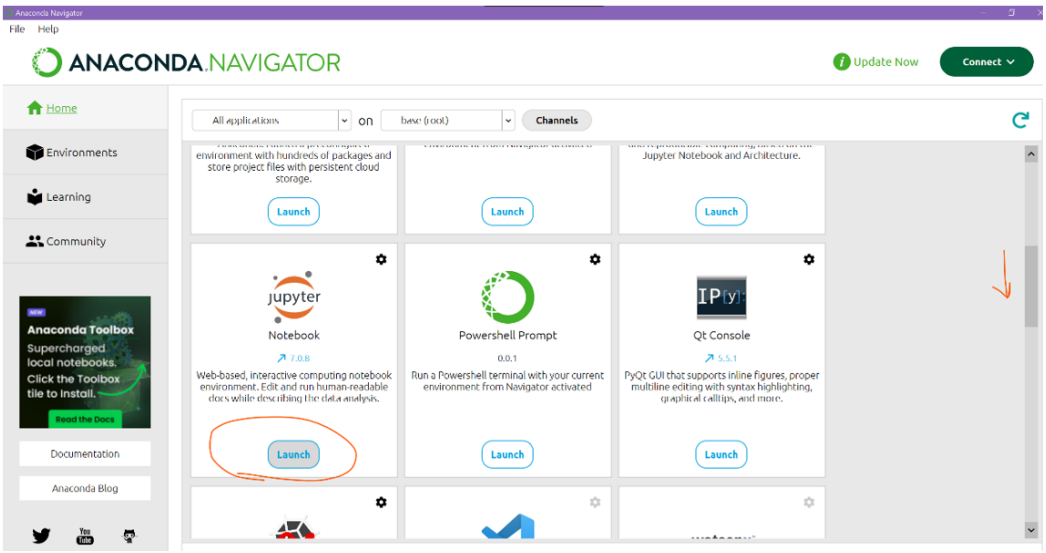


Рисунок 2 – Запуск Jupyter

Далее следует создать новую папку с любым названием для хранения файлов, внутри нее - новый файл с расширением «.ipynb». Помимо этого, в папку нужно перенести датасет, в данной практической работе используется датасет поездок метро в Англии (см. Рисунок 3 – Создание папки и файла).

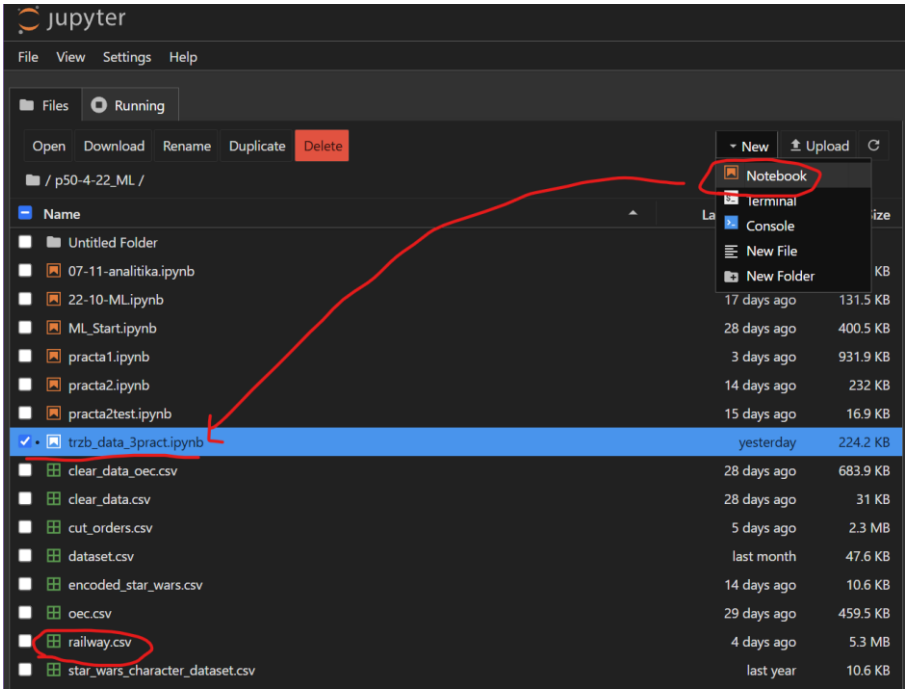


Рисунок 3 – Создание папки и файла

2. Импортирование библиотек и выгрузка датасета.

Датасет - это набор данных, организованный определенным образом, который обычно используется для анализа и обучения машинных моделей. Обычно в формате «.csv» (это простой текстовый формат для хранения табличных данных, где значения в каждой строке разделяются запятыми). Теперь, можно зайти в файл и начинать работу. Строки здесь могут быть двух основных видов - код (непосредственно python-код, который будет выполнять определенные операции над датасетом) или markdown - это простой язык разметки, который используется для форматирования текста в читаемый вид. Здесь также можно использовать html-теги.

Во-первых, необходимо импортировать библиотеки, которые будут использоваться в ходе выполнения практической работы. Для этого используется строка кода, внутри нее «import» - ключевое слово для импортирования в проект библиотек или зависимостей и «as» - для выдачи псевдонима удобного обращения. Для этой практической работы понадобится одна библиотека - **pandas** - используется для манипулирования данными, выполняет фильтрацию, сортировку, работу с разными видами значений, читает датасеты, хранит их и обрабатывает. Чтобы импортировать датасет следует создать отдельную переменную и воспользоваться функцией «read_csv» библиотеки «pandas» (см. Рисунок 4 – Импорт и просмотр датасета).

[3]:
```
import pandas as pd
```

Для того, чтобы импортировать датасет, надо создать отдельную переменную для его хранения и дальнейшего обращения, к примеру ds. Функцию **read_csv** библиотеки **pandas** позволяет читать датасет - для этого в качестве параметра передается название файла датасета вместе с расширением в одинарных кавычках.

Юпитер автоматически прочитает датасет из переменной при ее единичном вводе, но это по сути метод **display**.

[4]:
```
ds = pd.read_csv('railway.csv')
ds
```

[4]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	D
0	da8a6ba8-b3dc-4677-b176	2023-12-08	12:41:11	Online	Contactless	Adult	Standard	Advance	43	London Paddington	L
1	b0cdd1b0-f214-4197-be53	2023-12-16	11:23:01	Station	Credit Card	Adult	Standard	Advance	23	London Kings Cross	
2	f3ba7a96-f713-40d9-9629	2023-12-19	19:51:27	Online	Credit Card	NaN	Standard	Advance	3	Liverpool Lime Street	N

Рисунок 4 – Импорт и просмотр датасета

3. Просмотр информации по датасету.

Метод **info()** выводит краткие сведения о датасете, а именно:

- индекс датасета, который является RangeIndex - простой числовой индекс от 0 до 31652, всего записей 31653;

- количество столбцов, отсчет начинается, как и во всех массивах, с нуля;

- имена столбцов и их количество;

- количество непустых значений (non-null);

- тип данных в столбце - в данном случае 1 столбец типа float и 17 столбцов типа object (здесь - строковое значение);

- указание сколько датасет занимает в памяти - здесь 4.3+ МБ.

Метод **shape** выводит размерность датасета - 31653 строки - первое число в выводе и 18 столбцов - второе число в выводе (см. Рисунок 5 – Метод info() и shape).

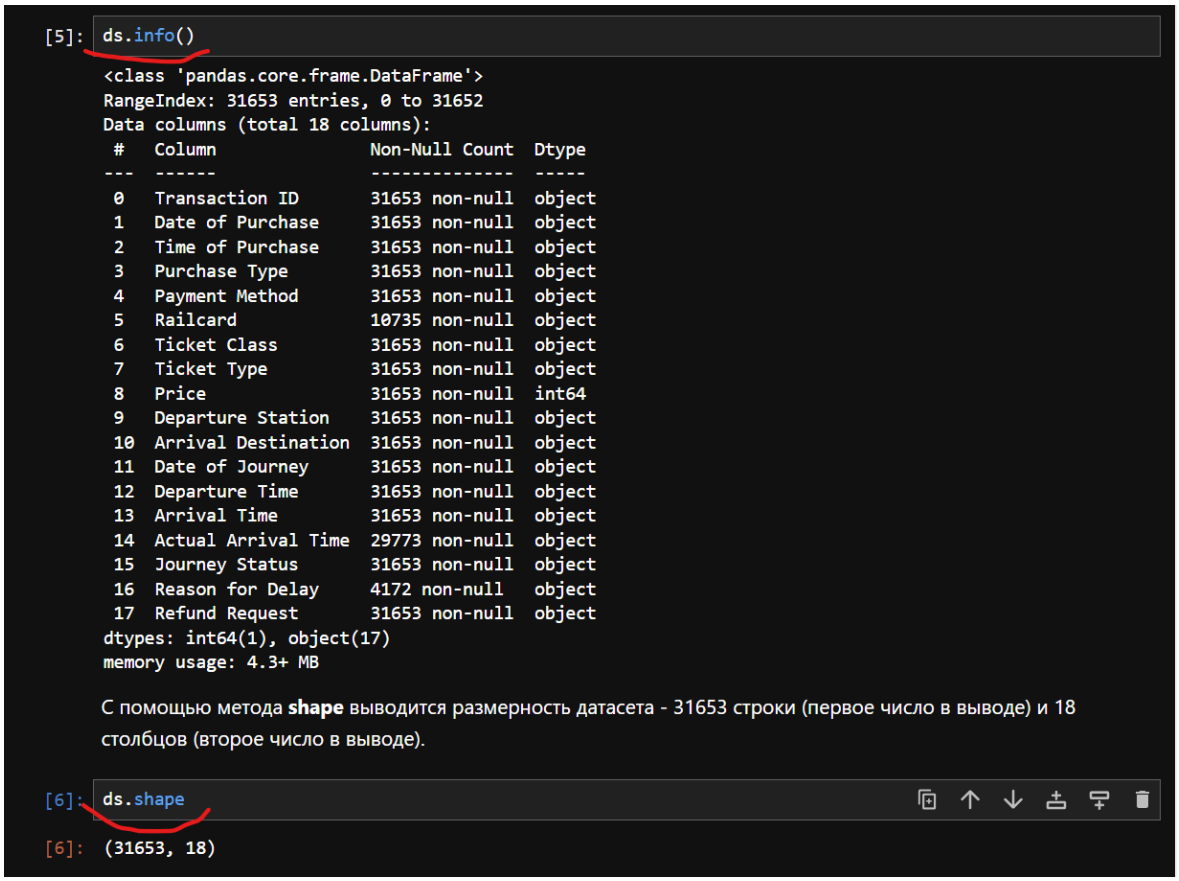


Рисунок 5 – Метод info() и shape

Метод **columns** выводит в виде массива названия всех столбцов (см. Рисунок 6 – Метод columns):

- Transaction ID - уникальный идентификатор для индивидуальной покупки билета на поезд.

- Date of Purchase - дата приобретения билета.

- Time of Purchase - время приобретения билета.

- Purchase Type - тип покупки билета, независимо от того, был ли билет приобретен онлайн или непосредственно на железнодорожном вокзале.

- Payment Method - способ оплаты, использованный при покупке билета (бесконтактный, кредитной или дебетовой картой).

- Railcard - железнодорожная карточка. Владельцы карт получают 1/3 скидки на покупку билетов.

- Ticket Class - класс места по билету (Стандартный или первый).

- Ticket Type - тип билета, когда вы купили билет или можете им воспользоваться. На предварительные билеты предоставляется скидка в размере 1/2 стоимости, и их необходимо приобрести как минимум за сутки до вылета. На билеты в нерабочее время действует скидка в размере 1/4 стоимости, и их необходимо использовать в нерабочее время (в будние дни с 18:00 до 16:00). Билеты в любое время продаются по полной цене и могут быть куплены и использованы в любое время суток.

- Price - окончательная стоимость билета.

- Departure Station - станция для посадки в поезд.

- Arrival Destination - станция для выхода из поезда.

- Date of Journey - дата отправления поезда.

- Departure Time - время отправления поезда.

- Arrival Time - время прибытия поезда в пункт назначения по расписанию (может быть на следующий день после отправления).

- Actual Arrival Time - фактическое время прибытия поезда в пункт назначения (может быть на следующий день после отправления).

- Journey Status - статус поездки, независимо от того, пришел ли поезд вовремя, задержался или был отменен.

- Reason for Delay - причина задержки или отмены бронирования.

- Refund Request - запросил ли пассажир возврат средств после задержки или отмены бронирования.

```
[7]: ds.columns

[7]: Index(['Transaction ID', 'Date of Purchase', 'Time of Purchase',
       'Purchase Type', 'Payment Method', 'Railcard', 'Ticket Class',
       'Ticket Type', 'Price', 'Departure Station', 'Arrival Destination',
       'Date of Journey', 'Departure Time', 'Arrival Time',
       'Actual Arrival Time', 'Journey Status', 'Reason for Delay',
       'Refund Request'],
      dtype='object')
```

Рисунок 6 – Метод columns

Анализ датасета можно провести с помощью функции **describe()** - статистическое описание числовых данных:

- count - количество непустых значений в столбцах с числовыми значениями, здесь их 31653, как и записей, то есть пропусков нет.

- mean - среднее арифметическое значение (сумма всех значений столбца, деленная на количество этих значений) - 23.449.

- std - стандартное отклонение относительно среднего значения, в столбце Price оно не такое большое (но и сравнить не с чем, это единственный столбец с числовым типом данных) - 29.998, следовательно, разброс по данным небольшой.

- min - показывает минимальное значение в столбце - 1.

- max - аналогично показывает максимальное значение в столбце 267.

- 25% - первый квартиль (нижний) - значение разделяет нижние 25% данных.

- 50% - медиана - значение разделяет данные пополам, центральное значение в упорядоченном наборе данных, второй квартиль совпадает с медианой - 50% (делит данные пополам).

- 75% - третий квартиль (верхний) - значение разделяет нижние 75% данных.

Квартили - значения, разделяющие набор данных на равные части.

Количество уникальных значений в каждом столбце выведет метод nunique (см. Рисунок 7 – Функция describe() и метод nunique()).

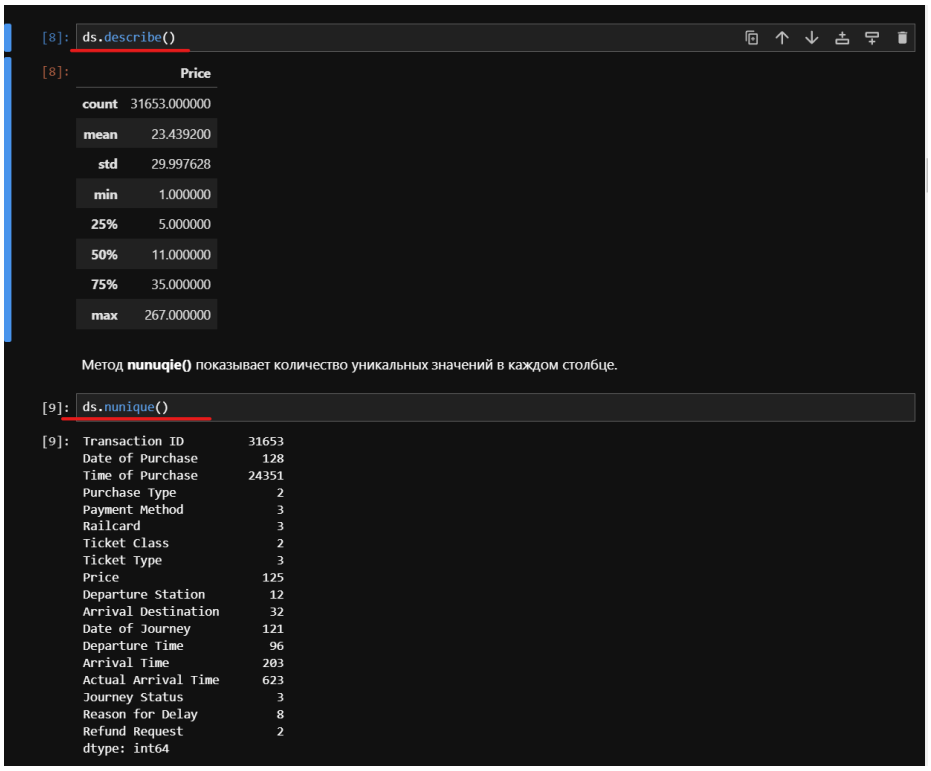


Рисунок 7 – Функция describe() и метод nunique()

Похожий метод на предыдущий – **unique()**, только он выводит конкретно все уникальные значения в виде массива, можно указать нужный столбец в квадратных скобках и одинарных кавычках после названия датасета. Исходя из результата вывода - всего три уникальных значения по столбцу с методом оплаты «Payment Method» - «Contactless», «Credit Card», «Debit Card». Также выводится тип данных, в этом случае – object (см. Рисунок 8 – Метод unique()).

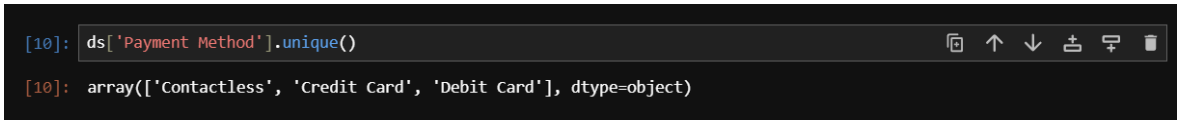


Рисунок 8 – Метод unique()

Метод **head()** выведет по умолчанию первые пять записей в датасете (см. Рисунок 9 – Метод head()).

```
[11]: ds.head()
```

[11]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination	Date of Journey
0	da8a6ba8-b3dc-4677-b176	2023-12-08	12:41:11	Online	Contactless	Adult	Standard	Advance	43	London Paddington	Liverpool Lime Street	2024-01-01
1	b0cdd1b0-f214-4197-be53	2023-12-16	11:23:01	Station	Credit Card	Adult	Standard	Advance	23	London Kings Cross	York	2024-01-01
2	f3ba7a96-f713-40d9-9629	2023-12-19	19:51:27	Online	Credit Card	NaN	Standard	Advance	3	Liverpool Lime Street	Manchester Piccadilly	2024-01-02
3	b2471f11-4fe7-4c87-8ab4	2023-12-20	23:00:36	Station	Credit Card	NaN	Standard	Advance	13	London Paddington	Reading	2024-01-01
4	2be00b45-0762-485e-a7a3	2023-12-27	18:22:56	Online	Contactless	NaN	Standard	Advance	76	Liverpool Lime Street	London Euston	2024-01-01

Рисунок 9 – Метод head()

Метод **tail()** противоположно вышеописанному методу выведет последние пять записей в датасете (см. Рисунок 10 – Метод tail()).

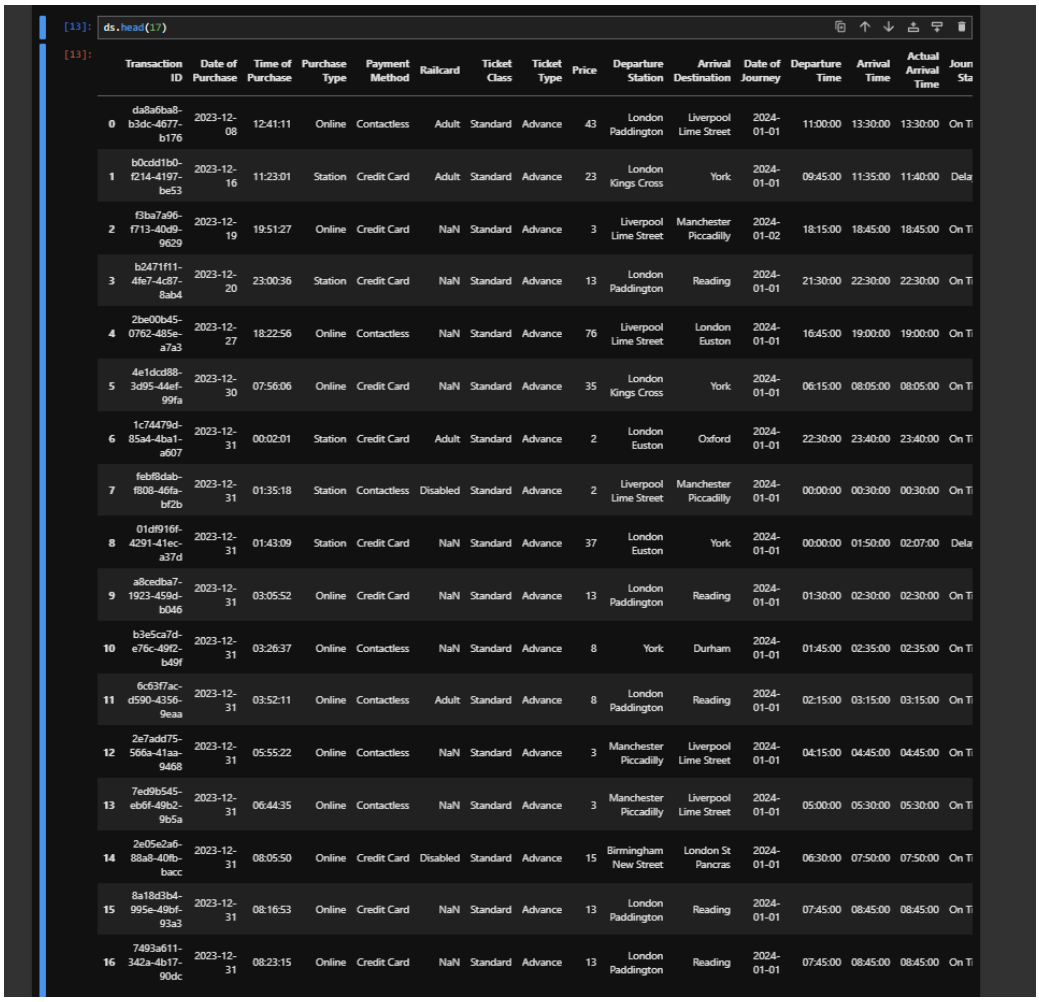
```
[12]: ds.tail()
```

[12]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination	Date Journ
31648	1304623d-b8b7-4999-8e9c	2024-04-30	18:42:58	Online	Credit Card	NaN	Standard	Off-Peak	4	Manchester Piccadilly	Liverpool Lime Street	20 04
31649	7da22246-f480-417c-bc2f	2024-04-30	18:46:10	Online	Contactless	NaN	Standard	Off-Peak	10	London Euston	Birmingham New Street	20 04
31650	add9debf-46c1-4c75-b52d	2024-04-30	18:56:41	Station	Credit Card	NaN	Standard	Off-Peak	4	Manchester Piccadilly	Liverpool Lime Street	20 04
31651	b92b047c-21fd-4859-966a	2024-04-30	19:51:47	Station	Credit Card	NaN	Standard	Off-Peak	10	London Euston	Birmingham New Street	20 04
31652	1d5d89a2-bde5-410f-8f91	2024-04-30	20:05:39	Station	Credit Card	Adult	Standard	Off-Peak	3	Liverpool Lime Street	Manchester Piccadilly	20 04

Рисунок 10 – Метод tail()

Оба вышеписанных метода принимают в себя параметр - число, то есть количество записей, которые нужно вывести с начала или с конца датасета. Например, можно в метод head() передать число 17 - он выведет первые 17 записей датасета (см. Рисунок 11 – Передача параметра в метод head()).



```
[13]: ds.head(17)
```

[13]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination	Date of Journey	Departure Time	Arrival Time	Actual Arrival Time	Jour Sta
0	da8a6ba8-b3dc-4677-b176	2023-12-08	12:41:11	Online	Contactless	Adult	Standard	Advance	43	London Paddington	Liverpool Lime Street	2024-01-01	11:00:00	13:30:00	13:30:00	On Ti
1	b0cdd1b0-f214-4197-be53	2023-12-16	11:23:01	Station	Credit Card	Adult	Standard	Advance	23	London Kings Cross	York	2024-01-01	09:45:00	11:35:00	11:40:00	Dela
2	f3ba7a96-f713-40d9-9629	2023-12-19	19:51:27	Online	Credit Card	NaN	Standard	Advance	3	Liverpool Lime Street	Manchester Piccadilly	2024-01-02	18:15:00	18:45:00	18:45:00	On Ti
3	b2471f11-4fe7-4c87-8ab4	2023-12-20	23:00:36	Station	Credit Card	NaN	Standard	Advance	13	London Paddington	Reading	2024-01-01	21:30:00	22:30:00	22:30:00	On Ti
4	2be00b45-0762-485e-a7a3	2023-12-27	18:22:56	Online	Contactless	NaN	Standard	Advance	76	Liverpool Lime Street	London Euston	2024-01-01	16:45:00	19:00:00	19:00:00	On Ti
5	4e1dcd88-3d95-44ef-99fa	2023-12-30	07:56:06	Online	Credit Card	NaN	Standard	Advance	35	London Kings Cross	York	2024-01-01	06:15:00	08:05:00	08:05:00	On Ti
6	1c74479d-85a4-4ba1-a607	2023-12-31	00:02:01	Station	Credit Card	Adult	Standard	Advance	2	London Euston	Oxford	2024-01-01	22:30:00	23:40:00	23:40:00	On Ti
7	febf8dab-f808-46fa-bf2b	2023-12-31	01:35:18	Station	Contactless	Disabled	Standard	Advance	2	Liverpool Lime Street	Manchester Piccadilly	2024-01-01	00:00:00	00:30:00	00:30:00	On Ti
8	01df916f-4291-41ec-a37d	2023-12-31	01:43:09	Station	Credit Card	NaN	Standard	Advance	37	London Euston	York	2024-01-01	00:00:00	01:50:00	02:07:00	Dela
9	a8cedba7-1923-459d-b046	2023-12-31	03:05:52	Online	Credit Card	NaN	Standard	Advance	13	London Paddington	Reading	2024-01-01	01:30:00	02:30:00	02:30:00	On Ti
10	b3e5ca7d-e76c-49f2-b49f	2023-12-31	03:26:37	Online	Contactless	NaN	Standard	Advance	8	York	Durham	2024-01-01	01:45:00	02:35:00	02:35:00	On Ti
11	6c63f7ac-d590-4356-9eaa	2023-12-31	03:52:11	Online	Contactless	Adult	Standard	Advance	8	London Paddington	Reading	2024-01-01	02:15:00	03:15:00	03:15:00	On Ti
12	2e7add75-566a-41aa-9468	2023-12-31	05:55:22	Online	Contactless	NaN	Standard	Advance	3	Manchester Piccadilly	Liverpool Lime Street	2024-01-01	04:15:00	04:45:00	04:45:00	On Ti
13	7ed9b545-eb6f-49b2-9b5a	2023-12-31	06:44:35	Online	Contactless	NaN	Standard	Advance	3	Manchester Piccadilly	Liverpool Lime Street	2024-01-01	05:00:00	05:30:00	05:30:00	On Ti
14	2e05e2a6-88a8-40fb-bacc	2023-12-31	08:05:50	Online	Credit Card	Disabled	Standard	Advance	15	Birmingham New Street	London St Pancras	2024-01-01	06:30:00	07:50:00	07:50:00	On Ti
15	8a18d3b4-995e-49bf-93a3	2023-12-31	08:16:53	Online	Credit Card	NaN	Standard	Advance	13	London Paddington	Reading	2024-01-01	07:45:00	08:45:00	08:45:00	On Ti
16	7493a611-342a-4b17-90dc	2023-12-31	08:23:15	Online	Credit Card	NaN	Standard	Advance	13	London Paddington	Reading	2024-01-01	07:45:00	08:45:00	08:45:00	On Ti

Рисунок 11 – Передача параметра в метод head()

4. Обработка, фильтрация, сортировка датасета.

С помощью метода isnull(), а также умножения среднего значения (mean) на 100 для получения процентного соотношения, можно посмотреть в нем же сводку по пропущенным значениям в датасете. Исходя из вывода - больше всего пропусков в столбцах «Railcard», «Actual Arrival Time» и «Reason for Delay», в остальных всё по нулям - пропусков нет (см. Рисунок 12 – Проверка на пустые значения).

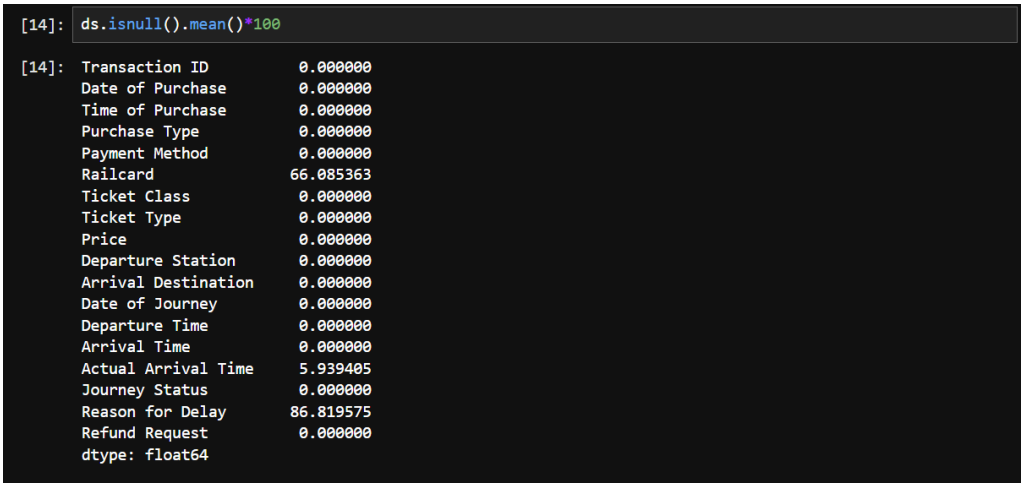
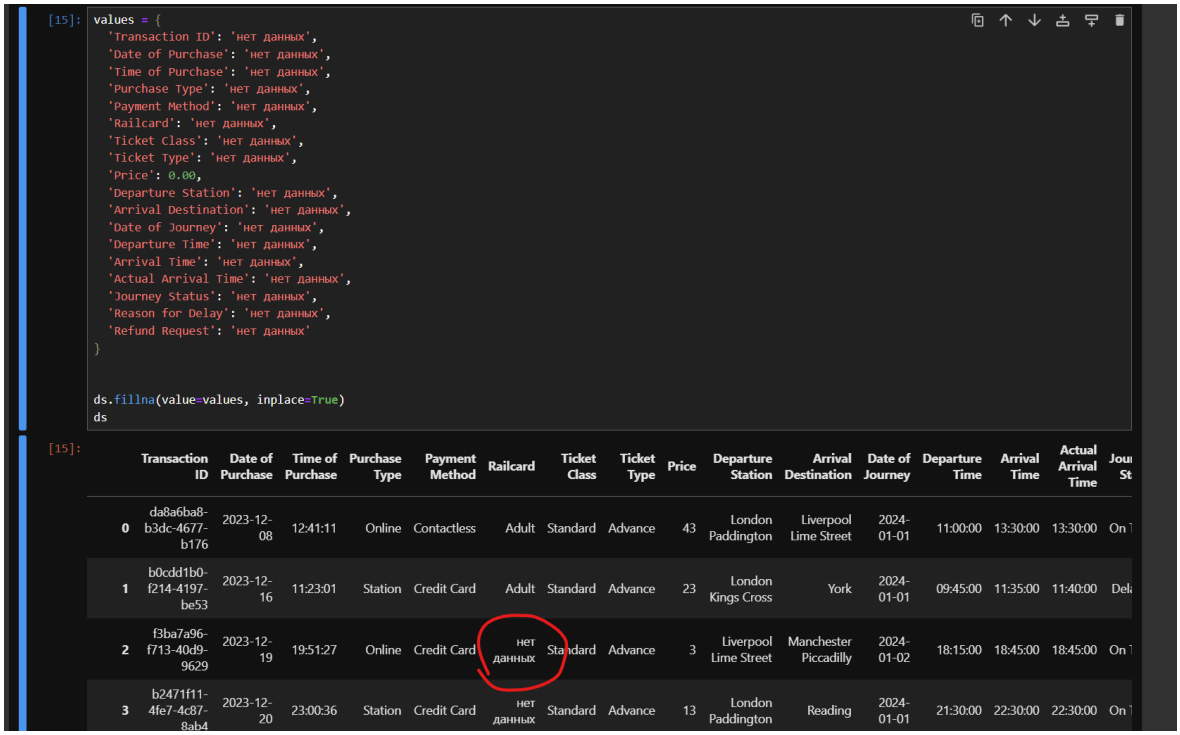


Рисунок 12 – Проверка на пустые значения

С помощью метода **fillna** в датасете пустые значения можно заменить на другие. Сам метод устанавливает два параметра - value - значения, на которые надо провести замену и inplace= True - внести значения в текущий датасет/False - создать новый, а текущий оставить без изменений. Для параметра values надо создать словарь в следующем формате – «столбец»: «данные, на которые будет проведена замена nan-значения». По сути нужно только три столбца - только в них найдены пустые значения, но для наглядности будут указаны все. 17 из 18 столбцов имеют значение типа object - строки, поэтому nan будут заменены на «нет данных». В столбце «Price» замена будет на 0.00 - это единственный столбец типа float. После повторного вывода данные будут изменены. Пустые значения заменены на «нет данных» (см. Рисунок 13 – Замена пустых значений).



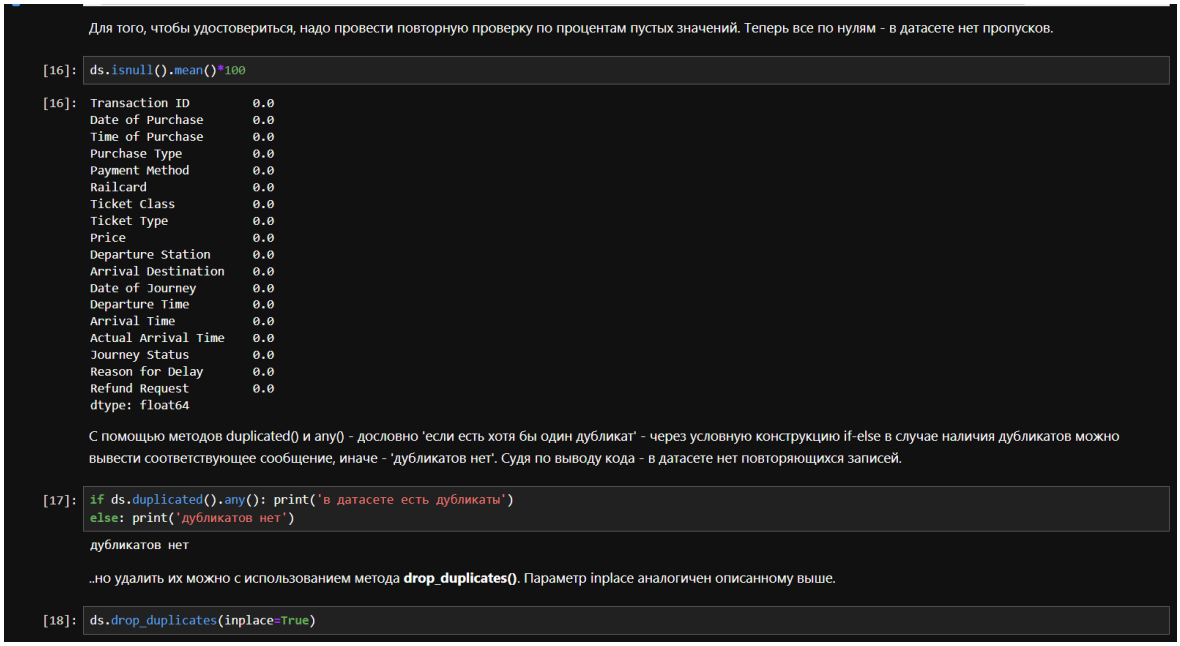
```
[15]: values = {
        'Transaction ID': 'нет данных',
        'Date of Purchase': 'нет данных',
        'Time of Purchase': 'нет данных',
        'Purchase Type': 'нет данных',
        'Payment Method': 'нет данных',
        'Railcard': 'нет данных',
        'Ticket Class': 'нет данных',
        'Ticket Type': 'нет данных',
        'Price': 0.00,
        'Departure Station': 'нет данных',
        'Arrival Destination': 'нет данных',
        'Date of Journey': 'нет данных',
        'Departure Time': 'нет данных',
        'Arrival Time': 'нет данных',
        'Actual Arrival Time': 'нет данных',
        'Journey Status': 'нет данных',
        'Reason for Delay': 'нет данных',
        'Refund Request': 'нет данных'
      }

      ds.fillna(value=values, inplace=True)
      ds
```

[15]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination	Date of Journey	Departure Time	Arrival Time	Actual Arrival Time	Jou St
0	da8a6ba8-b3dc-4677-b176	2023-12-08	12:41:11	Online	Contactless	Adult	Standard	Advance	43	London Paddington	Liverpool Lime Street	2024-01-01	11:00:00	13:30:00	13:30:00	On T
1	b0cdd1b0-f214-4197-be53	2023-12-16	11:23:01	Station	Credit Card	Adult	Standard	Advance	23	London Kings Cross	York	2024-01-01	09:45:00	11:35:00	11:40:00	Del
2	f3ba7a96-f713-40d9-9629	2023-12-19	19:51:27	Online	Credit Card	нет данных	Standard	Advance	3	Liverpool Lime Street	Manchester Piccadilly	2024-01-02	18:15:00	18:45:00	18:45:00	On T
3	b2471f11-4fe7-4c87-8ab4	2023-12-20	23:00:36	Station	Credit Card	нет данных	Standard	Advance	13	London Paddington	Reading	2024-01-01	21:30:00	22:30:00	22:30:00	On T

Рисунок 13 – Замена пустых значений

После повторной проверки на пустые значения всё будет по нулям – датасет полностью без пропусков. Дубликаты можно удалить методом «**drop_duplicates**» при их наличии в датасете (см. Рисунок 14 – Повторная проверка на пустые значения и удаление дубликатов).



Для того, чтобы удостовериться, надо провести повторную проверку по процентам пустых значений. Теперь все по нулям - в датасете нет пропусков.

```
[16]: ds.isnull().mean()*100
```

```
[16]: Transaction ID         0.0
      Date of Purchase       0.0
      Time of Purchase       0.0
      Purchase Type          0.0
      Payment Method         0.0
      Railcard               0.0
      Ticket Class           0.0
      Ticket Type            0.0
      Price                  0.0
      Departure Station      0.0
      Arrival Destination    0.0
      Date of Journey        0.0
      Departure Time         0.0
      Arrival Time           0.0
      Actual Arrival Time    0.0
      Journey Status         0.0
      Reason for Delay       0.0
      Refund Request         0.0
      dtype: float64
```

С помощью методов duplicated() и any() - дословно 'если есть хотя бы один дубликат' - через условную конструкцию if-else в случае наличия дубликатов можно вывести соответствующее сообщение, иначе - 'дубликатов нет'. Судя по выводу кода - в датасете нет повторяющихся записей.

```
[17]: if ds.duplicated().any(): print('в датасете есть дубликаты')
      else: print('дубликатов нет')
```

```
дубликатов нет
```

..но удалить их можно с использованием метода **drop_duplicates()**. Параметр inplace аналогичен описанному выше.

```
[18]: ds.drop_duplicates(inplace=True)
```

Рисунок 14 – Повторная проверка на пустые значения и удаление дубликатов

Чтобы провести срез данных, есть несколько способов.

Первый - выбрать столбцы из датасета и вывести нужные. Например, столбцы цены, типа оплаты и времени прибытия по расписанию. Получится датафрейм из трех столбцов (рядом индекс каждой записи), количество записей не менялось (см. Рисунок 15 – Срез: первый способ).

```
[19]: ds[['Price', 'Purchase Type', 'Arrival Time']]
```

[19]:

	Price	Purchase Type	Arrival Time
0	43	Online	13:30:00
1	23	Station	11:35:00
2	3	Online	18:45:00
3	13	Station	22:30:00
4	76	Online	19:00:00
...	...	...	...
31648	4	Online	20:30:00
31649	10	Online	21:35:00
31650	4	Station	20:45:00
31651	10	Station	22:35:00
31652	3	Station	22:00:00

31653 rows × 3 columns

Рисунок 15 – Срез: первый способ

Второй способ реализации среза данных - функция **loc**. Он выведет записи в указанном через двоеточие диапазоне, то же самое можно сделать и с количеством столбцов. В примере ниже выводятся строки от 5 до 50, также выводятся столбцы с «Purchase Type» по «Price». Всего вывелось шесть столбцов (см. Рисунок 16 – Срез: второй способ).

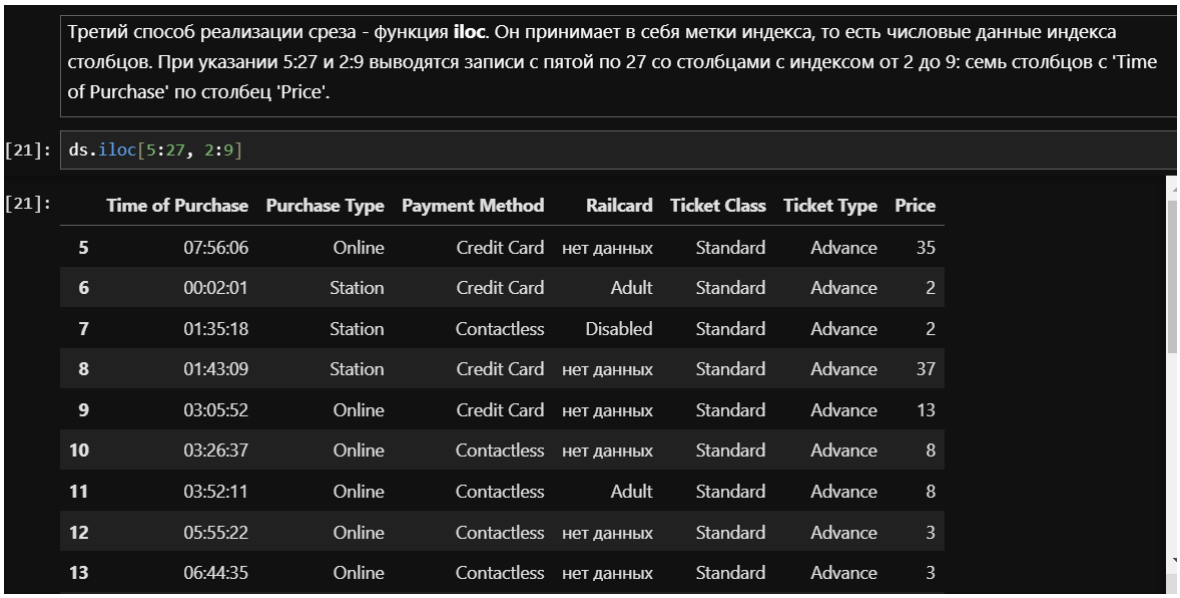
```
[20]: ds.loc[5:50, 'Purchase Type':'Price']
```

[20]:

	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price
5	Online	Credit Card	нет данных	Standard	Advance	35
6	Station	Credit Card	Adult	Standard	Advance	2
7	Station	Contactless	Disabled	Standard	Advance	2
8	Station	Credit Card	нет данных	Standard	Advance	37
9	Online	Credit Card	нет данных	Standard	Advance	13
10	Online	Contactless	нет данных	Standard	Advance	8
11	Online	Contactless	Adult	Standard	Advance	8
12	Online	Contactless	нет данных	Standard	Advance	3
13	Online	Contactless	нет данных	Standard	Advance	3
14	Online	Credit Card	Disabled	Standard	Advance	15
15	Online	Credit Card	нет данных	Standard	Advance	13

Рисунок 16 – Срез: второй способ

Третий способ реализации среза - функция **iloc**. Он принимает в себя метки индекса, то есть числовые данные индекса столбцов. При указании 5:27 и 2:9 выводятся записи с пятой по 27 со столбцами с индексом от 2 до 9: семь столбцов с «Time of Purchase» по столбец «Price» (см. Рисунок 17 – Срез: третий способ).



Третий способ реализации среза - функция **iloc**. Он принимает в себя метки индекса, то есть числовые данные индекса столбцов. При указании 5:27 и 2:9 выводятся записи с пятой по 27 со столбцами с индексом от 2 до 9: семь столбцов с 'Time of Purchase' по столбец 'Price'.

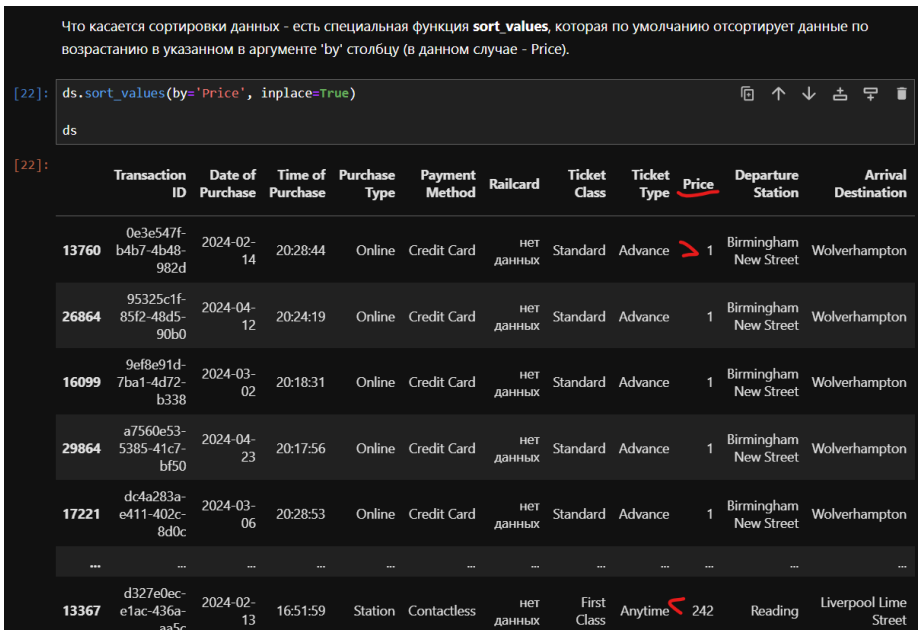
```
[21]: ds.iloc[5:27, 2:9]
```

[21]:

	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price
5	07:56:06	Online	Credit Card	нет данных	Standard	Advance	35
6	00:02:01	Station	Credit Card	Adult	Standard	Advance	2
7	01:35:18	Station	Contactless	Disabled	Standard	Advance	2
8	01:43:09	Station	Credit Card	нет данных	Standard	Advance	37
9	03:05:52	Online	Credit Card	нет данных	Standard	Advance	13
10	03:26:37	Online	Contactless	нет данных	Standard	Advance	8
11	03:52:11	Online	Contactless	Adult	Standard	Advance	8
12	05:55:22	Online	Contactless	нет данных	Standard	Advance	3
13	06:44:35	Online	Contactless	нет данных	Standard	Advance	3

Рисунок 17 – Срез: третий способ

Что касается сортировки данных - есть специальная функция **sort_values**, которая по умолчанию отсортирует данные по возрастанию в указанном в аргументе «by» столбцу, в данном случае – Price (см. Рисунок 18 – Сортировка 1).



Что касается сортировки данных - есть специальная функция **sort_values**, которая по умолчанию отсортирует данные по возрастанию в указанном в аргументе 'by' столбцу (в данном случае - Price).

```
[22]: ds.sort_values(by='Price', inplace=True)

      ds
```

[22]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination
13760	0e3e547f-b4b7-4b48-982d	2024-02-14	20:28:44	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
26864	95325c1f-85f2-48d5-90b0	2024-04-12	20:24:19	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
16099	9ef8e91d-7ba1-4d72-b338	2024-03-02	20:18:31	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
29864	a7560e53-5385-41c7-bf50	2024-04-23	20:17:56	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
17221	dc4a283a-e411-402c-8d0c	2024-03-06	20:28:53	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
...	...	...	...	...	...	...	...	...	...	...	...
13367	d327e0ec-e1ac-436a-aa5c	2024-02-13	16:51:59	Station	Contactless	нет данных	First Class	Anytime	242	Reading	Liverpool Lime Street

Рисунок 18 – Сортировка 1

Если в параметре ascending указать значение «True», то данные отсортируются также по возрастанию - от самого начала дня во времени оплаты и до самого его конца - до 23 часов и 59 минут (см. Рисунок 19 – Сортировка 2).

Если в параметре **ascending** указать значение 'True', то данные отсортируются также по возрастанию - от самого начала дня во времени оплаты и до самого его конца (до 23 часов и 59 минут).

```
23]: ds.sort_values(by='Time of Purchase', inplace=True, ascending=True)

     ds
```

23]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination	Da Jou
30437	8d30844c-9e88-4ce8-8299	2024-04-26	00:00:09	Online	Contactless	Adult	First Class	Off-Peak	36	York	Edinburgh	2 (
3061	b442b2cb-e097-4b6e-bdca	2024-01-13	00:00:09	Online	Credit Card	нет данных	Standard	Off-Peak	19	London Paddington	Reading	2 (
15558	b7ebc375-9e5c-4f68-bef9	2024-02-25	00:00:10	Station	Credit Card	Adult	Standard	Advance	2	London Euston	Oxford	2 (
21815	89445b1c-b1e0-4726-9855	2024-03-24	00:00:14	Online	Contactless	нет данных	Standard	Off-Peak	14	Manchester Piccadilly	Leeds	2 (
18887	31160c17-61b0-432c-a766	2024-03-13	00:00:20	Online	Contactless	нет данных	Standard	Off-Peak	14	Manchester Piccadilly	Leeds	2 (
...	...	...	...	...	...	...	...	...	...	...	...	
11394	2e58ff14-c0b6-4ef6-832d	2024-02-07	23:59:43	Station	Credit Card	Adult	Standard	Advance	56	Manchester Piccadilly	London Euston	2 (

Рисунок 19 – Сортировка 2

Если указать в параметре ascending значение False - будут выведены значения по убыванию, в данном случае от самой большой цены и до самой маленькой (см. Рисунок 20 – Сортировка 3).

```
[24]: ds.sort_values(by='Price', inplace=True, ascending=False)

      ds
```

[24]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination
711	092e5598-08de-42f5-b6b2	2024-01-04	06:35:01	Station	Credit Card	нет данных	First Class	Anytime	267	Manchester Piccadilly	London Euston
11849	de723682-3979-4d69-9664	2024-02-09	06:30:59	Station	Credit Card	нет данных	First Class	Anytime	267	Manchester Piccadilly	London Euston
2042	05193f47-2107-4bb8-8adb	2024-01-09	06:33:37	Station	Credit Card	нет данных	First Class	Anytime	267	Manchester Piccadilly	London Euston
13367	d327e0ec-e1ac-436a-aa5c	2024-02-13	16:51:59	Station	Contactless	нет данных	First Class	Anytime	242	Reading	Liverpool Lime Street
31434	a4bbac34-6ed7-4d71-b738	2024-04-29	16:49:20	Station	Contactless	нет данных	First Class	Anytime	242	Reading	Liverpool Lime Street
...	...	...	...	...	...	...	...	...	...	...	...
15183	7b5230e6-5413-4f34-9a0f	2024-02-21	20:21:16	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton

Рисунок 20 – Сортировка 3

Также с указанием столбца в датасете можно вывести только записи с определенном в нем значении. Для этого используется оператор двойного «равно». В выводе ниже будут показаны все записи датасета, где в столбце типа оплаты стоит значение «Online» (см. Рисунок 21 – Фильтрация).

```
[25]: ds[ds['Purchase Type'] == 'Online']
```

[25]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination
24288	7484b3b7-70dc-4ca7-9f93	2024-04-03	05:57:39	Online	Credit Card	нет данных	First Class	Anytime	216	London Euston	Manchester Piccadilly
29564	b8f91da1-380b-42c5-81b7	2024-04-22	17:21:36	Online	Credit Card	нет данных	First Class	Anytime	216	London Euston	Manchester Piccadilly
15645	df9d431f-02c5-4c85-bb6d	2024-02-26	05:00:07	Online	Debit Card	нет данных	First Class	Anytime	216	London Euston	Manchester Piccadilly
17698	16c9b1c4-1c53-4e61-ae8b	2024-03-08	17:24:48	Online	Credit Card	нет данных	First Class	Anytime	216	London Euston	Manchester Piccadilly
949	9c1e0d3f-caed-4775-8bdc	2024-01-05	05:01:20	Online	Debit Card	нет данных	First Class	Anytime	216	London Euston	Manchester Piccadilly
...	...	...	...	...	...	...	...	...	...	...	...

Рисунок 21 – Фильтрация

Метод **query** выведет все строки с заданным условием, например, все записи, где цена равна единице (см. Рисунок 22 – Метод query).

```
[26]: ds.query('Price == 1')
```

[26]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination
16099	9ef8e91d-7ba1-4d72-b338	2024-03-02	20:18:31	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
7286	5e651fd0-c290-4b81-974b	2024-01-27	20:16:46	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
29864	a7560e53-5385-41c7-bf50	2024-04-23	20:17:56	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
28221	6d0d4ead-07fc-4fe8-951f	2024-04-17	20:24:48	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
21287	4e0d5a64-7c36-45d0-a686	2024-03-21	20:16:38	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton
2488	39195f02-ae74-4902-bffd	2024-01-10	20:29:46	Online	Credit Card	нет данных	Standard	Advance	1	Birmingham New Street	Wolverhampton

Рисунок 22 – Метод query

5. Группировка данных и топ-5 записей.

Метод **groupby** сгруппирует данные по указанным условиям. В данном случае группировка идет по столбцу "Payment Method" - в нем всего три уникальных значения (Contactless, Credit Card и Debit Card). С помощью агрегатной функции size() вернет размерность столбца, то есть будут показаны количественные данные по каждому методу оплаты. Результат конвертируется в датафрейм функцией to_frame с добавлением названия для нового столбца с данными - Count. Если сложить все числа получится 31653 - это и есть количество записей в датасете, метод groupby просто сгруппировал их по каждому уникальному значению - по категории.

Сгруппировать данные можно также по количеству поездок в каждую дату, для этого используется агрегатная функция count. Столбец с получившимися данными переименовывается в count trips - количество поездок. Для подсчета используется столбец «Transaction ID», так как это по сути определитель факта происходящей поездки (см. Рисунок 23 – Группировка данных через grouby).

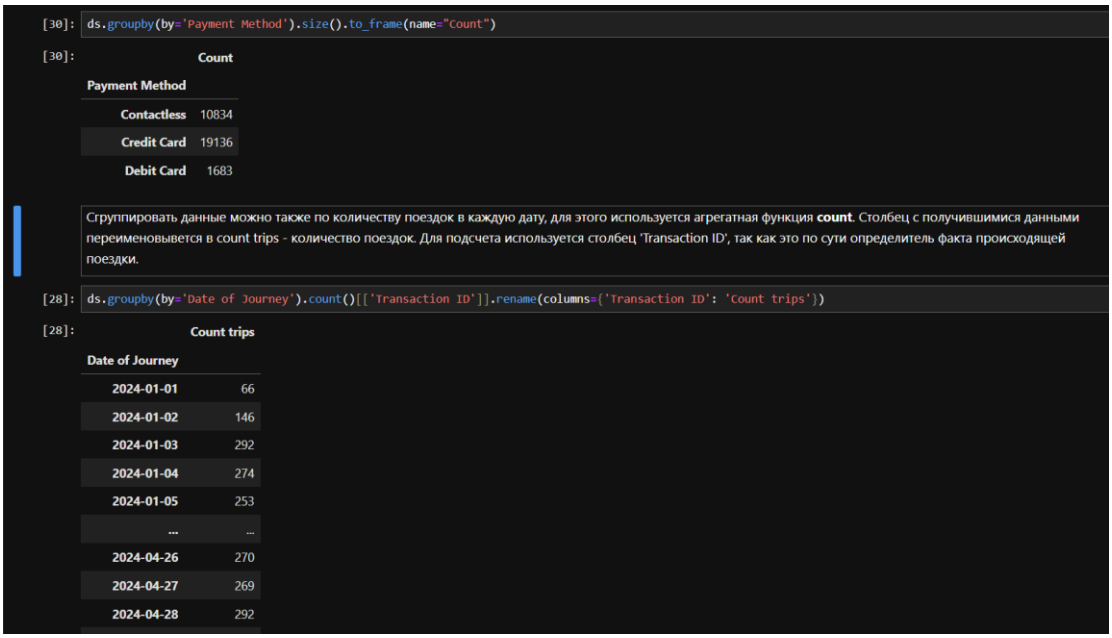
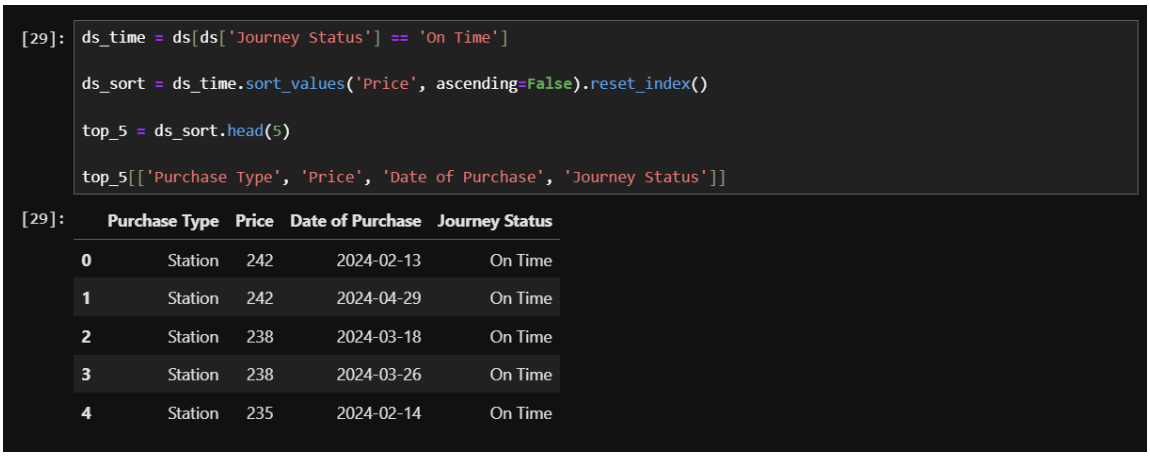


Рисунок 23 – Группировка данных через groupby

Для вывода топ-5 записей надо подготовить датасет. Сначала в отдельную переменную сохраняется датасет с отфильтрованным столбцом «Journey Status» по значению «On Time». Далее в новую переменную датасет сортируется по столбцу «Price» по убыванию, далее сбрасывается индекс (reset_index()), чтобы счет шел с нуля. Из данного датасета с фильтрацией и сортировкой берутся первые пять записей со столбцами «Purchase Type», «Price», «Date Of Purchase», «Journey Status».

В выводе оказываются пять записей - топ-5 самых дорогих поездок, совершенных «вовремя» - по расписанию. Самый дорогой билет - имеет значение 242, наиболее дешёвый из топа – 235. Выбор данных был основан на более важных столбцах – конкретно статус поездки – по расписанию или нет (такое можно предсказать и взять за целевую переменную), а также цена поездки как столбец, на который влияют остальные (см. Рисунок 24 – Топ-5 записей).



[29]:
```
ds_time = ds[ds['Journey Status'] == 'On Time']

ds_sort = ds_time.sort_values('Price', ascending=False).reset_index()

top_5 = ds_sort.head(5)

top_5[['Purchase Type', 'Price', 'Date of Purchase', 'Journey Status']]
```

[29]:

	Purchase Type	Price	Date of Purchase	Journey Status
0	Station	242	2024-02-13	On Time
1	Station	242	2024-04-29	On Time
2	Station	238	2024-03-18	On Time
3	Station	238	2024-03-26	On Time
4	Station	235	2024-02-14	On Time

Рисунок 24 – Топ-5 записей

Еще один топ-5 по одному столбцу можно вывести следующим образом: для начала сгруппировать столбец станций отправления по количеству поездок через функцию **count()** и метод **groupby**. Затем отсортировать по убыванию и взять первые пять записей – наиболее популярные станции отправления по поездкам. Результат помещается в отдельную переменную и выводится в виде датафрейма с подписями столбцов и с индексами (см. Рисунок 25 – Топ-5 записей по столбцу).

```
[42]: station_counts = ds.groupby("Departure Station")["Departure Station"].count()

      sorted_counts = station_counts.sort_values(ascending=False)

      top_5_2 = sorted_counts.head(5).to_frame(name="количество поездок")
      top_5_2.index.name = "станция отправления"

      top_5_2 = top_5_2.reset_index()

      top_5_2
```

[42]:

	станция отправления	количество поездок
0	Manchester Piccadilly	5650
1	London Euston	4954
2	Liverpool Lime Street	4561
3	London Paddington	4500
4	London Kings Cross	4229

Рисунок 25 – Топ-5 записей по столбцу

Сохранить датасет с внесенными изменениями можно с помощью метода «to_csv», выдав ему новое название отдельного файла. index=False говорит о том, что строковый индекс не будет записан в файл. Файл нового датасета можно вывести и увидеть, что он будет в той же папке, что и оригинальный датасет (см. Рисунок 26 – Сохранение датасета).

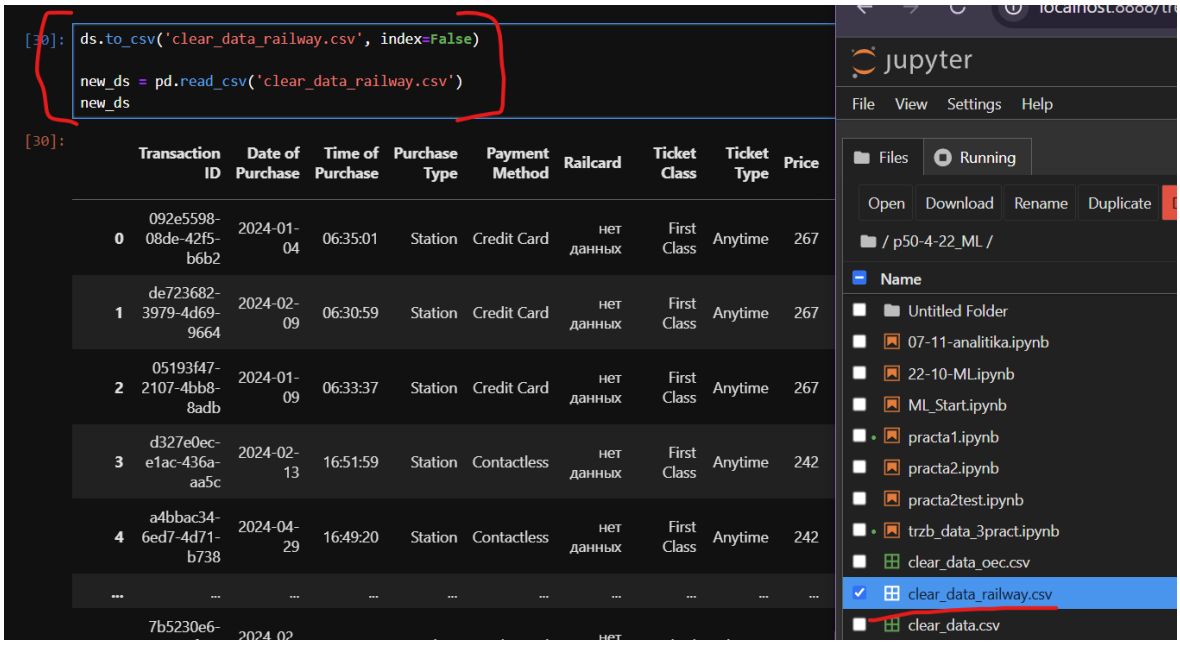


Рисунок 26 – Сохранение датасета

Вывод: в ходе выполнения данной практической работы с использованием python-библиотеки pandas была просмотрена информация по датасету, произведена его обработка, выведены первые и последние пять записей, проведён срез данных, фильтрация и сортировка, выведены первичные статистические характеристики датасета, проведена группировка по одному столбцу и вызвана одна из агрегатных функций, также был выведен и обоснован топ-5 записей, весь ход работы описан в отчёте.