Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5
дисциплины
«Искусственный интеллект и машинное обучение»
Вариант 12

Выполнил:
Рябинин Егор Алексеевич
2 курс, группа ИВТ-б-о-23-2,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г

Тема: Введение в pandas: изучение структуры DataFrame и базовых операций.

Цель: познакомить с основами работы с библиотекой pandas, в частности, со структурой данных DataFrame.

Порядок выполнения работы:

Ссылка на репозиторий GitHib:

https://github.com/EgorGorilla/Lab5_Artificial-Intelligence-and-Machine-Learning

Таблица 1 – Данные о сотрудниках

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	Иван	25	Инженер	IT	60000	2
1	2	Ольга	30	Аналитик	Маркетинг	75000	5
2	3	Алексей	40	Менеджер	Продажи	90000	15
3	4	Мария	35	Программист	IT	80000	7
4	5	Сергей	28	Специалист	HR	50000	3
5	6	Анна	32	Разработчик	IT	85000	6
6	7	Дмитрий	45	HR	HR	48000	12
7	8	Елена	29	Маркетолог	Маркетинг	70000	4
8	9	Виктор	31	Юрист	Юридический	95000	10
9	10	Алиса	27	Дизайнер	Дизайн	62000	5
10	11	Павел	33	Администратор	Администрация	55000	7
11	12	Светлана	26	Тестировщик	Тестирование	67000	2
12	13	Роман	42	Финансист	Финансы	105000	20
13	14	Татьяна	37	Редактор	Редакция	72000	9
14	15	Николай	39	Логист	Логистика	75000	11
15	16	Валерия	24	SEO-специалист	SEO	64000	3
16	17	Григорий	50	Бухгалтер	Бухгалтерия	110000	25
17	18	Юлия	45	Директор	Финансы	150000	20
18	19	Степан	35	Экономист	Экономика	98000	14
19	20	Василиса	31	Проект-менеджер	Продажи	88000	8

Таблица 2 – Данные о клиентах

	ID	Имя	Возраст	Город	Баланс на счете	Кредитная история
0	1	Иван	34	Москва	120000	Хорошая
1	2	Ольга	27	Санкт-Петербург	80000	Средняя
2	3	Алексей	45	Казань	150000	Плохая
3	4	Мария	38	Новосибирск	200000	Хорошая
4	5	Сергей	29	Екатеринбург	95000	Средняя
5	6	Анна	50	Воронеж	300000	Отличная
6	7	Дмитрий	31	Челябинск	140000	Средняя
7	8	Елена	40	Краснодар	175000	Хорошая
8	9	Виктор	28	Ростов-на-Дону	110000	Плохая
9	10	Алиса	33	Уфа	98000	Средняя
10	11	Павел	46	Омск	250000	Хорошая
11	12	Светлана	37	Пермь	210000	Отличная
12	13	Роман	41	Тюмень	135000	Средняя
13	14	Татьяна	25	Саратов	155000	Хорошая
14	15	Николай	39	Самара	125000	Средняя
15	16	Валерия	42	Волгоград	180000	Плохая
16	17	Григорий	49	Барнаул	275000	Отличная
17	18	Юлия	50	Иркутск	320000	Хорошая
18	19	Степан	30	Хабаровск	105000	Средняя
19	20	Василиса	35	Томск	90000	Плохая

Таблица 3 – Данные с пропусками

	ID	Имя	Возраст	Город	Баланс на счете	Кредитная история
0	1	Иван	34.0	Москва	120000.0	Хорошая
1	2	Ольга	27.0	Санкт-Петербург	NaN	Средняя
2	3	Алексей	NaN	Казань	150000.0	Плохая
3	4	Мария	38.0	NaN	200000.0	Хорошая
4	5	Сергей	29.0	Екатеринбург	NaN	NaN
5	6	NaN	50.0	Воронеж	300000.0	Отличная
6	7	Дмитрий	31.0	NaN	140000.0	Средняя
7	8	Елена	40.0	Краснодар	175000.0	Хорошая
8	9	Виктор	NaN	Ростов-на-Дону	110000.0	NaN
9	10	Алиса	33.0	Уфа	NaN	Средняя
10	11	Павел	46.0	Омск	250000.0	Хорошая
11	12	NaN	37.0	Пермь	210000.0	Отличная
12	13	Роман	41.0	NaN	135000.0	Средняя
13	14	Татьяна	25.0	Саратов	155000.0	NaN
14	15	Николай	NaN	Самара	125000.0	Средняя
15	16	Валерия	42.0	NaN	NaN	Плохая
16	17	Григорий	49.0	Барнаул	275000.0	Отличная
17	18	Юлия	NaN	Иркутск	320000.0	Хорошая
18	19	Степан	30.0	Хабаровск	105000.0	Средняя
19	20	Василиса	35.0	Томск	90000.0	Плохая

Задание №1. Создание DataFrame разными способами.

1.1. Создайте DataFrame из словаря списков с данными о пяти сотрудниках (табл. 1).

```
# 1. Создайте DataFrame из словаря списков с данными о пяти сотрудника
import numpy as np
import pandas as pd
from IPython.display import display
data = {
    'ID': [1,2,3,4,5],
    'Имя': ['Иван','Ольга','Алексей','Мария','Сергей'],
    "Возраст": [25, 30, 40, 35, 28],
    "Должность": ["Инженер", "Аналитик", "Менеджер", "Программист", "С
    'Отдел': ['IT',"Маркетинг","Продажи",'IT','HR'],
    'Зарплата': [60000, 75000, 90000, 80000, 50000],
    'Стаж работы': [2,5,15,7,3]
}
df = pd.DataFrame(data)
display(df)
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	Иван	25	Инженер	IT	60000	2
1	2	Ольга	30	Аналитик	Маркетинг	75000	5
2	3	Алексей	40	Менеджер	Продажи	90000	15
3	4	Мария	35	Программист	IT	80000	7
4	5	Сергей	28	Специалист	HR	50000	3

Рисунок 1 – Задание №1.1

1.2. Создайте DataFrame из списка словарей с теми же данными.

```
# 2. Создайте DataFrame из списка словарей с теми же данными.
data = [
    {"ID": "1", "Возраст": 25, "Должность": "Инженер", "Отдел": "IT",
    {"ID": "2", "Возраст": 30, "Должность": "Аналитик", "Отдел": "Марк
    {"ID": "3", "Возраст": 40, "Должность": "Менеджер", "Отдел": "Прод
    {"ID": "4", "Возраст": 35, "Должность": "Программист", "Отдел": "I
    {"ID": "5", "Возраст": 28, "Должность": "Специалист", "Отдел": "HR
]
df = pd.DataFrame(data)
display(df)
```

	ID	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	25	Инженер	IT	60000	2
1	2	30	Аналитик	Маркетинг	75000	5
2	3	40	Менеджер	Продажи	90000	15
3	4	35	Программист	IT	80000	7
4	5	28	Специалист	HR	50000	3

Рисунок 2 – Задание №1.2

1.3. Создайте DataFrame из массива NumPy, заполненного случайными числами от 20 до 60 (для возраста сотрудников).

```
# 3. Создайте DataFrame из массива NumPy, заполненного случайными числ
data = np.array([
    [1, "Иван", 25, "Инженер", "IT", 60000, 2],
    [2, "Ольга", 30, "Аналитик", "Маркетинг", 75000, 5],
    [3, "Алексей", 40, "Менеджер", "Продажи", 90000, 15],
    [4, "Мария", 35, "Программист", "IT", 80000, 7],
    [5, "Сергей", 28, "Специалист", "HR", 50000, 3],
    [6, "Анна", 32, "Разработчик", "IT", 85000, 6],
    [7, "Дмитрий", 45, "HR", "HR", 48000, 12],
    [8, "Елена", 29, "Маркетолог", "Маркетинг", 70000, 4],
    [9, "Виктор", 31, "Юрист", "Юридический", 95000, 10],
    [10, "Алиса", 27, "Дизайнер", "Дизайн", 62000, 5],
    [11, "Павел", 33, "Администратор", "Администрация", 55000, 7],
    [12, "Светлана", 26, "Тестировщик", "Тестирование", 67000, 2],
    [13, "Роман", 42, "Финансист", "Финансы", 105000, 20],
    [14, "Татьяна", 37, "Редактор", "Редакция", 72000, 9],
    [15, "Николай", 39, "Логист", "Логистика", 75000, 11],
    [16, "Валерия", 24, "SEO-специалист", "SEO", 64000, 3],
    [17, "Григорий", 50, "Бухгалтер", "Бухгалтерия", 110000, 25],
    [18, "Юлия", 45, "Директор", "Финансы", 150000, 20],
    [19, "Степан", 35, "Экономист", "Экономика", 98000, 14],
    [20, "Василиса", 31, "Проект-менеджер", "Продажи", 88000, 8]
])
data[:, 2] = np.random.randint(20, 61, size=data.shape[0])
df_random = pd.DataFrame(data, columns=["ID", "Имя", "Возраст", "Должн
display(df_random)
```

Рисунок 3 – Задание №1.3

1.4. Проверьте типы данных в каждом столбце с помощью .info().

```
df_random.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   ID           20 non-null     object
 1   Имя          20 non-null     object
 2   Возраст      20 non-null     object
 3   Должность    20 non-null     object
 4   Отдел        20 non-null     object
 5   Зарплата     20 non-null     object
 6   Стаж работы  20 non-null     object
dtypes: object(7)
memory usage: 1.2+ KB
```

Рисунок 4 – Задание №1.4

Задание №2. Чтение данных из файлов (CSV, Excel, JSON).

2.1. Сохраните Таблицу 1 в CSV и затем загрузите ее обратно в DataFrame.

```
df_random.to_csv("table_csv.csv", index=False)
df_csv = pd.read_csv("table_csv.csv")
display(df_csv.head(5))
```

Рисунок 5 – Задание №2.1

2.2. Запишите Таблицу 2 в Excel (data.xlsx, лист "Клиенты") и прочитайте ее в DataFrame.

```
# Создание DataFrame и сохранение его в Excel

data_2 = {
    "ID": list(range(1, 21)),
    "Имя": ["Иван", "Ольга", "Алексей", "Мария", "Сергей", "Анна", "Дм
            "Павел", "Светлана", "Роман", "Татьяна", "Николай", "Валер
    "Возраст": [34, 27, 45, 38, 29, 50, 31, 40, 28, 33, 46, 37, 41, 25
    "Город": ["Москва", "Санкт-Петербург", "Казань", "Новосибирск", "Е
              "Ростов-на-Дону", "Уфа", "Омск", "Пермь", "Тюмень", "Сар
              "Хабаровск", "Томск"],
    "Баланс на счете": [120000, 80000, 150000, 200000, 95000, 300000,
                        250000, 210000, 135000, 155000, 125000, 18000
    "Кредитная история": ["Хорошая", "Средняя", "Плохая", "Хорошая", "
                          "Плохая", "Средняя", "Хорошая", "Отличная",
                          "Отличная", "Хорошая", "Средняя", "Плохая"]
}
df2 = pd.DataFrame(data_2)
df2.to_excel("data.xlsx", sheet_name="Клиенты", index=False)

# Чтение data.xlsx в новый DataFrame

df_excel = pd.read_excel("data.xlsx", sheet_name="Клиенты")
display(df_excel.head(5))
```

Рисунок 6 – Задание №2.2

2.3. Экспортируйте Таблицу 1 в формат JSON и затем прочитайте ее обратно в DataFrame

```
df_random.to_json('table.json', orient='records', indent=4)
df_json = pd.read_json('table.json')
display(df_json.head(5))
```

Рисунок 7 – Задание №2.3

Задание №3. Доступ к данным (.iloc, .loc, .at, .iat).

3.1. Получите информация о сотруднике с ID = 5 с помощью .loc[].

```
display(df_csv.loc[df_csv["ID"] == 5])
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
4	5	Сергей	57	Специалист	HR	50000	3

Рисунок 8 – Задание №3.1

3.2. Выведите возраст третьего сотрудника в таблице с .iloc[].

```
print(df_csv.loc[2]["Возраст"])
```

```
23
```

Рисунок 9 – Задание №3.2

3.3. Выведите название отдела для сотрудника "Мария" с .at[].

```
print(df_csv.at[3, "Отдел"])
```

```
IT
```

Рисунок 10 – Задание №3.3

3.4. Выведите зарплату сотрудника, находящегося в четвертой строке и пятом столбце, используя .iat[].

```
print(df_csv.iat[3,5])
```

```
80000
```

Рисунок 11 – Задание №3.4

Задание №4. Добавление новых столбцов и строк.

4.1. Добавьте новый столбец "Категория зарплаты":

- "Низкая" - если Зарплата $<$ 60000.
- "Средняя" - если 60000 $\leq$ Зарплата $<$ 100000.
- "Высокая" - если Зарплата $\geq$ 100000.

```
df_csv["Категория зарплаты"] = df_csv["Зарплата"].apply(
    lambda x: "Низкая" if x < 60000
              else "Средняя" if 60000 <= x < 100000
              else "Высокая"
)
display(df_csv)
```

Рисунок 12 – Задание №4.1

4.2. Добавьте нового сотрудника с данными:

```
df_csv.loc[20] = [21,"Антон", 32, "Разработчик", "IT", 85000, 6,"Средн
display(df_csv.tail(3))
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы	Ка за
18	19	Степан	46	Экономист	Экономика	98000	14	
19	20	Василиса	23	Проект-менеджер	Продажи	88000	8	
20	21	Антон	32	Разработчик	IT	85000	6	

Рисунок 13 – Задание №4.2

4.3. Добавьте двух новых сотрудников одновременно, используя pd.concat()

```
new_data ={
    "ID": [22, 23],
    "Имя": ["Альфредо", "Егор"],
    "Возраст": [45, 19],
    "Должность": ["Разработчик", "SEO-специалист"],
    "Отдел": ["IT", "Продажи"],
    "Зарплата": [170000, 90000],
    "Стаж работы": [26, 2],
    "Категория зарплаты":["Высокая", "Средняя"]
}
couple = pd.DataFrame(new_data)
df_csv = pd.concat([df_csv, couple], ignore_index=True)
display(df_csv.tail())
```

Рисунок 14 – Задание №4.3

Задание №5. Удаление строк и столбцов.

5.1. Удалите столбец "Категория зарплаты" с .drop().

```
df_csv=df_csv.drop(columns=["Категория зарплаты"])
display(df_csv.head(3))
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	Иван	27	Инженер	IT	60000	2
1	2	Ольга	44	Аналитик	Маркетинг	75000	5
2	3	Алексей	23	Менеджер	Продажи	90000	15

Рисунок 15 – Задание №5.1

5.2. Удалите строку с ID = 10.

```
df_csv = df_csv[df_csv["ID"] != 10]
display(df_csv)
```

Рисунок 16 – Задание №5.2

5.3. Удалите все строки, где "Стаж работы" < 3 лет.

```
df_csv = df_csv[df_csv['Стаж работы'] >= 3]
display(df_csv)
```

Рисунок 17 – Задание №5.3

5.4. Удалите все столбцы, кроме - Имя, Должность, Зарплата.

```
df_csv = df_csv[["Имя", "Должность", "Зарплата"]]
display(df_csv)
```

Рисунок 18 – Задание №5.4

Задание №6. Фильтрация данных (query, isin, between).

6.1. Выберите всех клиентов из "Москва" или "Санкт-Петербург", используя .isin().

```
df_isin = df_excel[df_excel["Город"].isin(["Москва", "Санкт-Петербург"
display(df_isin)
```

	ID	Имя	Возраст	Город	Баланс на счете	Кредитная история
0	1	Иван	34	Москва	120000	Хорошая
1	2	Ольга	27	Санкт-Петербург	80000	Средняя

Рисунок 19 – Задание №6.1

6.2. Выберите клиентов, у которых Баланс на счете от 100000 до 250000, используя .between().

```
df_between = df_excel[df_excel["Баланс на счете"].between(100000,25000
display(df_between)
```

Рисунок 20 – Задание №6.2

6.3. Отфильтруйте клиентов, у которых "Кредитная история" - "Хорошая" и "Баланс на счете" - > 150000, используя .query().

```
df_query = df_excel.query('`Кредитная история` == "Хорошая" and `Балан
display(df_query)
```

	ID	Имя	Возраст	Город	Баланс на счете	Кредитная история
3	4	Мария	38	Новосибирск	200000	Хорошая
7	8	Елена	40	Краснодар	175000	Хорошая
10	11	Павел	46	Омск	250000	Хорошая
13	14	Татьяна	25	Саратов	155000	Хорошая
17	18	Юлия	50	Иркутск	320000	Хорошая

Рисунок 21 – Задание №6.3

Задание №7. Подсчет значений (count, value_counts, nunique).

7.1. Подсчитайте количество непустых значений в каждом столбце (.count()).

```
print(df_excel.count())
```

```
ID                   20
Имя                  20
Возраст              20
Город                20
Баланс на счете      20
Кредитная история    20
dtype: int64
```

Рисунок 22 – Задание №7.1

7.2. Определите частоту встречаемости значений в "Город" (.value_counts()).

```
print(df_excel["Город"].value_counts())
```

```
Город
Москва              1
Санкт-Петербург     1
Хабаровск           1
Иркутск             1
Барнаул             1
Волгоград           1
Самара              1
Саратов             1
Тюмень              1
Пермь               1
Омск                1
Уфа                 1
Ростов-на-Дону      1
Краснодар           1
Челябинск           1
Воронеж             1
Екатеринбург        1
Новосибирск         1
Казань              1
Томск               1
Name: count, dtype: int64
```

Рисунок 23 – Задание №7.2

7.3. Найдите количество уникальных значений в "Город" , "Возраст", "Баланс на счете" (.nunique()).

```
c_a_b = ["Город" , "Возраст" , "Баланс на счете"]
print(df_excel[c_a_b].nunique())
```

```
Город              20
Возраст            19
Баланс на счете    20
dtype: int64
```

Рисунок 24 – Задание №7.3

Задание №8. Обнаружение пропусков (isna, notna).

Используя Таблицу 3:

8.1. Подсчитайте количество NaN в каждом столбце (.isna().sum()).

```
nan = df_last.isna().sum()
print(nan)
```

```
ID                  0
Имя                 2
Возраст             4
Город               4
Баланс на счете     4
Кредитная история   3
dtype: int64
```

Рисунок 25 – Задание №8.1

8.2. Подсчитайте количество заполненных значений в каждом столбце (.notna().sum()).

```
filled_counts = df_last.notna().sum()
print(filled_counts)
```

```
ID                  20
Имя                 18
Возраст             16
Город               16
Баланс на счете     16
Кредитная история   17
dtype: int64
```

Рисунок 26 – Задание №8.2

8.3. Выведите DataFrame, оставив только строки, где нет пропущенных значений.

```
df_without_nan = df_last.dropna()
display(df_without_nan)
```

	ID	Имя	Возраст	Город	Баланс на счете	Кредитная история
0	1	Иван	34.0	Москва	120000.0	Хорошая
7	8	Елена	40.0	Краснодар	175000.0	Хорошая
10	11	Павел	46.0	Омск	250000.0	Хорошая
16	17	Григорий	49.0	Барнаул	275000.0	Отличная
18	19	Степан	30.0	Хабаровск	105000.0	Средняя
19	20	Василиса	35.0	Томск	90000.0	Плохая

Рисунок 27 – Задание №8.3

Индивидуальное задание.

Вариант 12. Использовать DataFrame, содержащий следующие колонки:

• фамилия;

• имя;

• номер телефона;

• дата рождения (список из трех чисел).

Написать программу, выполняющую следующие действия:

• ввод с клавиатуры данных и добавление строк в DataFrame;

• записи должны быть размещены по алфавиту;

• вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры, если таких нет - выдать на дисплей соответствующее сообщение.

Примечания:

• Приложение должно использовать интерфейс командной строки (модуль argparse) или графический интерфейс пользователя (модули tkinker, PySide2, Kivy и т.д);

• При работе с датой и временем использовать пакет datetime. Организовать чтение и сохранение данных из/в формат Parquet;

• Выполнить валидацию сохраненных данных с помощью сторонних приложений для работы с форматом Parquet;

• Огранизовать также удаление данных по одной из колонок DataFrame на выбор обучающихся.

Листинг программы к индивидуальному заданию:

```
import tkinter as tk
from tkinter import messagebox, ttk
import pandas as pd
from datetime import datetime
import pyarrow.parquet as pq
import pyarrow as pa
```

```
import os
FILE = "people.parquet"

def load_data():
    if os.path.exists(FILE):
        return pd.read_parquet(FILE)
    return pd.DataFrame(columns=["Фамилия", "Имя", "Телефон", "Дата рождения"])

def save_data(df):
    table = pa.Table.from_pandas(df)
    pq.write_table(table, FILE)

def add_entry():
    last = last_name_var.get()
    first = first_name_var.get()
    phone = phone_var.get()
    bdate = birthdate_var.get()
    try:
        date = datetime.strptime(bdate, "%d.%m.%Y")
        birth_list = [date.day, date.month, date.year]
    except ValueError:
        messagebox.showerror("Ошибка", "Дата в формате ДД.ММ.ГГГГ")
        return
    df = load_data()
    new_row = {"Фамилия": last, "Имя": first, "Телефон": phone, "Дата рождения": birth_list}
    df = pd.concat([df, pd.DataFrame([new_row])])
    df = df.sort_values(by=["Фамилия", "Имя"])
    save_data(df)
```

```
            messagebox.showinfo("Успех", "Запись добавлена!")

        def filter_by_month():
            try:
                month = int(month_var.get())
                if not 1 <= month <= 12:
                    raise ValueError
            except ValueError:
                messagebox.showerror("Ошибка", "Месяц должен быть числом от 1
до 12")
                return
            df = load_data()
            filtered = df[df["Дата рождения"].apply(lambda d: d[1] == month)]
            result_text.delete("1.0", tk.END)
            if filtered.empty:
                result_text.insert(tk.END, "Нет людей с днем рождения в этом
месяце.")
            else:
                result_text.insert(tk.END, filtered.to_string(index=False))

        def delete_by_column():
            col = delete_column_var.get()
            val = delete_value_var.get()
            df = load_data()
            if col not in df.columns:
                messagebox.showerror("Ошибка", f"Колонка '{col}' не найдена.")
                return
            before = len(df)
            df = df[df[col] != val]
            after = len(df)
```

```
            save_data(df)
            messagebox.showinfo("Готово", f"Удалено записей: {before - after}")

        okno = tk.Tk()
        okno.title("Учет данных о людях")
        okno.geometry("600x550")
        tk.Label(okno, text="Фамилия").pack()
        last_name_var = tk.StringVar()
        tk.Entry(okno, textvariable=last_name_var).pack()
        tk.Label(okno, text="Имя").pack()
        first_name_var = tk.StringVar()
        tk.Entry(okno, textvariable=first_name_var).pack()
        tk.Label(okno, text="Телефон").pack()
        phone_var = tk.StringVar()
        tk.Entry(okno, textvariable=phone_var).pack()
        tk.Label(okno, text="Дата рождения (ДД.ММ.ГГГГ)").pack()
        birthdate_var = tk.StringVar()
        tk.Entry(okno, textvariable=birthdate_var).pack()
        tk.Button(okno, text="Добавить", command=add_entry).pack(pady=5)
        tk.Label(okno, text="Фильтр по месяцу рождения (1-12)").pack()
        month_var = tk.StringVar()
        tk.Entry(okno, textvariable=month_var).pack()
        tk.Button(okno, text="Показать",
command=filter_by_month).pack(pady=5)
        result_text = tk.Text(okno, height=10, width=70)
        result_text.pack()
        tk.Label(okno, text="Удалить записи по колонке").pack()
        delete_column_var = tk.StringVar()
        delete_column_menu = ttk.Combobox(okno,
textvariable=delete_column_var)
```

```
        delete_column_menu['values'] = ["Фамилия", "Имя", "Телефон"]
        delete_column_menu.pack()
        delete_value_var = tk.StringVar()
        tk.Entry(okno, textvariable=delete_value_var).pack()
        tk.Button(okno, text="Удалить",
command=delete_by_column).pack(pady=5)
        okno.mainloop()
```

Результат работы программы:

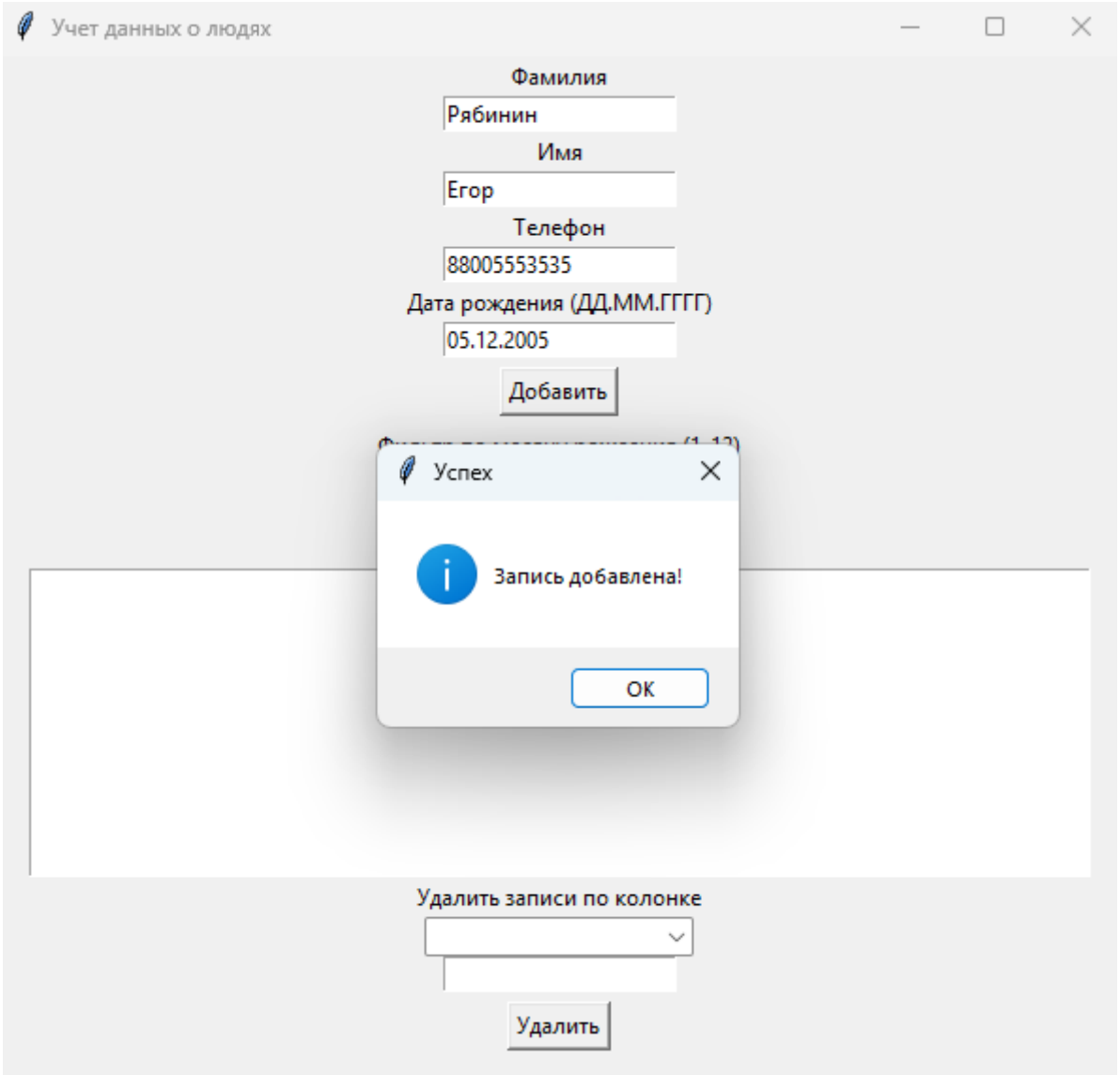

Рисунок 28 – Добавление записи

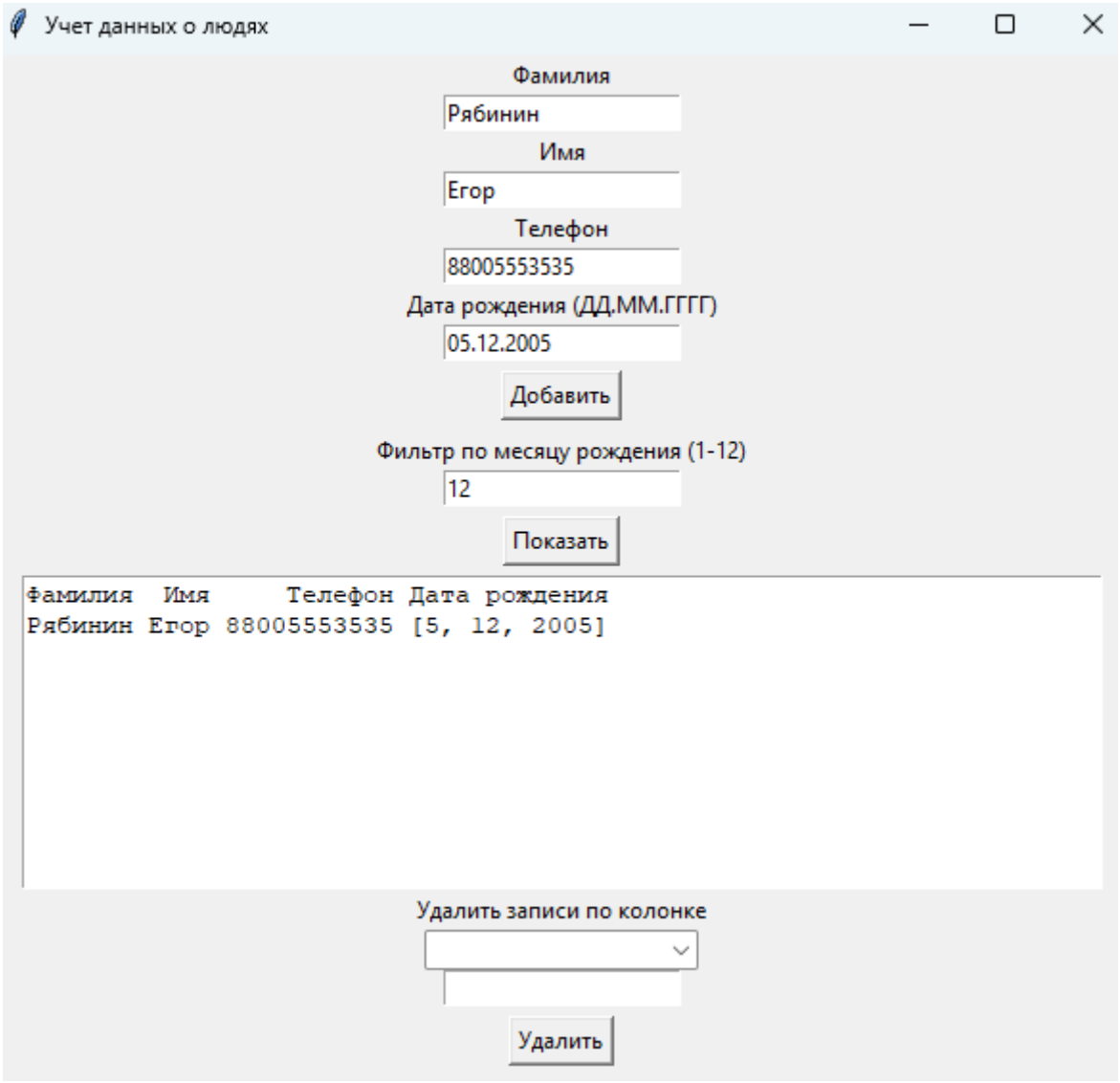

Рисунок 29 – Фильтрация

Фамилия	Имя	Телефон	Дата рождения
Петров	Петр	89093424364	[12,4,1996]
Рябинин	Егор	89097580527	[5,12,2005]

Рисунок 30 – Parquet-файл

Контрольные вопросы:

1. Как создать pandas.DataFrame из словаря списков?

Использовать: pandas.DataFrame(data), где data — это словарь, где ключи — имена столбцов, а значения — списки данных.

2. В чем отличие создания DataFrame из списка словарей и словаря списков?

Словарь списков: ключи — имена столбцов, значения — списки значений по столбцам.

Список словарей: каждый словарь представляет строку, ключи — названия столбцов.

3. Как создать pandas.DataFrame из массива NumPy?

Использовать: pandas.DataFrame(numpy_array)

4. Как загрузить DataFrame из CSV-файла, указав разделитель ; ?

pandas.read_csv('file.csv', sep=';')

5. Как загрузить данные из Excel в pandas.DataFrame и выбрать конкретный лист?

pandas.read_excel('file.xlsx', sheet_name='ИмяЛиста')

6. Чем отличается чтение данных из JSON и Parquet в pandas?

JSON — текстовый формат, читается через read_json.

Parquet — бинарный формат, эффективнее по скорости и размеру, читается через read_parquet.

7. Как проверить типы данных в DataFrame после загрузки?

Использовать: df.dtypes

8. Как определить размер DataFrame (количество строк и столбцов)?

Использовать: df.shape

9. В чем разница между .loc[] и .iloc[] ?

.loc[] — обращение по именованным индексам и меткам.

.iloc[] — обращение по числовым позициям (номерам строк и столбцов).

10. Как получить данные третьей строки и второго столбца с .iloc[] ?

df.iloc[2, 1]

11. Как получить строку с индексом "Мария" из DataFrame?

df.loc['Мария']

12. Чем .at[] отличается от .loc[] ?

.at[] — более быстрый доступ к одному элементу по метке.

.loc[] — доступ к срезам и множеству значений.

13. В каких случаях .iat[] работает быстрее, чем .iloc[] ?

.iat[] — быстрее при доступе к одному элементу по числовому индексу.

14. Как выбрать все строки, где "Город" равен "Москва" или "СПб", используя .isin() ?

df[df['Город'].isin(['Москва', 'СПб'])]

15. Как отфильтровать DataFrame, оставив только строки, где "Возраст" от 25 до 35 лет, используя .between() ?

df[df['Возраст'].between(25, 35)]

16. В чем разница между .query() и .loc[] для фильтрации данных?

.query() использует строковое выражение, удобно и читаемо.

.loc[] требует логического массива, более гибко при сложных условиях.

17. Как использовать переменные Python внутри .query() ?

Через символ @, например: df.query('Возраст > @min_age')

18. Как узнать, сколько пропущенных значений в каждом столбце DataFrame?

df.isna().sum()

19. В чем разница между .isna() и .notna() ?

.isna() — True для NaN.

.notna() — True для непустых значений.

20. Как вывести только строки, где нет пропущенных значений?

df.dropna()

21. Как добавить новый столбец "Категория" в DataFrame, заполнив его фиксированным значением "Неизвестно"?

df['Категория'] = 'Неизвестно'

22. Как добавить новую строку в DataFrame, используя .loc[]?

df.loc[новый_индекс] = [значение1, значение2, ...]

23. Как удалить столбец "Возраст" из DataFrame?

df.drop(columns=['Возраст'], inplace=True)

24. Как удалить все строки, содержащие хотя бы один NaN, из DataFrame?

df.dropna(inplace=True)

25. Как удалить столбцы, содержащие хотя бы один NaN, из DataFrame?

df.dropna(axis=1, inplace=True)

26. Как посчитать количество непустых значений в каждом столбце DataFrame?

df.count()

27. Чем .value_counts() отличается от .nunique() ?

.value_counts() возвращает частоты каждого значения.

.nunique() возвращает количество уникальных значений.

28. Как определить сколько раз встречается каждое значение в столбце "Город"?

df['Город'].value_counts()

29. Почему display(df) лучше, чем print(df), в Jupyter Notebook?

display(df) форматирует вывод как таблицу с прокруткой и стилем, print(df) — простой текст.

30. Как изменить максимальное количество строк, отображаемых в DataFrame в Jupyter Notebook?

pandas.set_option('display.max_rows', число)

Вывод: в ходе практической работы мы познакомились с основами работы с библиотекой pandas, в частности, со структурой данных DataFrame.