ГУАП

КАФЕДРА № 41

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ст. преподаватель |  |  |  | В.В. Боженко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |
| ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ |
| по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4216 |  |  |  | С.Д. Комолова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

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

**Индивидуальное задание:**

Вариант 6:

1. Группировка - тип образования по каждому семейному статусу (marital\_status). Результат должен иметь примерно следующий вид:
2. Группировка - тип образования по каждому семейному статусу (marital\_status). Создать датафрейм. Переименовать столбец с количеством в “сount”. Отсортировать по убыванию столбца “count”.
3. Сводная таблица (pivot\_table) - средний доход семьи по семейному положению. Отсортировать по убыванию. Округлить до двух знаков.
4. Сводная таблица (pivot\_table) - средний доход семьи по семейному положению - строки и уровню образованию - столбцы. Отсортировать по возрастанию marital\_status. Округлить до двух знаков.

**Ход работы:**

Ссылка на репозиторий github: <https://github.com/sufferix/data-analysis>

В данной лабораторной работе необходимо загрузить и обработать датасет, а также выполнить задания по варианту с помощью библиотеки `pandas` в `Python`. Для начала была импортирована данная библиотека с помощью функции `import`, затем был загружен датасет `clients.csv`

import pandas as pd

df = pd.read\_csv('clients.csv',sep=';')

Были выведены первые 20 строк с помощью функции `df.head(20)`(см. Рисунок 1)

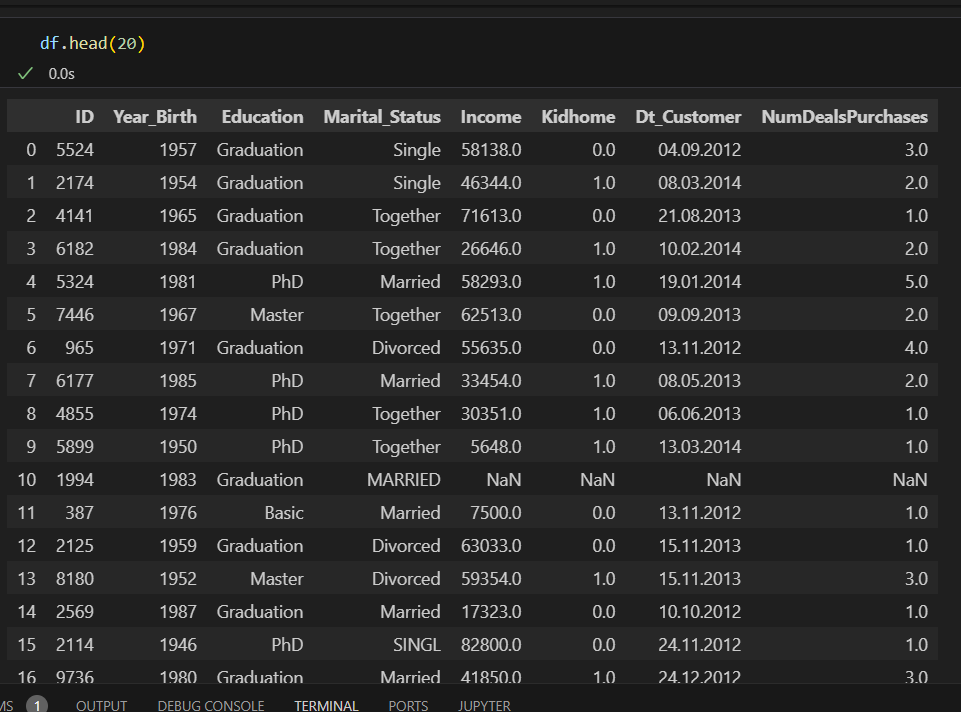


Рисунок 1 – Метод df.head()

Предметной областью по варианту является журнал клиентов с такой информацией о них:

* уникальный идентификатор
* год рождения
* уровень образования (базовый, выпускник, магистр, доктор наук)
* семейное положение (холостой, в отношениях, женатый, разведён, вдовый)
* семейная прибыль
* количество детей
* дата регистрации в компании
* количество покупок

Все эти характеристики отображены в столбцах данного датасета. Чтобы показать все столбцы, используется функция `columns` (см. Рисунок 2)

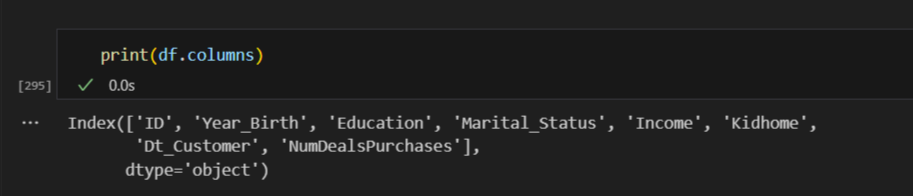


Рисунок 2 – Метод df.columns()

Когда все столбцы были выведены, есть необходимость заменить названия столбцов, так как некоторые не отображают контекст (к примеру, `Kidhome`) или имеют в названии ошибку (в `NumDealsPurchases` слова deals и purchases имеют схожий смысл, а также написание не похоже на остальные столбцы) (см. Рисунок 3)

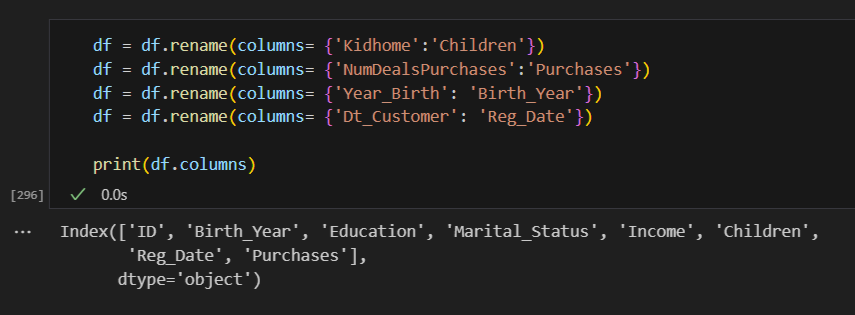


Рисунок 3 – Изменение названий столбцов

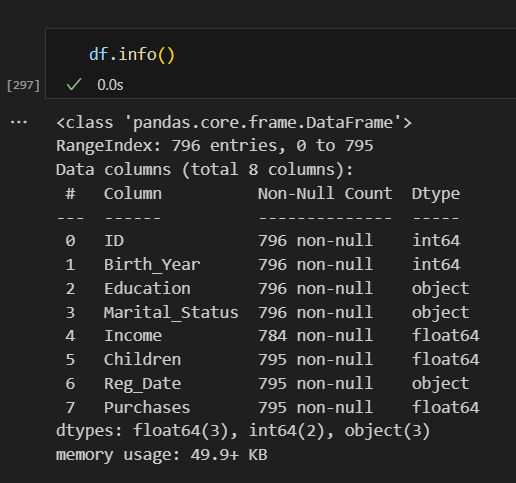
Для отображения информации о датасете используется функция `df.info()` (см. Рисунок 4)

Рисунок 4 – Метод df.info()

Всего записей в этом датасете 796, однако в столбцах Income, Children, Reg\_Date и Purchases наблюдаются меньше записей, так как там есть значения `NaN`. Чтобы вывести все записи, имеющие хотя бы одно нулевое значение, используется функция `.isnull()`. Функция `.any(axis=1)` указывает на поиск ненулевых значений именно по строкам (см. Рисунок 5)

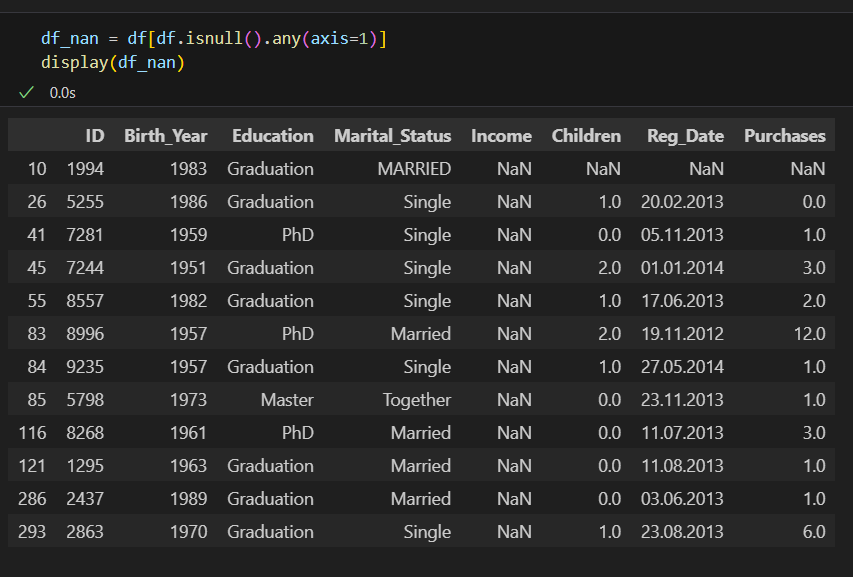


Рисунок 5 – Вывод записей с нулевыми значениями

Было принято решение удалить все записи, содержащие в себе хотя бы одно значение `NaN`, так как в дальнейшем отсутствие данных повлияло бы на значение среднего дохода семьи. За удаление строк с нулевыми значениями отвечает функция `.dropna()`. После этого была проведена проверка нулевых значений в столбцах с помощью функции `.isna()`, `.sum()` показывает сумму таких значений. Везде написано 0 (см. Рисунок 6)

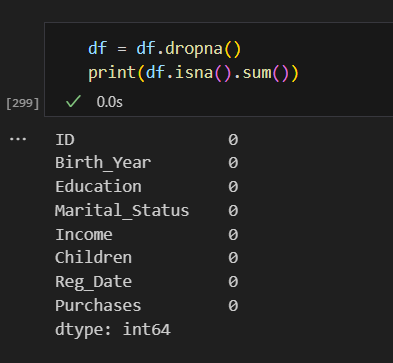


Рисунок 6 – Вывод количества нулевых значений

Также вывод функции `.info()` показал типы данных в столбцах. Столбцы `Children` и `Purchases` имеют тип `float`, однако значения в данных столбцах могут быть только дискретными, поэтому необходимо задать данным столбцам тип данных `int` с помощью функции `.astype()`

df['Children'] = df['Children'].astype(int)

df['Purchases'] = df['Purchases'].astype(int)

Также столбец `Reg\_Date` имеет тип данных `object`, но так как это столбец с датой, то данному столбцу необходимо присвоить тип данных `datetime` для правильного отображения с помощью функции `pd.to\_datetime()` и задать формат дд.мм.гг.

df['Reg\_Date'] = pd.to\_datetime(df['Reg\_Date'], format='%d.%m.%Y')

Теперь можно вывести информацию о датасете с изменёнными типами данных. (см. Рисунок 7)

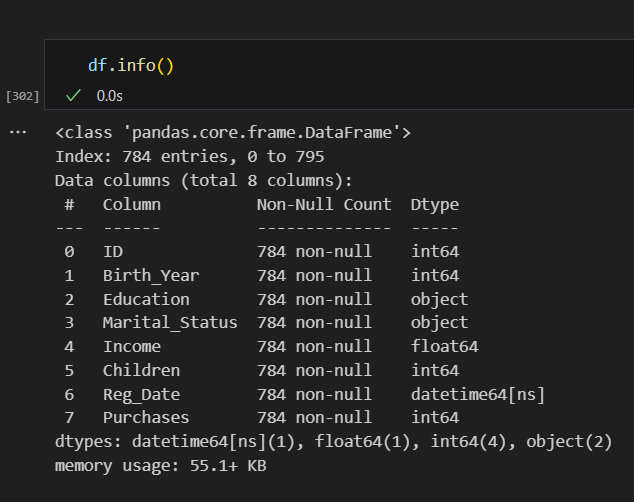


Рисунок 7 – Итоговый вывод метода df.info()

Метод `.describe()` используется для получения статистического обзора данных. Он предоставляет такие статистические показатели, как среднее значение, стандартное отклонение, минимальное и максимальное значения, а также квартильные значения (25%, 50%, и 75%) (см. Рисунок 8)

Столбец `Reg\_Date` тоже содержит числовые значения, но в виде даты, поэтому он был исключён из вывода, так как использовать его было бы некорректно.

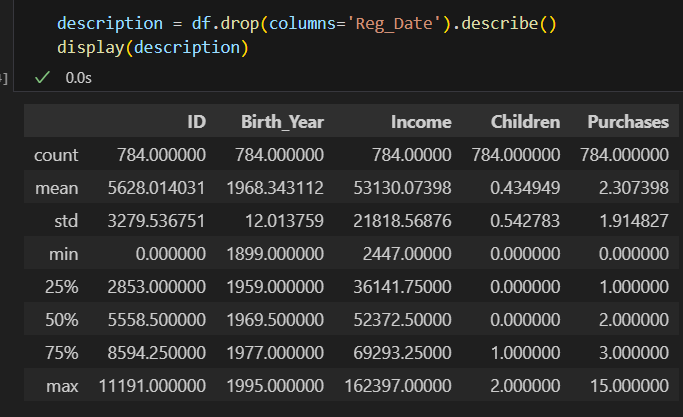


Рисунок 8 – Метод describe()

Далее необходимо проверить датасет на явные и неявные дубликаты. Для начала были проверены столбцы `Marital\_Status` и `Education` на наличие записей с одинаковыми значениями, но разные по написанию (регистр или синоним) с помощью функции `.unique()` (см. Рисунок 9)

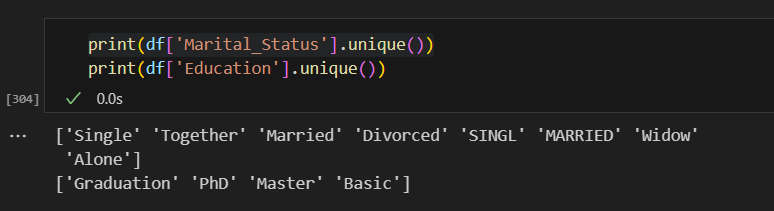


Рисунок 9 – Вывод всех уникальных значений в столбцах Marital\_Status и Education

В столбце `Education` нет каких-либо дублирующихся значений, однако в столбце `Marital\_Status` есть значения `SINGL` и `Alone`, которые дублируют `Single`, и `MARRIED`, дублирующий `Married`. Приведение записей с такими значениями в нормальный вид можно сделать с помощью функции `.replace()` (см. Рисунок 10)

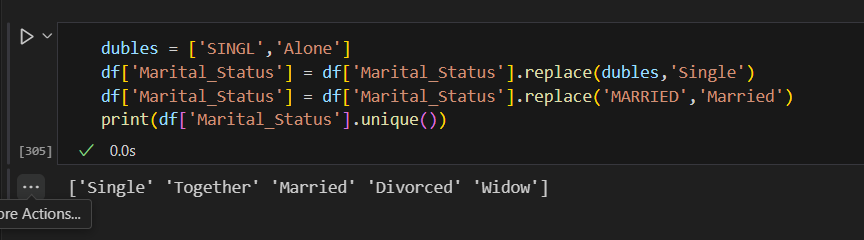


Рисунок 10 – Удаление всех неявных дубликатов

С помощью функции `.duplicated()` можно проверить, какие записи полностью повторяют уже существующую запись, но таким образом он выведет весь датасет со значениями `True/False`, поэтому для упрощения был добавлен метод `.sum()`

print(df.duplicated().sum())

df = df.drop\_duplicates()

В результате датасет был обработан и приведён в подходящий для анализа вид. Столбцы были переименованы и у некоторых из них были изменены типы данных. Датасет был очищен от дубликатов и от записей с неизвестными значениями.

**Задание 1:**

Группировка данных происходит с помощью метода `.groupby()`. Необходимо было сгруппировать данные по столбцам `Education` и `Marital\_Status`, и вывести количество записей, соответствующих определённому семейному положению на всех уровнях образования, выведя только столбец `ID` с методом `.count()` (см. Рисунок 11)

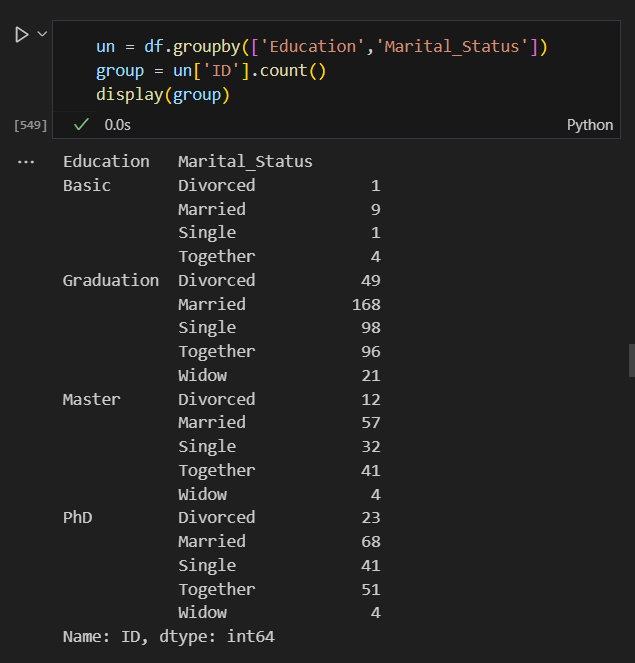


Рисунок 11 – Группировка семейного положения по образованию

Получилась таблица, сгруппированная по уровню образования, которая показывает количество клиентов с разным семейным положением на все уровнях образования

**Задание 2**

Так как группировка точно такая же, то можно работать с таблицей из прошлого задания. Но теперь необходимо из этой таблицы создать датафрейм с помощью метода `pd.DataFrame()`. Самой таблице была присвоена переменная `group`, на основе которой был создан датафрейм. После этого столбец `ID` был переименован в столбец `Count`

gr\_df = pd.DataFrame(group)

gr\_df = gr\_df.rename(columns={'ID':'Count'})

Далее значения столбца `Count` были отсортированы по убыванию, используя метод `sort\_values`. Стоит обратить внимание на то, что внутри метода пишется всегда атрибут `ascending`, даже если надо отсортировать по убыванию. Атрибут `ascending` принимает значения `True/False` (см. Рисунок 12)



Рисунок 12 – Группировка семейного положения по образованию и сортировка по убыванию

В результате получилась отсортирована таблица по количеству записей с определённым семейным положением на различных уровнях образования

**Задание 3**

Метод `.pivot\_table()` используется для создания сводных таблиц, которые позволяют агрегировать данные и представлять их в удобном формате. Основные атрибуты метода:

* `values`: Столбец или столбцы для агрегирования
* `index`: Столбец или столбцы, которые будут использоваться в качестве индексов строк
* `columns`: Столбец или столбцы, которые будут использоваться в качестве заголовков столбцов
* `aggfunc`: Функция или список функций для агрегации

В данном задании необходимо создать сводную таблицу среднего дохода только по семейному положению, поэтому из списка используются только атрибуты `values`, `index` и `aggfunc`. За значения берётся доход, за индексы - семейное положение и функция усреднения.

Также значения были отсортированы по убыванию и округлены до 2 знаков с помощью метода `.round(2)` (см. Рисунок 13)

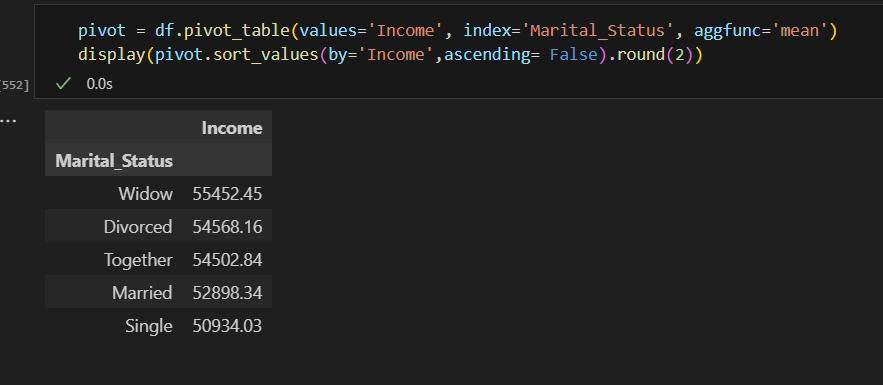


Рисунок 13 – Сводная таблица среднего дохода по семейному положению

Получившаяся таблица показывает среднее значение дохода семьи по семейному положению.

**Задание 4**

В данном задании необходимо сделать такую же сводную таблицу, но с использованием атрибута `columns`, в качестве которого используется столбец `Education`. Значения были отсортированы по возрастанию индексов `Marital\_Status` и округлены до 2 знаков. (см. Рисунок 14)

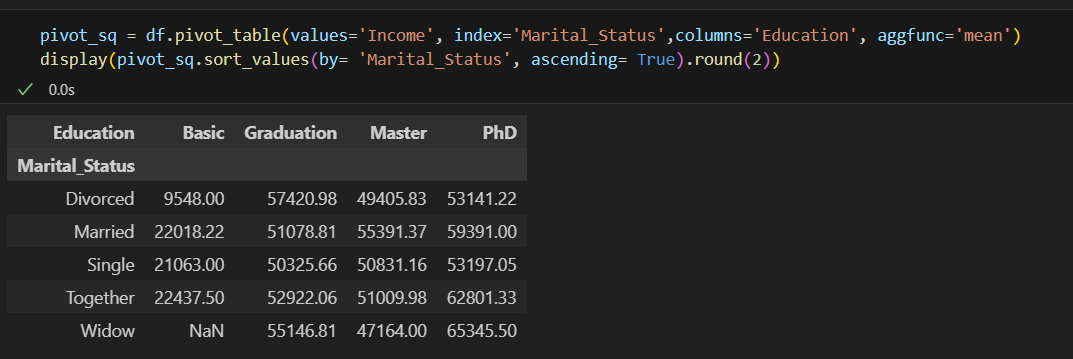


Рисунок 14 – Сводная таблица среднего дохода по семейному положению и образованию

Данная таблица отображает средние значения дохода семьи по разным уровням образования и семейному положению клиентов. Также в данной таблице можно заметить значение NaN. Это обусловлено тем, что в датасете не было заявлено записей вдовцов с базовым образованием.

Вывод:

В данной лабораторной работе была проделана работа по обработке датасета журнала клиентов, исключив ненужные записи и заменив некоторые значения, а также анализу данных с помощью группировки и создания сводной таблицы. Были освоены основные методы библиотеки `pandas` и варианты их использования:

1. Для вывода информации о датасете были использованы методы `.info()`, `.describe()`

2. Для вывода информации об определённых аспектах датасета были использованы методы `df.column`, `.isna()`, `.isnull()`, `.unique()`, `.duplicated()`

3. Для замены данных в датасете были использованы методы `.rename()` - для столбцов или значений внутри столбцов, `.astype()` и `pd.to\_datetime` - для изменения типа данных какого-либо столбца

Также были использованы методы `.groupby()` для группировки значений, `sort\_values()` для их сортировки, `.pivot\_table()` для создания сводной таблицы и `pd.DataFrame()` для создания датафрейма из таблицы.

При выполнении индивидуального задания были получены некоторые результаты группировок и сводных таблиц: большую часть клиентов составляют клиенты, которые закончили бакалавриат и находятся в браке, а также средняя зарплата выше у овдовевших людей, имеющих докторскую степень.