Подключение и загрузка данных

Команда / метод	Пояснение и пример
`import pandas as pd`	Подключение библиотеки Pandas под псевдонимом pd
`pd.read_csv()`	Считывание CSV-файла и загрузка данных в DataFrame
`df` общепринятое имя объекта, можно дать оригинальное	Табличная структура данных (DataFrame)

Просмотр, анализ данных и работа с датафреймом

Команда / метод	Пояснение и пример
`df.info()`	Вывод общей информации: количество строк, столбцов, типы данных
`df.head()`	Просмотр первых 5 строк таблицы
`df.head(n)`	Просмотр первых n строк таблицы
`df.tail()`	Просмотр последних 5 строк
`df.sample(5)`	Случайный выбор 5 строк
`df.describe()`	Основные статистические характеристики числовых данных каждого числового столбца: count — количество ненулевых значений mean — среднее значение std — стандартное отклонение min — минимальное значение

	25% — первый квартиль (значение, ниже которого 25% данных) 50% — медиана (второй квартиль) 75% — третий квартиль max — максимальное значение
`display(df)`	команда для красивого отображения DataFrame df в табличном виде
`df.describe(include='all')`	Описание всех столбцов, включая категориальные
`dfcopied = df.copy()`	копирование исходной датафрейм в новую переменную.
`df['Страна или регион'].unique()`	уникальное значение колонки датафрейма
`df['Страна или регион'].value_counts()`	такое же как unique(). Дополнительно подсчитывает количество раз, которое то или иное уникальное значение встречается в колонке,

Размер таблицы

Команда / метод	**Пояснение и пример**
`df.shape`	Возвращает кортеж (строки, столбцы)
`df.shape[0]`	Количество строк
`df.shape[1]` Для легкого запоминания: (строки ↓, столбцы →) (0 , 1)	Количество столбцов

Работа со строками

Команда / метод	**Пояснение и пример**

`df.loc[]`
 	
`df.loc[0, 'salary'] = 60000 # поменяем зарплату у первой строки. Здесь 0 – индекс строки, 'salary' – имя столбца`
	работа с диапазоном: выбирает строки

`df.index`	выводит индексы строк

`df.iloc[строки, столбцы]`
 	
`df.iloc[0]  # первая строка`
 	
`df.iloc[:, 2]  # третий столбец (нумерация с 0)`
 	
`df.iloc[0:3, 1:4]  # строки 0,1,2 и столбцы 1,2,3`
	Доступ к строкам по числовому индексу
	
	Всё, что идёт до запятой, относится к **строкам**, после запятой — к **столбцам**.

`df.iloc[0:100, 0:5]`	Задаем диапазон
`df.loc['Наиболее частое'] = df.mode().iloc[0]`	Для категориальных столбцов можно использовать mode() или заполнить константой берем первую строку, если несколько мод

`df.append()`
 	
пример:
 	
`new_row = {'age': 10, 'salary': '100', 'Баллы': 100}`
`df = df.append(new_row, ignore_index=True)`
	Добавление новой строки в таблицу

`df = df.append(df.sum(axis=0), ignore_index = True)`	Добавим в конце строку с суммами значений по каждому столбцу (Работает только для числовых столбцов.)
`df = df.append(df.mean(axis=0), ignore_index=True)`	Строка со средним по каждому столбцу
`df = df.append(df.median(axis=0), ignore_index=True)`	Строка с медианой по каждому столбцу

`df = df.drop(df.shape[0]-1, axis = 0)`
 	
`df = df.drop(0)  # удаляет строку с индексом 0`
 	
`df = df.drop([0, 2])  # удаляет несколько строк по индексам`
 	
`df = df[df['age'] >= 30]  # оставляем только строки, где age >= 30`
	Задав параметр axis = 0, можно удалить любую строку из датафрейма: для этого нужно написать ее номер в качестве первого аргумента в методе drop(). Удалим последнюю строчку (указываем ее индекс — это будет количество строк):

`df = df.drop([0, 2, 4], axis=0)`	Удаляем строки с индексами 0, 2 и 4

Фильтрация данных

Команда / метод	**Пояснение и пример**
`df[df['age'] > 30]`	Выбор строк по условию
Логические операторы `df[(df['age'] > 30) & (df['salary'] > 50000)]`	Используются & (И), | (ИЛИ)
`df[df['city']=='Москва']`	Фильтрация по конкретному значению
`df['Баллы_new'] = round(df['Баллы'»])`	округлим все значения в столбце «Баллы»

Работа со столбцами

Команда / метод	**Пояснение и пример**
`df['age']`	Обращение к одному столбцу

`df[['age','salary']]`
можно сохранить в новой переменной:
`new = df[['age', 'salary']]`
	Выбор нескольких столбцов

`df.columns`	Получение названий столбцов
`df.dtypes`	Типы данных столбцов (int64, float64, object…)

`astype()`
Пример:
`df['age'] = df['age'].astype(int)`
	Изменение типа данных

`rename()`
Пример:
`df.rename(columns={'old': 'new'}, inplace=True)`
	Переименование столбцов

`df['age_plus_salary'] = df['age'] + df['salary']  # сумма двух столбцов` `df['new_column'] = 0  # создаём столбец 'new_column' со значением 0 во всех строках`	Добавление столбца

`df['city'] = ['Paris', 'London', 'Berlin']  # длина списка должна совпадать с количеством строк`	
`df = df.drop(['Сумма'], axis = 1)`	Удаление столбца
`df.isnull().sum()`	Посчитать количество пропусков по столбцам

Очистка данных

Команда / метод	**Пояснение и пример**

`drop_duplicates()`
 	
`df.drop_duplicates(subset=None, inplace=False)`
 	
`subset=[столбцы] – проверять дубликаты только по указанным столбцам`
 	
inplace=True — изменить DataFrame на месте
 	
Если subset=None, проверяются все столбцы
 	
Пример:
 	
`df1.drop_duplicates(subset=['ВВП_new', 'Баллы_new'])`
	Удаление дубликатов
	
	Удаляем дубликаты по столбцам 'ВВП_new' и 'Баллы_new'

`df.fillna(значение на которое заменяем, inplace=False)`
 	
`df['age'].fillna(df['age'].mean(), inplace=True)`
	Замена пропущенных значений

`df_clean =df.dropna(axis=0, inplace=False)`
 	
axis=0 — удаляем строки с пропусками (параметр по умолчанию)
 	
axis=1 — удаляем столбцы с NaN
 	
`# Удаляем строки с пропусками в новый DataFrame`
 	
`df_clean = df.dropna(subset=['age', 'salary']) # удаление в конкретных столбцах`
 	
`df_clean = df.dropna(axis=1)`
	Удаление строк или столбцов с пропусками

print(df_clean) # удалить столбцы, где есть хотя бы один NaN df_clean = df.dropna()# удалить строки с хотя бы одним NaN	

Группировка данных

Команда / метод	**Пояснение и пример**
groupby() df.groupby('Баллы_new').count() # группируем строки по уникальным значениям столбца 'Баллы_new'. После группировки. count() считает количество ненулевых (не NaN) значений в каждом столбце для каждой группы. df.groupby('Баллы_new').sum() # сумма значений в каждой группе Несколько агрегатов: df_agg = df.groupby('Баллы_new').agg({ 'Баллы_new': 'count', 'Баллы_new': 'sum', 'Баллы_new': 'mean', 'Баллы_new': 'median' })	Группировка строк какого-либо столбца по одинаковым значениям
count()	Подсчёт количества значений в группе
sum()	Суммирование значений
mean()	Среднее значение
agg()	Применение нескольких агрегатных функций

Сортировка и преобразования

Команда / метод	**Пояснение и пример**
df.sort_values(by='имя_столбца', ascending=True, inplace=False) ascending=True — по возрастанию (по умолчанию) ascending=False — по убыванию	Сортировка данных по столбцу

inplace=True — сортировка происходит на месте без создания нового DataFrame Если inplace=False (по умолчанию), метод возвращает новый DataFrame	

`df['Столбец'].apply(функция)`
 	
Пример функция для перевода строки в нижний регистр:
 	
`def my_lower(row):`
 	
`    return row.lower()`
 	
`df['Страна или регион'] = df['Страна или регион'].apply(my_lower)`
	Применение функции к каждому элементу

`tolist()`
 	
Примеры:
 	
`df['Баллы'].tolist()`
 	
`df.columns.tolist()`
	Преобразование столбца в список Python
	
	Часто бывает нужно получить в виде списка названия столбцов датафрейма.

Сохранение данных

Команда / метод	**Пояснение и пример**
`to_csv()`	Сохранение DataFrame в CSV-файл
`to_excel()`	Сохранение DataFrame в Excel-файл