

Занятие 23

SQL, psycopg2

Программа “сложения” двух словарей

```
dict1 = {'яблоки': 100, 'бананы': 333, 'груши': 300, 'апельсины': 300}  
dict2 = {'яблоки': 300, 'груши': 200, 'малина': 777, 'ананасы': 12}
```

```
res = {}  
for key in dict1 | dict2:  
    res[key] = dict1.get(key, 0) + dict2.get(key, 0)  
print(res)
```

```
# Почему нельзя сразу писать res = dict1 | dict2
```

Задача 22-1

Вывести содержимое таблицы book, при этом авторов отсортировать по возрастанию, а цены книг по убыванию

Задача 22-2

Дана структура типа бинарное дерево. Все вершины пронумерованы от 1 до n . Дерево задано в виде списка кортежей: [(1, 2), (1, 3), (2, 4), (2, 5), (3, 6), (6, 7), (7, 8)]

Каждый кортеж (a, b) показывает, что вершина a соединена с вершиной b.

По определению в дереве невозможны циклы.

Найти максимальную длину от вершины (1) до конечной вершины.

Задача 22-3

В Python существуют ключевые слова, которые нельзя использовать для названия переменных, функций и классов. Для получения списка всех ключевых слов можно воспользоваться атрибутом `kwlist` из модуля `keyword`.

Приведенный ниже код:

```
import keyword  
print(keyword.kwlist)
```

ВЫВОДИТ:

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def',  
'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda',  
'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

Напишите программу, которая принимает строку текста и заменяет в ней все ключевые слова на `<kw>`

О примерной структуре презентации по работе

1. Титульный слайд – Название работы, автор, курс (лето 2024)
2. Содержание презентации или план
3. Цели
4. Задачи, спецификация, бизнес-проблема !!!
5. Особенности, специфика
6. Функционал, реализованный и планируемый к реализации
7. Теория, подходы к решению
8. Что реализовано, что предстоит в планах реализовать
9. Структуры, схемы, алгоритмы и прочее
10. И использованные технические средства, модули, и т.д.
11. Демонстрация работающей программы
12. Любые другие пункты, которые вы хотите включить. Может быть какие-то главные формулы или гениальные строки кода и т.д.
13. Последний слайд. Кто-то пишет Спасибо за внимание, кто-то Вопросы? Я обычно повторяю тему доклада, свое имя, электронный адрес, иногда телефон.

Комментарии

1. Презентация не должна быть полноценным докладом с полным текстом. Краткое изложение. **Основные тезисы**
2. **Графические** представления приветствуются
3. Это скорее **опорный конспект**, но если вы не уверены в себе, как докладчик, то можно написать ключевые фразы
4. Обязательно проговорите ее перед зеркалом **вслух**. Выяснится много интересных деталей, засекайте время – не больше 10 минут.
5. Если есть **сокращения**, то лучше их расшифровать, если это не общепринятые.
6. Это ваша презентация, сколько вам надо слайдов столько и делайте. Если надо больше, значит больше, если надо меньше, значит меньше

Физическая модель

- Специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных. Выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным, создание индексов, и т.д.

Создать таблицу

Наполним таблицу данными

book_id	title	author	price	amount
INT PRIMARY KEY	VARCHAR(50)	VARCHAR(30)	INT	INT
1	Мастер и Маргарита	Булгаков М.А.	670	3
2	Белая гвардия	Булгаков М.А.	540	5
3	Идиот	Достоевский Ф.М.	460	10
4	Братья Карамазовы	Достоевский Ф.М.	799	2
5	Стихотворения и поэмы	Есенин С.А.	650	15

SELECT

- `SELECT title, amount FROM book;`

Использовать названия

- `SELECT title AS Название, amount FROM book;`

Вычисляемые поля

- `SELECT title, author, price, amount, price * amount AS total FROM book;`

Функции

Функция	Описание	Пример
CEILING(x)	возвращает наименьшее целое число, большее или равное x (округляет до целого числа в большую сторону)	CEILING(4.2)=5 CEILING(-5.8)=-5
ROUND(x, k)	округляет значение x до k знаков после запятой, если k не указано – x округляется до целого	ROUND(4.361)=4 ROUND(5.86592,1)=5.9
FLOOR(x)	возвращает наибольшее целое число, меньшее или равное x (округляет до целого числа в меньшую сторону)	FLOOR(4.2)=4 FLOOR(-5.8)=-6
POWER(x, y)	возведение x в степень y	POWER(3,4)=81.0
SQRT(x)	квадратный корень из x	SQRT(4)=2.0 SQRT(2)=1.41...
DEGREES(x)	конвертирует значение x из радиан в градусы	DEGREES(3) = 171.8...
RADIANS(x)	конвертирует значение x из градусов в радианы	RADIANS(180)=3.14...
ABS(x)	модуль числа x	ABS(-1) = 1 ABS(1) = 1
PI()	pi = 3.1415926...	

Задание

Сосчитайте налоги и цену с налогом

- `SELECT title, price, (price*0.2/1.2 AS tax, price*1.2 AS price_tax FROM book;`

Округлите

- `SELECT title, price, ROUND((price*0.2)/1.2,2) AS tax, ROUND(price/(1.2),2) AS price_tax FROM book;`
- В конце года цену всех книг на складе пересчитывают – снижают ее на 30%. Написать SQL запрос, который из таблицы book выбирает названия, авторов, количества и вычисляет новые цены книг. Столбец с новой ценой назвать new_price, цену округлить до 2-х знаков после запятой

Условные выражения

```
SELECT title, amount,
```




```
CASE WHEN amount > 6 THEN 'Very good!'
```

```
  WHEN amount < 4 THEN 'Not so good'
```

```
  ELSE 'OK'
```

```
END AS comment
```

```
FROM book2
```

	title text 	amount integer 	comment text 
1	Тарас Бульба	2	Not so good
2	Мертвые души	100	Very good!
3	Ревизор	50	Very good!
4	Капитанская дочка	200	Very good!
5	Евгений Онегин	20	Very good!
6	Воскресение	5	OK
7	Война и мир	10	Very good!

Where

Выборка, если цена меньше 600

- `SELECT title, price FROM book WHERE price < 600;`

Полная стоимость, если цена больше 4000

- `SELECT title, author, price * amount AS total FROM book WHERE price * amount > 4000;`

Книги Булгакова, если цена больше 600

- `SELECT title, author, price FROM book WHERE price > 600 AND author = 'Булгаков М.А.';`

Или Булгаков, или Есенин и цена больше 600

- `SELECT title, author, price FROM book WHERE (author = 'Булгаков М.А.' OR author = 'Есенин С.А.') AND price > 600;`

Between, In, Order By, Asc, Desc

- `SELECT title, amount FROM book WHERE amount BETWEEN 5 AND 14;`

То же самое: `SELECT title, amount FROM book WHERE amount >= 5 AND amount <=14;`

- `SELECT title, price FROM book WHERE author IN ('Булгаков М.А.', 'Достоевский Ф.М.');`
- `SELECT title, author, price FROM book ORDER BY title;`
- `SELECT title, author, price FROM book ORDER BY 1;`
- `SELECT author, title, amount AS Количество FROM book WHERE price < 750 ORDER BY author, amount DESC;`
- `select author, title from book where amount between 2 and 14 order by author desc, title asc`

LIKE

Символ-шаблон	Описание	Пример
%	Любая строка, содержащая ноль или более символов	SELECT * FROM book WHERE author LIKE '%М.%' выполняет поиск и выдает все книги, инициалы авторов которых содержат «М.»
(подчеркивание)	Любой одиночный символ	SELECT * FROM book WHERE title LIKE 'Поэм' выполняет поиск и выдает все книги, названия которых либо «Поэма», либо «Поэмы» и пр.

```
SELECT title FROM book WHERE title LIKE 'Б%';
```

```
SELECT title FROM book WHERE title LIKE "_____";
```

```
SELECT title FROM book WHERE title LIKE "_____";
```

Вывести названия книг, которые состоят ровно из одного слова, если считать, что слова в названии отделяются друг от друга пробелами

```
SELECT title FROM book
```

```
WHERE title NOT LIKE "% %";
```


Distinct, Sum, Group By

```
SELECT DISTINCT author FROM book;
```

```
SELECT author, count(amount) FROM book GROUP BY author;
```

```
SELECT author, SUM(amount) FROM book GROUP BY author;
```

MIN, MAX, AVERAGE

```
SELECT author, MIN(price) AS min_price FROM book GROUP BY author;
```

```
select author, min(price) as Минимальная_цена, max(price) as  
Максимальная_цена, avg(price) as Средняя_цена  
from book group by author
```

```
SELECT author, SUM(price * amount) AS Стоимость FROM book  
GROUP BY author;
```

```
SELECT author, ROUND(AVG(price),2) AS Средняя_цена FROM book  
GROUP BY author;
```

```
SELECT SUM(amount) AS Количество FROM book;
```

```
SELECT SUM(amount) AS Количество, SUM(price * amount) AS  
Стоимость FROM book;
```

Задание

1. Найти минимальную и максимальную цену книг всех авторов, общая стоимость книг которых больше 5000.

```
SELECT author, MIN(price) AS Минимальная_цена, MAX(price) AS Максимальная_цена FROM  
book  
GROUP BY author  
HAVING SUM(price * amount) > 5000;
```

2. Найти минимальную и максимальную цену книг всех авторов, общая стоимость книг которых больше 5000. Результат вывести по убыванию минимальной цены.

```
SELECT author,  
       MIN(price) AS Минимальная_цена,  
       MAX(price) AS Максимальная_цена  
FROM book  
GROUP BY author  
HAVING SUM(price * amount) > 5000  
ORDER BY Минимальная_цена DESC;
```

HAVING

Сначала определяется таблица, из которой выбираются данные (FROM), затем из этой таблицы отбираются записи в соответствии с условием WHERE, выбранные данные агрегируются (GROUP BY), из агрегированных записей выбираются те, которые удовлетворяют условию после HAVING.

Потом формируются данные результирующей выборки, как это указано после SELECT (вычисляются выражения, присваиваются имена и пр.).

Результирующая выборка сортируется, как указано после ORDER BY.

```
SELECT author, MIN(price) AS Минимальная_цена, MAX(price) AS Максимальная_цена
FROM book
WHERE author <> 'Есенин С.А.'
GROUP BY author
HAVING SUM(amount) > 10;
```

Вложенные запросы

```
SELECT title, author, price, amount
```

```
FROM book
```

```
WHERE price = ( SELECT MIN(price) FROM book );
```

```
select author, title, price
```

```
from book
```

```
where price <= (select avg(price) from book)
```

```
order by price DESC
```

Вложенные запросы

Вывести информацию о книгах, количество экземпляров которых отличается от среднего количества экземпляров книг на складе более чем на 3. То есть нужно вывести и те книги, количество экземпляров которых меньше среднего на 3, или больше среднего на 3.

```
SELECT title, author, amount
```

```
FROM book
```

```
WHERE ABS(amount - (SELECT AVG(amount) FROM book)) >3;
```

Вложенные запросы

Вывести информацию о книгах тех авторов, общее количество экземпляров книг которых не менее 12.

```
SELECT title, author, amount, price
FROM book
WHERE author IN (
    SELECT author
    FROM book
    GROUP BY author
    HAVING SUM(amount) >= 12
);
```

Вложенные запросы

Вывести информацию о книгах, количество экземпляров которых отличается от среднего количества экземпляров книг на складе более чем на 3, а также указать среднее значение количества экземпляров книг.

```
SELECT title, author, amount,  
  (  
    SELECT AVG(amount)  
    FROM book  
  ) AS Среднее_количество  
FROM book  
WHERE abs(amount - (SELECT AVG(amount) FROM book)) >3;
```


Создайте еще одну таблицу с такими же полями, как и book

```
Create table supply  
(supply_id  INT PRIMARY KEY,  
title  VARCHAR(50),  
author    VARCHAR(30),  
price  INT,  
amount    INT)
```

Добавление данных из другой таблицы

- `INSERT INTO` book (title, author, price, amount) `VALUES` ('Война и мир', 'Толстой Л.Н.', 1070, 2), ('Анна Каренина', 'Толстой Л.Н.', 599, 3);

```
INSERT INTO book (title, author, price, amount)
SELECT title, author, price, amount FROM supply;
```

```
INSERT INTO book (title, author, price, amount)
SELECT title, author, price, amount
FROM supply
WHERE title NOT IN ( SELECT title FROM book );
```

Задание

Создать таблицу author следующей структуры:

- author_id INT PRIMARY KEY
- name_author VARCHAR(50)
- create table author
- (author_id INT PRIMARY KEY,
- name_author VARCHAR(50))

insert into author

- Values
- (1, 'Булгаков М.А.'),
- (2, 'Достоевский Ф.М.'),
- (3, 'Есенин С.А.'),
- (4, 'Пастернак Б.Л.');

Соединяем book и author, чтобы имена писателей не дублировались

```
ALTER TABLE book1 ADD COLUMN author_id int;
```

```
UPDATE book1 SET author_id = 1 WHERE author like 'Б%'
```

```
UPDATE book1 SET author_id = 2 WHERE author like 'Д%'
```

```
UPDATE book1 SET author_id = 3 WHERE author like 'Е%'
```

```
ALTER TABLE book1 DROP author
```

Добавим в таблицу book1 книгу “Лирика”, author_id = 4, цена 500, 2 штуки

Соединение INNER JOIN

```
SELECT title, name_author  
FROM author1 INNER JOIN book1  
ON author1.author_id = book1.author_id
```

Результат запроса формируется так:

каждая строка одной таблицы сопоставляется с каждой строкой второй таблицы;

для полученной «соединённой» строки проверяется условие соединения;

если условие истинно, в таблицу результата добавляется соответствующая «соединённая» строка

Внешнее соединение LEFT и RIGHT OUTER JOIN

```
SELECT title, author1.name_author  
FROM book1 LEFT JOIN author1  
ON author1.author_id = book1.author_id;
```

```
SELECT title, author1.name_author  
FROM book1 RIGHT JOIN author1  
ON author1.author_id = book1.author_id;
```

Результат запроса формируется так:

в результат включается внутреннее соединение (INNER JOIN) первой и второй таблицы в соответствии с условием;

затем в результат добавляются те записи первой таблицы, которые не вошли во внутреннее соединение на шаге 1, для таких записей соответствующие поля второй таблицы заполняются значениями NULL

Задание

Создайте запрос, который формирует отсортированный список авторов. Если книг авторов нет в таблице book, то на месте названия печатается Null.

(предварительно добавьте в список авторов Лермонтова).

```
SELECT name_author, title  
FROM author LEFT JOIN book ON author.author_id = book.author_id  
ORDER BY name_author;
```

Алгоритм взаимодействия с БД

1. Установка соединения с сервером БД функцией `connect()`
2. Выполнение запроса:
 - 2.1. Получить объект курсора методом `cursor()`
 - 2.2. Выполнить запрос методом `execute()`
 - 2.3. Если запрос на изменение данных или структуры БД, то нужно зафиксировать изменения методом `commit()`
 - 2.4. Если запрос на получение данных (SELECT):
Обработать в программе полученный результат
3. Закрывать соединение с сервером БД методом `close()`

Работа с Postgresql из Python программы

1. Install package psycopg2

2. Устанавливаем соединение:

```
import psycopg2
con = psycopg2.connect(
    database="postgres",
    user="postgres",
    password= "Здесь должен быть Ваш пароль",
    host="127.0.0.1",
    port="5432" # или другой порт
)
```

Создание таблицы

```
cur = con.cursor() # курсор
```

```
cur.execute("""CREATE TABLE STUDENT  
    (ADMISSION INT PRIMARY KEY NOT NULL,  
    NAME TEXT NOT NULL,  
    AGE INT NOT NULL,  
    COURSE CHAR(50),  
    DEPARTMENT CHAR(50));""")
```

```
con.commit() # commit
```

```
con.close() # закрыть соединение
```

```
#Проверьте, что таблица создана
```

Загрузка данных

```
cur = con.cursor()
```

```
cur.execute( "INSERT INTO STUDENT (ADMISSION,NAME,AGE,COURSE,DEPARTMENT) VALUES  
(3419, 'Abel', 17, 'Computer Science', 'ICT')")
```

```
cur.execute("INSERT INTO STUDENT (ADMISSION,NAME,AGE,COURSE,DEPARTMENT) VALUES  
(3421, 'Joel', 17, 'Computer Science', 'ICT')
```

```
cur.execute("INSERT INTO STUDENT (ADMISSION,NAME,AGE,COURSE,DEPARTMENT) VALUES  
(3422, 'Antony', 19, 'Electrical Engineering', 'Engineering')
```

```
cur.execute("INSERT INTO STUDENT (ADMISSION,NAME,AGE,COURSE,DEPARTMENT) VALUES  
(3423, 'Alice', 18, 'Information Technology', 'ICT')»)
```

```
con.commit()
```

```
con.close()
```

SELECT

```
cur = con.cursor()
```

```
cur.execute("SELECT admission, name, age, course, department from STUDENT")
```

```
rows = cur.fetchall() # возвращает список всех строк
```

```
for row in rows:
```

```
    print("ADMISSION =", row[0])
```

```
    print("NAME =", row[1])
```

```
    print("AGE =", row[2])
```

```
    print("COURSE =", row[3])
```

```
    print("DEPARTMENT =", row[4], "\n")
```

```
con.close()
```

Задание

Напишите программу, которая считывает информацию из таблицы book и печатает ее.

```
import psycopg2
con = psycopg2.connect(
    database="postgres",
    user="postgres",
    password = "*****",
    host="127.0.0.1",
    port="5433" # или 5432
)
cur = con.cursor()
cur.execute("SELECT * FROM book")
rows = cur.fetchall() # возвращает список всех строк
for row in rows:
    print(row)
con.close()
```

Pandas vs SQL

Operation	Pandas	SQL
Read CSV	<code>pd.read_csv(file)</code>	<code>LOAD DATA INFILE 'data.csv' INTO TABLE table FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 ROWS;</code>
Print first 10 (or k) rows	<code>df.head(10)</code>	<code>SELECT * FROM table LIMIT 10;</code>
Dimensions	<code>df.shape</code>	<code>SELECT count(*) FROM table;</code>
Datatype	<code>df.dtypes</code>	<code>DESCRIBE table;</code>
Filter Data	<code>df[df.column>10]</code>	<code>SELECT * FROM table where column>10;</code>
Select column(s)	<code>df.column</code>	<code>SELECT column FROM table;</code>
Sort	<code>df.sort_values("column")</code>	<code>SELECT * FROM table ORDER BY column;</code>
Fill NaN	<code>df.column.fillna(0)</code>	<code>UPDATE table SET column=0 WHERE column IS NULL;</code>
Join	<code>pd.merge(df1, df2, on = "col", how = "inner")</code>	<code>SELECT * FROM table1 JOIN table2 ON (table1.col = table2.col);</code>
Concatenate	<code>pd.concat((df1, df2))</code>	<code>SELECT * FROM table1 UNION ALL table2;</code>
Group	<code>df.groupby("column"). agg_col.mean()</code>	<code>SELECT column, avg(agg_col) FROM table GROUP BY column;</code>
Unique values	<code>df.column.unique()</code>	<code>SELECT DISTINCT column FROM table;</code>
Rename column	<code>df.rename(columns = {"old_name": "new_name"})</code>	<code>ALTER TABLE table RENAME COLUMN old_name TO new_name;</code>
Delete column	<code>df.drop(columns = ["column"])</code>	<code>ALTER TABLE table DROP COLUMN column;</code>

import collections

`collections.Counter` - вид словаря, который позволяет нам считать количество неизменяемых объектов (в большинстве случаев, [строки](#)).

`collections.deque(iterable, [maxlen])` - создаёт очередь из итерируемого объекта с максимальной длиной `maxlen`. Очереди очень похожи на списки, за исключением того, что добавлять и удалять элементы можно либо справа, либо слева. (`appendleft`, `popleft`)

`collections.OrderedDict` - ещё один похожий на словарь объект, но он помнит порядок, в котором ему были даны ключи (после версии 3.6 любой словарь помнит этот порядок)

`collections.namedtuple` позволяет создать тип данных, ведущий себя как кортеж, с тем дополнением, что каждому элементу присваивается имя, по которому можно в дальнейшем получать доступ

`collections.Counter`

```
import collections
```

```
a = collections.Counter('aabbbbccccdddeeeeabcdef')
```

```
print(f"{a=}")
```

```
b = dict(a)
```

```
print(f"{b=}")
```

```
a=Counter({'c': 5, 'd': 5, 'e': 5, 'b': 4, 'a': 3, 'f': 1})
```

```
b={'a': 3, 'b': 4, 'c': 5, 'd': 5, 'e': 5, 'f': 1}
```


Задача 23-1

Найдите длину наибольшей подстроки данной строки, которая является палиндромом.

Например, дана строка 'aabbccddcc' тогда длиной подстроки с наибольшим палиндромом является 6 (подстрока 'ccddcc')

Задача 23-2

1. Напишите программу, которая считывает информацию из таблицы book и печатает ее в виде таблицы, соответствующей таблице из базы данных.
2. Более сложный вариант составить DataFrame на основании считанной информации, соответствующий таблице из базы данных.

Задача 23-3

Создайте функцию, на вход которой подается список из целых положительных чисел, и которая в качестве результата возвращает самое большое число, которое можно составить из этих чисел.

Например, вход [1, 21, 3], результат 3211

Если вход [9, 81, 25], то результат 98125.