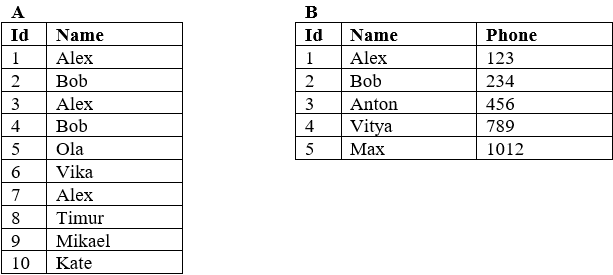
Задачи

SQL

**Задание 1.**

Есть две таблицы: A и B.



Сколько строк получится при следующих запросах:

**SELECT** \* **FROM** A LEFT **JOIN** B **ON** A**.Name** = B**.Name**  
**SELECT** \* **FROM** A RIGHT **JOIN** B **ON** A**.Name** = B**.Name**

Скрытый текст

1) 10 строк

2) 8 строк

**Задание 2.**

Вывести идентификаторы пользователей и последнюю пару вопрос-ответ по каждому пользователю, где есть текстовый ответ ассистента.

**import** duckdb **as** db  
  
conn = db.connect()  
conn.execute('''  
-- Есть таблица, в которой хранятся реквесты и респонсы:  
  
CREATE TABLE MESSAGES (  
ID integer NOT NULL,  
MESSAGE\_ID integer NOT NULL,  
USER\_ID varchar(10) NOT NULL,  
MESSAGE\_TYPE varchar(10) NOT NULL,  
QUERY varchar(40) NOT NULL,  
CREATED timestamp NOT NULL)  
;  
  
INSERT INTO MESSAGES VALUES  
(1, 111, 'user\_1', 'REQUEST', 'привет', '2021-07-21 12:51:39.0000'),  
(2, 222, 'user\_2', 'REQUEST', 'перевод', '2021-07-21 12:52:03.0000'),  
(3, 111, 'user\_1', 'RESPONSE', 'Приветствую тебя!', '2021-07-21 12:52:48.0000'),  
(4, 333, 'user\_3', 'REQUEST', 'включи yputube', '2021-07-21 12:53:57.0000'),  
(5, 444, 'user\_1', 'REQUEST', 'как дела', '2021-07-21 13:05:13.0000');  
''')

conn.query('''  
# Ваш код  
  
''').df()

Скрытый текст

conn.query('''  
WITH LastRequest AS (SELECT USER\_ID, MESSAGE\_ID, QUERY, CREATED, ROW\_NUMBER() OVER (PARTITION BY USER\_ID ORDER BY CREATED DESC) AS row\_num  
FROM MESSAGES  
WHERE MESSAGE\_TYPE = 'REQUEST')  
  
SELECT lr.USER\_ID, lr.MESSAGE\_ID, lr.QUERY AS LAST\_Request, res.QUERY AS Last\_Response  
FROM LastRequest lr  
JOIN (  
 SELECT MESSAGE\_ID, QUERY  
 FROM MESSAGES  
 WHERE MESSAGE\_TYPE = 'RESPONSE'  
) res  
ON lr.MESSAGE\_ID = res.MESSAGE\_ID  
WHERE lr.row\_num = 1  
ORDER BY lr.USER\_ID;  
''').df()

**Задание 3.**

Есть две таблицы:

Orders (order\_id – PK, promocode\_id – FK) - заказы

Promocodes (promocode\_id – PK, name – UQ, discount) – промокоды

Вопросы:

1. Какая доля заказов с промокодами?
2. Какой самый популярный промокод (название) и число его использований?

Скрытый текст

Ответ на вопрос №1:

**SELECT** p**.name** **AS** name\_promocode, **COUNT**(\*) **AS** usage\_count  
**FROM** orders o  
**JOIN** promocodes p **ON** o**.promocode\_id** = p**.promocode\_id**  
**GROUP** **BY** p**.promocode\_id**, p**.name**  
**ORDER** **BY** usage\_count **DESC**  
**LIMIT** 1

Ответ на вопрос №2:

**SELECT** p**.name** **AS** name\_promocode, **COUNT**(\*) **AS** usage\_count  
**FROM** orders o  
**JOIN** promocodes p **ON** o**.promocode\_id** = p**.promocode\_id**  
**GROUP** **BY** p**.promocode\_id**, p**.name**  
**ORDER** **BY** usage\_count **DESC**  
**LIMIT** 1

**Задание 4.**

Если в таблице "А" 10 строк, а в таблице "Б" – 100 строк. Сколько строк будет в SELECT  
(id – PK)?

**SELECT** \* **FROM** A LEFT **JOIN** B **ON** A**.id** = B**.id**

Скрытый текст

Количество строк в результате будет 10. Поскольку мы сравниваем строки по id.

**Задание 5.**

Напишите SQL-запрос, чтобы из таблицы изъять имена, которые начинаются на букву А.

Скрытый текст

**SELECT** name **FROM** **table** **WHERE** name **LIKE** "A%";

или

**SELECT** name **FROM** **table** **WHERE** LOWER(name) **LIKE** "a%";

Вопросы

SQL

1. **Назовите виды JOIN.**

Скрытый текст

**Существуют следующие типы соединений (5 видов):**

1. *INNER JOIN или просто JOIN* - внутреннее соединение. В результате остаются только те строки, для которых нашлось соответствие. До сих пор мы использовали только этот тип соединений.
2. *LEFT JOIN* - левое внешнее соединение. Работает как JOIN, но если для строки таблицы, находящейся по левую сторону ключевого слова LEFT JOIN, не нашлось ни одной строки в таблице, находящейся по правую сторону LEFT JOIN, то строка все равно добавляется в результат, а значения столбцов правой таблицы равны null.
3. *RIGHT JOIN* - правое внешнее соединение. Работает как JOIN, но если для строки таблицы, находящейся по правую сторону ключевого слова RIGHT JOIN, не нашлось ни одной строки в таблице, находящейся по левую сторону RIGHT JOIN, то строка все равно добавляется в результат, а значения столбцов левой таблицы равны null.
4. *FULL JOIN* - полное внешнее соединение. Если для какой-либо из таблиц не нашлось строки в другой таблице, то строка все равно попадает в результат, а значения столбцов другой таблицы равны null.
5. *CROSS JOIN* - перекрестное (или декартово) произведение. Каждая строка одной таблицы соединяется с каждой строкой второй таблицы, давая тем самым в результате все возможные сочетания строк двух таблиц. Аналогичного результата можно достичь просто перечислив таблицы в FROM через запятую.
6. **Что такое Primary Key и какой он может быть (из чего состоять)?**

Скрытый текст

*Первичный ключ (англ. primary key)* — это специальное поле (или набор полей) в таблице базы данных, которое уникально идентифицирует каждую запись (строку) в этой таблице. Простым языком, PK — это уникальный "идентификационный номер" для каждой строки в таблице, который не может повторяться.  
Он может быть одиночным, т.е. состоять из 1 столбца, а может быть составным, т.е. состоять из 2 и более столбцов.

**Основные характеристики первичного ключа:**

1. Уникальность: Значение первичного ключа должно быть уникальным для каждой строки в таблице. Никакие две строки не могут иметь одно и то же значение PK.
2. Не может быть пустым: Значение первичного ключа не может быть NULL.
3. Идентификация: Используется для быстрой и точной идентификации записи в таблице.
4. **В чем заключается различие между условиями HAVING и WHERE в SQL?**

Скрытый текст

В SQL условия HAVING и WHERE используются для фильтрации данных, но они применяются на разных этапах выполнения запроса и имеют разные цели:

1. **WHERE**:
   * Применяется для фильтрации строк перед их группировкой и агрегацией.
   * Используется с отдельными строками данных.
   * Пример: SELECT \* FROM table WHERE condition;
2. **HAVING**:
   * Применяется для фильтрации результатов после выполнения группировки (с использованием GROUP BY).
   * Используется для фильтрации агрегированных данных (например, с функциями SUM, COUNT, AVG и т.д.).
   * Пример: SELECT column1, COUNT(*) FROM table GROUP BY column1 HAVING COUNT(*) > 1;

Таким образом, WHERE используется для фильтрации исходных данных, а HAVING — для фильтрации агрегированных результатов.

1. **Объясните разницу между командами DELETE и TRUNCATE?**

Скрытый текст

Команда **DELETE** — это DML-операция, которая удаляет записи из таблицы, соответствующие заданному условию:

**DELETE** **FROM** table\_name **WHERE** condition;

При этом создаются логи удаления, то есть операцию можно отменить. А вот команда **TRUNCATE** — это DDL-операция, которая полностью пересоздаёт таблицу, и отменить такое удаление невозможно:

TRUNCATE **TABLE** table\_name;

1. **Как найти дубли в поле email?**

Скрытый текст

**SELECT** email, **COUNT**(email)   
**FROM** customers   
**GROUP** **BY** email   
**HAVING** **COUNT**(email) > 1;

Функция **COUNT()** возвращает количество строк из поля email. Оператор **HAVING** работает почти так же, как и **WHERE**, вот только применяется не для всех столбцов, а для набора, созданного оператором **GROUP BY**.

1. **Что такое выражение GROUP BY в SQL, и в каких ситуациях оно используется?**

Скрытый текст

Выражение **GROUP BY** в SQL используется для группировки строк в наборе результатов по одному или нескольким столбцам. Это особенно полезно, когда вам нужно выполнять агрегатные функции, такие как COUNT(), SUM(), AVG(), MAX(), или MIN(), на группах данных.

Пример использования:

Предположим, у вас есть таблица sales, которая содержит информацию о продажах, включая колонны salesperson (имя продавца) и amount (сумма продажи). Вы хотите узнать общую сумму продаж для каждого продавца. Для этого можно использовать выражение GROUP BY следующим образом:

**SELECT** salesperson, SUM(amount) **AS** total\_sales  
**FROM** sales  
**GROUP** **BY** salesperson;

Разбор примера:

* SELECT salesperson, SUM(amount): выбираем имя продавца и сумму всех продаж для этого продавца.
* FROM sales: из таблицы sales.
* GROUP BY salesperson: группируем строки по имени продавца.

Результат запроса будет содержать строки, где каждая строка представляет уникального продавца и общую сумму его продаж.

Когда использовать GROUP BY:

1. **Анализ данных:** Когда нужно вычислить статистические данные по группам, например, общее количество продаж по регионам или среднюю зарплату по должностям.
2. **Отчеты:** При создании отчетов, показывающих агрегированные данные, такие как суммарные продажи по месяцам или количество заказов по клиентам.
3. **Оптимизация:** Для уменьшения объема данных при работе с большими наборами данных, группируя и агрегируя информацию, прежде чем ее передать в приложение для дальнейшей обработки.
4. **Что такое индексы в SQL? Какие преимущества и недостатки?**

Скрытый текст

*Индексы в SQL* — это специальные структуры данных, которые используются для ускорения выполнения запросов. Они создаются на основе одного или нескольких столбцов таблицы и позволяют быстро находить строки соответствующие условиям запроса.

**Плюсы индексов:**

1. Ускорение запросов: Индексы значительно уменьшают время выполнения запросов SELECT, особенно тех, что включают WHERE, JOIN, и ORDER BY операторы.
2. Быстрый доступ к данным: Поиск данных по индексам осуществляется быстрее, так как они упорядочены.
3. Уменьшение нагрузок: Благодаря индексам сервер базы данных загружается меньше, так как нужные данные находятся быстрее.

**Минусы индексов:**

1. Дополнительное место: Индексы требуют дополнительного дискового пространства.
2. Замедление операций записи: Вставка, обновление и удаление данных могут замедляться, так как индексы должны обновляться соответственно.
3. Сложность управления: Неправильное или чрезмерное использование индексов может действительно навредить производительности базы данных.

Индексы являются мощным инструментом для оптимизации производительности запросов, однако важно балансировать между их числом и негативным влиянием на операции записи. Компетентное управление индексами может заметно повысить эффективность работы любой системы баз данных.

Проходя через эти вопросы и задачи, вы не просто готовитесь к собеседованию, а обретаете уверенность и навыки, которые пригодятся вам в профессиональной деятельности. И помните, что каждое интервью — это не только возможность получить работу, но и шанс узнать что-то новое и улучшить свои навыки.