Школа DS Reboot

Занятие 6: Основы SQL. Продвинутые запросы с собеседований: порядковые статистики, SELF JOIN, добавление столбцов без оконных функций. Работа с датами

Порядок исполнения SQL-запросов

SQL queries run in this order

FROM + JOIN

↓

WHERE

↓

GROUP BY

↓

HAVING

↓

SELECT (window functions happen here!)

↓

ORDER BY

↓

LIMIT

Вопросы с реальных собеседований

Порядковые статистики (без оконных функций)

Задача: найти человека который прочитал максимальное кол-во книг

family_members

id	name	species	num_books_read	num_legs
1	Dave	human	200	2
2	Mary	human	180	2
3	Pickles	dog	0	4

family_members

id	name	species	num_books_read	num_legs
1	Dave	human	200	2
2	Mary	human	180	2
3	Pickles	dog	0	4

SQL:

```
SELECT * FROM family_members
WHERE num_books_read = (SELECT
MAX(num_books_read) FROM family_members);
```

Result:

id	name	species	num_books_read	num_legs
1	Dave	human	200	2

Порядковые статистики (без оконных функций)

Задача: найти существо у которого наибольшее кол-во ног

family_members

id	name	species	num_books_read	num_legs
1	Dave	human	200	2
2	Mary	human	180	2
3	Pickles	dog	0	4

Решение

family_members

id	name	species	num_books_read	num_legs
1	Dave	human	200	2
2	Mary	human	180	2
3	Pickles	dog	0	4

SQL:

```
SELECT * FROM family_members WHERE num_legs =
(SELECT MAX(num_legs) FROM family_members);
```

Result:

id	name	species	num_books_read	num_legs
3	Pickles	dog	0	4

Порядковые статистики (без оконных функций)

Задача: найдите второй по цене товар в таблице Products

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

Решение: вложенные циклы в коррелированных подзапросах

```sql
SELECT DISTINCT ProductName, Price
FROM Products P1
WHERE 2 = (
    SELECT COUNT(DISTINCT Price)
    FROM Products P2
    WHERE P1.Price <= P2.Price
)
```

```
[2]  1  prices = np.random.permutation(np.arange(10))
     2  prices
```

```
array([7, 2, 0, 6, 5, 1, 3, 9, 8, 4])
```

```
1  k = 3
2  for i in range(len(prices)):
3      n_leq = 0
4      for j in range(len(prices)):
5          if prices[j] >= prices[i]:
6              n_leq += 1
7      if n_leq == k:
8          print(prices[i])
9          break
```

Работа с датами

1. **DATE** возвращает дату
2. **TIME** возвращает время
3. **DATETIME** возвращает дату и время

DATE

```
SELECT DATE('2018-11-01');
SELECT DATE('now');
SELECT DATE('now', 'localtime');
SELECT DATE('2018-11-01','-1 day');
SELECT DATE('2018-11-01','-1 month');
SELECT DATE('2018-11-01','+1 day');
```

TIME

```
SELECT TIME('10:20:30','+2 hours');
```

```
SELECT TIME('now','+2 hours');
```

```
SELECT TIME('2018-11-02 15:20:15');
```

Задача: Отобразить все заказы таблицы Orders сделанные хотя бы 23 года назад

Задача: Отобразить последний день предыдущего месяца

DATETIME

```
SELECT datetime('now');
```

```
CREATE TABLE referrals(
    id INTEGER PRIMARY KEY,
    source TEXT NOT NULL,
    created_at TEXT DEFAULT CURRENT_TIMESTAMP
);
```

DATE

Вы можете сравнивать даты при помощи операторов сравнения < и >

Задача: найдите тех звезд, которые родились раньше 1 января 1982

celebs_born

id	name	birthdate
1	Michael Jordan	1963-02-17
2	Justin Timberlake	1981-01-31
3	Taylor Swift	1989-12-13

Решение

celebs_born

id	name	birthdate
1	Michael Jordan	1963-02-17
2	Justin Timberlake	1981-01-31
3	Taylor Swift	1989-12-13

SQL:

```
SELECT * FROM celebs_born
WHERE birthdate < '1982-01-01'
```

Result:

id	name	birthdate
1	Michael Jordan	1963-02-17
2	Justin Timberlake	1981-01-31

DATE

Вы можете сравнивать даты при помощи операторов сравнения < и >

Задача: найдите тех звезд, которые родились в промежуток с 1 января 1980 по 1 декабря 2020

celebs_born

id	name	birthdate
1	Michael Jordan	1963-02-17
2	Justin Timberlake	1981-01-31
3	Taylor Swift	1989-12-13

Решение

celebs_born

id	name	birthdate
1	Michael Jordan	1963-02-17
2	Justin Timberlake	1981-01-31
3	Taylor Swift	1989-12-13

SQL:

```
SELECT * FROM celebs_born
WHERE (birthdate  BETWEEN '1980-01-01' AND '2013-
01-09')
```

Result:

id	name	birthdate
2	Justin Timberlake	1981-01-31
3	Taylor Swift	1989-12-13

Self join

Элемент **SELF JOIN** используется для объединения таблицы с ней самой таким образом, будто это две разные таблицы, временно переименовывая одну из них.

Запрос и использованием **SELF JOIN** имеет следующий вид:

```
SELECT a.имяколонки, b.имя_колонки...
FROM таблица1 a, таблица1 b
WHERE a.общее поле = b.общее поле;
```

Задание

Запрос и использованием **SELF JOIN** имеет следующий вид:

```
SELECT a.имяколонки, b.имя_колонки...
FROM таблица1 a, таблица1 b
WHERE a.общее поле = b.общее поле;
```

Задача: Сопоставьте каждому объекту пару в соответствии с defeats_id

rps

id	name	defeats_id
1	Rock	3
2	Paper	1
3	Scissors	2

Ожидаемый результат:

Result:

object	beats
Rock	Scissors
Paper	Rock
Scissors	Paper

Решение

rps

id	name	defeats_id
1	Rock	3
2	Paper	1
3	Scissors	2

SQL:

```
SELECT r1.name AS object, r2.name AS beats
FROM rps AS r1
INNER JOIN rps AS r2
ON r1.defeats_id = r2.id;
```

Result:

object	beats
Rock	Scissors
Paper	Rock
Scissors	Paper

Добавление константного столбца с числом строк в таблице (без оконных функций)

Задача. Подумайте, как добавить к произвольной таблице с известным первичным ключом константный столбец, содержащий число строк в ней. Если вы знаете, что такое оконные функции, то придумайте решение без них.