MySQL: Агрегирующие функции, группировки и объединения таблиц

1. Агрегирующие функции

Основные агрегирующие функции:

- COUNT() - подсчет количества строк
- SUM() - сумма значений
- AVG() - среднее значение
- MAX() - максимальное значение
- MIN() - минимальное значение

Примеры использования:

```
-- Подсчет всех строк в таблице
SELECT COUNT(*) FROM users;

-- Подсчет уникальных значений
SELECT COUNT(DISTINCT city) FROM users;

-- Среднее значение зарплаты
SELECT AVG(salary) FROM employees;

-- Максимальная и минимальная цена
SELECT MAX(price), MIN(price) FROM products;
```

2. Группировка данных (GROUP BY)

Синтаксис:

```
SELECT column1, AGG_FUNCTION(column2)
FROM table_name
GROUP BY column1;
```

Примеры использования:

```
-- Количество пользователей по городам
SELECT city, COUNT(*) as users_count
FROM users
GROUP BY city;

-- Средняя зарплата по отделам
SELECT department, AVG(salary) as avg_salary
FROM employees
GROUP BY department;
```

HAVING

Используется для фильтрации после группировки:

```
SELECT department, AVG(salary) as avg_salary
FROM employees
GROUP BY department
HAVING avg_salary > 50000;
```

3. Объединение таблиц (JOINs)

Типы объединений:

- INNER JOIN - только записи с совпадениями в обеих таблицах
- LEFT JOIN - все записи из левой таблицы и совпадения из правой
- RIGHT JOIN - все записи из правой таблицы и совпадения из левой
- FULL JOIN - все записи из обеих таблиц

Примеры:

```
-- INNER JOIN
SELECT users.name, orders.order_date
FROM users
INNER JOIN orders ON users.id = orders.user_id;

-- LEFT JOIN
SELECT users.name, orders.order_date
FROM users
LEFT JOIN orders ON users.id = orders.user_id;

-- Объединение нескольких таблиц
SELECT users.name, orders.order_date, products.name
FROM users
INNER JOIN orders ON users.id = orders.user_id
INNER JOIN products ON orders.product_id = products.id;
```

4. Подзапросы (Subqueries)

В условии WHERE:

```
SELECT name
FROM products
WHERE price > (SELECT AVG(price) FROM products);
```

В операторе FROM:

```
SELECT t.department, t.avg_salary
FROM (
    SELECT department, AVG(salary) as avg_salary
    FROM employees
    GROUP BY department
) t
WHERE t.avg_salary > 50000;
```

В операторе JOIN:

```
SELECT users.name
FROM users
INNER JOIN (
    SELECT user_id
    FROM orders
    GROUP BY user_id
    HAVING COUNT(*) > 5
) frequent_buyers
ON users.id = frequent_buyers.user_id;
```

5. Алиасы (псевдонимы)

Алиасы для таблиц:

```
-- Базовый синтаксис
SELECT u.* FROM users AS u;
-- Сокращенный синтаксис (без AS)
SELECT e.* FROM employees e;

-- В сложных запросах
SELECT
    e.name AS employee_name,
    d.name AS department_name
FROM employees e
JOIN departments d ON e.dept_id = d.id;
```

Алиасы для столбцов:

```
-- С использованием AS
SELECT
    first_name AS name,
    salary AS monthly_salary
FROM employees;

-- Без использования AS
SELECT
    first_name name,
    salary monthly_salary
FROM employees;
```

Алиасы в подзапросах:

```
SELECT avg_data.department, avg_data.average_salary
FROM (
    SELECT
        department,
        AVG(salary) AS average_salary
    FROM employees
    GROUP BY department
) AS avg_data;
```

6. Полезные советы

Лучшие практики использования алиасов:

- Используйте короткие, но понятные имена для алиасов таблиц
- Всегда указывайте алиасы при работе с несколькими таблицами
- Давайте осмысленные имена для вычисляемых полей
- Используйте префиксы таблиц для избежания неоднозначности

Оптимизация запросов:

- Избегайте SELECT * - выбирайте только нужные поля
- При больших объемах данных используйте LIMIT

Общие рекомендации:

- Всегда указывайте условие соединения в JOIN
- При группировке включайте в SELECT только сгруппированные поля и агрегатные функции