SQL - основные запросы

SELECT 'столбцы или * для выбора всех столбцов; обязательно'

FROM 'таблица; обязательно'

WHERE 'условие/фильтрация, например, city = 'Moscow'; необязательно'

GROUP BY 'столбец, по которому хотим сгруппировать данные; необязательно'

HAVING 'условие/фильтрация на уровне сгруппированных данных; необязательно'

ORDER BY 'столбец, по которому хотим отсортировать вывод; необязательно'

Порядок выполнения запросов:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

```
SELECT * FROM Customers;
```

Выбрать **все** (обозначается как *) из таблицы **Customers**

```
SELECT model, speed, ram, hd, price, screen
FROM Laptop
-- WHERE ram IN (32, 64) -- ананлогично WHERE ram = 32 OR ram = 64
-- WHERE ram = 32 OR ram = 64 AND price <= 970
WHERE price BETWEEN 700 AND 970 -- 700 и 970 включительно
```

Выбери **model, speed, ram, hd, price, screen** из таблицы **Laptop** где price от 700 **до** 970

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

Выведи **title** как **Название** (в выводе переименует атрибут title).

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

Выведи title, author, price, amount, и **вычисляемый столбец total, равный price * amount**.

Математические 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...	

```
SELECT author, title,
ROUND(IF(author = 'Булгаков М.А.', price * 1.1, IF(author = 'Есенин С.А.', price * 1.05, price) ), 2) AS new_price
FROM book;
```

Если автор Булгаков М.А. ⇒ новая цена поднимется на 10%

Если автор Есенин С.А. ⇒ новая цена поднимется на 5%

Вывод осуществляется в вычисляемый столбец **new_price**.

В PostgreSQL вместо `IF` используют конструкцию `CASE ... WHEN ... THEN ... ELSE ... END` или IIF и некоторые другие.

```
...
    CASE
        WHEN author = 'Text 1' THEN price * 2
        WHEN author = 'Text 2' THEN price * 3
```

```
        ELSE price END
...
```

Предика́т

— это утверждение, высказанное о субъекте. Субъектом высказывания называется то, о чём делается утверждение.

Логические операторы при отсутствии скобок, как и арифметические операторы, выполняются в соответствии с их старшинством. Одноместная операция NOT имеет наивысший приоритет, затем AND и только потом OR

Приоритеты операций:

1. круглые скобки
2. умножение (*), деление (/)
3. сложение (+), вычитание (-)
4. операторы сравнения (=, >, <, >=, <=, <>)
5. NOT
6. AND
7. OR

Примеры простых предикатов сравнения:

предикат	**описание**
price < 1000	Цена меньше 1000
type = ‘laptop’	Типом продукции является портативный компьютер
cd = ‘24x’	24-скоростной CD-ROM
color <> ’y’	Не цветной принтер
ram – 128 > 0	Объем оперативной памяти свыше 128 Мбайт
Price <= speed*2	Цена не превышает удвоенной частоты процессора

BETWEEN, IN

Логическое выражение после ключевого слова `WHERE` может включать операторы `BETWEEN` и `IN`. Приоритет у этих операторов такой же как у операторов сравнения, то есть они выполняются раньше, чем `NOT`, `AND`, `OR`.

Оператор `BETWEEN` позволяет отобрать данные, относящиеся к некоторому интервалу, включая его границы.

```
SELECT title, amount
FROM book
```

```
WHERE amount BETWEEN 5 AND 14;
```

Результат:

```
+---------------+--------+
| title         | amount |
+---------------+--------+
| Белая гвардия | 5      |
| Идиот         | 10     |
+---------------+--------+
```

Этот запрос можно реализовать по-другому, результат будет точно такой же.

```
SELECT title, amount
FROM book
WHERE amount >= 5 AND amount <=14;
```

Оператор `IN` позволяет выбрать данные, соответствующие значениям из списка.

```
SELECT title, price
FROM book
WHERE author IN ('Булгаков М.А.', 'Достоевский Ф.М.');
```

Результат:

```
+--------------------+--------+
| title              | price  |
+--------------------+--------+
| Мастер и Маргарита | 670.99 |
| Белая гвардия      | 540.50 |
| Идиот              | 460.00 |
| Братья Карамазовы  | 799.01 |
+--------------------+--------+
```

Этот запрос можно реализовать по-другому, результат будет точно такой же.

```
SELECT title, price
FROM book
WHERE author = 'Булгаков М.А.' OR author = 'Достоевский Ф.М.';
```

оператор LIKE

Оператор `LIKE` используется для сравнения строк. В отличие от операторов отношения равно (**=**) и не равно (**<>**), `LIKE` позволяет сравнивать строки не на полное совпадение (не совпадение), а в соответствии с шаблоном. Шаблон может включать **обычные символы** и **символы-шаблоны**. При сравнении с шаблоном, его обычные символы должны в точности совпадать с символами, указанными в строке. Символы-шаблоны могут совпадать с произвольными элементами символьной строки.

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.

Вывести названия книг, начинающихся с буквы «*Б*».

Запрос:

```
SELECT title
FROM book
WHERE title LIKE 'Б%';
/* эквивалентное условие
title LIKE 'б%'
*/
```

Результат:

```
+-------------------+
| title             |
+-------------------+
| Белая гвардия     |
| Братья Карамазовы |
+-------------------+
```

Предикат в языке SQL

может принимать одно из трех значений **TRUE** (истина), **FALSE** (ложь) или **UNKNOWN** (неизвестно).

Исключение, которые не могут принимать значение **UNKNOWN,** составляют следующие предикаты:

- **IS NULL** (отсутствие значения),
- **EXISTS** (существование),
- **UNIQUE** (уникальность)
- **MATCH** (совпадение).

Правила комбинирования всех трех истинностных значений легче запомнить, обозначив

- **TRUE** как **1**,
- **FALSE** как **0**
- **UNKNOWN** как **1/2** (где-то между истинным и ложным значениями).

AND с двумя истинностными значениями дает минимум этих значений. Например, **TRUE AND UNKNOWN** будет равно **UNKNOWN**.

OR с двумя истинностными значениями дает максимум этих значений. Например, **FALSE OR UNKNOWN** будет равно **UNKNOWN**.

Отрицание истинностного значения равно 1 минус данное истинностное значение. Например, **NOT UNKNOWN** будет равно **UNKNOWN**.

Работа с ТАБЛИЦАМИ

```
CREATE TABLE team
(
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name varchar(50) NOT NULL,
    last_name varchar(50) NOT NULL,
    language varchar(20),
    level varchar(20),
);
```

Создать таблицу **team** со **столбцами id, first_name, last_name, language, level**. И **указываем тип данных**, хранящийся в каждом столбце.

varchar (n | max) - строковые данные переменного размера. Используйте *n* , чтобы определить размер строки в байтах и может быть значением от 1 до 8000 или использовать **максимальное** значение

char (n) - строковые данные фиксированного размера. *n* определяет размер строки в байтах и должно иметь значение от 1 до 8000

- для MySql - INT PRIMARY KEY AUTO_INCREMENT / для **MS server** - **INT IDENTITY(1,1) PRIMARY KEY**

```
INSERT INTO team (id, first_name, last_name, language, level) VALUES ('1', 'Дмитрий', 'Васильев', 'python', 'junior');
```

```
INSERT INTO team_new (id, first_name, last_name, language, level) VALUES ('11111', 'Алекс', 'Жучариков', 'c#', 'junior');
```

Добавить строку в таблицу **team** по указанным полям.

```
DELETE FROM team_new WHERE id = '11111';
```

Удалить из таблицы **team_new строки**, где значение id = = '11111'

```
ALTER TABLE MyDataBase.dbo.users
ADD age INT;
```

Добавить столбец age с типом данных INT **в таблицу users**.

```
UPDATE team_new SET id = '11' WHERE last_name = 'Жучариков';
```

Заменить (обновить) данные в поле **id на значение 11**, в строках, **где значение last_name =** 'Жучариков'

Запросом `UPDATE` можно обновлять значения нескольких столбцов одновременно. В этом случае простейший запрос будет выглядеть так:

```
UPDATE таблица SET поле1 = выражение1, поле2 = выражение2
```

На складе, кроме хранения и получения книг, выполняется их оптовая продажа. Для реализации этого действия включим дополнительный столбец `buy` в таблицу `book`:

book_id	title	author	price	amount	buy
INT PRIMARY KEY AUTO_INCREMENT	**VARCHAR(50)**	**VARCHAR(30)**	**DECIMAL(8,2)**	**INT**	int
1	Мастер и Маргарита	Булгаков М.А.	670.99	3	0
2	Белая гвардия	Булгаков М.А.	540.50	5	3
3	Идиот	Достоевский Ф.М.	460.00	10	8
4	Братья Карамазовы	Достоевский Ф.М.	799.01	2	0
5	Стихотворения и поэмы	Есенин С.А.	650.00	15	18

Пример

В столбце `buy` покупатель указывает количество книг, которые он хочет приобрести. Для каждой книги, выбранной покупателем, необходимо уменьшить ее количество на складе на указанное в столбце `buy` количество, а в столбец `buy` занести 0.

Запрос:

```
UPDATE book
SET amount = amount - buy,
    buy = 0;

SELECT * FROM book;
```

Результат:

```
Affected rows: 3
Query result:
+---------+----------------------+------------------+--------+--------+-----+
| book_id | title                | author           | price  | amount | buy |
+---------+----------------------+------------------+--------+--------+-----+
| 1       | Мастер и Маргарита   | Булгаков М.А.    | 670.99 | 3      | 0   |
| 2       | Белая гвардия        | Булгаков М.А.    | 540.50 | 2      | 0   |
| 3       | Идиот                | Достоевский Ф.М. | 460.00 | 2      | 0   |
| 4       | Братья Карамазовы    | Достоевский Ф.М. | 799.01 | 2      | 0   |
| 5       | Стихотворения и поэмы | Есенин С.А.      | 650.00 | -3     | 0   |
+---------+----------------------+------------------+--------+--------+-----+
```

Задание

В таблице **book** необходимо скорректировать значение для покупателя в столбце **buy** таким образом, чтобы оно не превышало количество экземпляров книг, указанных в столбце **amount**. А цену тех книг, которые покупатель не заказывал, снизить на 10%.

UPDATE book
SET buy = IF(buy > amount, amount, buy),
price = IF(buy = 0, price*0.9, price);

SELECT * FROM book;

с условием

```
DROP TABLE team
```

Удалить таблицу

```
SELECT * INTO team_new FROM team
```

Копировать таблицу - создается таблица **team_new** и копируются все данные из таблицы **team**.

Добавление записей из другой таблицы

С помощью запроса на добавление можно не только добавить в таблицу конкретные значения (список `VALUES`), но и записи из другой таблицы, отобранные с помощью запроса на выборку. В этом случае

вместо раздела `VALUES` записывается запрос на выборку, начинающийся с `SELECT`. В нем можно использовать `WHERE, GROUP BY, ORDER BY`

Запрос:

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

SELECT * FROM book;
```

Результат:

```
Affected rows: 4
Query result:
+---------+----------------------+------------------+--------+--------+
| book_id | title                | author           | price  | amount |
+---------+----------------------+------------------+--------+--------+
| 1       | Мастер и Маргарита   | Булгаков М.А.    | 670.99 | 3      |
| 2       | Белая гвардия        | Булгаков М.А.    | 540.50 | 5      |
| 3       | Идиот                | Достоевский Ф.М. | 460.00 | 10     |
| 4       | Братья Карамазовы    | Достоевский Ф.М. | 799.01 | 2      |
| 5       | Стихотворения и поэмы | Есенин С.А.      | 650.00 | 15     |
| 6       | Лирика               | Пастернак Б.Л.   | 518.99 | 2      |
| 7       | Черный человек       | Есенин С.А.      | 570.20 | 6      |
| 8       | Белая гвардия        | Булгаков М.А.    | 540.50 | 7      |
| 9       | Идиот                | Достоевский Ф.М. | 360.80 | 3      |
+---------+----------------------+------------------+--------+--------+
Affected rows: 9
```

Пример

Занести из таблицы `supply` в таблицу `book` только те книги, названия которых отсутствуют в таблице `book.`

Запрос:

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

SELECT * FROM book;
```

Результат:

```
Affected rows: 2

Query result:
+---------+----------------------+------------------+--------+--------+
| book_id | title                | author           | price  | amount |
+---------+----------------------+------------------+--------+--------+
| 1       | Мастер и Маргарита   | Булгаков М.А.    | 670.99 | 3      |
| 2       | Белая гвардия        | Булгаков М.А.    | 540.50 | 5      |
```

```
| 3       | Идиот                 | Достоевский Ф.М. | 460.00 | 10     |
| 4       | Братья Карамазовы     | Достоевский Ф.М. | 799.01 | 2      |
| 5       | Стихотворения и поэмы | Есенин С.А.      | 650.00 | 15     |
| 6       | Лирика                | Пастернак Б.Л.   | 518.99 | 2      |
| 7       | Черный человек        | Есенин С.А.      | 570.20 | 6      |
+---------+-----------------------+------------------+--------+--------+
```

Сортировка и группировка

```
SELECT last_name, first_name, birthday FROM users
WHERE age >= 18
ORDER BY last_name
```

ORDER BY **сортирует** запрошенные поля **из таблицы users по last_name**. Так как last_name имеет строковый тип данных, то сортировка будет осуществлена по алфавиту.

```
SELECT last_name, first_name, birthday FROM users
WHERE age >= 18
--ORDER BY last_name
ORDER BY birthday DESC
```

ORDER BY **сортирует** запрошенные поля **из таблицы users по birthday**. Так как **birthday** имеет тип данных = дата, то сортировка будет осуществлена в хронологическом порядке, но атрибут **DESC** указывает на то, что сортировка осуществляется в обратном порядке.

```
SELECT * FROM products WHERE count > 0 ORDER BY price DESC LIMIT 5
```

Выберет все данные **из таблицы products где count > 0**, **отсортирует** в обратном порядке (**DESC**) по **price**, и выведет только **первые 5 строк**.

-- **SELECT TOP 5 * FROM products ORDER BY price DESC**
-- Инструкция LIMIT не поддерживается **MSSQL Express**, вместо нее нужно воспользоваться аналогом: TOP.

```
SELECT * FROM products WHERE count > 0 ORDER BY price DESC LIMIT 3, 5
```

Выберет все данные **из таблицы products где count > 0**, **отсортирует** в обратном порядке (**DESC**) по **price**, и выведет только **первые 5 строк пропуская первые 3**.

Выбор уникальных элементов столбца

Чтобы отобрать уникальные элементы некоторого столбца используется ключевое слово `DISTINCT`, которое размещается сразу после `SELECT`.

Пример

Выбрать различных авторов, книги которых хранятся в таблице `book`.

Запрос:

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

Результат:

```
+------------------+
| author           |
+------------------+
| Булгаков М.А.    |
| Достоевский Ф.М. |
| Есенин С.А.      |
+------------------+
```

Другой способ – использование оператора `GROUP BY`, который группирует данные при выборке, имеющие одинаковые значения в некотором столбце. Столбец, по которому осуществляется группировка, указывается после `GROUP BY` .

С помощью `GROUP BY` можно выбрать уникальные элементы столбца, по которому осуществляется группировка. Результат будет точно такой же как при использовании `DISTINCT`.

Запрос:

```
SELECT  author
FROM book
GROUP BY author;
```

групповые функции SUM и COUNT

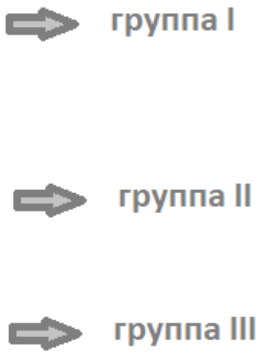
При группировке над элементами столбца, входящими в группу можно выполнить различные действия, например, просуммировать их или найти количество элементов в группе.

Подробно рассмотрим, как осуществляется группировка данных по некоторому столбцу и вычисления над группой на следующем примере:

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

1. В таблице `book` определяются строки, в которых в столбце `author` одинаковые значения:

book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	3
5	Игрок	Достоевский Ф.М.	480.50	10
6	Стихотворения и поэмы	Есенин С.А.	650.00	15

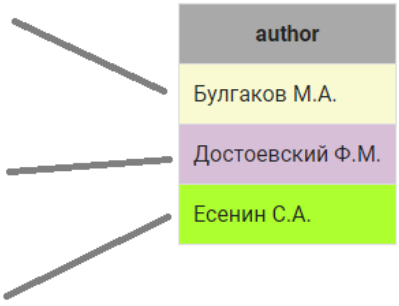


Получили 3 различные группы:

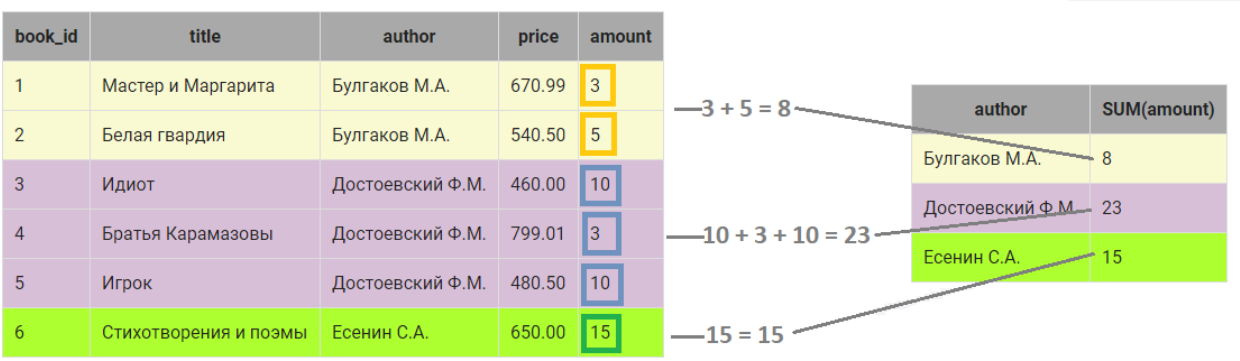
- **группа I** объединяет две записи, у которых в столбце `author` значение Булгаков М.А.;
- **группа II** объединяет три записи, у которых в столбце `author` значение Достоевский Ф.М.;
- **группа III** объединяет одну запись, у которой в столбце `author` значение Есенин С.А.

2. Вместо каждой группы в результирующий запрос включается одна запись. Запись как минимум включает значение столбца, по которому осуществляется группировка (в нашем случае это `author`):

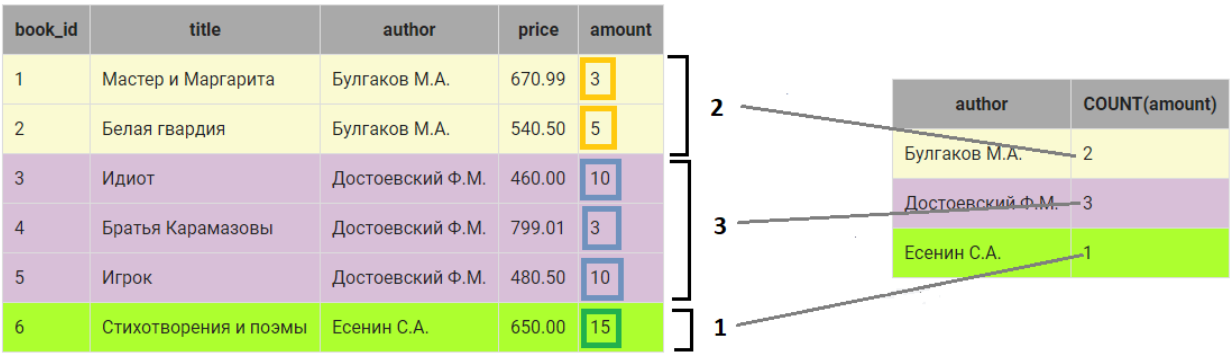
book_id	title	author	price	amount
1	Мастер и Маргарита	Булгаков М.А.	670.99	3
2	Белая гвардия	Булгаков М.А.	540.50	5
3	Идиот	Достоевский Ф.М.	460.00	10
4	Братья Карамазовы	Достоевский Ф.М.	799.01	3
5	Игрок	Достоевский Ф.М.	480.50	10
6	Стихотворения и поэмы	Есенин С.А.	650.00	15



3. Дальше можно выполнить вычисления над элементами КАЖДОЙ группы в отдельности, например, посчитать общее количество экземпляров книг каждого автора. Для этого используется групповая функция `SUM()`, а в скобках указывается столбец, по которому нужно выполнить суммирование (в нашем случае `amount`):



4. Также можно посчитать, сколько записей относится к группе. Для этого используется функция `COUNT()`, в скобках можно указать ЛЮБОЙ столбец из группы, если группа не содержит пустых значений (ниже приведен пример, в котором показано, как работает `COUNT()`, если в группе есть пустые значения):



Пример

Посчитать, сколько экземпляров книг каждого автора хранится на складе.

Запрос:

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

Результат:

```
+------------------+-------------+
| author           | SUM(amount) |
+------------------+-------------+
| Булгаков М.А.    | 8           |
| Достоевский Ф.М. | 23          |
| Есенин С.А.      | 15          |
+------------------+-------------+
```

Выборка данных, групповые функции MIN, MAX и AVG

К групповым функциям SQL относятся: `MIN()`, `MAX()` и `AVG()`, которые вычисляют минимальное, максимальное и среднее значение элементов столбца, относящихся к группе.

Пример

Вывести минимальную цену книги каждого автора

Запрос:

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

Результат:

```
+------------------+-----------+
| author           | min_price |
+------------------+-----------+
| Булгаков М.А.    | 540.50    |
| Достоевский Ф.М. | 460.00    |
| Есенин С.А.      | 650.00    |
+------------------+-----------+
```

Выборка данных по условию, групповые функции

В запросы с групповыми функциями можно включать условие отбора строк, которое в обычных запросах записывается после `WHERE`. В запросах с групповыми функциями
вместо `WHERE` используется ключевое слово `HAVING` , которое размещается после оператора `GROUP BY`.

Пример

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

Запрос:

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

Результат:

```
+------------------+------------------+-------------------+
| author           | Минимальная_цена | Максимальная_цена |
+------------------+------------------+-------------------+
| Достоевский Ф.М. | 460.00           | 799.01            |
| Есенин С.А.      | 650.00           | 650.00            |
+------------------+------------------+-------------------+
```

`WHERE` и `HAVING` могут использоваться в одном запросе. При этом необходимо учитывать **порядок выполнения SQL запроса на выборку на СЕРВЕРЕ**:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

Пример

Вывести максимальную и минимальную цену книг каждого автора, кроме Есенина, количество экземпляров книг которого больше 10.

```
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;
```

Результат:

```
+------------------+------------------+-------------------+
| author           | Минимальная_цена | Максимальная_цена |
+------------------+------------------+-------------------+
| Достоевский Ф.М. | 460.00           | 799.01            |
+------------------+------------------+-------------------+
```

Другим способом решения примера является запрос:

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

Не смотря на то что результат будет одинаковым, так делать не рекомендуется. «Потому что как написано - запрос сначала выбирает всех авторов, потом выводит данные, рассчитывая минимальное и максимальное значение цены для каждого, и только после всего убирает Есенина. Можно убрать Есенина в данном случае раньше и не использовать ресурсы базы для расчета его минимального и максимального значения, как это сделано в первом варианте. На небольшой базе быстродействия не ощутить, но если выполнять такое на продуктивной, то второй вариант значительно проигрывает

Вложенные запросы

SQL позволяет создавать вложенные запросы. Вложенный запрос (подзапрос, внутренний запрос) – это запрос внутри другого запроса SQL.

Вложенный запрос используется для выборки данных, которые будут использоваться в условии отбора записей основного запроса. Его применяют для:

- сравнения выражения с результатом вложенного запроса;
- определения того, включено ли выражение в результаты вложенного запроса;
- проверки того, выбирает ли запрос определенные строки.

Вложенный запрос имеет следующие компоненты:

- ключевое слово `SELECT` после которого указываются имена столбцов или выражения (чаще всего список содержит один элемент) ;
- ключевое слово `FROM` и имя таблицы, из которой выбираются данные;
- необязательное предложение `WHERE`;
- необязательное предложение `GROUP BY:`
- необязательное предложение `HAVING`.

Вложенные запросы могут включаться в WHERE или HAVING так (в квадратных скобках указаны необязательные элементы, через | – один из элементов):

- `WHERE | HAVING выражение` *`оператор_сравнения`* `(вложенный запрос)`;
- `WHERE | HAVING выражение, включающее вложенный запрос ;`
- `WHERE | HAVING выражение [NOT] IN (вложенный запрос)`;
- `WHERE | HAVING выражение` *`оператор_сравнения`* `ANY | ALL (вложенный запрос)`.

Также вложенные запросы могут вставляться в основной запрос после ключевого слова `SELECT`.

Пример

Вывести информацию о самых дешевых книгах, хранящихся на складе.

Для реализации этого запроса нам необходимо получить минимальную цену из столбца `price` таблицы `book`, а затем вывести информацию о тех книгах, цена которых равна минимальной. Первая часть – поиск минимума – реализуется вложенным запросом.

Запрос:

```
SELECT title, author, price, amount
FROM book
WHERE price = (
         SELECT MIN(price)
         FROM book
      );
```

Результат:

```
+-------+------------------+--------+--------+
| title | author           | price  | amount |
+-------+------------------+--------+--------+
| Идиот | Достоевский Ф.М. | 460.00 | 10     |
+-------+------------------+--------+--------+
```

Вложенный запрос определяет минимальную цену книг во всей таблице (это 460.00), а затем в основном запросе для каждой записи проверяется, равна ли цена минимальному значению, если равна, информация о книге включается в результирующую таблицу запроса.

Пример

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;
```

Пример
Вывести информацию (автора, название и цену) о тех книгах, цены которых превышают минимальную цену книги на складе не более чем на 150 рублей в отсортированном по возрастанию цены виде.

```
SELECT author, title, price
FROM book
WHERE ABS(price - (SELECT MIN(price) FROM book )) <= 150
ORDER BY price
```

Вложенный запрос, оператор IN

Вложенный запрос может возвращать несколько значений одного столбца. Тогда его можно использовать в разделе `WHERE` совместно с оператором `IN`.

```
WHERE имя_столбца IN (вложенный запрос, возвращающий один столбец)
```

Оператор `IN` определяет, совпадает ли значение столбца с одним из значений, содержащихся во вложенном запросе. При этом логическое выражение после `WHERE` получает значение истина. Оператор `NOT IN` выполняет обратное действие – выражение истинно, если значение столбца не содержится во вложенном запросе.

Пример

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

Запрос:

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

Результат:

```
+----------------------+------------------+--------+--------+
| title                | author           | amount | price  |
+----------------------+------------------+--------+--------+
| Идиот                | Достоевский Ф.М. | 10     | 460.00 |
| Братья Карамазовы    | Достоевский Ф.М. | 3      | 799.01 |
| Игрок                | Достоевский Ф.М. | 10     | 480.50 |
| Стихотворения и поэмы | Есенин С.А.     | 15     | 650.00 |
+----------------------+------------------+--------+--------+
```

Вложенный запрос отбирает двух авторов (Достоевского и Есенина). А в основном запросе для каждой записи таблицы `book` проверяется, входит ли автор книги в отобранный список, если входит - информация о книге включается в запрос.

Вложенный запрос, операторы ANY и ALL

Вложенный запрос, возвращающий несколько значений одного столбца, можно использовать для отбора записей с помощью операторов `ANY` и `ALL` совместно с операциями отношения (=, <>, <=, >=, <, >).

Операторы `ANY` и `ALL` используются в SQL для сравнения некоторого значения с результирующим набором вложенного запроса, состоящим из одного столбца. При этом тип данных столбца, возвращаемого вложенным запросом, должен совпадать с типом данных столбца (или выражения), с которым происходит сравнение.

При использовании оператора `ANY` в результирующую таблицу будут включены все записи, для которых выражение со знаком отношения верно хотя бы для одного элемента результирующего запроса. Как работает оператор `ANY`:

- `amount > ANY (10, 12)` эквивалентно `amount > 10`
- `amount < ANY (10, 12)` эквивалентно `amount < 12`
- `amount = ANY (10, 12)` эквивалентно `(amount = 10) OR (amount = 12)`, а также `amount IN (10,12)`
- `amount <> ANY (10, 12)` вернет все записи с любым значением `amount,` включая 10 и 12

При использовании оператора `ALL` в результирующую таблицу будут включены все записи, для которых выражение со знаком отношения верно для всех элементов результирующего запроса. Как работает оператор `ALL`:

- `amount > ALL (10, 12)` эквивалентно `amount > 12`
- `amount < ALL (10, 12)` эквивалентно `amount < 10`
- `amount = ALL (10, 12)` не вернет ни одной записи, так как эквивалентно `(amount = 10) AND (amount = 12)`
- `amount <> ALL (10, 12)` вернет все записи кроме тех, в которых `amount равно 10 или 12`

Важно! Операторы `ALL` и `ANY` можно использовать т**олько с вложенными запросами**. В примерах выше (10, 12) приводится как результат вложенного запроса просто для того, чтобы показать как эти операторы работают. В запросах так записывать нельзя.

Пример

Вывести информацию о тех книгах, количество которых меньше самого маленького среднего количества книг каждого автора.

Запрос:

```
SELECT title, author, amount, price
FROM book
WHERE amount < ALL (
        SELECT AVG(amount)
        FROM book
        GROUP BY author
      );
```

Результат:

```
+--------------------+------------------+--------+--------+
| title              | author           | amount | price  |
+--------------------+------------------+--------+--------+
| Мастер и Маргарита | Булгаков М.А.    | 3      | 670.99 |
| Братья Карамазовы  | Достоевский Ф.М. | 3      | 799.01 |
+--------------------+------------------+--------+--------+
```

Пояснение

1. Вложенный запрос

```
SELECT AVG(amount)
        FROM book
        GROUP BY author
```

отбирает следующие записи:

```
+-------------+
| AVG(amount) |
+-------------+
| 4.0000      |
| 7.6667      |
| 15.0000     |
+-------------+
```

2. Условие отбора в основном запросе

```
amount < ALL (
        SELECT AVG(amount)
        FROM book
        GROUP BY author
      )
```

можно переписать (если заменить вложенный запрос списком отобранных значений):

```
amount < ALL ( 4.0000, 7.6667, 15.0000)
```

что в соответствии с определением `ALL`, это значит, что подходят все `amount` меньшие **4.000**.

Таким образом, наш запрос отобрал все книги книги **Мастер и Маргарита** и **Братья Карамазовы**, количество которых равно 3.

Вложенный запрос после SELECT

Вложенный запрос может располагаться после ключевого слова `SELECT`
. В этом случае результат выполнения запроса выводится в отдельном столбце результирующей таблицы. При этом результатом запроса может быть только одно значение, тогда оно будет повторяться во всех строках. Также вложенный запрос может использоваться в выражениях.

Запрос:

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

Результат:

```
+----------------------+------------------+--------+--------------------+
| title                | author           | amount | Среднее_количество |
+----------------------+------------------+--------+--------------------+
| Мастер и Маргарита   | Булгаков М.А.    | 3      | 7.6667             |
| Братья Карамазовы    | Достоевский Ф.М. | 3      | 7.6667             |
| Стихотворения и поэмы | Есенин С.А.     | 15     | 7.6667             |
+----------------------+------------------+--------+--------------------+
```

Во вложенном запросе вычисляется среднее количество экземпляров книг на складе. Этот запрос используется и в условии отбора, и для создания столбца **Среднее_количество** в результирующей таблице запроса. Значения столбца одинаковы во всех строках, поскольку вложенный запрос возвращает одно значение.