**Основные команды SQL**

Основные команды SQL не ограничиваются стандартными CREATE, UPDATE и DELETE. Данная статья будет полезна тем, кто хочет [освежить свои знания по SQL перед собеседованием на работу](https://tproger.ru/articles/sql-interview-questions/).

Обратите внимание, что в примерах используется MySQL, поэтому в конце каждого оператора ставится точка с запятой.

Примечание Вы читаете улучшенную версию некогда выпущенной нами статьи.

**Настройка базы данных**

Перед началом создайте БД с тестовыми данными. Для работы вам понадобится скачать два файла: [DLL.sql](https://drive.google.com/file/d/0B_oq3-doZhC-ME1lUlR3a3pYRU0/view?resourcekey=0-flfPMq0i6E6-i5BX27Se4g" \t "_blank) и [InsertStatements.sql](https://drive.google.com/file/d/0B_oq3-doZhC-TV9ud1JubkVDaXM/view?resourcekey=0-CToHONw06G5QQ6DuRf68Ng" \t "_blank). После установите MySQL, откройте терминал и войдите в консоль MySQL с помощью команды:

mysql -u root -p

Затем введите пароль и выполните следующую команду. Назовём базу данных «university»:

CREATE DATABASE university;

USE university;

SOURCE <path\_of\_DLL.sql\_file>;

SOURCE <path\_of\_InsertStatements.sql\_file>;

**SHOW DATABASES**

SQL-команда, которая отвечает за просмотр доступных баз данных.

**CREATE DATABASE**

Команда для создания новой базы данных.

**USE**

С помощью этой SQL-команды USE <database\_name> выбирается база данных, необходимая для дальнейшей работы с ней.

**SOURCE**

А SOURCE <file.sql> позволит выполнить сразу несколько SQL-команд, содержащихся в файле с расширением .sql.

**DROP DATABASE**

Стандартная SQL-команда для удаления целой базы данных.

**SHOW TABLES**

С помощью этой несложной команды можно увидеть все таблицы, которые доступны в базе данных.

**CREATE TABLE**

SQL-команда для создания новой таблицы:

CREATE TABLE <table\_name1> (

<col\_name1><col\_type1>,

<col\_name2><col\_type2>,

<col\_name3><col\_type3>

PRIMARY KEY(<col\_name1>),

FOREIGN KEY(<col\_name2>) REFERENCES <table\_name2>(<col\_name2>));

**Ограничения целостности при использовании CREATE TABLE**

Может понадобиться создать ограничения для определённых столбцов в таблице. При создании таблицы можно задать следующие ограничения:

* ячейка таблицы не может иметь значение NULL;
* первичный ключ — PRIMARY KEY(col\_name1, col\_name2, …);
* внешний ключ — FOREIGN KEY(col\_namex1, …, col\_namexn) REFERENCES table\_name(col\_namex1, …, col\_namexn).

Можно задать больше одного первичного ключа. В этом случае получится составной первичный ключ.

**Пример**

Создайте таблицу «instructor»:

CREATE TABLE instructor (

ID CHAR(5),

name VARCHAR(20) NOT NULL,

dept\_name VARCHAR(20),

salary NUMERIC(8,2),

PRIMARY KEY (ID),

FOREIGN KEY (dept\_name) REFERENCES department(dept\_name));

**DESCRIBE**

С помощью DESCRIBE <table\_name> можно просмотреть различные сведения (тип значений, является ключом или нет) о столбцах таблицы.

**INSERT**

Команда INSERT INTO <table\_name> в SQL отвечает за добавление данных в таблицу:

INSERT INTO <table\_name> (<col\_name1>, <col\_name2>, <col\_name3>, …)

VALUES (<value1>, <value2>, <value3>, …);

При добавлении данных в каждый столбец таблицы не требуется указывать названия столбцов.

INSERT INTO <table\_name>

VALUES (<value1>, <value2>, <value3>, …);

**UPDATE**

SQL-команда для обновления данных таблицы:

UPDATE <table\_name>

SET <col\_name1> = <value1>, <col\_name2> = <value2>, ...

WHERE <condition>;

**DELETE**

SQL-команда DELETE FROM <table\_name> используется для удаления данных из таблицы.

**DROP TABLE**

А так можно удалить всю таблицу целиком.

**SELECT**

Далее мы рассмотрим основные команды SQL, которые позволяют работать непосредственно с данными. К одной из таких SQL-команд относится SELECT для получения данных из выбранной таблицы:

SELECT <col\_name1>, <col\_name2>, …

FROM <table\_name>;

Следующей командой можно вывести все данные из таблицы:

SELECT \* FROM <table\_name>;

**SELECT DISTINCT**

В столбцах таблицы могут содержаться повторяющиеся данные. Используйте SELECT DISTINCT для получения только неповторяющихся данных.

SELECT DISTINCT <col\_name1>, <col\_name2>, …

FROM <table\_name>;

**WHERE**

Можно использовать ключевое слово WHERE в SELECT для указания условий в запросе:

SELECT <col\_name1>, <col\_name2>, …

FROM <table\_name>

WHERE <condition>;

В запросе можно задавать следующие условия:

* сравнение текста;
* сравнение численных значений;
* логические операции AND (и), OR (или) и NOT (отрицание).

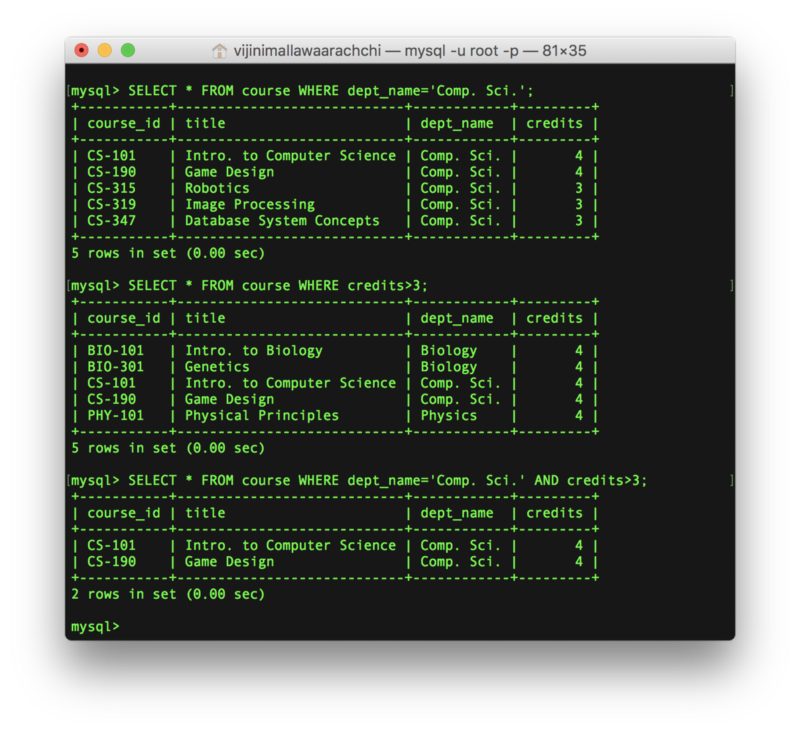
**Пример**

Попробуйте выполнить следующие команды. Обратите внимание на условия, заданные в WHERE:

SELECT \* FROM course WHERE dept\_name=’Comp. Sci.’;

SELECT \* FROM course WHERE credits>3;

SELECT \* FROM course WHERE dept\_name='Comp. Sci.' AND credits>3;

**GROUP BY**

Оператор GROUP BY часто используется с агрегатными функциями, такими как COUNT, MAX, MIN, SUM и AVG, для группировки выходных значений.

SELECT <col\_name1>, <col\_name2>, …

FROM <table\_name>

GROUP BY <col\_namex>;

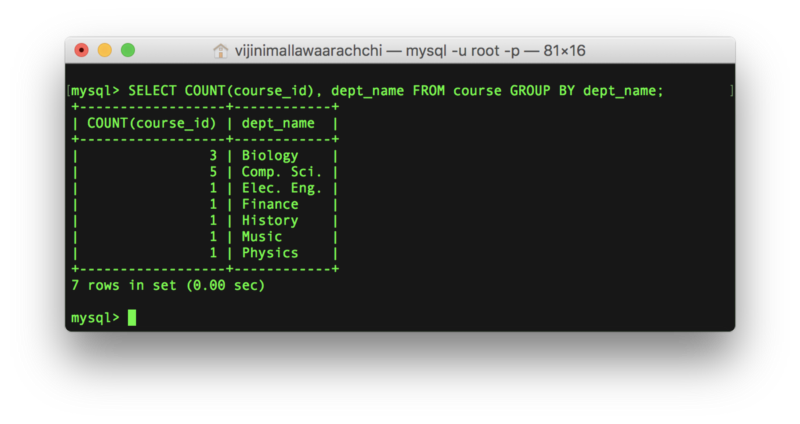
**Пример**

Выведем количество курсов для каждого факультета:

SELECT COUNT(course\_id), dept\_name

FROM course

GROUP BY dept\_name;



**HAVING**

Ключевое слово HAVING было добавлено в SQL по той причине, что WHERE не может использоваться для работы с агрегатными функциями.

SELECT <col\_name1>, <col\_name2>, ...

FROM <table\_name>

GROUP BY <column\_namex>

HAVING <condition>

**Пример**

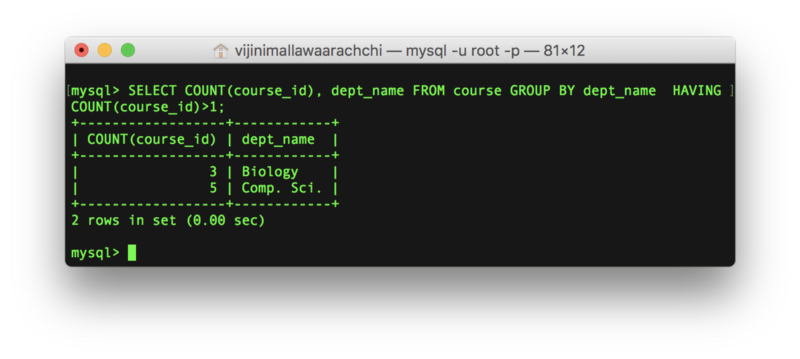
Выведем список факультетов, у которых более одного курса:

SELECT COUNT(course\_id), dept\_name

FROM course

GROUP BY dept\_name

HAVING COUNT(course\_id)>1;

**ORDER BY**

ORDER BY используется для сортировки результатов запроса по убыванию или возрастанию. ORDER BY отсортирует по возрастанию, если не будет указан способ сортировки ASC или DESC.

SELECT <col\_name1>, <col\_name2>, …

FROM <table\_name>

ORDER BY <col\_name1>, <col\_name2>, … ASC|DESC;

**Пример**

Выведем список курсов по возрастанию и убыванию количества кредитов:

SELECT \* FROM course ORDER BY credits;

SELECT \* FROM course ORDER BY credits DESC;

**BETWEEN**

BETWEEN используется для выбора значений данных из определённого промежутка. Могут быть использованы числовые и текстовые значения, а также даты.

SELECT <col\_name1>, <col\_name2>, …

FROM <table\_name>

WHERE <col\_namex> BETWEEN <value1> AND <value2>;

**Пример**

Выведем список инструкторов, чья зарплата больше 50 000, но меньше 100 000:

SELECT \* FROM instructor

WHERE salary BETWEEN 50000 AND 100000;

**LIKE**

Оператор LIKE используется в WHERE, чтобы задать шаблон поиска похожего значения.

Есть два свободных оператора, которые используются в LIKE:

* % (ни одного, один или несколько символов);
* \_ (один символ).

SELECT <col\_name1>, <col\_name2>, …

FROM <table\_name>

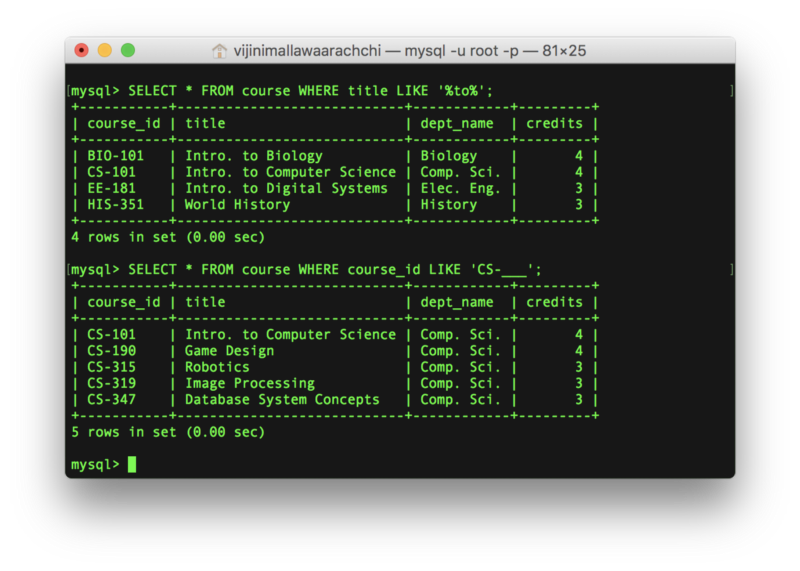
WHERE <col\_namex> LIKE <pattern>;

**Пример**

Выведем список курсов, в имени которых содержится «to», и список курсов, название которых начинается с «CS-»:

SELECT \* FROM course WHERE title LIKE ‘%to%’;

SELECT \* FROM course WHERE course\_id LIKE 'CS-\_\_\_';



**IN**

С помощью IN можно указать несколько значений для оператора WHERE:

SELECT <col\_name1>, <col\_name2>, …

FROM <table\_name>

WHERE <col\_namen> IN (<value1>, <value2>, …);

**Пример**Выведем список студентов с направлений Comp. Sci., Physics и Elec. Eng.:

SELECT \* FROM student

WHERE dept\_name IN (‘Comp. Sci.’, ‘Physics’, ‘Elec. Eng.’);

**JOIN**

JOIN используется для связи двух или более таблиц с помощью общих атрибутов внутри них. На изображении ниже показаны различные способы объединения в SQL. Обратите внимание на разницу между левым внешним объединением и правым внешним объединением:



SELECT <col\_name1>, <col\_name2>, …

FROM <table\_name1>

JOIN <table\_name2>

ON <table\_name1.col\_namex> = <table2.col\_namex>;

**Пример**

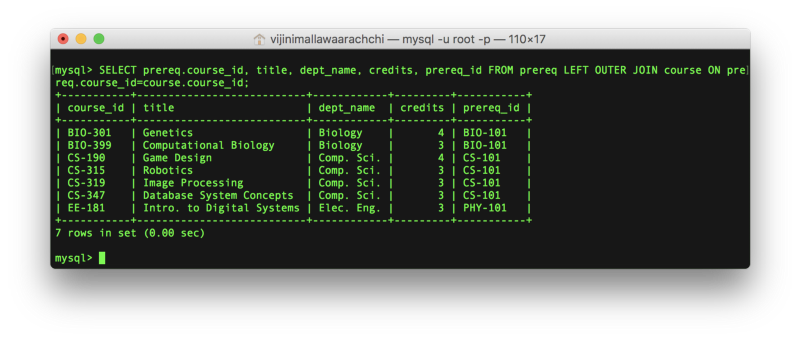
Выведем список всех обязательных курсов и детали о них:

SELECT prereq.course\_id, title, dept\_name, credits, prereq\_id

FROM prereq

LEFT OUTER JOIN course

ON prereq.course\_id=course.course\_id;

**VIEW**

VIEW — это виртуальная таблица SQL, созданная в результате выполнения выражения. Она содержит строки и столбцы и очень похожа на обычную SQL-таблицу. VIEW всегда показывает самую свежую информацию из базы данных.

**Создание**

CREATE VIEW <view\_name> AS

SELECT <col\_name1>, <col\_name2>, …

FROM <table\_name>

WHERE <condition>;

**Удаление**

DROP VIEW <view\_name>

**Агрегатные функции**

Это не совсем основные команды SQL, однако знать их тоже желательно. Агрегатные функции используются для получения совокупного результата, относящегося к рассматриваемым данным:

* COUNT(col\_name) — возвращает количество строк;
* SUM(col\_name) — возвращает сумму значений в данном столбце;
* AVG(col\_name) — возвращает среднее значение данного столбца;
* MIN(col\_name) — возвращает наименьшее значение данного столбца;
* MAX(col\_name) — возвращает наибольшее значение данного столбца.

**Вложенные подзапросы**

Вложенные подзапросы — это SQL-запросы, которые включают выражения SELECT, FROM и WHERE, вложенные в другой запрос.

**Пример**

Найдём курсы, которые преподавались осенью 2009 и весной 2010 годов:

SELECT DISTINCT course\_id

FROM section

WHERE semester = ‘Fall’ AND year= 2009 AND course\_id IN (

SELECT course\_id

FROM section

WHERE semester = ‘Spring’ AND year= 2010);