# SQL

## [SQL](https://www.codecademy.com/learn/learn-sql) : [Queries](https://www.codecademy.com/learn/learn-sql/modules/learn-sql-queries)

**ANDOperator**

The AND operator allows multiple conditions to be combined. Records must match both conditions that are joined by AND to be included in the result set. The example query will match any car that is blue and made after 2014.

SELECT model FROM cars WHERE color = 'blue'AND year >2014;

**ASClause**

Columns or tables in SQL can be *aliased* using the AS clause. This allows columns or tables to be specifically renamed in the returned result set. The given query will return a result set with the column for name renamed to movie\_title.

SELECT name AS'movie\_title'FROM movies;

**OROperator**

The OR operator allows multiple conditions to be combined. Records matching either condition joined by the OR are included in the result set. The given query will match customers whose state is either ca or ny.

SELECT name FROM customers WHERE state = "ca"OR state = "ny";

**%Wildcard**

The % wildcard can be used in a LIKE operator pattern to match zero or more unspecified character(s). The example query will match any movie that begins with The, followed by zero or more of any characters.

SELECT name FROM movies WHERE name LIKE'The%';

**SELECTStatement**

The SELECT \* statement returns all columns from the provided table in the result set. The given query will fetch all columns and records (rows) from the movies table.

SELECT \* FROM movies;

**\_Wildcard**

The \_ wildcard can be used in a LIKE operator pattern to match any single unspecified character. The given query will match any movie which begins with a single character, followed by ove.

SELECT name FROM movies WHERE name LIKE'\_ove';

**ORDER BYClause**

The ORDER BY clause can be used to sort the result set by a particular column either alphabetically or numerically. It can be ordered in ascending (default) or descending order with ASC/DESC. In the example, all the rows of the contacts table will be ordered by the birth\_date column in descending order.

SELECT \* FROM contacts ORDERBY birth\_date DESC;

**LIKEOperator**

The LIKE operator can be used inside of a WHERE clause to match a specified pattern. The given query will match any movie that begins with Star in its title.

SELECT name FROM movies WHERE name LIKE'Star%';

**DISTINCTClause**

Unique values of a column can be selected using a DISTINCT query. For a table contact\_details having five rows in which the city column contains Chicago, Madison, Boston, Madison, and Denver, the given query would return:

* Chicago
* Madison
* Boston
* Denver

SELECTDISTINCT city FROM contact\_details;

**BETWEENOperator**

The BETWEEN operator can be used to filter by a *range* of values. The range of values can be text, numbers or date data. The given query will match any movie made between the years 1980 and 1990, inclusive.

SELECT \* FROM movies WHERE year BETWEEN1980AND1990;

**LIMITClause**

The LIMIT clause is used to narrow, or *limit*, a result set to the specified number of rows. The given query will limit the result set to 5 rows.

SELECT \* FROM movies LIMIT5;

**NULLValues**

Column values in SQL records can be NULL, or have no value. These records can be matched (or not matched) using the IS NULL and IS NOT NULL operators in combination with the WHERE clause. The given query will match all addresses where the address has a value or is not NULL.

SELECT address FROM records WHERE address ISNOTNULL;

**WHEREClause**

The WHERE clause is used to filter records (rows) that match a certain condition. The given query will select all records where the pub\_year equals 2017.

SELECT title FROM library WHERE pub\_year = 2017;

## [SQL](https://www.codecademy.com/learn/learn-sql) : [Manipulation](https://www.codecademy.com/learn/learn-sql/modules/learn-sql-manipulation)

[Download as a PDF](https://codecademy-content.s3.amazonaws.com/cheatsheets/learn-sql/learn-sql-manipulation.pdf)

**Column Constraints**

Column constraints are the rules applied to the values of individual columns:

* PRIMARY KEY constraint can be used to uniquely identify the row.
* UNIQUE columns have a different value for every row.
* NOT NULL columns must have a value.
* DEFAULT assigns a default value for the column when no value is specified.

There can be only one PRIMARY KEY column per table and multiple UNIQUE columns.

CREATETABLE student ( id INTEGER PRIMARY KEY, name TEXT UNIQUE, grade INTEGERNOTNULL, age INTEGER DEFAULT 10);

**CREATE TABLEStatement**

The CREATE TABLE statement is used to create a new table in a database. It allows one to specify the name of the table and the name of each column in the table.

CREATETABLE table\_name ( column1 datatype, column2 datatype, column3 datatype );

**INSERTStatement**

The INSERT INTO statement is used to add a new record (row) to a table.

It has two forms as shown in the code block:

* Insert values based on the order of the columns in the table.
* Define the columns to insert values into.

-- Insert into columns in order:INSERTINTO table\_name VALUES(value1, value2, value3);-- Insert into columns by name:INSERTINTO table\_name (column1, column2, column3)VALUES(value1, value2, value3);

**ALTER TABLEStatement**

The ALTER TABLE statement is used to modify the columns of an existing table. When combined with the ADD COLUMN clause, it is used to add a new column to a table.

-- Syntax:ALTERTABLE table\_name ADD column\_name datatype;-- Example:ALTERTABLE employees ADD first\_name TEXT;

**DELETEStatement**

The DELETE statement is used to delete records (rows) in a table. This statement does not delete the whole table.

Inside, the WHERE clause specifies which record or records that should be deleted. If the WHERE clause is omitted, all records will be deleted.

DELETEFROM table\_name WHERE some\_column = some\_value;

**UPDATEStatement**

The UPDATE statement is used to edit records (rows) in a table. It usually includes a SET clause that indicates the column to edit and a WHERE clause for specifying which record(s) should be updated.

UPDATE table\_name SET column1 = value1, column2 = value2 WHERE some\_column = some\_value;

## [SQL](https://www.codecademy.com/learn/learn-sql) : [Aggregate Functions](https://www.codecademy.com/learn/learn-sql/modules/learn-sql-aggregate-functions)

[Download as a PDF](https://codecademy-content.s3.amazonaws.com/cheatsheets/learn-sql/learn-sql-aggregate-functions.pdf)

**Column References**

The GROUP BY and ORDER BY clauses can reference the selected columns by number in which they appear in the SELECT statement. The example query will count the number of movies per rating, and will:

* GROUP BY column 2 (rating)
* ORDER BY column 1 (total\_movies)

SELECTCOUNT(\*)AS'total\_movies', rating FROM movies GROUPBY2ORDERBY1;

**SUM()Aggregate Function**

The SUM() aggregate function takes the name of a column as an argument and returns the sum of all the value in that column.

SELECT SUM(salary)FROM salary\_disbursement;

**MAX()Aggregate Function**

The MAX() aggregate function in SQL takes the name of a column as an argument and returns the largest value in a column. The given query will return the largest value from the amount column.

SELECT MAX(amount)FROM transactions;

**COUNT()Aggregate Function**

The COUNT() aggregate function in SQL returns the total number of rows that match the specified criteria. For instance, to find the total number of employees who have less than 5 years of experience, the given query can be used.

**Note:** A column name of the table can also be used instead of \*. Unlike COUNT(\*), this variation COUNT(column) will not count NULL values in that column.

SELECTCOUNT(\*)FROM employees WHERE experience <5;

**GROUP BYClause**

The GROUP BY clause will group records in a result set by identical values in one or more columns. It is often used in combination with aggregate functions to query information of similar records. The GROUP BY clause can come after FROM or WHERE but must come before any ORDER BY or LIMIT clause.

The given query will count the number of movies per rating.

SELECT rating,COUNT(\*)FROM movies GROUPBY rating;

**MIN()Aggregate Function**

The MIN() aggregate function in SQL returns the smallest value in a column. For instance, to find the smallest value of the amount column from the table named transactions, the given query can be used.

SELECT MIN(amount)FROM transactions;

**AVG()Aggregate Function**

The AVG() aggregate function returns the average value in a column. For instance, to find the average salary for the employees who have less than 5 years of experience, the given query can be used.

SELECT AVG(salary)FROM employees WHERE experience <5;

**HAVINGClause**

The HAVING clause is used to further filter the result set groups provided by the GROUP BY clause. HAVING is often used with aggregate functions to filter the result set groups based on an aggregate property. The given query will select only the records (rows) from only years where more than 5 movies were released per year.

SELECTyear,COUNT(\*)FROM movies GROUPBYyearHAVINGCOUNT(\*)>5;

**ROUND()Function**

The ROUND() function will round a number value to a specified number of places. It takes two arguments: a number, and a number of decimal places. It can be combined with other aggregate functions, as shown in the given query. This query will calculate the average rating of movies from 2015, rounding to 2 decimal places.

SELECTyear, ROUND(AVG(rating),2)FROM movies WHEREyear=2015;

## [SQL](https://www.codecademy.com/learn/learn-sql) : [Multiple Tables](https://www.codecademy.com/learn/learn-sql/modules/learn-sql-multiple-tables)

[Download as a PDF](https://codecademy-content.s3.amazonaws.com/cheatsheets/learn-sql/learn-sql-multiple-tables.pdf)

**Outer Join**

An outer join will combine rows from different tables even if the join condition is not met. In a LEFT JOIN, every row in the left table is returned in the result set, and if the join condition is not met, then NULL values are used to fill in the columns from the right table.

SELECT column\_name(s)FROM table1 LEFT JOIN table2 ON table1.column\_name= table2.column\_name;

**WITHClause**

The WITH clause stores the result of a query in a temporary table (temporary\_movies) using an alias.

Multiple temporary tables can be defined with one instance of the WITH keyword.

WITH temporary\_movies AS(SELECT\*FROM movies )SELECT\*FROM temporary\_movies WHEREyearBETWEEN2000AND2020;

**UNIONClause**

The UNION clause is used to combine results that appear from multiple SELECT statements and filter duplicates.

For example, given a first\_names table with a column name containing rows of data “James” and “Hermione”, and a last\_names table with a column name containing rows of data “James”, “Hermione” and “Cassidy”, the result of this query would contain three names: “Cassidy”, “James”, and “Hermione”.

SELECT name FROM first\_names UNIONSELECT name FROM last\_names

**CROSS JOINClause**

The CROSS JOIN clause is used to combine each row from one table with each row from another in the result set. This JOIN is helpful for creating all possible combinations for the records (rows) in two tables.

The given query will select the shirt\_color and pants\_color columns from the result set, which will contain all combinations of combining the rows in the shirts and pants tables. If there are 3 different shirt colors in the shirts table and 5 different pants colors in the pants table then the result set will contain 3 x 5 = 15 rows.

SELECT shirts.shirt\_color, pants.pants\_colorFROM shirts CROSS JOIN pants;

**Inner Join**

The JOIN clause allows for the return of results from more than one table by joining them together with other results based on common column values specified using an ON clause. INNER JOIN is the default JOIN and it will only return results matching the condition specified by ON.

SELECT\*FROM books JOIN authors ON books.author\_id= authors.id;

## COMMANDS

### ALTER TABLE

ALTERTABLE table\_name

ADD column\_name datatype;

ALTER TABLE lets you add columns to a table in a database.

### AND

SELECT column\_name(s)

FROM table\_name

WHERE column\_1 = value\_1

AND column\_2 = value\_2;

AND is an operator that combines two conditions. Both conditions must be true for the row to be included in the result set.

### AS

SELECT column\_name AS'Alias'

FROM table\_name;

AS is a keyword in SQL that allows you to rename a column or table using an alias.

### AVG()

SELECT AVG(column\_name)

FROM table\_name;

AVG() is an aggregate function that returns the average value for a numeric column.

### BETWEEN

SELECT column\_name(s)

FROM table\_name

WHERE column\_name BETWEEN value\_1 AND value\_2;

The BETWEEN operator is used to filter the result set within a certain range. The values can be numbers, text or dates.

### CASE

SELECT column\_name,

CASE

WHEN condition THEN 'Result\_1'

WHEN condition THEN 'Result\_2'

ELSE 'Result\_3'

END

FROM table\_name;

CASE statements are used to create different outputs (usually in the SELECT statement). It is SQL’s way of handling if-then logic.

### COUNT()

SELECTCOUNT(column\_name)

FROM table\_name;

COUNT() is a function that takes the name of a column as an argument and counts the number of rows where the column is not NULL.

### CREATE TABLE

CREATETABLE table\_name (

column\_1 datatype,

column\_2 datatype,

column\_3 datatype

);

CREATE TABLE creates a new table in the database. It allows you to specify the name of the table and the name of each column in the table.

### DELETE

DELETEFROM table\_name

WHERE some\_column = some\_value;

DELETE statements are used to remove rows from a table.

### GROUP BY

SELECT column\_name,COUNT(\*)

FROM table\_name

GROUPBY column\_name;

GROUP BY is a clause in SQL that is only used with aggregate functions. It is used in collaboration with the SELECT statement to arrange identical data into groups.

### HAVING

SELECT column\_name,COUNT(\*)

FROM table\_name

GROUPBY column\_name

HAVINGCOUNT(\*)> value;

HAVING was added to SQL because the WHERE keyword could not be used with **aggregate functions**.

### INNER JOIN

SELECT column\_name(s)

FROM table\_1

JOIN table\_2

ON table\_1.column\_name= table\_2.column\_name;

An inner join will combine rows from different tables if the join condition is true.

### INSERT

INSERTINTO**table\_name (column\_1, column\_2, column\_3)**

**VALUES(value\_1,'value\_2', value\_3);**

INSERT statements are used to add a new row to a table.

### IS NULL / IS NOT NULL

SELECT column\_name(s)

FROM table\_name

WHERE column\_name ISNULL;

IS NULL and IS NOT NULL are operators used with the WHERE clause to test for empty values.

### LIKE

SELECT column\_name(s)

FROM table\_name

WHERE column\_name LIKE pattern;

LIKE is a special operator used with the WHERE clause to search for a specific pattern in a column.

### LIMIT

SELECT column\_name(s)

FROM table\_name

LIMIT number;

LIMIT is a clause that lets you specify the maximum number of rows the result set will have.

### MAX()

SELECT MAX(column\_name)

FROM table\_name;

MAX() is a function that takes the name of a column as an argument and returns the largest value in that column.

### MIN()

SELECT MIN(column\_name)

FROM table\_name;

MIN() is a function that takes the name of a column as an argument and returns the smallest value in that column.

### OR

SELECT column\_name

FROM table\_name

WHERE column\_name = value\_1

OR column\_name = value\_2;

OR is an operator that filters the result set to only include rows where either condition is true.

### ORDER BY

SELECT column\_name

FROM table\_name

ORDERBY column\_name ASC | DESC;

ORDER BY is a clause that indicates you want to sort the result set by a particular column either alphabetically or numerically.

### OUTER JOIN

SELECT column\_name(s)

FROM table\_1

LEFT JOIN table\_2

ON table\_1.column\_name= table\_2.column\_name;

An outer join will combine rows from different tables even if the join condition is not met. Every row in the left table is returned in the result set, and if the join condition is not met, then NULL values are used to fill in the columns from the right table.

### ROUND()

SELECT ROUND(column\_name,integer)

FROM table\_name;

ROUND() is a function that takes a column name and an integer as arguments. It rounds the values in the column to the number of decimal places specified by the integer.

### SELECT

SELECT column\_name

FROM table\_name;

SELECT statements are used to fetch data from a database. Every query will begin with SELECT.

### SELECT DISTINCT

SELECTDISTINCT column\_name

FROM table\_name;

SELECT DISTINCT specifies that the statement is going to be a query that returns unique values in the specified column(s).

### SUM

SELECT SUM(column\_name)

FROM table\_name;

SUM() is a function that takes the name of a column as an argument and returns the sum of all the values in that column.

### UPDATE

UPDATE table\_name

SET some\_column = some\_value

WHERE some\_column = some\_value;

UPDATE statements allow you to edit rows in a table.

### WHERE

SELECT column\_name(s)

FROM table\_name

WHERE column\_name operator value;

WHERE is a clause that indicates you want to filter the result set to include only rows where the following condition is true.

### WITH

WITH temporary\_name AS(

SELECT\*

FROM table\_name)

SELECT\*

FROM temporary\_name

WHERE column\_name operator value;

WITH clause lets you store the result of a query in a temporary table using an alias. You can also define multiple temporary tables using a comma and with one instance of the WITH keyword.

The WITH clause is also known as common table expression (CTE) and subquery factoring.

## Top 50 SQL Interview Questions & Answers

**1. What is DBMS?**

A Database Management System (DBMS) is a program that controls creation, maintenance and use of a database. DBMS can be termed as File Manager that manages data in a database rather than saving it in file systems.

**2. What is RDBMS?**

RDBMS stands for Relational Database Management System. RDBMS store the data into the collection of tables, which is related by common fields between the columns of the table. It also provides relational operators to manipulate the data stored into the tables.

**Example: SQL Server.**

**3. What is SQL?**

SQL stands for Structured Query Language , and it is used to communicate with the Database. This is a standard language used to perform tasks such as retrieval, updation, insertion and deletion of data from a database.

Standard SQL Commands are Select.

**4. What is a Database?**

Database is nothing but an organized form of data for easy access, storing, retrieval and managing of data. This is also known as structured form of data which can be accessed in many ways.

Example: School Management Database, Bank Management Database.

**5. What are tables and Fields?**

A table is a set of data that are organized in a model with Columns and Rows. Columns can be categorized as vertical, and Rows are horizontal. A table has specified number of column called fields but can have any number of rows which is called record.

Example:.

Table: Employee.

Field: Emp ID, Emp Name, Date of Birth.

Data: 201456, David, 11/15/1960.

**6. What is a primary key?**

A primary key is a combination of fields which uniquely specify a row. This is a special kind of unique key, and it has implicit NOT NULL constraint. It means, Primary key values cannot be NULL.

**7. What is a unique key?**

A Unique key constraint uniquely identified each record in the database. This provides uniqueness for the column or set of columns.

A Primary key constraint has automatic unique constraint defined on it. But not, in the case of Unique Key.

There can be many unique constraint defined per table, but only one Primary key constraint defined per table.

**8. What is a foreign key?**

A foreign key is one table which can be related to the primary key of another table. Relationship needs to be created between two tables by referencing foreign key with the primary key of another table.

**9. What is a join?**

This is a keyword used to query data from more tables based on the relationship between the fields of the tables. Keys play a major role when JOINs are used.

**10. What are the types of join and explain each?**

There are various types of join which can be used to retrieve data and it depends on the relationship between tables.

* **Inner Join.**

Inner join return rows when there is at least one match of rows between the tables.

* **Right Join.**

Right join return rows which are common between the tables and all rows of Right hand side table. Simply, it returns all the rows from the right hand side table even though there are no matches in the left hand side table.

* **Left Join.**

Left join return rows which are common between the tables and all rows of Left hand side table. Simply, it returns all the rows from Left hand side table even though there are no matches in the Right hand side table.

* **Full Join.**

Full join return rows when there are matching rows in any one of the tables. This means, it returns all the rows from the left hand side table and all the rows from the right hand side table.

**11. What is normalization?**

Normalization is the process of minimizing redundancy and dependency by organizing fields and table of a database. The main aim of Normalization is to add, delete or modify field that can be made in a single table.

**12. What is Denormalization.**

DeNormalization is a technique used to access the data from higher to lower normal forms of database. It is also process of introducing redundancy into a table by incorporating data from the related tables.

**13. What are all the different normalizations?**

The normal forms can be divided into 5 forms, and they are explained below -.

* **First Normal Form (1NF):.**

This should remove all the duplicate columns from the table. Creation of tables for the related data and identification of unique columns.

* **Second Normal Form (2NF):.**

Meeting all requirements of the first normal form. Placing the subsets of data in separate tables and Creation of relationships between the tables using primary keys.

* **Third Normal Form (3NF):.**

This should meet all requirements of 2NF. Removing the columns which are not dependent on primary key constraints.

* **Fourth Normal Form (4NF):.**

Meeting all the requirements of third normal form and it should not have multi- valued dependencies.

**14. What is a View?**

A view is a virtual table which consists of a subset of data contained in a table. Views are not virtually present, and it takes less space to store. View can have data of one or more tables combined, and it is depending on the relationship.

**15. What is an Index?**

An index is performance tuning method of allowing faster retrieval of records from the table. An index creates an entry for each value and it will be faster to retrieve data.

**16. What are all the different types of indexes?**

There are three types of indexes -.

* **Unique Index.**

This indexing does not allow the field to have duplicate values if the column is unique indexed. Unique index can be applied automatically when primary key is defined.

* **Clustered Index.**

This type of index reorders the physical order of the table and search based on the key values. Each table can have only one clustered index.

* **NonClustered Index.**

NonClustered Index does not alter the physical order of the table and maintains logical order of data. Each table can have 999 nonclustered indexes.

**17. What is a Cursor?**

A database Cursor is a control which enables traversal over the rows or records in the table. This can be viewed as a pointer to one row in a set of rows. Cursor is very much useful for traversing such as retrieval, addition and removal of database records.

**18. What is a relationship and what are they?**

Database Relationship is defined as the connection between the tables in a database. There are various data basing relationships, and they are as follows:.

* One to One Relationship.
* One to Many Relationship.
* Many to One Relationship.
* Self-Referencing Relationship.

**19. What is a query?**

A DB query is a code written in order to get the information back from the database. Query can be designed in such a way that it matched with our expectation of the result set. Simply, a question to the Database.

**20. What is subquery?**

A subquery is a query within another query. The outer query is called as main query, and inner query is called subquery. SubQuery is always executed first, and the result of subquery is passed on to the main query.

**21. What are the types of subquery?**

There are two types of subquery – Correlated and Non-Correlated.

A correlated subquery cannot be considered as independent query, but it can refer the column in a table listed in the FROM the list of the main query.

A Non-Correlated sub query can be considered as independent query and the output of subquery are substituted in the main query.

**22. What is a stored procedure?**

Stored Procedure is a function consists of many SQL statement to access the database system. Several SQL statements are consolidated into a stored procedure and execute them whenever and wherever required.

**23. What is a trigger?**

A DB trigger is a code or programs that automatically execute with response to some event on a table or view in a database. Mainly, trigger helps to maintain the integrity of the database.

Example: When a new student is added to the student database, new records should be created in the related tables like Exam, Score and Attendance tables.

**24. What is the difference between DELETE and TRUNCATE commands?**

DELETE command is used to remove rows from the table, and WHERE clause can be used for conditional set of parameters. Commit and Rollback can be performed after delete statement.

TRUNCATE removes all rows from the table. Truncate operation cannot be rolled back.

**25. What are local and global variables and their differences?**

Local variables are the variables which can be used or exist inside the function. They are not known to the other functions and those variables cannot be referred or used. Variables can be created whenever that function is called.

Global variables are the variables which can be used or exist throughout the program. Same variable declared in global cannot be used in functions. Global variables cannot be created whenever that function is called.

**26. What is a constraint?**

Constraint can be used to specify the limit on the data type of table. Constraint can be specified while creating or altering the table statement. Sample of constraint are.

* NOT NULL.
* CHECK.
* DEFAULT.
* UNIQUE.
* PRIMARY KEY.
* FOREIGN KEY.

**27. What is data Integrity?**

Data Integrity defines the accuracy and consistency of data stored in a database. It can also define integrity constraints to enforce business rules on the data when it is entered into the application or database.

**28. What is Auto Increment?**

Auto increment keyword allows the user to create a unique number to be generated when a new record is inserted into the table. AUTO INCREMENT keyword can be used in Oracle and IDENTITY keyword can be used in SQL SERVER.

Mostly this keyword can be used whenever PRIMARY KEY is used.

**29. What is the difference between Cluster and Non-Cluster Index?**

Clustered index is used for easy retrieval of data from the database by altering the way that the records are stored. Database sorts out rows by the column which is set to be clustered index.

A nonclustered index does not alter the way it was stored but creates a complete separate object within the table. It point back to the original table rows after searching.

**30. What is Datawarehouse?**

Datawarehouse is a central repository of data from multiple sources of information. Those data are consolidated, transformed and made available for the mining and online processing. Warehouse data have a subset of data called Data Marts.