

General Commands

SELECT Select data from a database.
SELECT column1,
 column2
 FROM table_name;

SELECT DISTINCT Return only distinct values.
SELECT DISTINCT
 column1, column2
 FROM table_name;

WHERE Extract records that fulfill a specified condition.
SELECT column1,
 column2
 FROM table_name
 WHERE condition;

AND, OR and NOT Filter records based on more than one condition. Combined with WHERE.

The AND operator displays a record if all the conditions separated by AND are TRUE.

SELECT column1,
 column2
 FROM table_name
 WHERE condition1 AND
 condition2;

The OR operator displays a record if any of the conditions separated by OR is TRUE.

SELECT column1,
 column2
 FROM table_name
 WHERE condition1 OR
 condition2;

The NOT operator displays a record if the condition(s) is NOT TRUE

SELECT column1,
 column2
 FROM table_name
 WHERE NOT condition;

ORDER BY Sort the result-set in ascending or descending order.
Ascending order is by default.

SELECT column1,
 column2
 FROM table_name
 ORDER BY column1
 ASC|DESC;

General Commands (cont)

INSERT INTO Insert new records in a table.

1. **INSERT INTO** table_name
Specify both the column names and the values.
 (column1, column2,
 column3)
 VALUES (value1, value2)

If you are adding values for all the columns of the table, no need to specify the column names.

INSERT INTO table_name
 VALUES (value1, value2);

NULL A field with a NULL value is a field with no value. A field with a NULL value is one that has been left blank during record creation.

IS NULL Syntax

SELECT column_names
 FROM table_name
 WHERE column_name IS
 NULL;

IS NOT NULL Syntax

SELECT column_names
 FROM table_name
 WHERE column_name IS
 NOT NULL;

UPDATE Modify the existing records in a table.

UPDATE table_name
 SET column1 = value1,
 column2 = value2
 WHERE condition;

DELETE Delete existing records in a table.

DELETE FROM
 table_name
 WHERE condition;

General Commands (cont)

LIMIT Specify the number of records to return.

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

IN Allows to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.

```
SELECT column_name(s)
FROM table_name
WHERE column_name
IN (SELECT STATEMENT);
```

BETWEEN Selects values within a given range. The values can be numbers, text, or dates. *Is inclusive.*

```
SELECT column_name(s)
FROM table_name
WHERE column_name
BETWEEN value1 AND value2;
```

GROUP BY Groups rows that have the same values into summary rows. Is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

General Commands (cont)

HAVING Was added to SQL because the WHERE keyword cannot be used with aggregate functions

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

EXISTS Test for the existence of any record in a subquery. Returns TRUE if the subquery returns one or more records.

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name
FROM table_name WHERE condition);
```

ANY Allow to perform a comparison between a single column value and a range of other values.

1.Returns a boolean value as a result 2. Returns TRUE if ANY of the subquery values meet the condition.

ANY means that the condition will be true if the operation is true for any of the values in the range.

```
SELECT column_name(s)
FROM table_name
WHERE column_name
operator ANY
(SELECT column_name
FROM table_name
WHERE condition);
```



General Commands (cont)

ALL

Returns:

1. A boolean value as a result
2. Returns TRUE if ALL of the subquery values meet the condition
3. Is used with SELECT, WHERE and HAVING statements.

ALL means that the condition will be true only if the operation is true for all values in the range.

ALL Syntax With SELECT

```
SELECT ALL column_name(s)
FROM table_name
WHERE condition;
```

ALL Syntax With WHERE or HAVING

```
SELECT column_name(s)
FROM table_name
WHERE column_name
operator ALL
(SELECT column_name
FROM table_name
WHERE condition);
```

General Commands (cont)

INSERT INTO SELECT Copies data from one table and Inserts it into another table. Requires that the data types in source and target tables matches. The existing records in the target table are unaffected.

Copy all columns from one table to another table

```
INSERT INTO table2
SELECT * FROM table1
WHERE condition;
```

Copy only some columns from one table into another table

```
INSERT INTO table2 (col1,
col2 ...)
SELECT col1, col2 ...
FROM table1
WHERE condition;
```

General Commands (cont)

CASE Goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause. If there is no ELSE part and no conditions are true, it returns NULL.

```

CASE
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  WHEN conditionN THEN resultN
  ELSE result
END;

```

Database

CREATE DATABASE	Create a new SQL database.	CREATE DATABASE db;
SHOW DATABASES	Show the list of databases.	SHOW DATABASES;
DROP DATABASE	Drop a SQL database.	DROP DATABASE db;
BACKUP DATABASE	Create a full backup of an existing SQL database. 'E:\testDB.bak'	BACKUP DATABASE db TO DISK = 'filepath';

A differential back up only backs up the parts of the database that have changed since the last full database backup.

```

BACKUP
DATABASE db
TO DISK =
'filepath'
WITH DIFFERENTIAL;

```

Database (cont)

CREATE TABLE Create a new table in a database.

```

CREATE
TABLE table (
  column1
  datatype,
  column2
  datatype,
  columnN
  datatype );

```

A copy of an existing table can also be created using CREATE TABLE. the new table will be filled with the existing values from the old table.

```

CREATE
TABLE
new_table
AS
SELECT
column1,
column2,...
FROM existi-
ng_table
WHERE ....;

```

DROP TABLE Drop an existing table in a database.

```

DROP TABLE
table_name;

```

TRUNCATE TABLE delete the data inside a table, but not the table itself.

```

TRUNCATE
TABLE table_-
name;

```

ALTER TABLE is used to add, delete, or modify columns in an existing table. is also used to add and drop various constraints on an existing table.

```

ADD Column
ALTER TABLE
table_name
ADD column-
_name
datatype;

```

```

DROP
COLUMN
ALTER TABLE
table_name
DROP
COLUMN
column_name;

```



Database (cont)

RENAME COLUMN ALTER TABLE table_name RENAME COLUMN old_name to new_name;

Constraints

Numeric Functions

MIN() Returns the minimum value in a set of values.

MAX() Returns the maximum value in a set of values.

COUNT() Returns the number of records. NULL values are not counted.

AVG() Returns the average value of an expression. NULL values are ignored.

SUM() Calculates the sum of a set of values. NULL values are ignored.

Arithmetic Operators

+ Add

- Subtract

***** Multiply

/ Divide

% Modulo

Bitwise Operators

& Bitwise AND

| Bitwise OR

^ Bitwise exclusive OR

Comparison Operators

= Equal to

> Greater than

< Less than

>= Greater than or equal to

<= Less than or equal to

<> Not equal to

Compound Operators

+= Add equals

-= Subtract equals

***=** Multiply equals

/= Divide equals

%= Modulo equals

&= Bitwise AND equals

^.= Bitwise exclusive equals

|*= Bitwise OR equals

Logical Operators

ALL TRUE if all of the subquery values meet the condition

AND TRUE if all the conditions separated by AND is TRUE

ANY TRUE if any of the subquery values meet the condition

BETWEEN TRUE if the operand is within the range of comparisons

EXISTS TRUE if the subquery returns one or more records

IN TRUE if the operand is equal to one of a list of expressions

LIKE TRUE if the operand matches a pattern

NOT Displays a record if the condition(s) is NOT TRUE

OR TRUE if any of the conditions separated by OR is TRUE

SOME TRUE if any of the subquery values meet the condition

Joining Tables

INNER JOIN Selects records that have matching values in both tables.

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name =
table2.column_name;
```



Joining Tables (cont)

LEFT JOIN	Returns all records from the left table (table1), and the matching records (if any) from the right table (table2).	SELECT column_name(s) FROM table1 LEFT JOIN table2 ON table1.column_name = table2.column_name;
RIGHT JOIN	Returns all records from the right table (table2), and the matching records (if any) from the left table (table1).	SELECT column_name(s) FROM table1 RIGHT JOIN table2 ON table1.column_name = table2.column_name;
CROSS JOIN	Returns all records from both tables (table1 and table2).	SELECT column_name(s) FROM table1 CROSS JOIN table2;
SELF JOIN	A self join is a regular join, but the table is joined with itself.	SELECT column_name(s) FROM table1 T1, table1 T2 WHERE condition;
UNION	Combine the result-set of two or more SELECT statements. 1) Every SELECT statement within UNION must have the same number of columns 2) The columns must also have similar data types 3) The columns in every SELECT statement must also be in the same order	

Joining Tables (cont)

UNION Syntax.	<i>Selects only distinct values by default. To allow duplicate values, use UNION ALL</i>	SELECT column_name(s) FROM table1 UNION SELECT column_name(s) FROM table2;
UNION ALL		SELECT column_name(s) FROM table1 UNION ALL SELECT column_name(s) FROM table2;

Date Functions

ADDDATE() OR DATE_ADD()	adds a time/date interval to a date and then returns the date. <i>ADDDATE(date, INTERVAL value addunit)</i> <i>ADDDATE(date, days)</i> <i>DATE_ADD(date, INTERVAL value addunit)</i>
ADDTIME()	Adds a time interval to a time/datetime and then returns the time/datetime. <i>ADDTIME(datetime, addtime)</i>
CURDATE()	Returns the current date. The date is returned as "-YYYY-MM-DD" (string) or as YYYYMMDD (numeric). This function equals the CURDATE() function. <i>CURDATE()</i>
CURRENT_DATE()	Returns the current date. The date is returned as "-YYYY-MM-DD" (string) or as YYYYMMDD (numeric). This function equals the CURRENT_DATE() function. <i>CURRENT_DATE()</i>



Date Functions (cont)

CURRENT_TIME()	Returns the current time. The time is returned as "HH-MM-SS" (string) or as HHMMSS.uuuuuu (numeric). This function equals the CURTIME() function. <i>CURRENT_TIME()</i>
CURTIME()	Returns the current time. The time is returned as "HH-MM-SS" (string) or as HHMMSS.uuuuuu (numeric). This function equals the CURRENT_TIME() function. <i>CURTIME()</i>
CURRENT_TIMESTAMP()	Returns the current date and time. The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric). <i>CURRENT_TIMESTAMP()</i>
DATE()	Extracts the date part from a datetime expression. <i>DATE(expression)</i>
DATEDIFF()	Returns the number of days between two date values. <i>DATEDIFF(date1, date2)</i>
DATE_FORMAT()	Formats a date as specified. <i>DATE_FORMAT(date, format)</i>
DATE_SUB()	Subtracts a time/date interval from a date and then returns the date. <i>DATE_SUB(date, INTERVAL value interval)</i>
DAY() OR DAYOFMONTH	returns the day of the month for a given date (a number from 1 to 31). <i>DAY(date) DAYOFMONTH-H(date)</i>

Date Functions (cont)

DAYNAME()	returns the weekday name for a given date. <i>DAYNAME(date)</i>
DAYOFWEEK()	returns the weekday index for a given date (a number from 1 to 7). 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday. <i>DAYOFWEEK(date)</i>
DAYOFWEEK(date)	returns the day of the year for a given date (a number from 1 to 366). <i>DAYOFYEAR(date)</i>
EXTRACT()	extracts a part from a given date. <i>EXTRACT(part FROM date)</i>
FROM_DAYS()	returns a date from a numeric datevalue.is to be used only with dates within the Gregorian calendar. is the opposite of the TO_DAYS() function. <i>FROM_DAYS(number)</i>
HOUR()	returns the hour part for a given date (from 0 to 838). <i>HOUR(datetime)</i>
LAST_DAY()	extracts the last day of the month for a given date. <i>LAST_DAY(date)</i>
LOCALTIME() OR LOCALTIMESTAMP()	returns the current date and time. The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric). <i>LOCALTIME()</i>
MAKEDATE()	creates and returns a date based on a year and a number of days value. <i>MAKEDATE(year, day)</i>



Date Functions (cont)

MAKETIME() Create and return a time value based on an hour, minute, and second value. **MAKETIME(hour, minute, second)**

MICROSECOND() Return the microsecond part of a datetime. **MICROSECOND(datetime)**



By **ArturPuiu**
cheatography.com/arturpuiu/

Not published yet.
Last updated 19th November, 2023.
Page 8 of 8.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>