

SQL Cheatsheet

Basic Queries & Operators

SELECT c1, c2 FROM t;

Query data in columns c1, c2 from a table.

SELECT * FROM t;

Query all rows and columns from a table

SELECT c1, c2 FROM t

WHERE condition;

Query data and filter rows using a boolean condition: =, <, <=, >, >=, <>.

SELECT c1, c2 FROM t1

WHERE c1[NOT] LIKE pattern;

Query rows using pattern matching. Use with % or _

SELECT c1, c2 FROM t

WHERE c1 [NOT] IN value_list;

Filter rows with values equals to those in the value_list.

SELECT c1, c2 FROM t

WHERE c1 BETWEEN limit1 AND limit2;

Filter rows with values between the two limits.

SELECT c1, c2 FROM t

WHERE c1 IS [NOT] NULL;

Filter NULL values.

SELECT DISTINCT c1 FROM t

WHERE condition;

Returns distinct rows from a table

SELECT c1, c2 FROM t

LIMIT n;

Returns the first n rows.

JOINS

SELECT c1, c2

FROM t1

INNER JOIN t2 ON condition;

Inner join t1 and t2

SELECT c1, c2

FROM t1

LEFT JOIN t2 ON condition;

Left join t1 and t2

SELECT c1, c2

FROM t1

RIGHT JOIN t2 ON condition;

Right join t1 and t2

SELECT c1, c2

FROM t1

FULL OUTER JOIN t2 ON condition;

Full outer join t1 and t2

SELECT c1, c2

FROM t1

CROSS JOIN t2;

Cross join t1 and t2. Results also called: Cartesian Product.

SELECT c1, c2

FROM t1 A

INNER JOIN t1 B ON condition;

Join t1 to itself using INNER JOIN. Also called: SELF JOIN.

Order, Group, Aggregate

SELECT c1, c2 FROM t

ORDER BY c1 [ASC][DESC];

Sort the results in ascending or descending order.

SELECT c1, aggregate(c2)

FROM t

GROUP BY c1;

Group rows using an aggregate function.

SELECT c1, aggregate(c2)

FROM t

GROUP BY c1;

HAVING condition;

Filter groups using HAVING operator.

AGGREGATE FUNCTIONS

AVG returns the average of a list

COUNT returns the number of elements of a list

SUM returns the total of a list

MAX returns the maximum value in a list

MIN returns the minimum value in a list

SQL Cheatsheet

DDL - Data Definition Language

```
CREATE TABLE t (  
  id INT PRIMARY KEY,  
  c1 VARCHAR NOT NULL,  
  c2 INT  
);
```

Create a new table with three columns

```
DROP TABLE t;
```

Delete table from the database

```
ALTER TABLE t ADD column;
```

Add a new column to the table

```
ALTER TABLE t1 RENAME c1 TO c2;
```

Rename column c1 to c2

```
ALTER TABLE t DROP COLUMN c;
```

Remove column c from the table

```
ALTER TABLE t RENAME TO tt;
```

Rename a table from t to tt

```
TRUNCATE TABLE t;
```

Remove all data in a table

DML - Data Manipulation Language

```
INSERT INTO t(column_list)  
VALUES (value_list);
```

Insert one record into a table.

```
INSERT INTO t1(column_list)  
SELECT column_list  
FROM t2;
```

Insert rows from table t2 into table t1. Columns types must match.

```
UPDATE t  
SET c1= new_value,  
      c2 = new_value  
/*c3, c4, ... */;
```

Update values in the column c1 and c2 for all rows.

```
UPDATE t  
SET c1 = new_value,  
      c2 = new_value  
WHERE condition;
```

Update values in the column c1, c2 that match the condition

```
DELETE FROM t;
```

Delete all data in a table

```
DELETE FROM t  
WHERE condition;
```

Delete rows that match the condition.

Constraints, Views, Triggers

CONSTRAINTS DEFINITION

```
CREATE TABLE t1(  
  c1 INT PRIMARY KEY, -- primary key constraint  
  c2 INT NOT NULL, -- NOT NULL constraint  
  FOREIGN KEY (c2) REFERENCES t2(c2), -- Foreign Key  
  c3 INT,  
  UNIQUE(c3), -- UNIQUE constraint  
  CHECK (c3 > 0 AND c3 >= c2) -- CHECK constraint  
);
```

VIEWS

```
CREATE [TEMPORARY] VIEW v(c1,c2)  
AS  
SELECT c1, c2  
FROM t;
```

Create a new view that consists of two columns from table t.

```
DROP VIEW v;
```

Delete the view

TRIGGERS

```
CREATE [OR ALTER] TRIGGER trigger_name  
BEFORE [OR AFTER] EVENT  
ON table_name FOR EACH ROW [OR STATEMENT]  
EXECUTE stored_procedure;
```

Create or modify a trigger.
EVENT values: **INSERT**, **UPDATE**, **DELETE**

```
DROP TRIGGER tr;
```

Delete a specific tr.