# SQL Cheatsheet

## Basic Queries & Operators

```
SELECT column1, column2 FROM table;
```
Query data in columns **column1**, column2 from a table.

```
SELECT * FROM table;
```
Query all rows and columns from a table

```
SELECT column1, column2 FROM table
WHERE condition;
```
Query data and filter rows using a boolean condition: =, <, <=, >, >=, <>.

```
SELECT column1, column2 FROM table1
WHERE column1[NOT] LIKE pattern;
```
Query rows using pattern matching.  Use with % or _

```
SELECT column1, column2 FROM table
WHERE column1 [NOT] IN value_list;
```
Filters rows with values equals to those in the value_list.

```
SELECT column1, column2 FROM table
WHERE column1 BETWEEN limit1 AND limit2;
```
Filters rows with values between the two limits.

```
SELECT column1, column2 FROM table
WHERE column1 IS [NOT] NULL;
```
Filters NULL values.

```
SELECT DISTINCT column1 FROM table
WHERE condition;
```
Returns distinct rows from a table

```
SELECT column1, column2 FROM table
WHERE rownum<n;
```
Returns the first n rows.

## JOINs

```
SELECT column1, column2
FROM table1
INNER JOIN table2 ON condition;
```
Inner join table1 and table2.

```
SELECT column1, column2
FROM table1
LEFT JOIN table2 ON condition;
```
Left join table1 and table2.

```
SELECT column1, column2
FROM table1
RIGHT JOIN table2 ON condition;
```
Right join table1 and table2

```
SELECT column1, column2
FROM table1
FULL OUTER JOIN table2 ON condition;
```
Full outer join table1 and table2

```
SELECT column1, column2
FROM table1
CROSS JOIN table2;
```
Cross join table1 and table2.

Results also called as ⇒ CARTESIAN PRODUCT

```
SELECT column1, column2
FROM table1 A
INNER JOIN table1 B ON condition;
```
Join table1 to itself using INNER JOIN.Also called as⇒SELF JOIN

## Order, Group, Aggregate

```
SELECT column1, column2 FROM table
ORDER BY column1 [ASC][DESC];
```
Sorts the results in ascending or descending order.

```
SELECT column1, aggregate_function_name(column2)
FROM table
GROUP BY column1;
```
Groups rows using an aggregate function.

```
SELECT column1, aggregate_function_name(column2)
FROM table
GROUP BY column1;
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 table_name(
    id NUMBER PRIMARY KEY,
    column_name1 VARCHAR2 NOT NULL,
    column_name2 DATE
);
```

Creates a new table with three columns.

```
DROP TABLE table_name;
```

Deletes table from the database

**ALTER TABLE table_name ADD column_name;**

Adds a new column to the table.

```
ALTER TABLE table_name1 RENAME
column_name1 TO column_name2;
```

Renames column column_name1(old name) to column_name2(new name).

**ALTER TABLE table_name DROP COLUMN column_name;**

Removes column column_name from the table.

**ALTER TABLE old_table_name RENAME**
**TO new_table_name;**

Renames a table from old_table_name to new_table_name.

**TRUNCATE TABLE table_name;**

Removes all data in a table.

## DML - Data Manipulation Language

**INSERT INTO table_name(column_list)**
**VALUES (value_list);**

Inserts one record into a table.

**INSERT INTO table1(column_list)**
**SELECT column_list**
**FROM table2;**

Inserts rows from table table2 into table table1.
Columns types must match!

```
UPDATE table
SET column1 = new_value,
    column2 = new_value
  /*column3, column4, ... */;
```

Updates values in the column column1 and column2 for all rows.

```
UPDATE table
SET column1 = new_value,
    column2 = new_value
WHERE condition;
```

Updates values in the column column1, column2 that match the condition.

**DELETE FROM table_name;**

Deletes all data in a table.

**DELETE FROM table_name**
**WHERE condition;**

Deletes rows that match the condition.

## Constraints, Views, Triggers

### CONSTRAINTS DEFINITION

```
CREATE TABLE table1(
   col1 NUMBER PRIMARY KEY, -- primary key constraint
   col2 NUMBER NOT NULL, -- NOT NULL constraint
   FOREIGN KEY (col2) REFERENCES table2(col2),-- Foreign Key
   col3 NUMBER,
   UNIQUE(col3), -- UNIQUE constraint
   CHECK (col3> 0 AND col3 >= col2) -- CHECK constraint
);
```

### VIEWS

```
CREATE [TEMPORARY] VIEW view_name(col1,col2)
AS
SELECT col1, col2
FROM table;
```

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

**DROP VIEW view_name;**

Deletes the view.

### TRIGGERS

```
CREATE [OR ALTER] TRIGGER trigger_name
BEFORE [OR AFTER] EVENT
ON table_name FOR EACH ROW [OR STATEMENT]
BEGIN
   ...
END;
```

Create or modify a trigger.

EVENT values: INSERT, UPDATE, DELETE

DROP TRIGGER trigger_name;

Deletes trigger.