SQL Basics Cheatsheet

Find All Columns and Rows in a Table

```
SELECT * FROM <table name>;
```

The asterisk or star symbol (`*`) means all columns.

The semi-colon (`;`) terminates the statement like a period in sentence or question mark in a question.

Examples:

```
SELECT * FROM books;
SELECT * FROM products;
SELECT * FROM users;
SELECT * FROM countries;
```

Retrieving Specific Columns of Information

Retrieving a single column:

```
SELECT <column name> FROM <table name>;
```

Examples:

```
SELECT email FROM users;
SELECT first_name FROM users;
SELECT name FROM products;
SELECT zip_code FROM addresses;
```

Retrieving multiple columns:

```
SELECT <column name 1>, <column name 2>, ... FROM <table na
```

Examples:

```
SELECT first_name, last_name FROM customers;
SELECT name, description, price FROM products;
SELECT title, author, isbn, year_released FROM books;
SELECT name, species, legs FROM pets;
```

Aliasing Column Names

```
SELECT <column name> AS <alias> FROM <table name>;
SELECT <column name> <alias> FROM <table name>;
```

Examples:

```
SELECT username AS Username, first_name AS "First Name" FRO
SELECT title AS Title, year AS "Year Released" FROM movies;
SELECT name AS Name, description AS Description, price AS "
SELECT name Name, description Description, price "Current P
```

Finding the Data You Want

```
SELECT <columns> FROM <table> WHERE <condition>;
```

Equality Operator

Find all rows that a given value matches a column's value.

```
SELECT <columns> FROM <table> WHERE <column name> = <value>
```

Examples:

```
SELECT * FROM contacts WHERE first_name = "Andrew";
SELECT first_name, email FROM users WHERE last_name = "Chal
SELECT name AS "Product Name" FROM products WHERE stock_cou
SELECT title "Book Title" FROM books WHERE year_published =
```

Inequality Operator

Find all rows that a given value doesn't match a column's value.

```
SELECT <columns> FROM <table> WHERE <column name> != <value
SELECT <columns> FROM <table> WHERE <column name> <> <value
```

The not equal to or inequality operator can be written in two ways `!=` and `<>`. The latter is *less* common.

Examples:

```
SELECT * FROM contacts WHERE first_name != "Kenneth";
SELECT first_name, email FROM users WHERE last_name != "L:o
SELECT name AS "Product Name" FROM products WHERE stock_cou
SELECT title "Book Title" FROM books WHERE year_published !
```

Relational Operators

There are several relational operators you can use:

- < less than
- <= less than or equal to
- > greater than
- >= greater than or equal to

These are primarily used to compare *numeric* and *date/time* types.

```
SELECT <columns> FROM <table> WHERE <column name> < <value>
SELECT <columns> FROM <table> WHERE <column name> <= <value
SELECT <columns> FROM <table> WHERE <column name> > <value>
SELECT <columns> FROM <table> WHERE <column name> >= <value
```

Examples:

```
SELECT first_name, last_name FROM users WHERE date_of_birth
SELECT title AS "Book Title", author AS Author FROM books W
SELECT name, description FROM products WHERE price > 9.99;
SELECT title FROM movies WHERE release_year >= 2000;
```

More Than One Condition

You can compare multiple values in a `WHERE` condition. If you want to test that *both* conditions are true use the `AND` keyword, or *either* conditions are true use the `OR` keyword.

```
SELECT <columns> FROM <table> WHERE <condition 1> AND <cond
SELECT <columns> FROM <table> WHERE <condition 1> OR <condi
```

Examples:

```
SELECT username FROM users WHERE last_name = "Chalkley" AND
SELECT * FROM products WHERE category = "Games Consoles" AN
SELECT * FROM movies WHERE title = "The Matrix" OR title =
SELECT country FROM countries WHERE population < 1000000 OR
```

Searching in a Set of Values

```
SELECT <columns> FROM <table> WHERE <column> IN (<value 1>,
```

Examples:

```
SELECT name FROM islands WHERE id IN (4, 8, 15, 16, 23, 42)
SELECT * FROM products WHERE category IN ("eBooks", "Books"
SELECT title FROM courses WHERE topic IN ("JavaScript", "Da
SELECT * FROM campaigns WHERE medium IN ("email", "blog", "
```

To find all rows that are not in the set of values you can use `NOT IN`.

```
SELECT <columns> FROM <table> WHERE <column>  NOT IN (<valu
```

Examples:

```
SELECT answer FROM answers WHERE id IN (7, 42);
SELECT * FROM products WHERE category NOT IN ("Electronics"
SELECT title FROM courses WHERE topic NOT IN ("SQL", "NoSQL
```

Searching within a Range of Values

```
SELECT <columns> FROM <table> WHERE <column> BETWEEN <lesse
```

Examples:

```
SELECT * FROM movies WHERE release_year BETWEEN 2000 AND 20
SELECT name, description FROM products WHERE price BETWEEN
SELECT name, appointment_date FROM appointments WHERE appoi
```

Pattern Matching

Placing the percent symbol (`%`) any where in a string in conjunction with the `LIKE` keyword will operate as a wildcard. Meaning it can be substituted by any number of

characters, including zero!

```
SELECT <columns> FROM <table> WHERE <column> LIKE <pattern>
```

Examples:

```
SELECT title FROM books WHERE title LIKE "Harry Potter%Fire
SELECT title FROM movies WHERE title LIKE "Alien%";
SELECT * FROM contacts WHERE first_name LIKE "%drew";
SELECT * FROM books WHERE title LIKE "%Brief History%";
```

PostgreSQL Specific Keywords

`LIKE` in PostgreSQL is case-sensitive. To do case-insensitive searches use `ILIKE`.

```
SELECT * FROM contacts WHERE first_name ILIKE "%drew";
```

Missing Values

```
SELECT * FROM <table> WHERE <column> IS NULL;
```

Examples:

```
SELECT * FROM people WHERE last_name IS NULL;
SELECT * FROM vhs_rentals WHERE returned_on IS NULL;
SELECT * FROM car_rentals WHERE returned_on IS NULL AND loc
```

To filter out missing values use can use `IS NOT NULL`.

```
SELECT * FROM <table> WHERE <column> IS NOT NULL;
```

Examples

```
SELECT * FROM people WHERE email IS NOT NULL;
SELECT * FROM addresses WHERE zip_code IS NOT NULL;
```