**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.

Example:

SELECT \* FROM books;

**Retrieving Specific Columns of Information**

Retrieving a single column:

SELECT <column name> FROM <table name>;

Example:

SELECT email FROM users;

Retrieving multiple columns:

SELECT <column name 1>, <column name 2>, ... FROM <table name>;

Example:

SELECT first\_name, last\_name FROM customers;

**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" FROM users;

SELECT name Name, price "Current Price" FROM products;

**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 = "Chalkley";

SELECT name AS "Product Name" FROM products WHERE stock\_count = 0;

**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.

**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.

Examples:

SELECT first\_name FROM users WHERE date\_of\_birth < '1998-12-01';

SELECT title AS "Book Title" FROM books WHERE year\_released <= 2015;

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, using the AND and OR keywords.

SELECT <columns> FROM <table> WHERE <condition 1> AND <condition 2> ...;

SELECT <columns> FROM <table> WHERE <condition 1> OR <condition 2> ...;

Examples:

SELECT username FROM users WHERE last\_name = "Chalkley" AND first\_name = "Andrew";

SELECT country FROM countries WHERE population < 1000000 OR population > 100000000;

**Searching in a Set of Values**

SELECT <columns> FROM <table> WHERE <column> IN (<value 1>, <value 2>, ...);

Examples:

SELECT name FROM islands WHERE id IN (4, 8, 15, 16, 23, 42);

SELECT \* FROM products WHERE category IN ("eBooks", "Books", "Comics");

SELECT title FROM courses WHERE topic IN ("JavaScript", "Databases", "CSS");

SELECT \* FROM campaigns WHERE medium IN ("email", "blog", "ppc");

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 (<value 1>, <value 2>, ...);

Examples:

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 <lesser value> AND <greater value>;

Examples:

SELECT \* FROM movies WHERE release\_year BETWEEN 2000 AND 2010;

SELECT name, appointment\_date FROM appointments WHERE appointment\_date BETWEEN "2015-01-01" AND "2015-01-07";

**Pattern Matching**

Placing the percent symbol (%) anywhere 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 \* 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 location = "PDX";

To filter out missing values you 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;