My Ultimate SQL Cheatsheet

SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table.col1 = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

SELECT Keywords

DISTINCT: Removes duplicate results

```
SELECT DISTINCT product_name
FROM product;
```

BETWEEN: Matches a value between two other values(inclusive)

```
SELECT product_name
FROM product
WHERE price BETWEEN 50 AND 100;
```

IN: Matches to any of the values in a list

```
SELECT product_name
FROM product
WHERE category IN
('Electronics', 'Furniture');
```

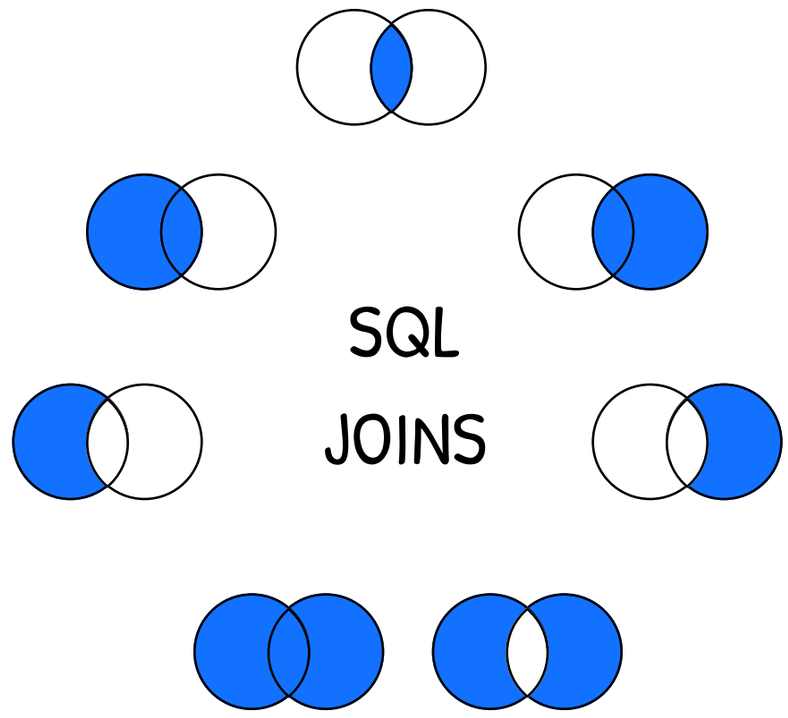
LIKE: Performs wildcard matches using _or%

```
SELECT product_name
FROM product
WHERE product_name
LIKE '%Desk%';
```

Joins

```
SELECT t1.*, t2.*
FROM t1
JOIN_type t2 ON t1 = t2.col;
```

Table 1	Table 2
A	A
B	B
C	D



INNER JOIN: show all matching records in both tables.

A	A
B	B

LEFT JOIN: show all records from left table, and any matching records from right table.

A	A
B	B
C	

RIGHT JOIN: show all records from right table, and any matching records from left table.

A	A
B	B
	D

FULL JOIN: show all records from both tables, whether there is a match or not.

A	A
B	B
C	
	D

CASE Statement

Simple Case

```
CASE name
   WHEN 'John' THEN 'Name John'
   WHEN 'Steve' THEN 'Name Steve'
   ELSE 'Unknown'
END
```

Searched Case

```
CASE
   WHEN name='John' THEN 'Name John'
   WHEN name='Steve' THEN 'Name Steve'
   ELSE 'Unknown'
END
```

Common Table Expression

```
WITH queryname AS (
SELECT col1, col2
FROM firsttable)
SELECT col1, col2..
FROM queryname
```

Modifying Data

Insert

```
INSERT INTO tablename (col1, col2...)
VALUES (val1, val2);
```

Insert from a Table

```
INSERT INTO tablename (col1, col2...)
SELECT col1, col2...
```

Insert Multiple Rows

```
INSERT INTO tablename(col1, col2...)
VALUES
(valA1, valB1),
(valA2, valB2),
(valA3, valB3);
```

Update

```
UPDATE tablename
SET col1 =val1
WHERE condition;
```

Update with a Join

```
UPDATE t
SET col1 =val1
FROM tablename t
INNER JOIN table x
ON t.id =x.tid
WHERE condition;
```

Delete

```
DELETE FROM tablename
WHERE condition;
```

Indexes

Create Index

```
CREATE INDEX indexname
ON tablename (cols);
```

Drop Index

```
DROP INDEX indexname;
```

Indexes

UNION: Shows unique rows from two result sets

UNION ALL: Shows all rows from two result sets

INTERSECT: Shows rows that exist in both result sets

MINUS: Shows rows that exist in the first result set but not the second.

Aggregate Functions

- **SUM:** Finds a total of the numbers provided
- **COUNT:** Finds the number of records
- **AVG:** Finds the average of the numbers provided
- **MIN:** Finds the lowest of the numbers provided
- **MAX:** Finds the highest of the numbers provided

Common Functions

- **LENGTH(string):** Returns the length of the provided string

- **INSTR(string, substring):** Returns the position of the substring within the specified string.
- **ADDDATE(input_date, days):** Adds a number of days to a specified date.
- **NOW:** Returns the current date, including time.
- **CEILING(input_val):** Returns the smallest integer greater than the provided number.
- **FLOOR(input_val):** Returns the largest integer less than the provided number
- **ROUND(input_val, [round_to]):** Rounds a number to a specified number of decimal places.
- **TRUNCATE(input_value, num_decimals):** Truncates a number to a number of decimals.
- **REPLACE(whole_string.string_to_replace, replacement_string):** Replace one string inside the whole string with another string.
- **SUBSTRING(string, start_position):** Returns part of a value, based on a position and length.

Create Table

Create Table

```
CREATE TABLE tablename (
column_name data_type
);
```

Create Table with Constraints

```
CREATE TABLE tablename (
column_name data_type NOT NULL,
CONSTRAINT pkname PRIMARY KEY (col),
CONSTRAINT pkname FOREIGN KEY (col),
```

```
REFERENCES other_table(col_in_other_table),
        CONSTRAINT ucname UNIQUE (col),
        CONSTRAINT ckname CHECK (consitions)
);
```

Create Temporary Table

```
CREATE TEMPORARY TABLE
tablename (
    colname datatype
);
```

Drop Table

```
DROP TABLE tablename;
```

Alter Table

Add Column

```
ALTER TABLE tablename
ADD columnname datatype;
```

Drop Column

```
ALTER TABLE tablename
DROP COLUMN columnname;
```

Modify Column

```
ALTER TABLE tablename CHANGE
columnname newcolumnname newdatatype;
```

Rename Column

```
ALTER TABLE tablename CHANGE
COLUMN currentname TO newname;
```

Add Constraint

```
ALTER TABLE tablename ADD
CONSTRAINT constraintname
constrainttype (column);
```

Drop Constraint

```
ALTER TABLE tablename DROP
constraint_type constraintname;
```

Rename Table

```
ALTER TABLE tablename
RENAME TO newtablename;
```

Window/Analytic Functions

```
functions_name (arguments) OVER (
    [query_partition _clause]
    [ORDER BY order_by_clause
    [windowing_clause]])
```

Example using RANK, showing the students details and their rank according to the fees_paid, grouped by gender;

```
SELECT
student_id, first_name, last_name, gender, fees_paid, RANK() OVER(
    PARTITION BY gender ORDER BY fees_paid) AS rank_val
FROM student;
```

Window/Analytic Functions

Single Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary =(
    SELECT MAX(salary)
    FROM employee
    WHERE last_name LIKE 'c%'
);
```

Multi Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary IN (
    SELECT salary
    FROM employee
    WHERE last_name LIKE 'c%'
);
```