SQL Quick Reference

1. Finding Data Queries

SELECT: used to select data from a database

- SELECT * FROM table_name;

DISTINCT: filters away duplicate values and returns rows of specified column

- SELECT DISTINCT column_name;

WHERE: used to filter records/rows

- SELECT column1, column2 FROM table_name WHERE condition;
- SELECT * FROM table_name WHERE condition1 AND condition2;
- SELECT * FROM table_name WHERE condition1 OR condition2;
- SELECT * FROM table_name WHERE NOT condition;
- SELECT * FROM table_name WHERE condition1 AND (condition2 OR condition3);
- SELECT * FROM table_name WHERE EXISTS (SELECT column_name FROM table_name WHERE condition);

ORDER BY: used to sort the result-set in ascending or descending order

- SELECT * FROM table_name ORDER BY column;
- SELECT * FROM table_name ORDER BY column DESC;
- SELECT * FROM table_name ORDER BY column1 ASC, column2 DESC;

SELECT TOP: used to specify the number of records to return from top of table

- SELECT TOP number columns_names FROM table_name WHERE condition;
- SELECT TOP percent columns_names FROM table_name WHERE condition;

- Not all database systems support `SELECT TOP`. The MySQL equivalent is the `LIMIT` clause
- `SELECT` column_names `FROM` table_name `LIMIT` offset, count;

LIKE: operator used in a WHERE clause to search for a specific pattern in a column

- % (percent sign) is a wildcard character that represents zero, one, or multiple characters
- _ (underscore) is a wildcard character that represents a single character
- `SELECT` column_names `FROM` table_name `WHERE` column_name `LIKE` pattern;
- `LIKE` 'a%' (find any values that start with "a")
- `LIKE` '%a' (find any values that end with "a")
- `LIKE` '%or%' (find any values that have "or" in any position)
- `LIKE` '_r%' (find any values that have "r" in the second position)
- `LIKE` 'a_%_%' (find any values that start with "a" and are at least 3 characters in length)
- `LIKE` '[a-c]%' (find any values starting with "a", "b", or "c"

IN: operator that allows you to specify multiple values in a WHERE clause

- essentially the IN operator is shorthand for multiple OR conditions
- `SELECT` column_names `FROM` table_name `WHERE` column_name `IN` (value1, value2, ...);
- `SELECT` column_names `FROM` table_name `WHERE` column_name `IN` (`SELECT STATEMENT`);

BETWEEN: operator selects values within a given range inclusive

- `SELECT` column_names `FROM` table_name `WHERE` column_name `BETWEEN` value1 `AND` value2;
- `SELECT` * `FROM` Products `WHERE` (column_name `BETWEEN` value1 `AND` value2) `AND NOT` column_name2 `IN` (value3, value4);
- `SELECT` * `FROM` Products `WHERE` column_name `BETWEEN` #01/07/1999# AND #03/12/1999#;

NULL: values in a field with no value

- `SELECT * FROM table_name WHERE column_name IS NULL;`
- `SELECT * FROM table_name WHERE column_name IS NOT NULL;`

AS: aliases are used to assign a temporary name to a table or column

- `SELECT column_name AS alias_name FROM table_name;`
- `SELECT column_name FROM table_name AS alias_name;`
- `SELECT column_name AS alias_name1, column_name2 AS alias_name2;`
- `SELECT column_name1, column_name2 + ', ' + column_name3 AS alias_name;`

UNION: set operator used to combine the result-set of two or more SELECT statements

- Each SELECT statement within UNION must have the same number of columns
- The columns must have similar data types
- The columns in each SELECT statement must also be in the same order
- `SELECT columns_names FROM table1 UNION SELECT column_name FROM table2;`
- `UNION` operator only selects distinct values, `UNION ALL` will allow duplicates

INTERSECT: set operator which is used to return the records that two SELECT statements have in common

- Generally used the same way as **UNION** above
- `SELECT columns_names FROM table1 INTERSECT SELECT column_name FROM table2;`

EXCEPT: set operator used to return all the records in the first SELECT statement that are not found in the second SELECT statement

- Generally used the same way as **UNION** above
- `SELECT columns_names FROM table1 EXCEPT SELECT column_name FROM table2;`

ANY|ALL: operator used to check subquery conditions used within a WHERE or HAVING clauses

- The `ANY` operator returns true if any subquery values meet the condition
- The `ALL` operator returns true if all subquery values meet the condition

- `SELECT` columns_names `FROM` table1 `WHERE` column_name operator (`ANY`|`ALL`) (`SELECT` column_name `FROM` table_name `WHERE` condition);

GROUP BY: statement often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns

- `SELECT` column_name1, COUNT(column_name2) `FROM` table_name `WHERE` condition `GROUP BY` column_name1 `ORDER BY` COUNT(column_name2) DESC;

HAVING: this clause was added to SQL because the WHERE keyword could not be used with aggregate functions

- `SELECT COUNT`(column_name1), column_name2 `FROM` table `GROUP BY` column_name2 `HAVING COUNT`(column_name1) > 5;

WITH: often used for retrieving hierarchical data or re-using temp result set several times in a query. Also referred to as "Common Table Expression"

- ```
  WITH RECURSIVE cte AS (
    SELECT c0.* FROM categories AS c0 WHERE id = 1 # Starting point
    UNION ALL
    SELECT c1.* FROM categories AS c1 JOIN cte ON c1.parent_category_id = cte.id
  )
  SELECT *
  FROM cte
  ```

2. Data Modification Queries

INSERT INTO: used to insert new records/rows in a table

- `INSERT INTO` table_name (column1, column2) `VALUES` (value1, value2);
- `INSERT INTO` table_name `VALUES` (value1, value2 ...);

UPDATE: used to modify the existing records in a table
  ```

- UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;
- UPDATE table_name SET column_name = value;

DELETE: used to delete existing records/rows in a table

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

3. Reporting Queries

COUNT: returns the # of occurrences

- SELECT COUNT (DISTINCT column_name);

MIN() and MAX(): returns the smallest/largest value of the selected column

- SELECT MIN (column_names) FROM table_name WHERE condition;
- SELECT MAX (column_names) FROM table_name WHERE condition;

AVG(): returns the average value of a numeric column

- SELECT AVG (column_name) FROM table_name WHERE condition;

SUM(): returns the total sum of a numeric column

- SELECT SUM (column_name) FROM table_name WHERE condition;

4. Join Queries

INNER JOIN: returns records that have matching value in both tables

- SELECT column_names FROM table1 INNER JOIN table2 ON table1.column_name=table2.column_name;

- SELECT table1.column_name1, table2.column_name2, table3.column_name3 FROM ((table1 INNER JOIN table2 ON relationship) INNER JOIN table3 ON relationship);

LEFT (OUTER) JOIN: returns all records from the left table (table1), and the matched records from the right table (table2)

- SELECT column_names FROM table1 LEFT JOIN table2 ON table1.column_name=table2.column_name;

RIGHT (OUTER) JOIN: returns all records from the right table (table2), and the matched records from the left table (table1)

- SELECT column_names FROM table1 RIGHT JOIN table2 ON table1.column_name=table2.column_name;

FULL (OUTER) JOIN: returns all records when there is a match in either left or right table

- SELECT column_names FROM table1 FULL OUTER JOIN table2 ON table1.column_name=table2.column_name;

Self JOIN: a regular join, but the table is joined with itself

- SELECT column_names FROM table1 T1, table1 T2 WHERE condition;

5. View Queries

CREATE: create a view

- CREATE VIEW view_name AS SELECT column1, column2 FROM table_name WHERE condition;

SELECT: retrieve a view

- SELECT * FROM view_name;

DROP: drop a view

- `DROP VIEW` view_name;

6. Altering Table Queries

ADD: add a column

- `ALTER TABLE` table_name `ADD` column_name column_definition;

MODIFY: change data type of column

- `ALTER TABLE` table_name `MODIFY` column_name column_type;

DROP: delete a column

- `ALTER TABLE` table_name `DROP COLUMN` column_name;

7. Creating Table Query

CREATE: create a table

- `CREATE TABLE` table_name `(`

```
column1 datatype,
column2 datatype,
column3 datatype,
column4 datatype,
);
```

Reference:

Enoch Tang, 2018: https://github.com/enochtangg/quick-SQL-cheatsheet

---