SQL Cheat Sheet

A quick reference guide to SQL, covering essential commands, data types, and functions for database management and querying.

Basic SQL Commands

Data Definition Language (DDL)

`CREATE TABLE table_name (column1 datatype, column2 datatype, ...);`	Creates a new table in the database.
`ALTER TABLE table_name ADD column_name datatype;`	Adds a new column to an existing table.
`ALTER TABLE table_name DROP COLUMN column_name;`	Deletes a column from an existing table.
`ALTER TABLE table_name MODIFY COLUMN column_name datatype;`	Modifies the data type of a column.
`DROP TABLE table_name;`	Deletes a table from the database.
`TRUNCATE TABLE table_name;`	Removes all rows from a table, but keeps the table structure.

Data Manipulation Language (DML)

`INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);`	Inserts a new row into a table.
`UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;`	Updates existing rows in a table based on a condition.
`DELETE FROM table_name WHERE condition;`	Deletes rows from a table based on a condition.
`SELECT column1, column2 FROM table_name WHERE condition;`	Retrieves data from one or more tables.
`SELECT * FROM table_name;`	Retrieves all columns from a table.

Data Control Language (DCL)

`GRANT privilege ON object TO user;`	Grants privileges to a user on a specific database object.
`REVOKE privilege ON object FROM user;`	Revokes privileges from a user on a specific database object.

SQL Querying

Basic SELECT Statement

`SELECT column1, column2 FROM table_name WHERE condition ORDER BY column1 ASC/DESC LIMIT number;`

- `WHERE`: Filters rows based on a condition.
- `ORDER BY`: Sorts the result set.
- `ASC`: Ascending order.
- `DESC`: Descending order.
- `LIMIT`: Limits the number of rows returned.

Aggregate Functions

`COUNT(column)`	Returns the number of rows.
`SUM(column)`	Returns the sum of values in a column.
`AVG(column)`	Returns the average value of a column.
`MIN(column)`	Returns the minimum value in a column.
`MAX(column)`	Returns the maximum value in a column.

GROUP BY and HAVING

`GROUP BY column`	Groups rows that have the same values in a column into summary rows.
`HAVING condition`	Filters the results of a `GROUP BY` query.
Example	`SELECT department, COUNT(*) FROM employees GROUP BY department HAVING COUNT(*) > 5;`

Joins and Subqueries

Joins

Joins are used to combine rows from two or more tables based on a related column.

- `INNER JOIN`: Returns rows when there is a match in both tables.
- `LEFT JOIN`: Returns all rows from the left table, and the matched rows from the right table.
- `RIGHT JOIN`: Returns all rows from the right table, and the matched rows from the left table.
- `FULL OUTER JOIN`: Returns all rows when there is a match in either left or right table.

Join Syntax

`SELECT columns FROM table1 INNER JOIN table2 ON table1.column = table2.column;`	Inner Join Example
`SELECT columns FROM table1 LEFT JOIN table2 ON table1.column = table2.column;`	Left Join Example
`SELECT columns FROM table1 RIGHT JOIN table2 ON table1.column = table2.column;`	Right Join Example
`SELECT columns FROM table1 FULL OUTER JOIN table2 ON table1.column = table2.column;`	Full Outer Join Example

Subqueries

A subquery is a query nested inside another SQL query. Subqueries can be used in `SELECT`, `FROM`, and `WHERE` clauses.

Example:
`SELECT column1 FROM table_name WHERE column2 IN (SELECT column2 FROM another_table);`

Transactions and Indexing

Transactions

A transaction is a sequence of SQL operations that are performed as a single logical unit of work.

- `START TRANSACTION;` - Begins a transaction.
- `COMMIT;` - Saves the changes made during the transaction.
- `ROLLBACK;` - Reverts the changes made during the transaction if an error occurs.

Transaction Examples

`START TRANSACTION; UPDATE accounts SET balance = balance - 100 WHERE account_id = 1; UPDATE accounts SET balance = balance + 100 WHERE account_id = 2; COMMIT;`	Transfers $100 from account 1 to account 2.
`START TRANSACTION; UPDATE accounts SET balance = balance - 100 WHERE account_id = 1; UPDATE accounts SET balance = balance + 100 WHERE account_id = 2; ROLLBACK;`	If any error occurs, all changes are rolled back.

Indexing

Indexes are special lookup tables that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table.

- `CREATE INDEX index_name ON table_name (column1, column2, ...);` - Creates an index on a table.
- `DROP INDEX index_name ON table_name;` - Deletes an index from a table.