Cheat Sheet for comprehensive SQL

SQL Basics

Data Types

- **Numeric**: INT, FLOAT, DECIMAL(p,s)
- **Character**: CHAR(n), VARCHAR(n), TEXT
- **Date/Time**: DATE, TIME, DATETIME, TIMESTAMP
- **Boolean**: BOOLEAN (TRUE/FALSE)

Basic Commands

- **SELECT**: Retrieve data from a table

```
SELECT column1, column2 FROM table_name;
```

- **WHERE**: Filter records

```
SELECT * FROM table_name WHERE condition;
```

- **ORDER BY**: Sort results

```
SELECT * FROM table_name ORDER BY column ASC|DESC;
```

- **LIMIT**: Restrict the number of rows returned

```
SELECT * FROM table_name LIMIT 10;
```

Data Manipulation

INSERT

- Add new records to a table

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

UPDATE

- Modify existing records

```
UPDATE table_name SET column1 = value1 WHERE condition;
```

DELETE

- Remove records from a table

```
DELETE FROM table_name WHERE condition;
```

Aggregation and Grouping

Aggregate Functions

- **COUNT**: Count the number of rows

```
SELECT COUNT(column) FROM table_name;
```

- **SUM**: Sum of values

```
SELECT SUM(column) FROM table_name;
```

- **AVG**: Average value

```
SELECT AVG(column) FROM table_name;
```

- **MIN/MAX**: Minimum/Maximum value

```
SELECT MIN(column), MAX(column) FROM table_name;
```

GROUP BY

- Group rows that have the same values

```
SELECT column, AGG_FUNC(column) FROM table_name GROUP BY column;
```

HAVING

- Filter groups (similar to WHERE but for groups)

```
  SELECT column, AGG_FUNC(column) FROM table_name GROUP BY column
HAVING condition;
```

Joins and Relationships

INNER JOIN

- Return only matching rows

```
  SELECT * FROM table1 INNER JOIN table2 ON table1.column =
table2.column;
```

LEFT JOIN

- Return all rows from the left table, and the matched rows from the right table

```
  SELECT * FROM table1 LEFT JOIN table2 ON table1.column =
table2.column;
```

RIGHT JOIN

- Return all rows from the right table, and the matched rows from the left table

```
  SELECT * FROM table1 RIGHT JOIN table2 ON table1.column =
table2.column;
```

FULL OUTER JOIN

- Return all rows when there is a match in either left or right table

```
  SELECT * FROM table1 FULL OUTER JOIN table2 ON table1.column =
table2.column;
```

Subqueries and Common Table Expressions (CTEs)

Subqueries

- Queries embedded within other queries

```
  SELECT column FROM table WHERE column IN (SELECT column FROM table
WHERE condition);
```

CTEs

- Temporary result sets that can be referenced within a SELECT, INSERT, UPDATE, or DELETE statement

```
WITH cte_name AS (
    SELECT column FROM table WHERE condition
)
SELECT * FROM cte_name;
```

Indexes and Performance

CREATE INDEX

- Speed up data retrieval

```
CREATE INDEX index_name ON table_name (column);
```

DROP INDEX

- Remove an index

```
DROP INDEX index_name ON table_name;
```

Transactions

BEGIN TRANSACTION

- Start a transaction

```
BEGIN TRANSACTION;
```

COMMIT

- Save changes permanently

```
COMMIT;
```

ROLLBACK

- Undo changes

```
ROLLBACK;
```

Views and Stored Procedures

CREATE VIEW

- Virtual table based on the result-set of an SQL statement

```
CREATE VIEW view_name AS SELECT column FROM table WHERE condition;
```

CREATE PROCEDURE

- Stored procedure to encapsulate SQL code

```
CREATE PROCEDURE procedure_name AS BEGIN SQL_statements END;
```

Advanced Features

CASE Statement

- Conditional logic within a query

```
SELECT column,
       CASE
           WHEN condition1 THEN result1
           WHEN condition2 THEN result2
           ELSE result3
       END AS alias
FROM table_name;
```

UNION

- Combine the result set of two or more SELECT statements

```
SELECT column FROM table1 UNION SELECT column FROM table2;
```

EXCEPT/INTERSECT

- EXCEPT: Return distinct rows from the first query that are not in the second query

```
SELECT column FROM table1 EXCEPT SELECT column FROM table2;
```

- INTERSECT: Return distinct rows that are present in both queries

```
SELECT column FROM table1 INTERSECT SELECT column FROM table2;
```

Tips and Tricks

- **Use EXPLAIN**: Analyze query execution plan

```
EXPLAIN SELECT * FROM table_name WHERE condition;
```

- **Avoid SELECT***: Specify columns to improve performance

- **Use EXISTS**: Check for existence of rows

```
SELECT column FROM table WHERE EXISTS (SELECT 1 FROM table WHERE condition);
```

- **Batch Inserts**: Use multiple VALUES for faster inserts

```
INSERT INTO table_name (column1, column2) VALUES (value1, value2), (value3, value4);
```

Common Errors and Solutions

- **Syntax Errors**: Double-check SQL syntax and keywords

- **Duplicate Entries**: Use UNIQUE constraints or check before insert

- **Performance Issues**: Analyze query execution plan, use indexes, and optimize joins

Resources

- **Documentation**: Official SQL documentation for your database system

- **Online Tutorials**: Websites like W3Schools, SQLZoo, and Khan Academy

- **Community Forums**: Stack Overflow, DBA Stack Exchange

This cheat sheet provides a comprehensive overview of essential SQL commands, functions, and best practices. Use it as a quick reference to enhance your SQL skills and efficiency.

By Ahmed Baheeg Khorshid

ver 1.0