SQL CHEATSHEET

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
SQL
STRUCTURED QUERY LANGUAGE
???

WHAT IS SQL?

Structured Query Language (SQL) is the standard language used to communicate with relational databases, enabling users to query, manipulate, and manage data. SQL is the foundation for retrieving and updating data in databases, making it an essential skill for database administrators, developers, and data analysts.

This cheat sheet is specifically tailored for Oracle SQL, highlighting the unique keywords and functions that set Oracle apart from other database systems. Whether you're a beginner looking to get familiar with Oracle's syntax or an experienced user needing a quick reference, this guide covers the key commands and features essential for working with Oracle databases efficiently.

KEYWORDS

SELECT

```
SELECT col1, col2
FROM table
WHERE condition
GROUP BY cols
HAVING condition
ORDER BY col;
```

ORDER OF PROCESSING

1. FROM
2. JOIN
3. WHERE
4. GROUP BY
5. HAVING
6. SELECT
7. DISTINCT
8. ORDER BY

9. FETCH

SELECT KEYWORDS

DISTINCT: Removes duplicate results

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

IN: Matches a value to one of many values

LIKE: Performs partial/wildcard matches

DATA TYPES

VARCHAR2(size): Variable-length character data.

NUMBER(p,s): Numeric data type.

DATE: Date and time.

CLOB: Character Large Object, stores large amounts of text.

BLOB: Binary Large Object, stores large amounts of binary data.

CONSTRAINTS

NOT NULL: Ensures a column cannot have NULL value.

UNIQUE: Ensures all values in a column are unique.

PRIMARY KEY: Uniquely identifies each record in a table.

FOREIGN KEY: Ensures referential integrity between tables.

CHECK: Ensures that the value in a column meets a specific condition.

DEFAULT: Sets a default value for a column when no value is specified.

MODIFYING DATA

INSERT:

```
INSERT INTO tablename (col1, col2...)
VALUES (val1, val2);
INSERT From Table:
INSERT INTO tablename (col1, col2…)
SELECT col1, col2…
```

UPDATE:

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

DELETE:

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

TRUNCATE:

```
TRUNCATE TABLE tablename;
```

UPDATE with Join:

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

INSERT Multiple Rows:

```
INSERT
INTO tablename (col1, col2) VALUES
(valA1, valB1)
INTO tablename (col1, col2) VALUES
(valA2, valB2)
SELECT * FROM dual;
```

MERGE:

```
MERGE INTO table_name
USING table_name
ON (condition)
WHEN MATCHED THEN update_clause
DELETE where_clause
WHEN NOT MATCHED THEN insert_clause;
```

JOINS

```
SELECT t1.*, t2.*
```

```
FROM t1
join_type t2 ON t1.col = t2.col;
```

INNER JOIN: show all matching records in both tables.

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

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

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

CROSS JOIN: show all combinations of records from both tables.

SELF JOIN: join a table to itself. Used for hierarchical data.

```
SELECT p.*, c.*
FROM yourtable p
INNER JOIN yourtable c ON p.id =
c.parent_id;
```

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 fkname FOREIGN KEY (col)
REFERENCES
other_table(col_in_other_table),
CONSTRAINT ucname UNIQUE (col),
CONSTRAINT ckname CHECK (conditions)
);
```

Drop Table:

```
DROP TABLE tablename;
```

Create Temporary Table:

```
CREATE GLOBAL TEMPORARY TABLE tname (
colname data_type
) ON COMMIT DELETE ROWS
```

ALTER TABLE

Add Column

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

Drop Column

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

Modify Column

```
ALTER TABLE tablename MODIFY columnname
newdatatype;
```

Rename Column

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

Add Constraint

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

Drop Constraint

```
ALTER TABLE tablename DROP CONSTRAINT
constraintname;
```

Rename Table

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

INDEXES

Create Index:

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

Drop Index:

```
DROP INDEX indexname;
```

SET OPERATORS

UNION: Shows unique rows from two result sets.

```
SELECT column1 FROM table1 UNION SELECT column1 FROM table2;
```

UNION ALL: Shows all rows from two result sets.

```
SELECT column1 FROM table1 UNION ALL SELECT column1 FROM table2;
```

INTERSECT: Shows rows that exist in both result sets.

```
SELECT column1 FROM table1 INTERSECT SELECT column1 FROM table2;
```

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

```
SELECT column1 FROM table1 MINUS SELECT column1 FROM table2;
```

SUBQUERIES

Single-row subquery:

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

Multi-row subquery:

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

Correlated subquery:

```
SELECT column FROM table1 WHERE column1 = (SELECT column2 FROM
table2 WHERE table1.column = table2.column);
```

AGGREGATE FUNCTIONS

COUNT

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

SUM

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

AVG

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

MAX

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

MIN

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

GROUPING AND FILTERING

GROUP BY

```
SELECT column1, COUNT(column2) FROM table GROUP BY column1;
```

HAVING

```
SELECT column1, COUNT(column2) FROM table GROUP BY column1
HAVING COUNT(column2) > 1;
```

VIEWS

Create View

```
CREATE VIEW view_name AS SELECT column1, column2 FROM table
WHERE condition;
```

Drop View

```
DROP VIEW view_name;
```

SEQUENCES

Create Sequence

```
CREATE SEQUENCE sequence_name START WITH 1 INCREMENT BY 1;
```

Next Value in Sequence

```
SELECT sequence_name.NEXTVAL FROM dual;
```

Drop Sequence

```
DROP SEQUENCE sequence_name;
```

TRANSACTIONS

BEGIN TRANSACTION

```
BEGIN;
```

COMMIT

```
COMMIT;
```

ROLLBACK

```
ROLLBACK;
```

PL/SQL BLOCK

Anonymous Block

```
DECLARE
  -- Declarations
BEGIN
  -- Statements
EXCEPTION
  -- Exception handling
END;
```

Stored Procedure

```
CREATE OR REPLACE PROCEDURE procedure_name AS
BEGIN
  -- Procedure code
END procedure_name;
```

COMMON ORACLE-SPECIFIC FUNCTIONS

TO_DATE

```
SELECT TO_DATE('2024-08-11', 'YYYY-MM-DD') FROM dual;
```

TO_CHAR

```
SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS') FROM dual;
```

NVL

```
SELECT NVL(column, 'default_value') FROM table;
```

DECODE

```
SELECT DECODE(column, 'value1', 'result1', 'value2', 'result2',
'default') FROM table;
```

COALESCE

```
SELECT COALESCE(column1, column2, 'default_value') FROM table;
```

TO_DATE

```
SELECT TO_DATE('11-Aug-2024', 'DD-Mon-YYYY') AS Date_Value FROM
DUAL;
```

SYSDATE

```
SELECT SYSDATE AS Current_Date FROM DUAL;
```

CEIL

```
SELECT CEIL(3.14) AS Ceil_Value FROM DUAL;
```

FLOOR

```
SELECT FLOOR(3.14) AS Floor_Value FROM DUAL;
```

SUBSTR

```
SELECT SUBSTR('Oracle SQL', 8, 3) AS Substring_Value FROM DUAL;
```

USER MANAGEMENT

Create User

```
CREATE USER username IDENTIFIED BY password;
```

Grant Privileges

```
GRANT privilege TO username;
```

Revoke Privileges

```
REVOKE privilege FROM username;
```

Drop User

```
DROP USER username CASCADE;
```