Oracle Database SQL Cheatsheet

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

Basic SQL Commands

Data Retrieval

`SELECT`	Retrieves data from one or more tables. **Example:** `SELECT column1, column2 FROM table_name;`
`FROM`	Specifies the table(s) to retrieve data from. **Example:** `SELECT * FROM employees;`
`WHERE`	Filters the rows based on a specified condition. **Example:** `SELECT * FROM employees WHERE salary > 50000;`
`ORDER BY`	Sorts the result-set in ascending or descending order. **Example:** `SELECT * FROM employees ORDER BY last_name ASC;`
`GROUP BY`	Groups rows that have the same values in specified columns into summary rows. **Example:** `SELECT department, COUNT(*) FROM employees GROUP BY department;`
`HAVING`	Filters the groups based on a specified condition (used with `GROUP BY`). **Example:** `SELECT department, COUNT(*) FROM employees GROUP BY department HAVING COUNT(*) > 10;`
`DISTINCT`	Returns only distinct (unique) values. **Example:** `SELECT DISTINCT department FROM employees;`

Data Manipulation

`INSERT`	Inserts new rows into a table. **Example:** `INSERT INTO employees (first_name, last_name) VALUES ('John', 'Doe');`
`UPDATE`	Modifies existing rows in a table. **Example:** `UPDATE employees SET salary = 60000 WHERE employee_id = 123;`
`DELETE`	Deletes rows from a table. **Example:** `DELETE FROM employees WHERE employee_id = 123;`
`MERGE`	Combines `INSERT`, `UPDATE`, and `DELETE` operations into a single statement. **Example:** `MERGE INTO target_table t USING source_table s ON (t.id = s.id) WHEN MATCHED THEN UPDATE SET t.column1 = s.column1 WHEN NOT MATCHED THEN INSERT (id, column1) VALUES (s.id, s.column1);`

Advanced SQL Concepts

Joins

`INNER JOIN`	Returns rows when there is a match in both tables. **Example:** `SELECT * FROM employees INNER JOIN departments ON employees.department_id = departments.department_id;`
`LEFT JOIN`	Returns all rows from the left table, and the matched rows from the right table. If there is no match, the result is NULL on the right side. **Example:** `SELECT * FROM employees LEFT JOIN departments ON employees.department_id = departments.department_id;`
`RIGHT JOIN`	Returns all rows from the right table, and the matched rows from the left table. If there is no match, the result is NULL on the left side. **Example:** `SELECT * FROM employees RIGHT JOIN departments ON employees.department_id = departments.department_id;`
`FULL OUTER JOIN`	Returns all rows when there is a match in one of the tables. **Example:** `SELECT * FROM employees FULL OUTER JOIN departments ON employees.department_id = departments.department_id;`
`CROSS JOIN`	Returns the Cartesian product of the sets of rows from the joined tables. **Example:** `SELECT * FROM employees CROSS JOIN departments;`

Subqueries

Definition	A query nested inside another query. **Example:** `SELECT * FROM employees WHERE department_id IN (SELECT department_id FROM departments WHERE location = 'New York');`
Types	• **Scalar subquery:** Returns a single value. • **Multiple-row subquery:** Returns multiple rows. • **Multiple-column subquery:** Returns multiple columns.

Set Operations

`UNION`	Combines the result-set of two or more `SELECT` statements (removes duplicate rows). **Example:** `SELECT column1 FROM table1 UNION SELECT column1 FROM table2;`
`UNION ALL`	Combines the result-set of two or more `SELECT` statements (includes duplicate rows). **Example:** `SELECT column1 FROM table1 UNION ALL SELECT column1 FROM table2;`
`INTERSECT`	Returns the intersection of two `SELECT` statements (returns common rows). **Example:** `SELECT column1 FROM table1 INTERSECT SELECT column1 FROM table2;`
`MINUS`	Returns the difference between two `SELECT` statements (returns rows from the first `SELECT` that are not in the second `SELECT`). In other SQL dialects, this may be called `EXCEPT`. **Example:** `SELECT column1 FROM table1 MINUS SELECT column1 FROM table2;`

Data Types and Functions

Common Data Types

`VARCHAR2(size)`	Variable-length character string. **Example:** `VARCHAR2(50)`
`NUMBER(p, s)`	Numeric data type with precision `p` and scale `s`. **Example:** `NUMBER(10, 2)`
`DATE`	Stores date and time values. **Example:** `DATE '2024-01-01'`
`CLOB`	Character Large Object; stores large blocks of text. **Example:** `CLOB`
`BLOB`	Binary Large Object; stores large binary data. **Example:** `BLOB`
`TIMESTAMP`	Stores date and time with fractional seconds. **Example:** `TIMESTAMP '2024-01-01 12:00:00.000'`

String Functions

`UPPER(string)`	Converts a string to uppercase. **Example:** `UPPER('hello')` returns `'HELLO'`
`LOWER(string)`	Converts a string to lowercase. **Example:** `LOWER('WORLD')` returns `'world'`
`SUBSTR(string, start, length)`	Extracts a substring from a string. **Example:** `SUBSTR('Oracle', 3, 4)` returns `'acle'`
`LENGTH(string)`	Returns the length of a string. **Example:** `LENGTH('SQL')` returns `3`
`TRIM(string)`	Removes leading and trailing spaces from a string. **Example:** `TRIM('  Oracle  ')` returns `'Oracle'`
`REPLACE(string, old_string, new_string)`	Replaces occurrences of a substring with another substring. **Example:** `REPLACE('SQL Tutorial', 'SQL', 'Oracle SQL')` returns `'Oracle SQL Tutorial'`

Number Functions

`ROUND(number, decimals)`	Rounds a number to a specified number of decimal places. **Example:** `ROUND(123.456, 2)` returns `123.46`
`TRUNC(number, decimals)`	Truncates a number to a specified number of decimal places. **Example:** `TRUNC(123.456, 2)` returns `123.45`
`MOD(number1, number2)`	Returns the remainder of a division operation. **Example:** `MOD(10, 3)` returns `1`
`CEIL(number)`	Returns the smallest integer greater than or equal to a number. **Example:** `CEIL(4.2)` returns `5`
`FLOOR(number)`	Returns the largest integer less than or equal to a number. **Example:** `FLOOR(4.8)` returns `4`

Date Functions

`SYSDATE`	Returns the current date and time. **Example:** `SELECT SYSDATE FROM dual;`
`TO_CHAR(date, format)`	Converts a date to a character string using a specified format. **Example:** `TO_CHAR(SYSDATE, 'YYYY-MM-DD')`
`TO_DATE(string, format)`	Converts a character string to a date using a specified format. **Example:** `TO_DATE('2024-01-01', 'YYYY-MM-DD')`
`ADD_MONTHS(date, number)`	Adds a specified number of months to a date. **Example:** `ADD_MONTHS(SYSDATE, 3)`
`MONTHS_BETWEEN(date1, date2)`	Returns the number of months between two dates. **Example:** `MONTHS_BETWEEN(SYSDATE, DATE '2023-01-01')`

PL/SQL Basics

PL/SQL Structure

```
DECLARE
  -- Declaration section (variables, constants, etc.)
BEGIN
  -- Executable statements
EXCEPTION
  -- Exception handling (optional)
END;
/
```

Variables and Data Types

Declaration	`variable_name data_type [NOT NULL] [:= initial_value];`
	
	Example:
	`counter NUMBER := 0;`
	`message VARCHAR2(100) NOT NULL := 'Hello';`

Common Data Types	• `NUMBER` • `VARCHAR2` • `DATE` • `BOOLEAN` • `%TYPE` (reference data type)

Control Structures

IF Statement	`IF condition THEN`
	`  -- Statements`
	`[ELSIF condition THEN`
	`  -- Statements]`
	`[ELSE`
	`  -- Statements]`
	`END IF;`

Looping Statements	• `FOR` loop • `WHILE` loop • `LOOP` statement (infinite loop with `EXIT` condition)

Example - FOR Loop	`FOR i IN 1..10 LOOP`
	`  -- Statements`
	`END LOOP;`

Exception Handling

Syntax	`EXCEPTION`
	`  WHEN exception_name THEN`
	`  -- Exception handling statements`
	`  WHEN OTHERS THEN`
	`  -- Default exception handling`
	`END;`

Common Exceptions	• `NO_DATA_FOUND` • `TOO_MANY_ROWS` • `DUP_VAL_ON_INDEX` • `OTHERS` (catch-all exception)