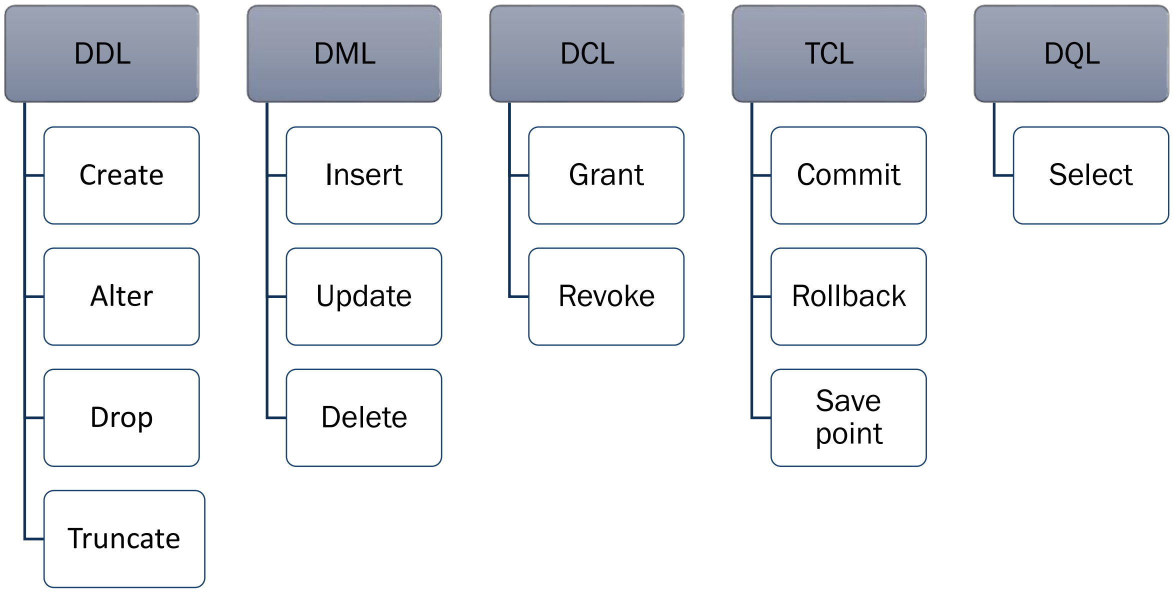
Structured Query language (SQL)

SQL COMMAND

The standard SQL commands to interact with relational databases are **CREATE**, **SELECT**, **INSERT**, **UPDATE**, **DELETE** and **DROP**.

These commands can be classified into the following groups based on their nature:



- **DDL: Data Definition Language**
- **DML: Data Manipulation Language**
- **DCL: Data Control Language**
- **TCL: Transaction Control Language**
- **DQL: Data Query Language**

DDL - Data Definition Language

COMMAND	DESCRIPTION
CREATE	The CREATE Command is used to create a new table, a view of a table, or other object in the database. **Syntax:** CREATE TABLE Table_name (Column1 datatype, Column2 datatype, Column3 datatype, ... ColumnN datatype, PRIMARY KEY (one or more columns)); **Example:** CREATE TABLE Employee (employee_id NUMBER PRIMARY KEY, first_name VARCHAR(50), last_name VARCHAR(50), phone_number VARCHAR(20), hire_date DATE, salary NUMBER(10, 2), department_id NUMBER);
ALTER	The ALTER command is used to modify an existing database object, such as a table. ➢ To Add a column in a table, use the following. ALTER TABLE Employee ADD Email VARCHAR(100); ➢ To Drop a column from a table, use the following. ALTER TABLE Employee DROP COLUMN Email; ➢ To rename a column in a table, use the following:

	ALTER TABLE Employee RENAME COLUMN Email To Email_ID;
	
	➢ To change the data type of a column in a table, use the following:
	
	ALTER TABLE Employee MODIFY COLUMN Email VARCHAR(40);

DROP	The DROP command is used to drop an existing table in a database. DROP TABLE Employee;
TRUNCATE	The TRUNCATE used to delete all the rows from the table, but not the table itself and free the space containing the table. TRUNCATE TABLE Employee;

DML - Data Manipulation Language

COMMAND	DESCRIPTION

INSERT	The INSERT into statement is used to insert new records in a table.
	
	We can write INSERT INTO statement in two ways:
	
	➢ Specify both the column names and the values to be inserted:
	
	INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
	VALUES (value1, value2, value3, ...);
	
	➢ If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table.
	
	INSERT INTO table_name
	VALUES (*value1*, *value2*, *value3*, ...);

UPDATE	The UPDATE statement is used to modify the existing records in a table.
	
	UPDATE table_name
	SET column1 = value1, column2 = value2, ...
	WHERE *condition*;

DELETE	The **DELETE** statement is used to delete existing records in a table. **DELETE FROM** table_*name* **WHERE** condition;

DCL - Data Control Language

COMMAND	DESCRIPTION
GRANT	It is used to give user access privileges to a database. **GRANT SELECT, UPDATE ON** MY_TABLE TO SOME_USER, ANOT HER_USER;
REVOKE	It is used to take back permissions from the user. **REVOKE SELECT, UPDATE ON** MY_TABLE FROM USER1, USER2.

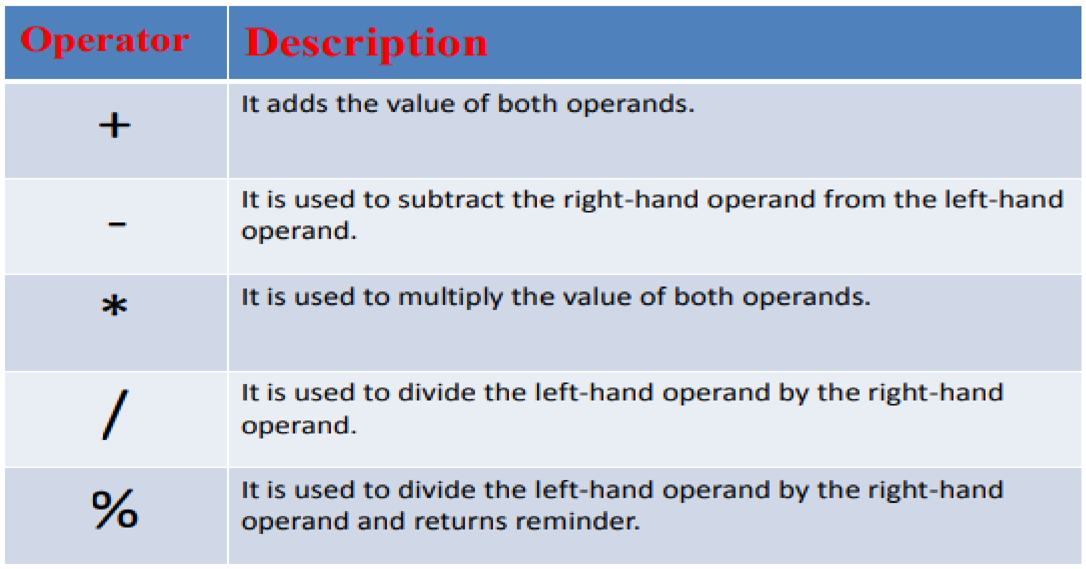
TCL - Transaction Control Language

COMMAND	DESCRIPTION
COMMIT	**COMMIT** command is used to save all the transactions to the database.
ROLLBACK	**ROLLBACK** command is used to undo transactions that have not already been saved to the database.
SAVE POINT	**SAVEPOINT** is used to roll the transaction back to a certain point without rolling **back** the entire transaction.

DQL - Data Query Language

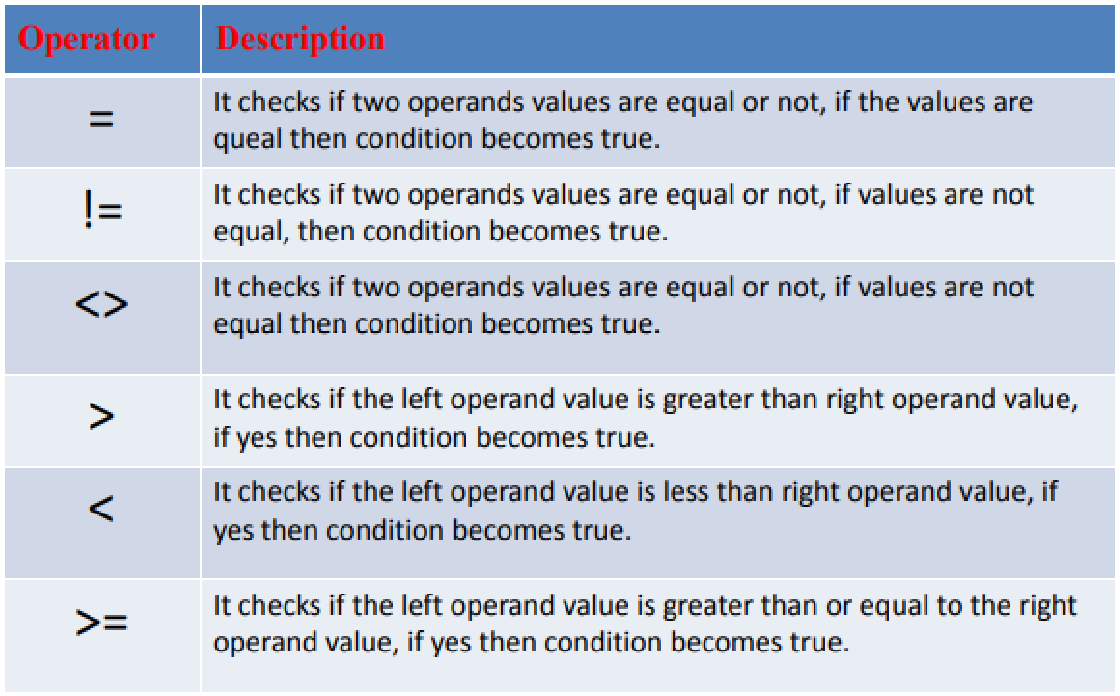
COMMAND	DESCRIPTION
SELECT	**SELECT** retrieves certain records from one or more tables. **Select** * from employee;

SQL OPERATOR

NAME	DESCRIPTION
ARITHMETIC OPERATOR	

Operator	Description
+	It adds the value of both operands.
-	It is used to subtract the right-hand operand from the left-hand operand.
*	It is used to multiply the value of both operands.
/	It is used to divide the left-hand operand by the right-hand operand.
%	It is used to divide the left-hand operand by the right-hand operand and returns reminder.

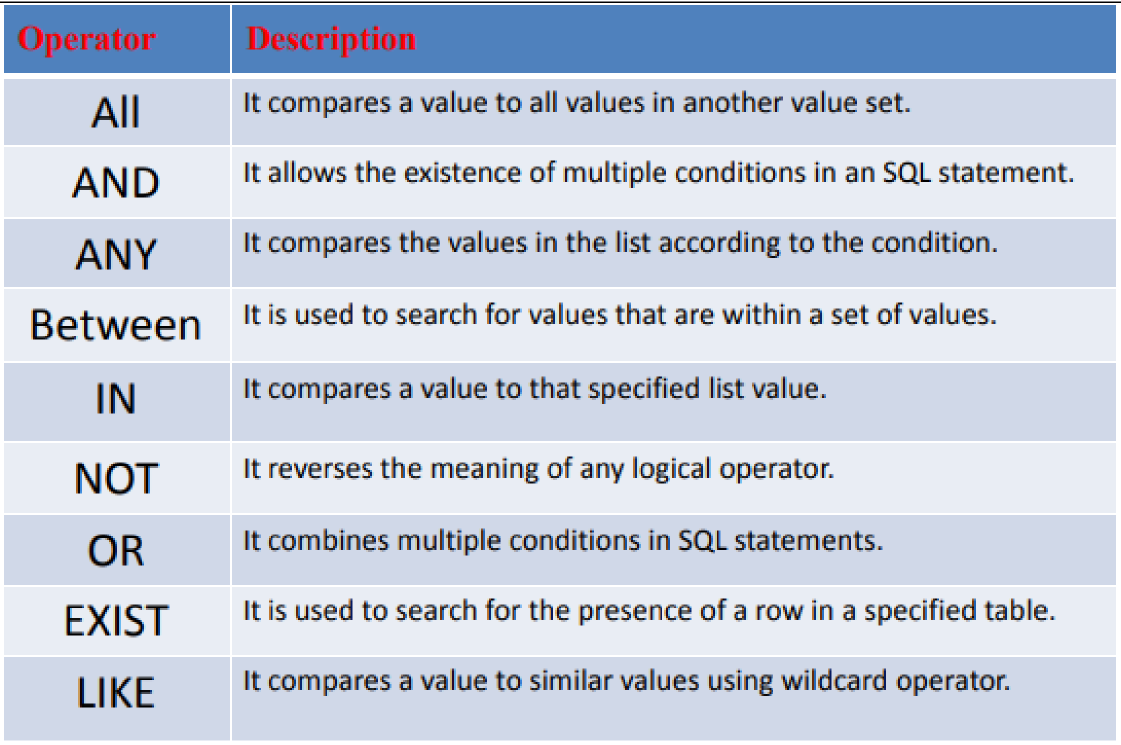
```
SELECT 15+10-5*5/5 FROM dual;
```

NAME	DESCRIPTION
COMPARISION OPERATOR	

Operator	Description
=	It checks if two operands values are equal or not, if the values are queal then condition becomes true.
!=	It checks if two operands values are equal or not, if values are not equal, then condition becomes true.
<>	It checks if two operands values are equal or not, if values are not equal then condition becomes true.
>	It checks if the left operand value is greater than right operand value, if yes then condition becomes true.
<	It checks if the left operand value is less than right operand value, if yes then condition becomes true.
>=	It checks if the left operand value is greater than or equal to the right operand value, if yes then condition becomes true.

```
SELECT * FROM Employee WHERE salary OPERATOR 10000;
```

LOGICAL OPERATOR



Operator	Description
All	It compares a value to all values in another value set.
AND	It allows the existence of multiple conditions in an SQL statement.
ANY	It compares the values in the list according to the condition.
Between	It is used to search for values that are within a set of values.
IN	It compares a value to that specified list value.
NOT	It reverses the meaning of any logical operator.
OR	It combines multiple conditions in SQL statements.
EXIST	It is used to search for the presence of a row in a specified table.
LIKE	It compares a value to similar values using wildcard operator.

SELECT * **FROM** Employee **WHERE** salary **OPERATOR ….;**

WHERE CONDITION

SQL SELECT Statement:

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

SQL DISTINCT Clause:

```
SELECT DISTINCT column1, column2....columnN
FROM   table_name;
```

SQL WHERE Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION;
```

SQL AND/OR Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION-1 {AND|OR} CONDITION-2;
```

SQL IN Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name IN (val-1, val-2,...val-N);
```

SQL BETWEEN Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name BETWEEN val-1 AND val-2;
```

SQL LIKE Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name LIKE { PATTERN };
```

SQL ORDER BY Clause:

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION
ORDER BY column_name {ASC|DESC};
```

SQL GROUP BY Clause:

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name;
```

SQL COUNT Clause:

```
SELECT COUNT(column_name)
FROM   table_name
WHERE  CONDITION;
```

SQL HAVING Clause:

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name
HAVING (arithematic function condition);
```

SQL DESC Statement:

```
DESC table_name;
```

IS NULL Statement:

```
SELECT Column1, Column2...
FROM Table_Name
WHERE Column IS NULL
```

IS NOT NULL Statement:

```
SELECT Column1, Column2...
FROM Table_Name
WHERE Column IS NOT NULL
```