

SOL :- SQL Stands for Structured Query Language

- It is used for storing and managing data in RDBMS
- It is a standard language for relational Database System. It enables a user to create, read, update and delete relational database and tables.
- SQL is not a database system, but it is a query language.
- Suppose you want to perform the queries of SQL language on the stored data in the database. You are required to install any DBMS in your systems. For ex: → Oracle, MySQL
- This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.

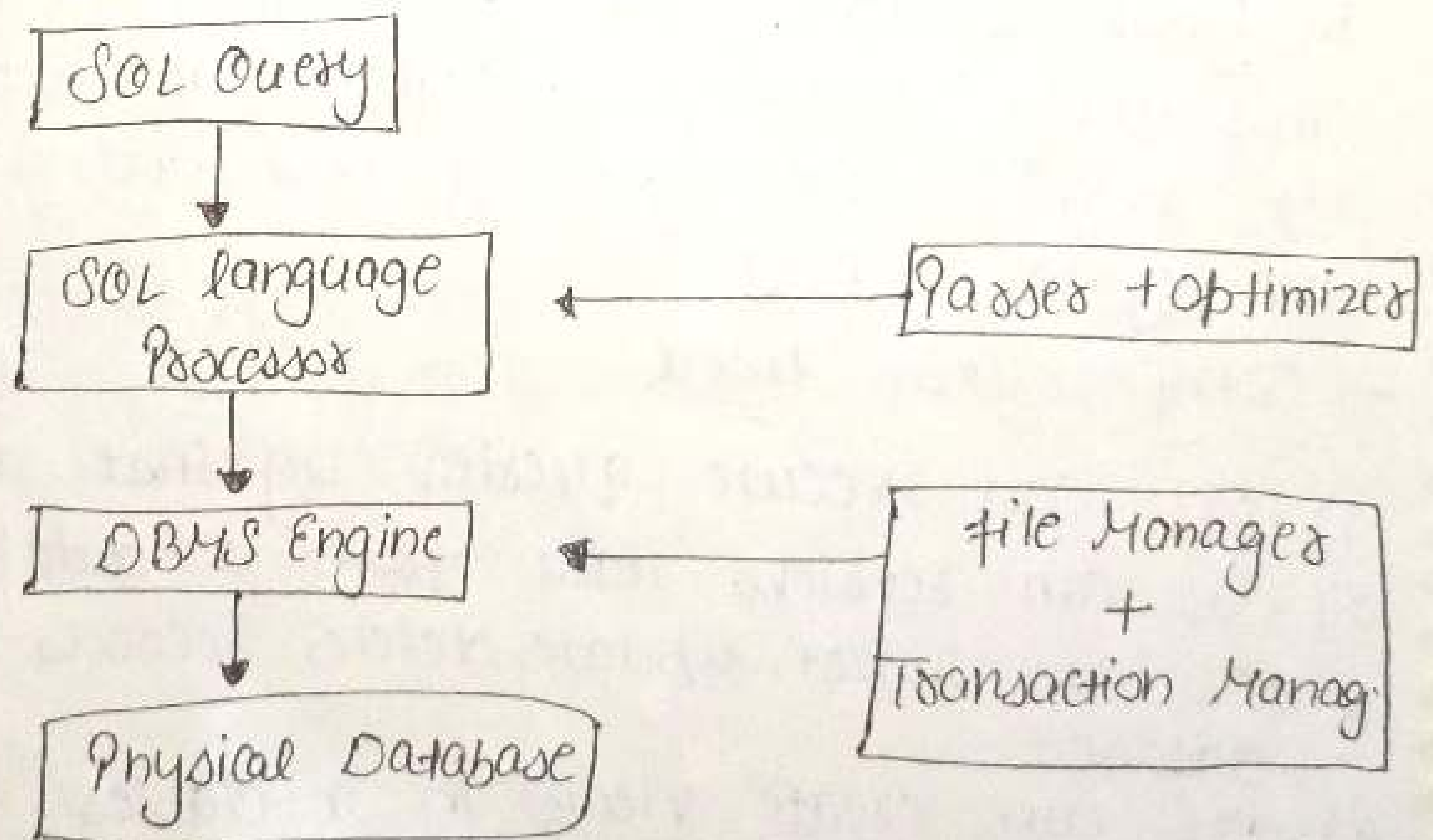
→ Why SQL used

- 1) SQL can execute queries against a database.
- 2) SQL can retrieve data from a database.
- 3) SQL can insert, update, delete, records from a database.
- 4) SQL can create views in a tables.
- 5) Manily represent in tables

Copyrighted by Codecap.io
All rights reserved.

SOL Process

- When we executing the command of SOL on any relational Database Management System, then the system automatically finds the best routine to carry out our request and the SOL engine.
- Structured Query language contain the following four component.
 - 1) Query Dispatcher
 - 2) Optimization Engines
 - 3) Classic query engines
 - 4) SOL Query Engine etc.
- All the non-SOL queries are handled by the classic query engine, but SOL query engine won't handle logical files



Rules for SQL

- Structure Query Language is not case sensitive. Generally, keywords of SQL are written in uppercase.
- Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text lines.
- SQL depends on tuple relational calculus and relational algebra.

Advantages of SQL

- 1) High Speed
- 2) No coding needed
- 3) Well defined standards
- 4) Portability
- 5) Multiple data view

Disadvantage

- 1) Complex interface
- 2) Cost
- 3) Partial control

Application of SQL

- 1) Data retrieval and analysis
- 2) Data Manipulation
- 3) Database Administration
- 4) Business intelligence
- 5) Web development

SQL

- 1) It is a relational database management system
- 2) SQL uses a relational data model with tables
- 3) The Schema of SQL databases is pre-defined, fixed, and static.
- 4) These databases are vertically scalable

No-SQL

- 1) It is a non-relational or distributed database system
- 2) Non-SQL databases use data models with various formats.
- 3) The Schema of No-SQL databases is a dynamic schema for unstructured data.
- 4) These databases are horizontally scalable

SQL Constraints

SQL constraints are rules that are defined for a table column to ensure the data in the table is accurate, consistent and valid.

- Constraints also help to maintain the integrity of the database
- They applied to a single column or a group of columns in a table.
- There are several types of constraints in SQL:

- 1) NOT NULL Constraint: It ensures that a column cannot have a NULL value.

Copyrighted by Codecap.io
All rights reserved.

2) Unique Constraint: Ensures that each value in a column is unique across all rows in the table.

3) Primary key Constraint: A combination of NOT NULL and UNIQUE constraints that identifies a unique row in a table.

4) Foreign key: Ensures that a value in a column matches the primary key of another table.

5) Check Constraint: Ensures that the values in a column meet a specific condition.

6) DEFAULT Constraint: Specifies a default value for a column when no value is specified.

- Every constraint starts or written as large letter.

- By using these constraints, you can create a structured and reliable database that can prevent data inconsistencies and improve the quality of data.

Data integrity → Set of rules and data in database is valid.

Data integrity in RDBMS refers to the accuracy, completeness, and consistency of data stored in a database. It ensures that the data is correct and reliable, and that it can be trusted for use in business operations and decision making.

There are several types of data integrity

- 1) Entity integrity: This ensures that each row in a table is unique and can be identified by a primary key.
• It prevents duplicate data from being inserted into the table.
- 2) Referential integrity: Rows cannot be deleted, which are used by other records.
- 3) Domain integrity: This ensures that data values in each column of a table are within the defined range of valid values or data types.
→ data in column is valid and meet certain criteria, correct format
- 4) User-defined integrity: This allows the database designer to specify additional constraints or rules to ensure that the data is consistent and accurate.
→ allows user to define their own rules

Data Types

Data types are used to represent the nature of the data that can be stored in the database table.
→ type of each column in a table can have specified data type

• for ex: in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column.

• String Data types

Numeric Data types

Date and time data types

types of Data types

- Binary data type → [binary, varbinary, image], small file
- Approximate [float, real] & Exact [int, smallint, bit, decimal]

→ String Data types → allow to store a group of character
- numbers, letters, symbol.
- fixed characters, variable.

1) CHAR (Size) → It is used to specify a fixed
Character data type length string that can contain
numbers, letters and special characters.
• Its size can be 0 to 255 characters Default
is 1.

2) VARCHAR (Size) → It is used to specify a variable
Char length string that can contain
numbers, letters, and special characters.
• Its size can be from 0 to 65535 characters.

3) BINARY (Size) → It is equal to CHAR() but
Binary data type stores binary byte strings.
• Its size parameter specifies the column
length in the bytes. Default is 1. Used to store binary data

4) VARBINARY (Size) → It is equal to VARCHAR() but stores binary byte strings.
• Its size parameter specifies the max. column
length in bytes.

5) TEXT (Size) → It holds a string that can contain
Character a maximum length of 255 characters.

6) TINY (TEXT) → It holds a string with a
maximum length of 255 characters.

7) MEDIUM TEXT

8) LONG TEXT

9) SET (val 1, val 2, val 3, ...)

10) BLOB (Size)

Copyrighted by CodeLeap.io
All rights reserved.

→ Numeric Data types → used for numeric values
→ widely used DT in SQL

1) BIT (Size): → It is used for a bit-value type.
The number of bits per value is specified in size.

• It size can be 1 to 64. Used to store numbers, such as integers, decimals, float num.

2) INT (Size): → It is used for the integer values.

3) FLOAT (Size, d): → It is used to specify a floating point number. Its size parameter specifies the total number of digits.

• The no. of digits after the decimal point is specified by d parameter.

4) BOOL → It is used to specify Boolean values true and false.

- Zero considered false.
- Non-zero considered True.

[bit, tinyint → boolean data type
smallint
bigint → large integer

→ Date and Time Data Types → used for contain both date and time
Date: yyyy-mm-dd
Time: hh:mm:ss

1) DATE: → It is used to specify date format yyyy-mm-dd.

• It supported range is from '1000-01-01' to '9999-12-31'.

2) DATETIME: → It is used to specify date and time combination.

• '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.

3) TIME STAMP (fsp): → It is used to specify the timestamp.

• Its value is stored as the no. of seconds.

• yyyy-mm-dd hh:mm:ss

4) TIME (fsp) :- It is used to specify the time format

• hh:mm:ss

5) Year :- It is used to specify a year in four-digit format.

Others data types

Copyrighted by Codecap.io
All rights reserved.

SQL Server data types

1) char(n) → fixed width character string data type.

• size upto 8000 characters

2) varchar(n) → It is a variable width character string data type.

3) varchar(max) :- variable width character string data types.

4) Text → Variable width character string DT.

5) binary(n) → fixed width binary string data type

SQL

- SQL is a language used to communicate with database and perform operation.
- A language with database
 - ANSI Standard operation
 - keyword cannot be abbreviated.
 - Statement manipulate data and table definitions in the database

SQL Plus

- Tool used to interact with database using Oracle [Tool] SQL comm
- An environment
 - Oracle proprietary
 - keywords can be abbreviated.
 - Commands do not allow manipulation of values in the database.
→ is used to execute those queries and operation against a Oracle database

#SQL Syntax

- When you want to do some operations on the data in the database, then you must have to write the query in the pre-defined syntax of SQL.
- The syntax of the structured query language is a unique set of rules and guidelines, which is non case-sensitive.
- Its syntax is defined and maintained by the ISO and ANSI standards.
- Some points about SQL keywords.
 - 1) You can write the keywords of SQL in both uppercase and lowercase, but writing the SQL keywords in uppercase improves the readability of the SQL query.
keywords (eg SELECT, GRANT, DELETE)

[~~SELECT~~] [SELECT * FROM EMPLOYEES;]

[Select * FROM EMPLOYEES;]

[select * FROM EMPLOYEES;]

Above three statements are equal.

- SQL syntax depends on relational algebra and tuple relational calculus.
- SQL statements or syntax are dependent on text lines. We can place a single SQL statement on one or multiple text lines.

SOL Statement

SOL Statement tell the database what operation you want to perform on the structured data

[SELECT "Column-name" FROM "table-name";]

- each SOL Statement begins with any of the SOL keywords and ends with the semicolon (;).
 - Used for separating the Multiple SOL Statements which are going to execute in the same call.
- 1) SELECT: Retrieves data from one or more tables
 - 2) INSERT INTO: add new records to a table
 - 3) UPDATE: Modifies existing records in a table.
 - 4) DELETE: removes records from a table.
 - 5) CREATE Table: Creates a new table.
 - 6) ALTER TABLE: Modifies the structure of an existing table.
 - 7) ORDER By: Sorts the data in ascending or descending order based on one or more columns.
 - 8) GROUP By: groups data based on one or more columns and performs aggregate function like SUM, AVG, COUNT etc.
 - 9) JOIN: Combines data from multiple tables based on a common column or condition
 - 10) DROP TABLE: delete a table and its data from the database.
-

SQL Constraints

SQL constraints are used to specify rules for the data in a table.

- Constraints can be column level or table level
 - Column level → Constraints apply to column.
 - Table level → Constraints apply to whole table.
- Constraints can be specified:

1) When a table is created with the CREATE TABLE Statement.

2) we can use a ALTER TABLE Statement to create constraints even after the table is created. [of any more constraints add then use ALTER TABLE]

Syntax → CREATE TABLE table-name (

column1 datatype constraints,

column2 datatype constraints,

column3 datatype

.....

);

Constraints-name → NOT NULL, UNIQUE, PRIMARY KEY.

1) NOT NULL: NULL means empty, i.e., the value is not available.

- Whenever a table's column is declared as NOT NULL, then the value for that column cannot be empty for any of the table's records.

[NULL does not mean zero. NULL means empty column, not even zero.]

Syntax: Create table table-name (column-name-1 datatype-1 NOT NULL, column-name-2 data type-2);

Copyrighted by Codecap.io
All rights reserved.

Example: CREATE TABLE Student

```
(  
  ID int(6) NOT NULL,  
  NAME varchar(10) NOT NULL,  
  ADDRESS varchar(20)  
);
```

2) UNIQUE: Duplicate value are not allowed in the columns to which the UNIQUE constraint is applied.

- The column with the unique constraint will always contain a unique value.
- This constraint can be applied to one or more than one column of a table, which means more than one unique constraint can exist on a single table.
- Using the UNIQUE constraint, you can also modify the already created tables.

Syntax: Create table table-name (column-name1 data type-1 unique,
column-name-2 data type-2)

Example: CREATE TABLE Student

```
(  
  ID int(6) NOT NULL UNIQUE,  
  NAME varchar(10),  
  ADDRESS varchar(20),  
  PRIMARY KEY(ID)  
);
```

3) PRIMARY KEY: → Primary key constraint is a combination of NOT NULL and Unique constraints.

- NOT NULL constraint and a UNIQUE constraint together forms a PRIMARY constraint.
- The column to which we have applied the primary constraint will always contain a unique value and will not allow null values

Syntax: Create table table-name (column-name-1 datatype primary key, column-name-2);

Example: CREATE TABLE Student

```
(
  ID int (6) NOT NULL UNIQUE,
  NAME varchar (10),
  ADDRESS varchar (20),
  PRIMARY KEY (ID)
);
```

Copyrighted by CodeCap.io
All rights reserved.

4) FOREIGN KEY: A foreign key is used for referential integrity.

When we have two tables, and one table has reference from another table, i.e. the same column is present in both the tables and that column act as a primary key in one table.

Example: CREATE TABLE Customers (

```
ID INT NOT NULL,
NAME varchar (20) NOT NULL,
AGE int NOT NULL,
ADDRESS char (25),
PRIMARY KEY (ID)
);
```

```
CREATE TABLE orders
(
  ID INT NOT NULL,
  DATE DATETIME,
  CUSTOMER-ID INT REFERENCES CUSTOMERS(ID),
  AMOUNT double,
  PRIMARY KEY(ID)
);
```

Copyrighted by Codeleap.io
All rights reserved.

5) CHECK: Whenever a check constraint is applied to the table's column, and the user want to insert the value in it, then the value will first be checked for certain condⁿ before inserting the value into that column.

for ex: Create table table-name (column-name, datatype-1 CHECK (condition) - - - - -);

Example: CREATE TABLE Student

```
(
  ID int (6) NOT NULL,
  NAME varchar (10) NOT NULL,
  AGE int NOT NULL CHECK (AGE >= 18)
);
```

The above query creates a table Student and specifies the condⁿ for the field AGE as AGE >= 18. That is, the user will not be allowed to enter any record in the table with AGE < 18.

6) DEFAULT: Whenever a default constraint is applied to the table's column, and the user has not specified the value to be inserted in it, then

default value which was specified while applying the default constraint will be inserted into that particular column.

Syntax: Create table table-name (column-name 1, datatype, column-name 2 datatype-2, DEFAULT Value);

Example: CREATE TABLE Student
(
ID int (6) NOT NULL,
NAME varchar (10) NOT NULL,
AGE int DEFAULT 18
);

Example: CREATE TABLE Emp
(
EmpID int NOT NULL,
Name varchar (20) NOT NULL,
Age int NOT NULL,
Address varchar (100) DEFAULT
Salary numeric (10,2) NOT NULL,
PRIMARY KEY (Emp-ID),
FOREIGN KEY (Dept-ID), REFERENCES Dept (ID),
CHECK (Age >= 18)
);

SQL Syntax

Commands

Syntax

• SELECT Statement:

Select Statement used to retrieves information from a table.

Syntax:

```
SELECT column1, column2, ..., columnN  
FROM table-name;
```

• DISTINCT Clause

To eliminate all the duplicate records and fetching only unique records.

Syntax:

```
SELECT DISTINCT column1, col2, ...  
FROM table-name;
```

• IN Clause

Select values that match any value in a given list of value.

Syntax:

```
SELECT Column1, Column2, ..., N  
FROM table-name;
```

Where Column-name IN (val-1, val-2, ..., N)

• Between Clause

define a range of values that the column values must fall in to make the condⁿ true.

Syntax:

```
SELECT Column1, Column2, ..., N  
FROM table-name
```

```
WHERE Column-name BETWEEN val-1  
AND val-2;
```

• Like Clause

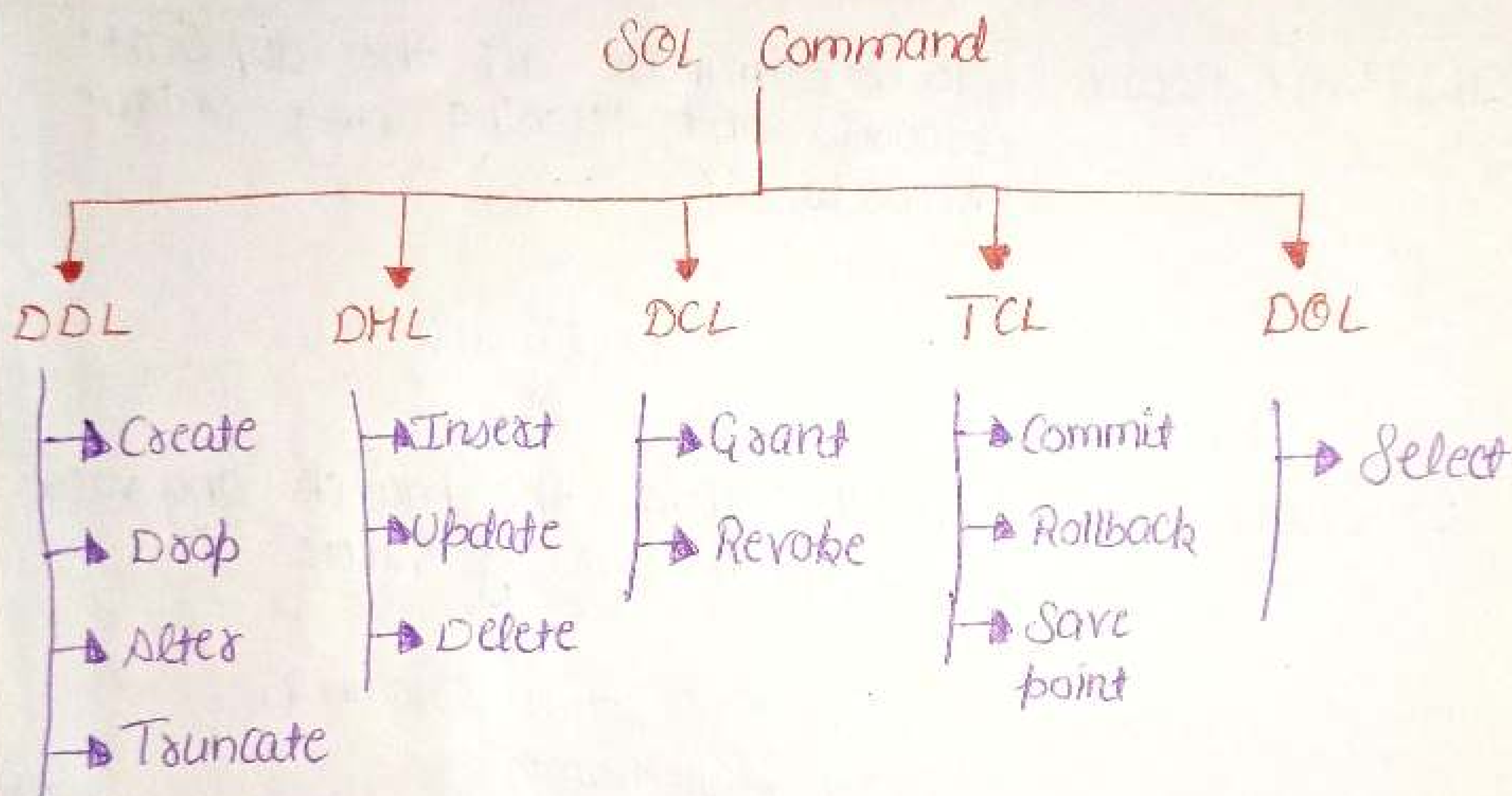
% , _ Operator

```
SELECT Column1, Column2, ..., N  
FROM table-name
```

```
WHERE Column-name LIKE {PATTERN};
```

SOL Commands

- SQL Commands are instructions that are used to communicate with the database.
- It is also used to perform specific tasks, functions and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop, Modify.



1) DDL : Data Definition language

- It is used for creating and modifying the database objects such as tables, indices, views.
- DDL Commands are used to define the structure and schema of the database.
- All the commands of DDL are auto-committed that means it permanently save all the changes in the database.

• Some Commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

1) CREATE: to create new table or database. When a table is created its column are name, datatype and sizes are supplied for each column, each table must have atleast one column.

• CREATE Statement can be used for

- Creating a database
- Creating a table

→ Creating a table

It specifies column names of the table, its data types (eg: varchar, integer, date etc) and can also specify integrity constraints (eg: primary key, foreign key, unique)

Syntax:
CREATE TABLE table-name
(
 column-name1 datatype,
 column-name 2 datatype,
 ...
 column-name n datatype
 (integrity constraints)
);

Example:

CREATE TABLE Emp (Emp-ID , int , data type)

attributes → Emp-ID int , → data type
Name varchar(20), → Size

Age int,

Address varchar(100), → [length is not fixed]

Salary numeric(10,2)

);

Output:

Emp-ID	Name	Age	Address	Salary

Note: DESC or DESCRIBE Statement:

To describe the details of the table structure.

Syntax: - DESC table-name;

Example: - DESC Emp; [Create database, datab-name;

• SHOW Statement

To see existing databases and tables.

• Show database;

• Show tables;

• USE Statement

To use or select any existing database.

Syntax: USE database-name;

Example: USE my db;

2) ALTER: To alter (modify) the structure of table.
• It is also used to add and drop various constraints on an existing table.

1) ALTER TABLE - ADD COLUMN

• for adding new column in a table

Syntax: ALTER TABLE table-name
ADD column-name(s) datatype; indicate add multiple data and column.

Example: ALTER TABLE Emp
ADD Email VARCHAR(100);

Emp-id	Name	Age	address	Salary	Email

Add new
column email

2) ALTER TABLE - DROP COLUMN

• for removing existing columns in a table

Syntax: ALTER TABLE table-name
DROP column-name(s);

Example: ALTER TABLE Emp
DROP Email;

Emp-id	Name	Age	address	Salary

∴ remove the above column
i.e E-mail.

3) ALTER TABLE - MODIFY COLUMN

To modify existing column in a table.

→ To change data type of any column or to modify its size.

Syntax: → ALTER TABLE table-name
MODIFY column-name datatype;

Example: ALTER TABLE Emp
MODIFY Name varchar(100);

It size is [20] → Check it with
change into desc command
[100] i.e Modify

4) ALTER TABLE - RENAME COLUMN

To rename an existing column in a table

Syntax: → ALTER TABLE table-name
RENAME old-column-name To new column;

Example: ALTER TABLE Emp.
MODIFY Address To Location;

Empid	Name	Age	Location	Salary

↓ [only Column Name
change
Address rename to Location.

3) DROP: } completely removes a table from the database.

- This command will destroy the table structure and the data stored in it. [delete table with structure]

Syntax: DROP TABLE table-name;

Example: DROP TABLE Emp;

- DROP command is also be used as Database.
- To delete the complete database

Syntax: DROP DATABASE database-name;

1) TRUNCATE: } used to remove all rows from a table. it is similar to the DELETE Statement with no WHERE clause.

[only delete the value not table structure]

Syntax: TRUNCATE TABLE table-name;

Example: TRUNCATE TABLE Emp;

Emp:

Emp-id	Name	Age	Location
1	A	32	Delhi
2	B	25	Pune



Truncate command

Emp id	Name	Age	Location

→ Structure present
→ only records delete

2) DML: Data manipulation language.

- DML commands are used for accessing and manipulating the data stored in the database.
- The DML commands are not auto-committed that means it can't permanently save all the changes in the database.
- They can be rollback

Copyrighted by Codecap.io
All rights reserved.

1) INSERT: The INSERT INTO Statement is used to insert new records in a table.

Syntax 1: Specify both the column values and the values to be inserted.

INSERT INTO table-name (column1, 2, 3, ... N)
VALUES (value 1, value 2, ... value N)

Syntax 2: Specify only values to be inserted.
But needed to remember the column order.

INSERT INTO table-name
VALUES (value 1, value 2, ... values N);

• Uses to Modification of database.

Example: Emp.

Emp-ID	Name	Age	Address	Salary
1	Rahul	32	Delhi	20000
2	Karan	25	Pune	30000
3	Kiran	23	Patna	30000
4	Mohi	26	Goa	45000

1st Syntax Specifies column

```
INSERT INTO Emp (Emp-ID, Name, Age, Address, Salary)
VALUES (1, 'Rahul', 32, 'Delhi', 2000);
```

2nd Syntax without column

```
INSERT INTO Emp VALUES (4, 'Chisag', 25, 'Mumbai', 600);
```

To Check

```
SELECT *
FROM Emp;
```

[To see table]

2) UPDATE: To modify the existing record in a table.

• always use the WHERE clause with the UPDATE statement to update the selected rows, otherwise all the rows would be affected.

Syntax

UPDATE table-name.

SET Column 1 = value 1, Column 2 = value 2 - - -

WHERE Condition;

Example: To update the address of Emp-ID: 3 to Chennai

```
UPDATE Emp
```

```
SET address = 'chennai'
```

```
WHERE Emp-ID = 3;
```

Copyrighted by Codeleap.io
All rights reserved.

Then Move on Emp table.

3	Karan	23	Chennai	2000
---	-------	----	---------	------

```
SELECT *
FROM Emp;
```

↓
Update.

3) DELETE } The DELETE Statement is used to delete existing records in a table.

• # Syntax: DELETE FROM table-name
WHERE Condition;

• Same WHERE Clause use.

Example: To delete the employee records of Pune location.

DELETE FROM Emp
WHERE Address = 'Pune';

To Check
= SELECT *
FROM Emp;

Copyrighted by Codecap.io
All rights reserved.

Emp-ID	Name	Age	Address	Salary
1	Rahul	32	Delhi	20000
3	Kiran	23	Patna	30000
4	Mani	26	Goa	45000

[CRUD] → Create, Read, Update, Delete]

3) DCL } Data Control language

• DCL command are used for controlling the user access in a database

• It is used to control privilege in Databases. i.e. it provide right to create a particular type of SQL.

ii) also gives right to connect the database and create a table in your Schema

- If you not have privilege not create table.
- Component of SQL.
- DDL Commands

1) GRANT: } Both are used in DDL and DML.

- Used to add permissions to users

Syntax:

GRANT <privilege list> | ALL
ON <object>
TO <user-list> | Public
[with GRANT OPTION];

Example:

GRANT ALL ON
Employee TO ABC;
[with GRANT OPTION];

Where:

Privilege list: } specifies the permission to be granted to users like ALTER, DELETE etc.

- All :-> Specifies all the permission.
- Object :-> Specifies name of table or their column.
- User-list :-> Specifies name of user Permission given.
- PUBLIC :-> Specifies all users.
- With GRANT OPTION :-> Specifies that the user in user.

3) REVOKE: Used to remove permissions from users.

Syntax:
REVOKE <privilege-list> | ALL
ON <table name> [Column - comma - list]
FROM <Users - list> | Public;

eg: REVOKE update on employee from
user(ABC);

4) TCL: Transaction Control language.

- TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.
- These operations are automatically committed in the database.
- The change in the database that we made using DML commands are either performed or rollbacked using TCL.

1) COMMIT: Used to save the change permanently.

Syntax: COMMIT;

2) ROLLBACK: Used to undo one or more changes.

Syntax: Rollback;

Copyrighted by Codeleap.io
All rights reserved.

3) SAVEPOINT: Used to save the changes temporarily so that it can be rolled back to a particular point.

Syntax: SAVEPOINT <savepoint-name>;

5) DQL: Data Query language.

- used to fetch the data from the data-base

1) SELECT: used to retrieve data from a database table.

- also called DQL because it is used to query information from a database table.
- used with different Conditions and CLAUSES

Syntax To retrieve selected field from the table

```
SELECT Column1, Column2, ... N  
FROM table-name(s);
```

Example:

Emp-ID	Name	age	address	Salary
1	A	32	Delhi	20000
2	B	25	Pune	15000
3	C	23	Pune	2000
4	D	25	Mumbai	6500

```
SELECT Emp-ID, Name  
FROM Emp;
```

↓

Emp-ID	Name
1	A
2	B
3	C
4	D

Copyrighted by Codeleap.io
All rights reserved.

Syntax: To retrieve all the field from table.

Select * → Means all value shows
from Emp; [SELECT *
FROM table(s);]

↓
Example: [left or complete table form]

# Difference <u>full form</u>	<u>DDL</u> Data Definition language	<u>DML</u> Data manipulation language.
<u>Basic</u>	used to create the database schema.	used to manipulate database
<u>Commands</u>	CREATE, ALTER, DROP etc.	Select, INSERT, UPDATE etc.

# <u>Difference</u> Basis	<u>ALTER</u> ALTER Command is a DDL command.	<u>UPDATE</u> UPDATE Command is a DML Command.
<u>Purpose</u>	ALTER add, delete, modify the attributes of relations in the tab database.	UPDATE Command modify one or more records in the relations.
<u>values</u>	initializes values.	specified value
<u>Syntax</u>	—	Copyrighted by CodeCap.io All rights reserved.

DELETE and DROP

Difference	DELETE	DROP
Basic	DELETE remove some or all the tuple from a table.	Can remove entire schema from the database.
Language	DML Command	DDL Command.
Clause	WHERE Clause used	No Clause used.
Roll-back.	Can be roll-backed action perform	Can-not be roll-backed action perform.

Delete and Truncate

Diff	DELETE	TRUNCATE
Language	DML	DDL
Basic	Specify the tuple that you want to delete.	Deletes all the tuples from a relation.
Speed	Slower	Faster
Restore	Followed commit or rollback.	Can't be restore.
Syntax	— —	—

Copyrighted by Codecap.io
All rights reserved.

Basic Query Structure

- SQL is based on set and relational operation with certain modifications.
- A typical SQL query has the form:

Select $A_1, A_2, \dots, A_n$
from $\delta_1, \delta_2, \dots, \delta_n$
where φ

- A_i represents attributes
- δ_i represents relations i.e table.
- φ is a predicate (or condⁿ)
- This SQL query is equivalent to the relational algebra expression.

$$\pi_{A_1, A_2, \dots, A_n}(\sigma_{\varphi}(\delta_1 \times \delta_2 \times \dots \times \delta_m))$$

Clauses - helps to specify a condⁿ on the column of a table.

- Clauses are in-built functions available to us in SQL. With the help of clauses, we can deal with data easily stored in the table.
- Clauses help us filter and analyze data quickly. When we have large amount of data stored in the database, we use clauses to query and get data required by the user.
- Some of the examples of clauses are:→
Where, and, or, like etc.

1) GROUP BY: GROUP BY Statement is used for organizing similar data into groups.

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN) to group the result-set by one or more columns.

→ GROUP BY clause is used with SELECT statement

→ GROUP BY clause is placed after WHERE clause ^{in SQL query}

→ GROUP BY clause is placed before ORDER BY clause in SQL

Syntax SELECT ^{Same} column-name (s), aggregate-function (column-name)

FROM table-name

[WHERE condition]

GROUP BY ^{Same} column-name (s)

[ORDER BY column-name (s)];

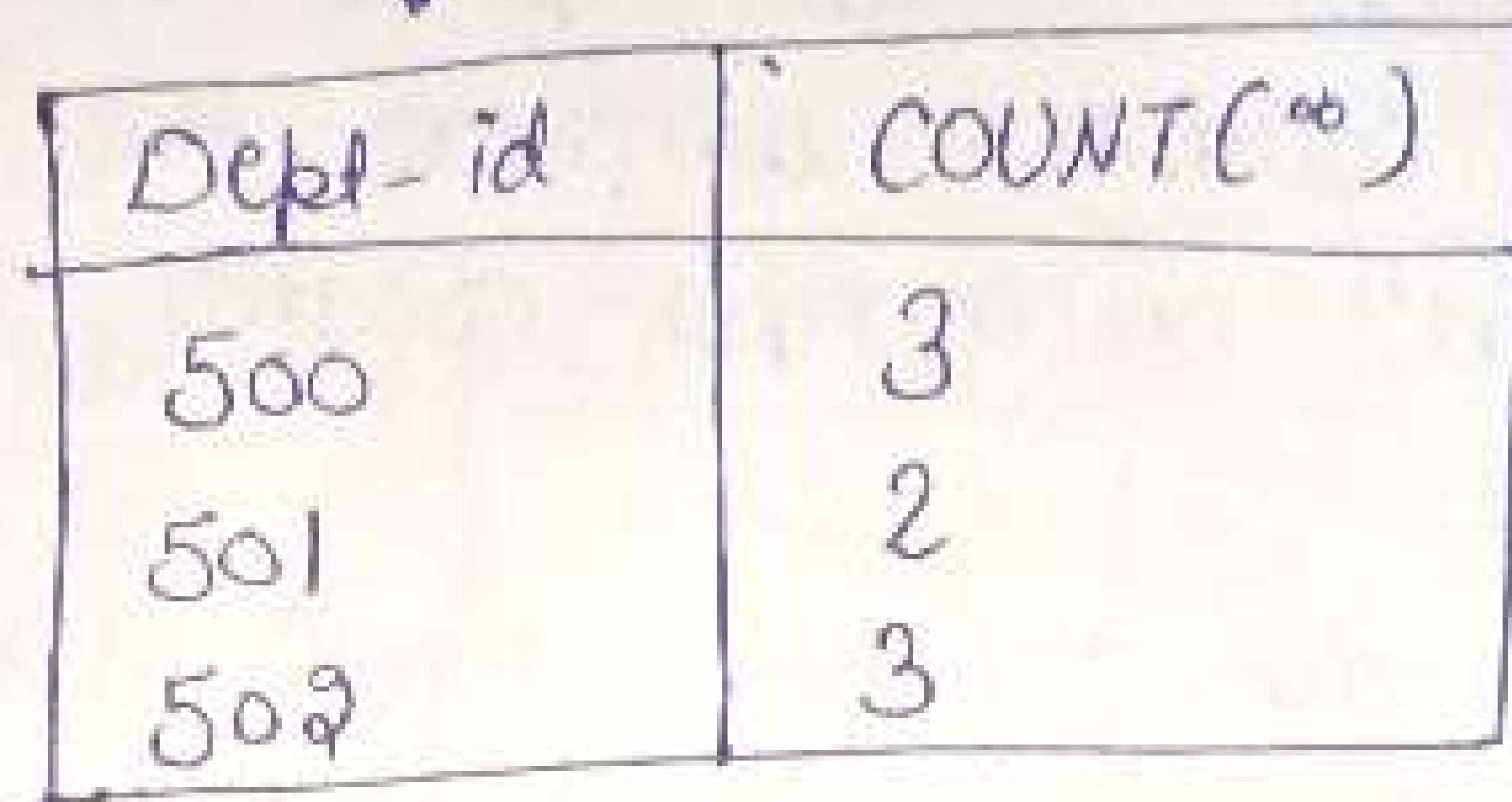
Example:

Emp-ID	Name	Age	Dept-ID
1	A	32	500
2	B	25	500
3	C	23	501
4	D	25	502
5	E	24	500
6	F	29	502
7	G	30	501
8	H	31	502

• (Group By with COUNT aggregate function)

→ SELECT Dept-id, COUNT(*)
FROM Emp
GROUP BY Dept-id

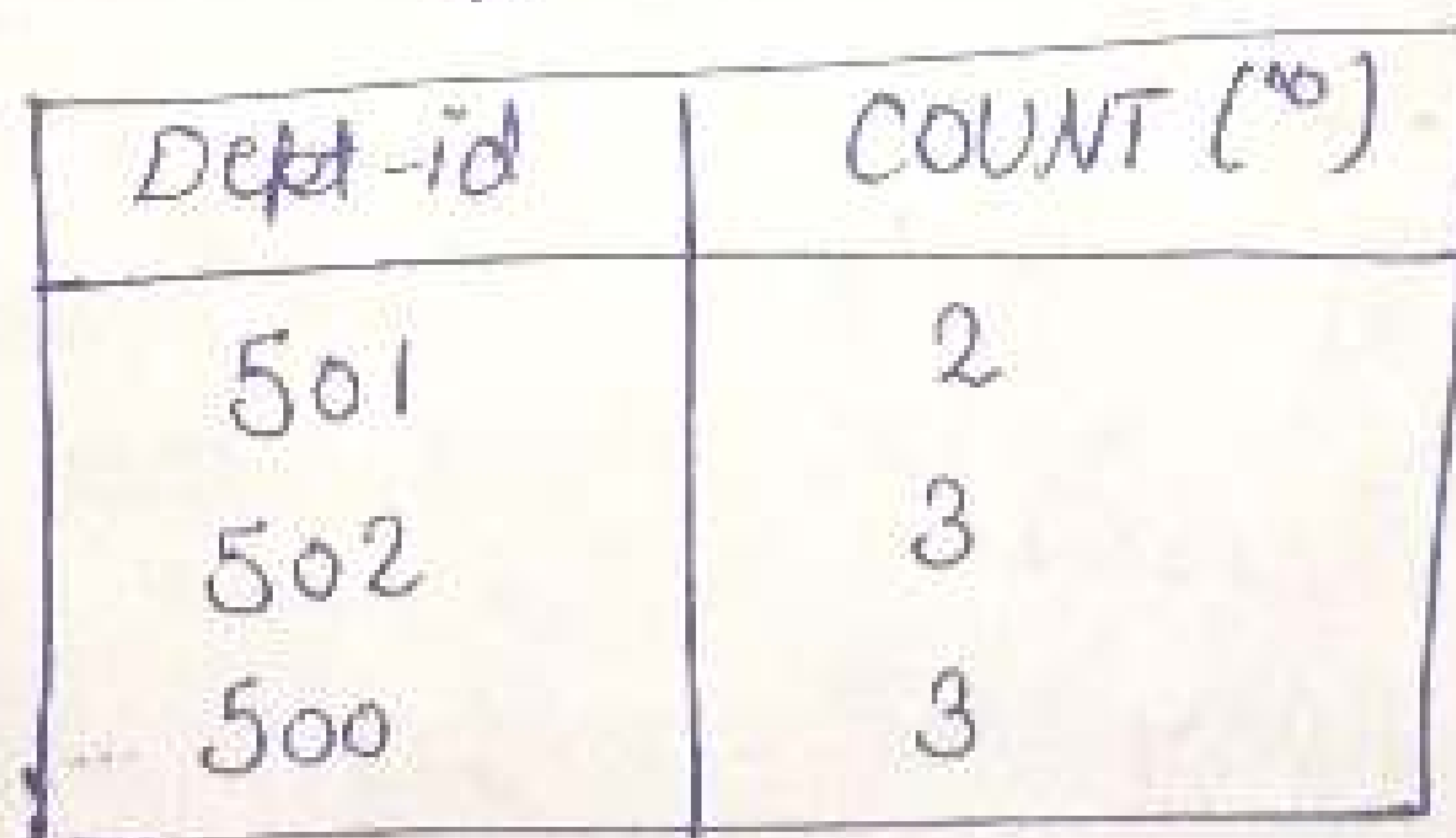
Copyrighted by Codecap.io
All rights reserved.



Dept-id	COUNT(*)
500	3
501	2
502	3

• GROUP BY with COUNT and ORDER BY

→ SELECT Dept-id, COUNT(*)
FROM Emp
GROUP BY Dept-id
ORDER BY COUNT(*);



Dept-id	COUNT(*)
501	2
502	3
500	3

2) HAVING : HAVING Clause is used to specify a search condⁿ for a group or an aggregate.

Copyrighted by Codecap.io
All rights reserved.

[
• HAVING Clause apply on groups
• WHERE Clause apply on complete table.]

→ HAVING Clause is placed after the GROUP BY Clause.

→ The WHERE Clause place condⁿ on the Selected Column, whereas the HAVING Clause places condⁿ on groups created by the GROUP BY Clause.

[Have groups, apply condⁿ, which group satisfy the condⁿ, that group result show]

Syntax:

```
SELECT Column name (s), aggregate-function ( )  
FROM table-name  
[WHERE Condition]  
GROUP BY Column-name (s)  
HAVING Condition  
[ORDER BY Column-name (s)];
```

Order of Clauses

- 1) SELECT
- 2) FROM
- 3) WHERE [apply on complete table]
- 4) GROUP BY → apply condⁿ
- 5) HAVING
- 6) ORDER BY

Copyrighted by CodeLeap.io
All rights reserved.

Example:

```
SELECT Dept-id, COUNT(*)  
FROM Emp  
GROUP BY Dept-id  
HAVING COUNT(*) > 2;
```

[Not apply where
because we need
only groupwise
not in complete
table]

↓

Dept ID	COUNT(*)
500	3
502	3

3) ORDER BY:

The output of the SELECT queries do not have any specific order. The ORDER BY clause allows us to specify order for the query output.

- ORDER BY Clause is used with SELECT Statement for arranging retrieved data in sorted order.
- The ORDER BY clauses by Default sort the retrieved data in ascending order [ASC]
- The sort data in descending order, [DESC] keyword is used with ORDER BY Clause.

Syntax:

```
SELECT Column-list  
FROM table-name  
[WHERE conditions]
```

```
ORDER BY Column1, Column2 .. ColumnN
```

```
[ASC/DESC];
```

Example Table = Emp

Emp-ID	Name	Age	Salary
1	A	32	2000
2	B	25	1500
3	C	23	6500
4	D	26	8500
5	E	22	4500

- Eg: Ascending order - Default [by Salary]

SELECT *
FROM Emp
ORDER BY Salary;

add Multiple column
[Name]
Primary ordering
Secondary ordering

1st column
Primary ordering
2nd by Secondary
ordering

Emp-ID	Name	Age	Salary
1	B	25	1500
2	A	32	2000
5	E	22	4500
3	C	23	6500
4	D	26	8500

- Descending order by name

SELECT *
FROM Emp
ORDER BY Name DESC;

Copyrighted by Codecap.io
All rights reserved.

4) WHERE CLAUSE is used to select a particular record based on a condition. It is used to filter records.

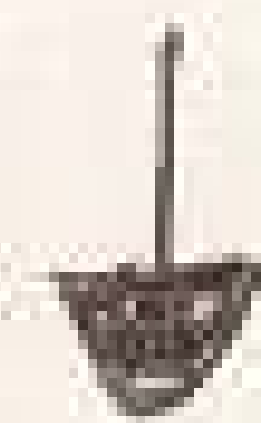
- WHERE Clause is used to specify a condition while fetching the data from a single table or by joining with multiple conditions.
- The WHERE Clause is not only used by SELECT Statement, but it is also used in the UPDATE, DELETE Statement.

Syntax: "SELECT with WHERE Clause"

```
SELECT column1, Column 2, ... Column N  
FROM table-name(s)  
WHERE - condition;
```

Example: To retrieve emp-id, Name from emp table whose salary is greater than 2000.

```
SELECT Emp-ID, Name  
FROM Emp  
WHERE Salary > 2000
```



Emp-ID	Name
5	E
3	C
4	D

5) DISTINCT Clause?

The SELECT DISTINCT Statement is used to return only distinct (different) values or simply it is used to return only unique values.

- DISTINCT Clause is used to remove duplicates from result set of a SELECT Statement.

Syntax: SELECT DISTINCT column 1, column 2 ... N
FROM table-name;

Ex: SELECT Without DISTINCT
Previous Table use.

SELECT Address
FROM Emp;

Address
Delhi
Pune
Pune
Mumbai
Patna

• SELECT With DISTINCT

SELECT DISTINCT Address
FROM Emp;

Address
Delhi
Pune
Mumbai
Patna

→ only unique values show.

AND operator

- WHERE Clause can be combined AND operator.
- It is used to combine multiple condⁿ with WHERE clause, in SELECT, UPDATE, DELETE statements.
- All the condⁿ must be satisfied for a record to be selected. [True]

Syntax:
SELECT Column1, Column2, ... ColumnN
FROM table-name
WHERE Condition 1 = "value" AND Condition 2 = "value" ... AND Condition N
Ex: Salary > 200 AND Age < 25;

OR operator

- C1 or C2 are True condⁿ.
- Any one of the condⁿ must be satisfied with record to be selected.

Syntax:
SELECT Column1, Column2, ... ColumnN
FROM table-name
WHERE Condition 1 OR Condition 2 ... ;
eg. Salary > 2000 OR Age < 25;

NOT operator

- It is used to negate the condⁿ with the WHERE clause, in SELECT, UPDATE or DELETE statements.
- It displays a record if the condⁿ(s) is NOT TRUE. [NOT IN, NOT between]

Syntax:
SELECT Column1, Column2, ... ColumnN
FROM table-name
WHERE NOT Condition;

NOT address = 'Humbol';

LIKE operator

- The LIKE operator is used to WHERE Clause to search for a specified pattern in a column.
- The LIKE operator is used to match string pattern values using two wildcard characters.
- Wildcard Characters: It is used to substitute one or more characters in a string.
 - The percent sign (%) :- It represents zero, one or multiple characters.
 - The underscore (-) :- It represents only a single character.

Syntax

```
SELECT column1, column2, ---  
FROM table-name.  
WHERE column LIKE pattern;
```

Ex: It select all employees with a Name starting with "K"

```
SELECT * FROM Emp.  
WHERE Name LIKE "K%";
```

→ with a Name that have 'a' in 2nd pos.

```
SELECT * FROM Emp.  
WHERE Name LIKE "_a%";
```

SOL operators → used to perform various operation on data in SOL statement

The SOL reserved words and Characters are called operators, which are used with a WHERE Clause in a SOL query.

Operators are used to specify Condⁿ in an SOL Statement.

[retrieve, merge data in database

- 1) Arithmetic operators
- 2) Comparison operators
- 3) Logical operators.

1) Arithmetic operators → It perform the mathematical operation on the numerical data of the SOL tables.

- 1) SOL Addition operator (+)
- 2) SOL Subtraction operator (-)
- 3) SOL Multiplication operator (*)
- 4) SOL Division operator (/)
- 5) SOL Modulus operator (%)

1) SOL Addition operator (+): It is used to perform addition operation on the data items, items include either single column or Multiple Columns.

SELECT column1 + column2 AS SUM FROM table name

Syntax: operand 1 [+] operand 2;

Example:
SELECT emp-id, emp-name, salary, salary + 100
AS "Salary + 100" FROM addition;

Output:

emp id	emp-name	salary	salary + 100
1	A	25000	25100
2	B	25000	25100
3	C	52000	52100

→ operation applied on single column.

2) Subtraction operator: Subtract right hand operand from left hand operand.

• Implementation

```
SELECT emp-id, emp-name, salary, salary - 100  
AS "Salary - 100" FROM subtraction;
```

Example same not salary minus.

3) Multiplication operator: It is use to perform multiplication of data.

```
# SELECT emp-id, emp-name, salary, salary * 100  
AS "Salary * 100" FROM addition;
```

5) Division operator: Divides left hand operand by right hand operand.

Same implementation "Salary / 100"

6) Modulus operator: It is use to get remainder when one data is divided by another.

Salary / 25000 < Take one data from table

2) Comparison operators

The comparison operators in SQL compare two different data of SQL table and check whether they are the same, greater and lesser.

The SQL comparison operators are used with the WHERE clause in the SQL queries.

1) (=) Equal

2) (!=) Not equal

3) (>) Greater than

4) (>=) Greater than or equal to

5) (<=) less than or equal to

6) (<) less than

Compare two values and return a boolean value (True, false)

Copyrighted by Codecap.io
All rights reserved.

1) Equal operator $\Rightarrow$ (=).

This operator is highly used in SQL queries.

Check if the values of two operands are equal or not if yes then condⁿ becomes true.

Syntax

SELECT * FROM TABLE-NAME WHERE
ATTRIBUTE Condition-Operator Given-values

Ex $\Rightarrow$

Emp-Id	Emp Name	Emp Salary
201	A	3000
202	B	4000
203	C	3000
204	D	3000

• SELECT * FROM Salary WHERE Emp-Salary = 3000

Output:

Emp-id	Emp-Name	Emp-Salary
201	A	3000

2) (!=): } Check if the value of two operands are equal or not, if values are not equal then condⁿ becomes True.

Syntax: SELECT * FROM Emp-details WHERE Salary != 40;

3) (>): } Greater than, Check if the value of two operand is greater than the value of right operand, if yes then condⁿ becomes true

Syntax: SELECT * FROM Emp-details WHERE Salary > 3000

Output:

Emp-id	Emp-Name	Emp-Salary
202	B	4000

4) (<): } Less than, Check if the value of two operand is greater than the value of right operand, if yes then condⁿ become true.

Query: SELECT * FROM Emp-details WHERE Salary < 40k.

Output: less than 40 Salary employees show

5) $\geq$ } Check if the value of left operand is greater than or equal to the value of right operand. If yes then condⁿ becomes true.

Query: `SELECT * FROM Emp-details WHERE Salary >= 40k;`
Ex: also include 40k value or other greater value

6) $\leq$ } Same definition.

• `SELECT * FROM Emp-details WHERE Salary <= 40k;`

Ex: Same above concept

3) Logical operators } The logical operators are those that are true or false. They return true or false values to combine one or more true or false values.

1) ALL [Compare a value to all value in another value set]

2) AND [Both T and f Boolean Table]

3) OR

4) NOT [only one Boolean expression Table]

5) ANY

6) BETWEEN

7) IN

8) EXISTS

9) SOME

10) UNIQUE [No duplicacy]

Copyrighted by CodeCap.io
All rights reserved.

String functions

SQL String functions are the performed pre-defined functions that allow the database user for string manipulation.

- String functions are used to perform an operation on input string and return an output string.

1) ASCII () :-> Return the ASCII value for the specific character.

Syntax: ASCII ('A');

Output -> A -> 65

2) Char-length () -> Returns the length of a string (in characters).

Syntax: Select Char-length ('Tanna');

Output = 5 [also count space]

3) Concat () :-> Add two or more expression together

Syntax :-> Select Concat ('Tanna', 'Dhanda');

Output :-> TannaDhanda

4) Concat_ws () :-> Add two or more expression together with a separator.

Syntax: Concat_ws ('_', 'Tanna', 'Dhanda');

Output: Tanna - Dhanda.

5) field () , find-in-set () , format () , Insert ()

Set operations

The SQL Set operation is used to combine the two or more SQL SELECT statements.

- 1) Union
- 2) Intersect
- 3) Minus

1) Union :- } The SQL Union operation is used to combine the result of two or more SQL SELECT queries.

It eliminates the duplicate rows.

Syntax :-
SELECT column-name FROM table1
UNION
SELECT column-name FROM table2;

Ex:->

Id	Name
1	A
2	B
3	C

1st table

ID	Name
3	C
4	D
5	E

2nd Table

SELECT * FROM first
UNION
SELECT * FROM second;

ID	Name
1	A
2	B
3	C
4	D
5	E

Copyrighted by Codecap.io
All rights reserved.

3) Intersect: It is used to combine two SELECT Statement. The intersect operation returns the common rows from both the SELECT Statements.

• Syntax:
SELECT column-name FROM table1
INTERSECT
SELECT column-name FROM table2;
• Show unique value.

4) Minus: It combines the result of two SELECT Statement. Minus operator is used to display the rows which are present in the first query but absent in the second query.

• Syntax:
SELECT column-name FROM table1
MINUS
SELECT column-name FROM table2;

Ex:

Id	name
1	A
2	B

Aggregate functions

Aggregate functions performs calculations on set of values and return a single values.

• Aggregate functions ignore NULL values, except Count(*).

- 1) COUNT()
- 2) SUM()
- 3) AVG()
- 4) MAX()
- 5) MIN()

Copyrighted by Codecap.io
All rights reserved.

Aggregate functions are often used by GROUP BY clause of the SELECT statement to group multiple rows together as input to form a single value output

1) COUNT() function: Count function returns the number of rows that matches the specified criterion.

Syntax: SELECT COUNT (column name) [∞ $\rightarrow$ all data]
FROM table-name
WHERE condition;

eg: $\Rightarrow$ Total no. of student records

```
SELECT COUNT( $\infty$ )
FROM Student;
```

↓

COUNT(∞)
6

ID	Name	Marks
1	A	90
2	B	40
3	C	70
4	D	80
5	E	70
6	F	NULL

- ignore NULL value
- Count of entered marks.

```
SELECT COUNT (Marks)
FROM Student;
```

→

COUNT (Marks)
5

2) SUM(): SUM function returns total sum of a selected numeric columns.

Syntax:
SUM (column - name)
FROM table - name
WHERE condition;

Ex: Sum of Student marks

SELECT SUM(Marks)
FROM Student;

→

SUM(Marks)
330

• If apply (Distinct) → Duplicate value don't take.

3) AVG(): Average function returns the average value of selected numeric column.

Syntax:
SELECT AVG (column-name)
FROM table - name
WHERE Condition;

Ex: → average mark of student

SELECT AVG(Marks)
FROM Students;

→

AVG(Marks)
66

$$AVG(Marks) = SUM / COUNT(Marks)$$

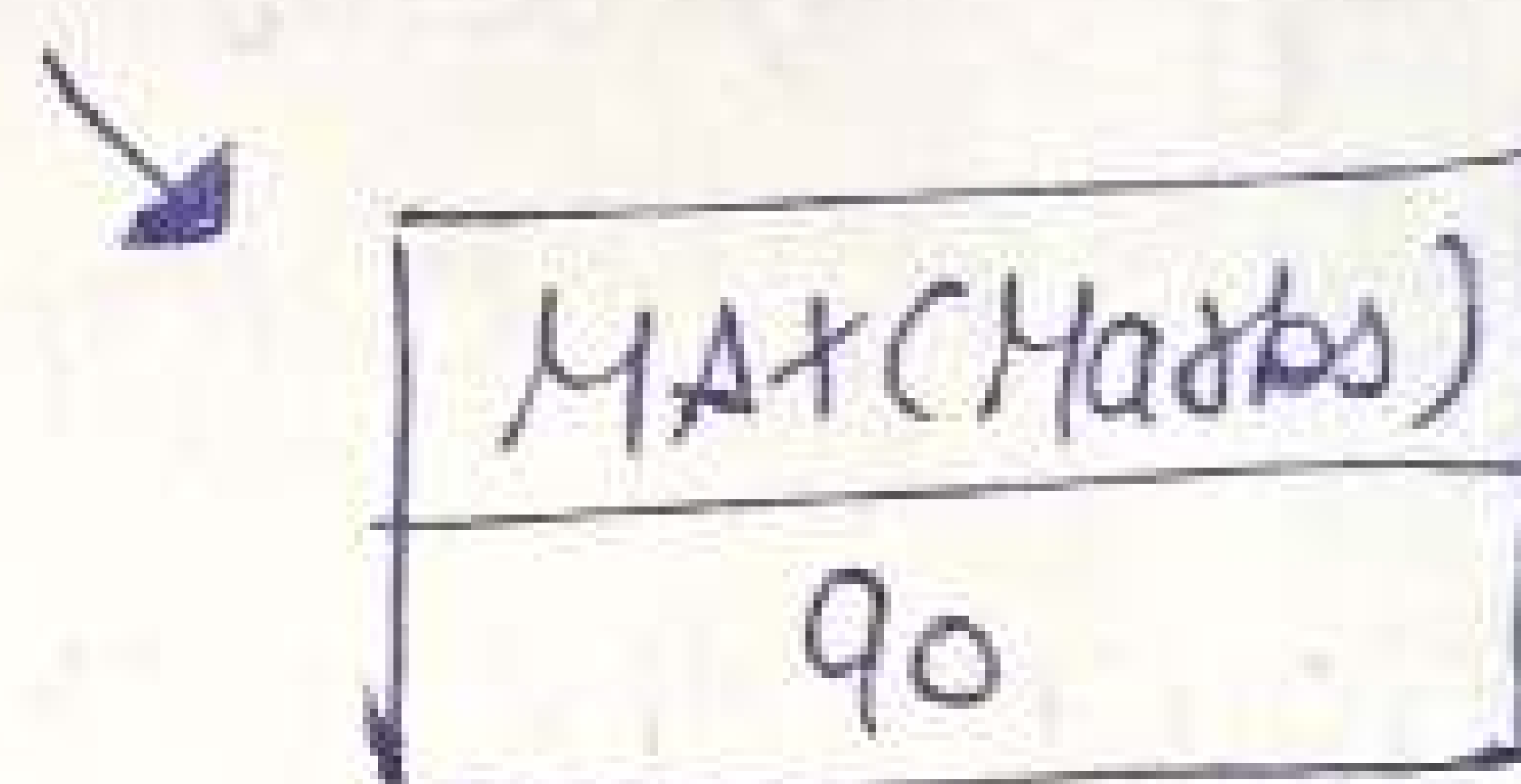
$$= \frac{330}{5} = 66$$

4) MAX(): } Maximum function returns maximum values of selected column.

Syntax:
SELECT MAX (column-name)
FROM table-name
WHERE Condition;

Example: SELECT MAX (marks)
FROM Student;

Copyrighted by Codeleap.io
All rights reserved.



MAX(Marks)
90

5) MIN(): } Minimum function returns minimum value of selected column.

Syntax:
SELECT MIN (column-name)
FROM table-name
WHERE Condition;

Example: SELECT MIN (Marks)
FROM Student;

6) SQRT(): } It is used to find out the square root of any number.

Syntax:
SELECT SQRT (NUMBER);

Example:
SELECT SQRT(25);

Indexes → one or more column of a table and contain a sorted list of value

- The index in SQL is a special table used to speed up the searching of the data in the database tables.
- The index concept in SQL is same as the index concept in the noraal or a book.
- It is the best SQL technique for improving the performance of queries.
- The drawbacks of using indexes
They slow down the execution time of UPDATE and INSERT statements.

• Advantage =

they speed up the execution time of SELECT and WHERE statements.

→ How it is created

CREATE INDEX Statement is used to create indexes in tables and to indicate wheather the index is in ascending or descending order.

Syntax

CREATE INDEX index_name ON Table-name (column);

→ Index name → name of that index that we want to create.

→ Table-name → name of the table on which the index is to be created.

→ Column-name → represents the name of the column on which index is to be applied.

- If we want to create an index on the combination of two or more columns, then the syntax:

CREATE INDEX Index-name ON Table-Name (column(s));

- Create UNIQUE INDEX

Unique index is the same as the primary key in SQL. The unique index does not allow selecting those columns which contain duplicate values.

- It maintains data integrity.

Syntax:

CREATE UNIQUE INDEX index name ON Table Name;

→ When should indexes be created

- A column contains a wide range of values.
- Does not contain a large number of null values.

- One or more columns are frequently used together in a where clause.

• When should indexes be avoided

- The table is small.

- The columns are not often used as a condition in the query.

- The column is updated frequently.

Types of SQL indexing

- 1) Clustered index
- 2) Non-Clustered index

1) Clustered index:-> Clustered index is the index that will arrange the rows physically in the memory in sorted order.

- When Primary key is created automatically the Clustered Index is created.
- We can create only one clustered on a table.

Example:
CREATE TABLE Employee-Index
(
Emp-ID INT,
Emp-name VARCHAR(100),
Emp-Sal int
)

Insert data into table.

INSERT INTO Employee-index VALUES (1, 'Tannu', 1000)

INSERT INTO " " " (3, 'Tanu2', 1050)

INSERT INTO " " " (7, 'Anu', 1570)

Select * from employee-index

output

Emp id	Name	Sal
1	Tannu	1000
3	Tan 2	1050
2	Tn	1570

→ Not create
a Primary key
Then data is
unsorted order

Now create Cluster index

Create CLUSTERED index ix - Employee - index on Employee -
index (Emp-ID)

Then, data is come into sorted order

1	Tan	1000
2	T	1050
3	S	1570

← output

2) Non-Clustered : When a Unique key is
created automatically, when Non-Clustered
index is created.

A non-Clustered index is an index that
will not arrange the rows physically in
the memory in sorted order.

We can create one or more Non-Clustered
index in a table.

Ex Same above example
Not sorted element

Copyrighted by Codecap.io
All rights reserved.

Then use un-Clustered index

Create NonClustered index ix - table name on table
-name (column-name ASC)

after apply Non-Clustered index order
is not sorted.

DROP INDEX Command

Sy: DROP INDEX index-name;

Ex: DROP INDEX abc;

When performance may be slowed or
improved.

1) Single-Column Indexes:- Create only one table column.

Syntax:- CREATE INDEX index-name
ON table-name (column-name);

2) Unique Indexes:- Not used for performance,
but also used for data integrity.
- Not allow duplicates values.

Syntax: CREATE INDEX index-name
ON table-name (column-name);

3) Composite Indexes:- is an index on two or
more columns of a table.

Syntax: CREATE INDEX index-name
ON table-name (column1, column2);

4) Implicit Indexes:- that are automatically
created by the database server
when an object is created.

Tables in SQL

Table is a collection of data, organized in
terms of rows and columns.

- In DBMS, table is known as relation

Syntax: CREATE table "table-name"
(
 "Column1" "data type",
 "Column2" "data type",
 ...
 "ColumnN" "data type");

Copyrighted by CodeCap.io
All rights reserved.

Ex: employee table

Emp-Name	Address	Salary
T	Lucknow	15K
B	Allahabad	18K
C	Delhi	20K

• In the above table, "Employee" is the table name, "EMP-NAME", "ADDRESS" and "SALARY" are the column names.

• The combination of data of multiple columns form a row.

eg:→ ('A', 'Delhi' and 20K) are the data of one row

• The SQL Table variable is used to create, modify, rename, copy and delete tables.

Type of Tables in SQL Server

- 1) User Tables (Regular Tables)
- 2) Local Temporary Tables
- 3) Global Temporary Table.
- 4) Table variable

Views

Any relation that is made visible to a user as a "virtual relation" is called a view.

• It provides limited access to DB.

• Views are the virtual relations, through which a selective portion of data from one or more relation (tables) can be seen.

• Uses of View

- 1) It helps in query processing
- 2) to hide data complexity.
- View are stored in the data dictionary in the tables called USER-VIEWS.

→ Define

A view is defined using the create view statement.

Create view V as <query expression>

- In SQL:
- View are kind of virtual tables.
 - A views also has rows and columns as they are in a real table in the database.
 - We can create a view by selecting fields from one or more tables present in the database.

Tables

Student Details

S-ID	Name	Address
1	ABC	Delhi
2	CDE	Noida
3	EFG	Punjab
4	GHI	Bihar

• not a physical table, but's logical repⁿ of data from one or more table.

Student Marks

Same
columns

Marks	Age
90	19
80	13
79	19
70	21

Creating Views

- We can create view using CREATE VIEW Statement.
- A view can be created from a single table or multiple tables

Syntax

```
CREATE VIEW view-name AS  
SELECT column1, column2 . . . .  
FROM table-name  
WHERE condition;
```

view-name: Name of the view

table-name: Name of table

condition: condⁿ to select rows.

→ view creating from a single table

- Create a view named Details View from the table Student Details.

Query:
CREATE VIEW Details View AS
SELECT Name, Address,
FROM Student Details
WHERE S-ID < 2;

- To see the data in the view, we can query the view in the same manners.

```
SELECT * FROM Detail view;
```

Output:

Name	Address
ABC	Delhi
CDE	Noida

→ Creating View from multiple tables

- Two tables [Student Details and Student Marks]

Query → CREATE VIEW Marks View AS
SELECT StudentDetails.Name, StudentDetails.Addr,
Student.Marks

FROM StudentDetails, StudentMarks
WHERE StudentDetails.Name = StudentMarks.Name;

- To display data of View Marks View:

SELECT * FROM Marks View;

Output:

Name	Address	Marks
ABC	Delhi	90
CDE	Noida	20

Deleting Views

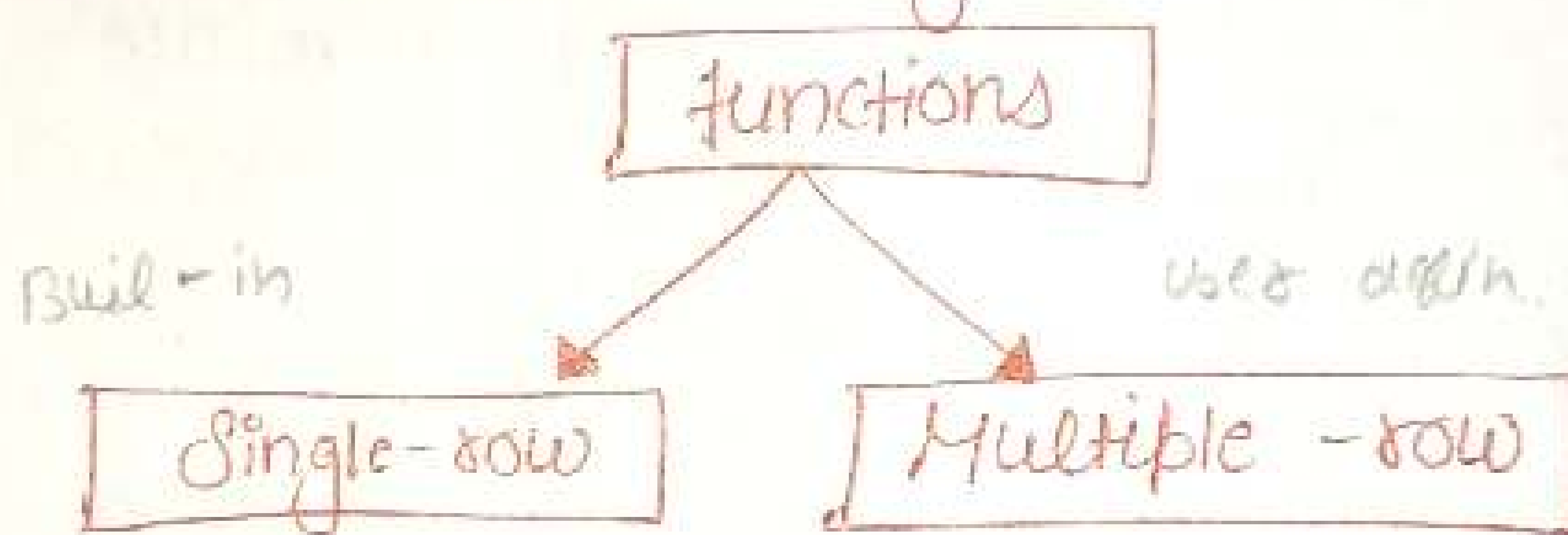
In SQL, we can delete or drop a view using the DROP Statement.

Syntax: DROP VIEW view-name;

view-name → which you want to delete.

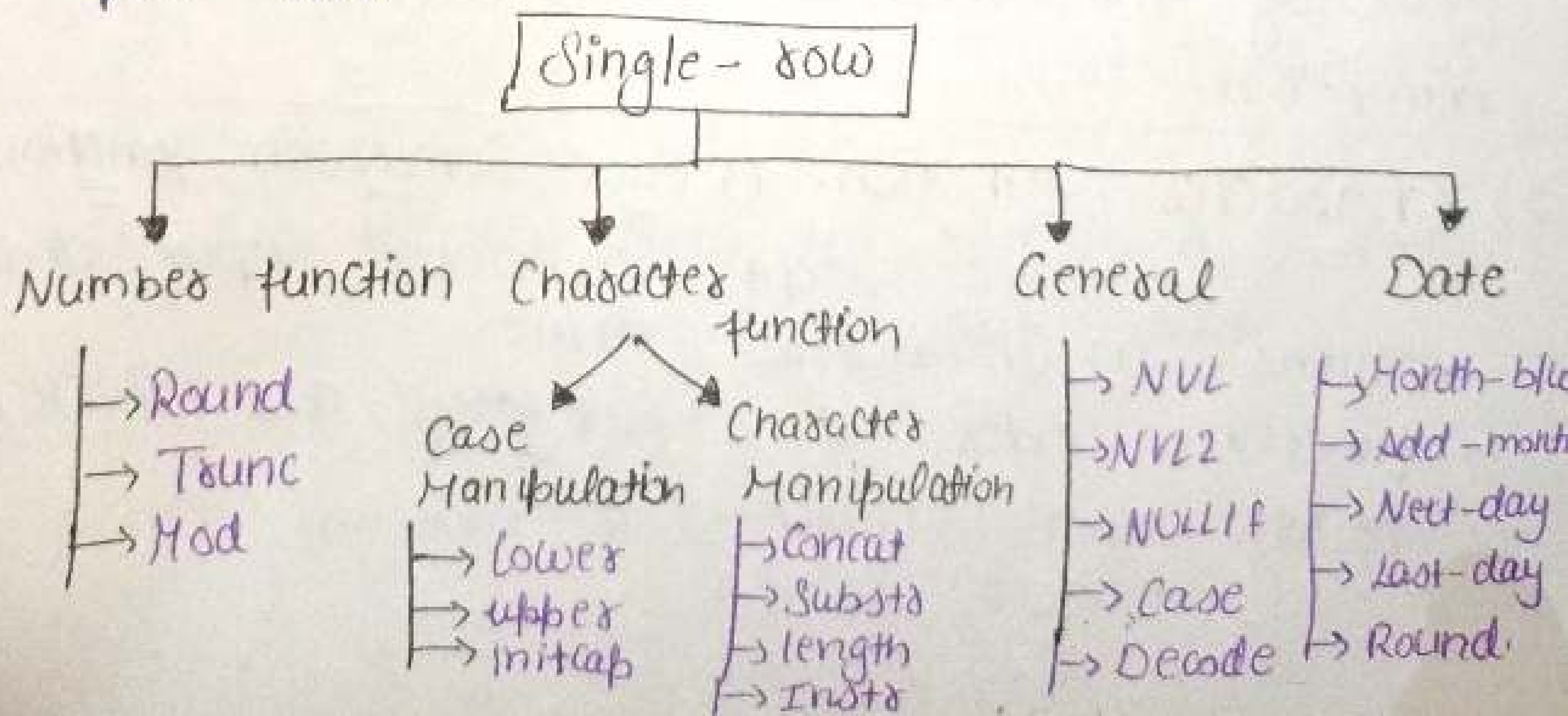
functions in SQL

- functions are the object in SQL which take one or more input from user, take operation on input and produce one or more output.
- A function is a special type of predefined command set that performs some operation and returns a single value.
- function operates on zero, one, two or more values that are provided to them. The values that are provided to functions are called **parameters** and **arguments**.



1) Single-row functions :

Single row functions are the one who work on single row and return one output per row.



- Single row functions can be Character functions, numeric functions, data functions and conversion functions. [Used to manipulate data items]
- Single row function can be used in SELECT Statement, WHERE and ORDER BY clause.

• Single row functions

1) Number functions: Accepts numeric input and returns numeric values.

• function under the category are ROUND, TRUNC and MOD

• ROUND and TRUNC function are used to round and truncate the number value.

eg:→ SELECT ROUND (1372.427)

FROM dual;

ROUND (1372.427)

1372.5

• MOD is used to return the remainder of the division operation b/w two numbers.

2) Character function: 1) Case Conversion function Accepts character input and returns a character value.

• function under the category are UPPER, LOWER and INITCAP

Copyrighted by CodeCap.io
All rights reserved.

• UPPER function: Converts a string to upper case

eg:→ SQL COURSE → SQL COURSE
[string] [upper function]

• lower function: Converts a string to lower case

eg:→ SQL Course → sql course

• INITCAP function: Convert first letter to each word to uppercase and remaining letters to lowercase.

eg:→ SQL Course → Sql Course

2) Character functions: Accept character input and return number or character value.

• functions are CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD.

• CONCAT function: Joins value together

eg:→ ('Good', 'string') → Goodstring

• SUBSTR function: return a portion of a string from a given start point to an end point.

• LENGTH function: Show the length of a string as a numeric value

eg:→ ('string') → 6

• INSTR function: finds the numeric position of a named character

eg:→ ('string', 's') → 3

3) General function :- Usually contains NULL handling functions. The functions under NVL, NVL2, NULLIF, CASE, DECODE

4) Date function :- Date arithmetic operations return date or numeric values. Function under the category are MONTHS - B/w, ADD, MONTHS, NEXT - DAY, LAST - DAY.

- MONTHS - BETWEEN function returns the count of months b/w the two dates.
- ADD - months function add 'n' number of months to an input date.
- NEXT - DAY function returns the next day of the date specified.

2) Multi-Row functions :- These functions are also called "Aggregate functions" or group functions.

- All these row functions will process multiple records.
- Applied on group of records and returns one value from the entire group.

→ Aggregate functions

- 1) MAX()
- 2) MIN()
- 3) AVG()
- 4) COUNT()

Copyrighted by Codecap.io
All rights reserved.

Nested Queries

- also known as subquery and inner queries
- The subquery or inner query is an SQL query placed inside another SQL query. A query is written inside a query. The result of inner query is used in execution of outer query.
- It is embedded in the HAVING or WHERE clause of the SQL statements.

• Important rule

1) The SQL subqueries can be used with the following statement along with the SQL expression operators.

- SELECT Statement
 - UPDATE Statement
 - INSERT Statement
 - DELETE Statement
- [operators like $=, <, >, >=$ etc]

2) The subqueries in SQL are always enclosed in the parenthesis and placed on the right side of the operators.

⇒ These are mainly two type of nested queries

1) Independent Nested Queries : In independent nested queries, the execution order is from the innermost query to the outer query.

- operators such as IN, NOT IN, ALL are used to write independent nested queries.

IN → If a column value in the outer query's result is present in the inner query's result

- NOT IN: Checks if a column value in the outer query's result is not present in the inner query result

Executed last

→ Select id, Name

FROM Employees

WHERE ID IN (SELECT Emp-ID FROM AWARDS)

Nested Query

[Executed first]

2) Co-related Nested Queries: the inner query uses the values from the outer query to execute the inner query for every row processed by the outer query.

Syntax

Copyrighted by CodeCup.io
All rights reserved.

SELECT Column-name

FROM table-name

WHERE Column-name IN, NOT IN expression operator

(SELECT Column-name from table-name WHERE...);

Ex: → Select name, rollno, dept → Column name

FROM Student → Table Name

WHERE rollno IN → expression operator

(Select rollno from Student WHERE dept = 'CSE');

column

table name

cond'n

name	roll	dept
x	1	CSE
y	2	IT
z	3	CSE
w	4	IT



name	roll no	dept
x	1	CSE
z	3	CSE

PL/SQL: PL/SQL [Procedural Language / Structured Query Language] is a programming language used to create and manage database appl in Oracle database.

- Procedural Language extensions to the Structured Query Language by adding control structure like loops and conditional statement.
- Oracle uses a PL/SQL engine to process the PL/SQL statements.
- PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements ^{which mean that code is organized into blocks}.
- PL/SQL includes procedural language elements like conditions and loops.

⇒ functionalities

- PL/SQL is not case sensitive so you are free to use lower case letter or upper case letters. Except within string and character literals.

- A line of PL/SQL text contains group of characters known as lexical units

- Delimiters
- Literals
 - variable
 - control statement
- Identifiers
- Comments
 - cursor
 - exception handling

- /~~oo~~ and ~~oo~~/ → Multiline comment
- - - → Single line comment

- := assignment

- ; to indicate end of line

Copyrighted by CodeCap.io
All rights reserved.

SQL use aggregate function

PL/SQL used to handle errors exception

- SQL is a single query that is used to perform DML and DDL operations.
- It is declarative, that defines what need to be done, rather than how things need to be done.
- Execute as a single statement.
- Mainly used to manipulate data.
- Cannot contain PL/SQL code in it.
→ used to retrieve, insert, update data from database.

- PL/SQL is a block of codes that used to write the entire program blocks, function etc.
- PL/SQL is procedural that defines how the things needs to be done.
→ write code that specifies how you want the database
- Execute as a whole block.
- Mainly used to create an application.
- It is an extension of SQL, so it can contain SQL inside it.
→ can to create stored procedure, function, trigger that can be executed.

Advantage of PL/SQL

- Support for SQL code reusability → no need to write code again again
- Performance code engine, looping statement, SQL statement, process multiple → error handling
- High security
- High portability
- High productivity
- Support object oriented programming concepts
- PL/SQL is tightly integrated with SQL.
- It supports the development of web application
- It offers a variety of programming structure

Disadvantages

- Requiring specialized knowledge
- Being limited to Oracle databases
- It uses high memory for storage
- Lacks debugging functionality

• More complex and difficult to learn and read.

• More difficult to debug.

Block/Structure/Genesis in PL/SQL

- The basic unit in PL/SQL is a block.
- All PL/SQL programs are made up of blocks, which are nested within each other.

PL/SQL Block

— Declare
— Begin
— Exception
— End.

Copyrighted by Codecap.io
All rights reserved.

- A PL/SQL Block is defined by the keywords `DECLARE`, `BEGIN`, `EXCEPTION`, `END`
- Typically, each block performs a logical action in the program.
- All statements are classified into units that is called blocks.
- PL/SQL blocks are of two types

Anonymous

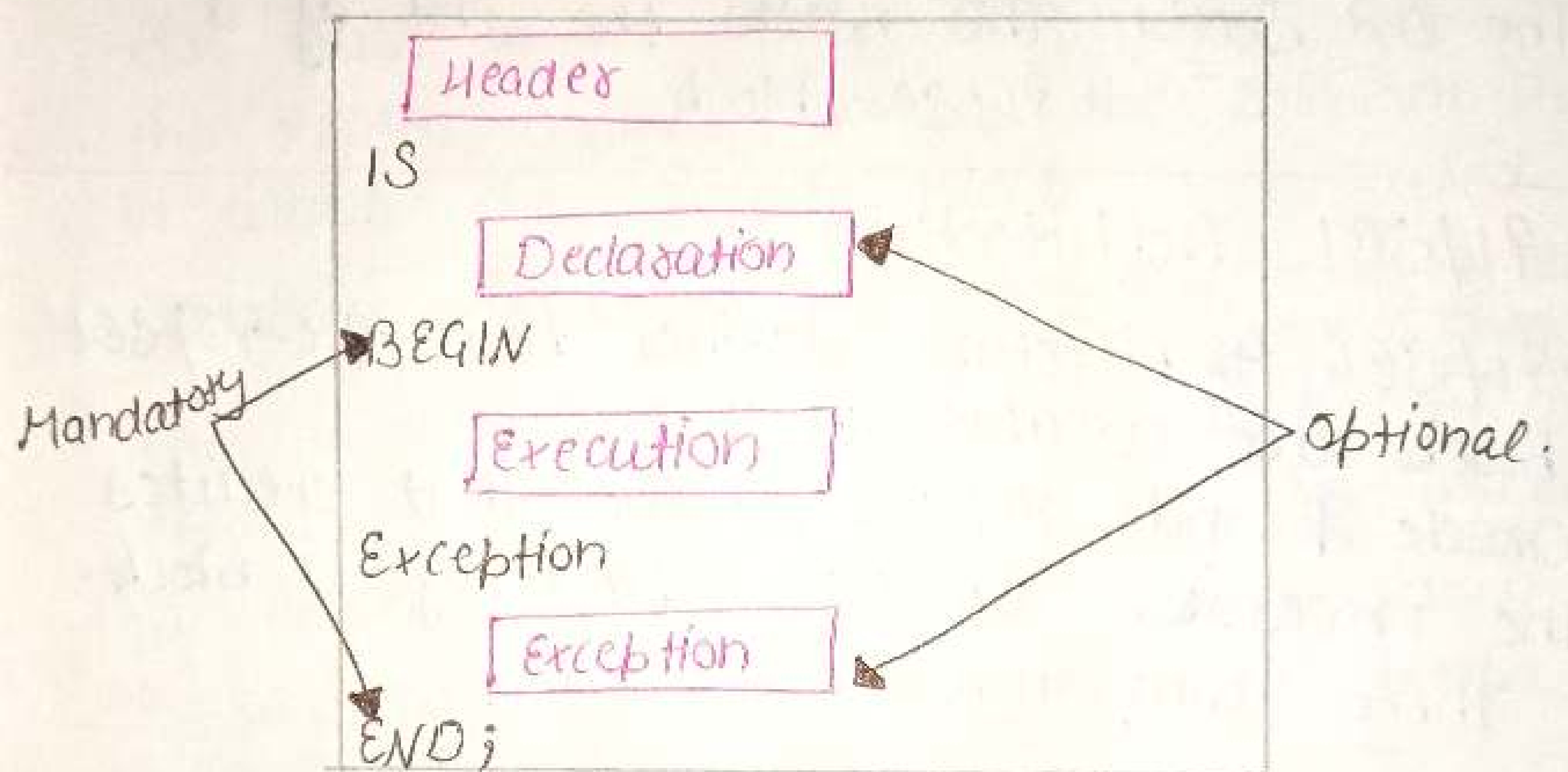
Named block

1) Anonymous Block: Block which is not have header are known as anonymous block.

- These block do not form the body of a function or triggers or procedure.

2) Named Block: Block which having header or labels.

- These blocks can either be subprograms like functions, procedures.



• Declaration Section: This is the first section of the PL/SQL blocks and optional part.

- Code block start with a declaration section, in which memory variables, constants, cursors and other oracle objects can be declared.

- This section starts with keyword 'DECLARE'

- Begin Section: This section starts with the keyword 'BEGIN'.

- Execution part is the main and mandatory part which actually executes the code that is written inside it.

- Exception Section: This section deals with handling errors that arise during execution of data manipulation statements which make up PL/SQL code blocks.

- This is an optional section of PL/SQL blocks.

- The End Section: This marks the end of a PL/SQL block.

PL/SQL Architecture

- PL/SQL Architecture defines how the PL/SQL blocks are executed in Oracle.

- Oracle provides PL/SQL engine that executes the procedural language part of a block.

- Three Components

- 1) PL/SQL Block

- 2) PL/SQL Engine.

Oracle engine

- 3) Database Server.

1) PL/SQL Block: This is the component which has the actual PL/SQL code.

- It also contains the SQL instruction that is used to interact with the

Database Server

- All the PL/SQL units are treated as PL/SQL Blocks, and this is the starting stage of the architecture which serves as the primary input.

- 2) PL/SQL Engine . When a PL/SQL program is compiled, the PL/SQL engine compiles the code.
- PL/SQL engine is the component where the actual processing of the codes takes place.
 - PL/SQL engine separates PL/SQL units and SQL part in the unit.
 - the PL/SQL engine provides the environment in which the PL/SQL Block is executed.

- 3) Database Server → includes SQL engine, which processes SQL statements and data which stores information about database objects, tables, views.
- This is the most important component of PL/SQL unit which stores the data.
 - The PL/SQL engine uses the SQL from PL/SQL units to interact with the database server.

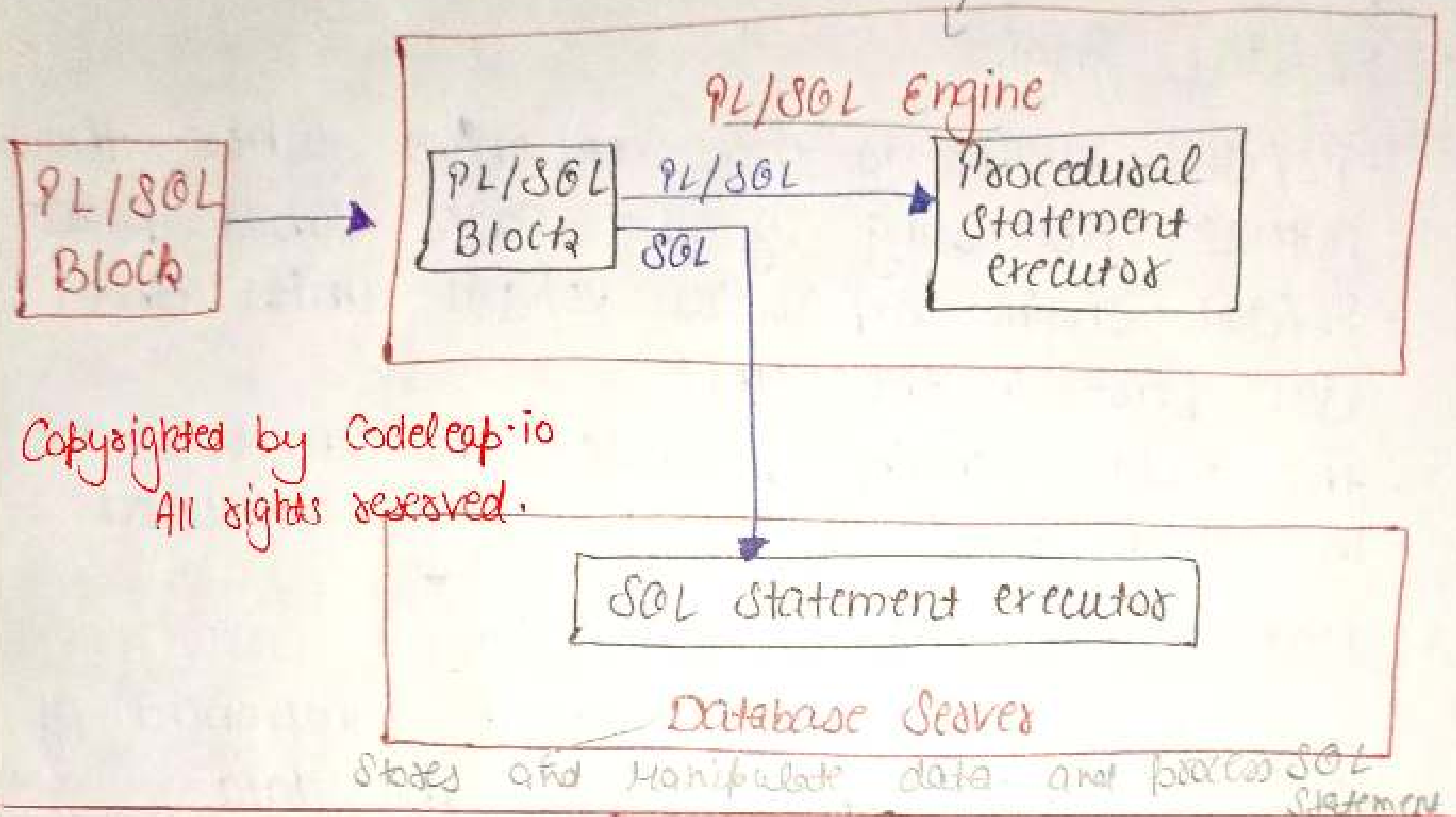
- It consists of SQL executor which passes the input SQL statement and execute the same.

```
PL/SQL Block
DECLARE
  Declaration Statement;
BEGIN
  SQL Statement;
  Procedural Statement;
EXCEPTION
  Exception handling Statement;
END;
```

- When a PL/SQL program is created, the PL/SQL engine sends SQL statements to the Oracle database engine to retrieve or manipulate data.

PL/SQL Architecture

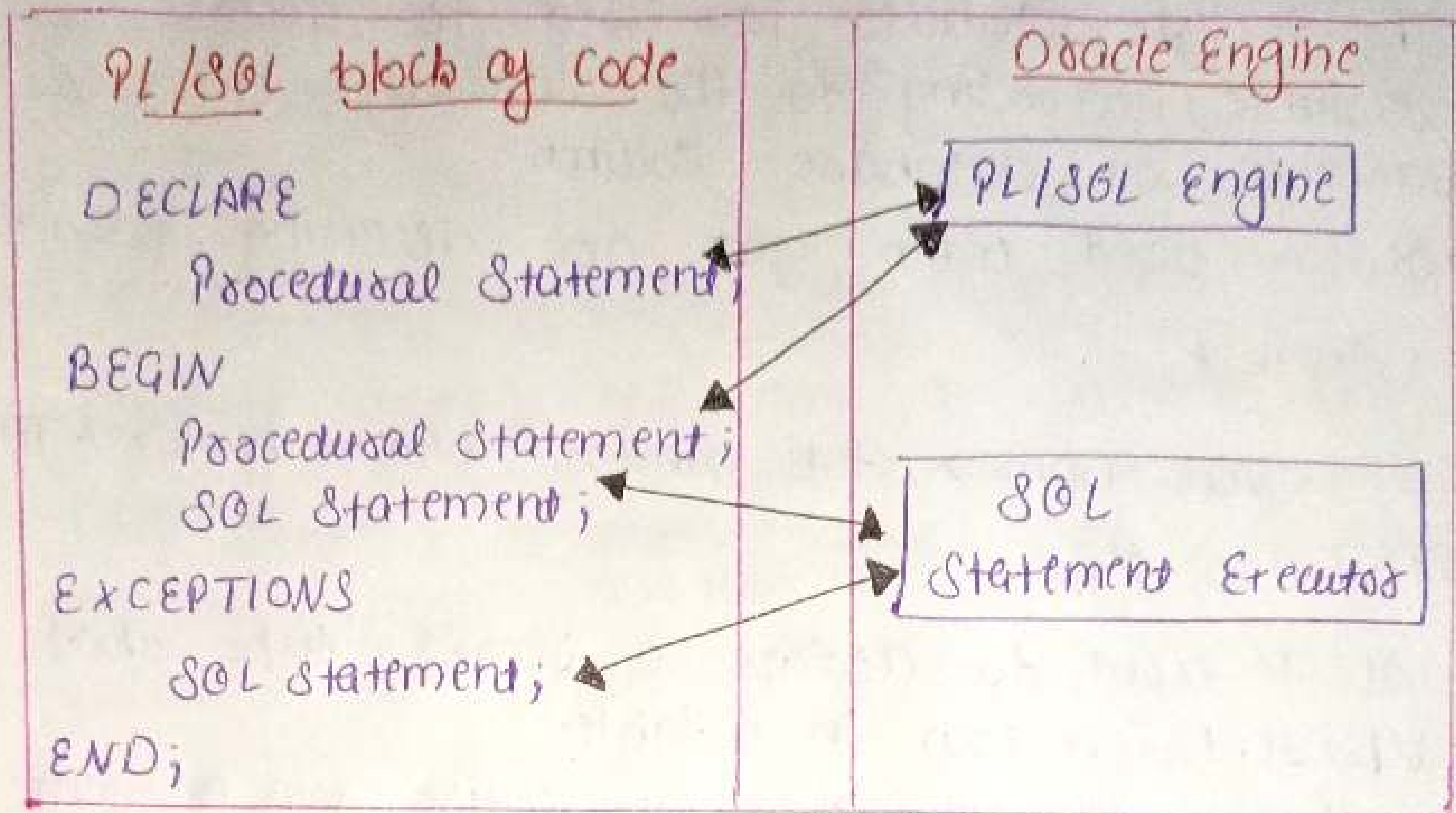
Executes PL/SQL Code



PL/SQL Execution Environment

- PL/SQL technology is required the PL/SQL engine accepts any valid PL/SQL block as input where PL/SQL code is executed. It includes the PL/SQL engine, which compiles and executes the code.
- PL/SQL in the Oracle Engine. The Oracle database engine stores the code. PL/SQL engine resides in the Oracle Engine, the Oracle engine can process not only single SQL statement but also entire PL/SQL blocks.
- These blocks are sent to the PL/SQL engine where procedural statements are executed. And SQL statements are sent to the

SQL executor in the Oracle Engine.



• The PL/SQL Execution Environment

• It explain how these statement are executed and how convenient it is to bundle SQL code.

Oracle PL/SQL engine, Oracle engine and PL/SQL runtime. These components work together to form

Variable Attribute a powerful and flexible platform
are used to store data and
manipulate value in program logic

• When we declare PL/SQL variables to hold column values, it must be of correct datatype and precision, otherwise error will occur during execution. So PL/SQL provides the facility to declare variable without having to specify a particular datatype

1) % Type

2) % Row Type

[:=] [index value
operator clause]

% → attributes indicator

1) % Type Attribute → same data type

The % Type Attribute is used to declare variable according to the already declared variable or database column.

• It is used when you are declaring individual

→ Syntax

`<var_name> <tab_name> . <column_name> % TYPE;`

2) % Row Type → same structure as a specified table row

• It is used to declare a record type that represents a row in a table.

• The record can store an entire row or some specific data selected from the table.

→ Syntax:

`<var_name> <tab_name> . ROW % TYPE;`

→ example: Employee Emp. % ROW TYPE;

Operators in PL/SQL

- Arithmetic [+ , - , / , *]
- Relational [True or False]
[= , != , < , > , <= , >=]

Copyrighted by Codecap.io
All rights reserved.

- Logical [AND , OR , NOT]
- Comparison [Like , Between , IN , IS NULL]

Constants

A constant is a value used in a PL/SQL block that remains unchanged throughout the program.

• Syntax

constant-name CONSTANT datatype := VALUE;

- Constant-name: is the name of constant just like variable name. The constant word is a reserved word and its value does not change.
- VALUE: It is a value which is assigned to a constant when it is declared. It cannot be assigned later.

Data types

1) Scalar Type : {

• Number

• Character and String Types

• Boolean

• Date, Time.

2) LOB Type : {

• NCLOB

• CLOB

• BFILE

• BLOB

Copyrighted by CodeCup.io
All rights reserved.

3) Reference Type : {

REFS

• REF Cursors

4) Composite Type : { Element of Array

Cursors

- When the SQL statement is processed, Oracle creates a memory area known as context area.
- A cursor is a pointer to this context area.
- It contains all information needed for processing the statement.
- In PL/SQL, the context area is controlled by cursor.

[help in SQL row-wise work.
[used in programming language.

• Two types of Cursors

[Implicit Cursors]

[Explicit Cursors]

1) Implicit Cursors: The implicit cursors are automatically generated by Oracle while an SQL statement is executed.

- These are created by default to process the statement when DML statement like INSERT, UPDATE, DELETE etc. are executed.
- Oracle provides some attributes known as implicit cursor's attributes to check the status of DML operation.

- They are used to retrieve and manipulate a single row of data.

→ % FOUND

→ % NOT FOUND

→ % ROW COUNT

→ % ISOPEN

returns

→ No. of Row effected

by DML Statement

so returned by SELECT

in To Statement

1. FOUND $\Rightarrow$ It returns value is TRUE if DML Statements like INSERT, DELETE and UPDATE affect at least one row or more rows or a SELECT INTO Statement returned one or more rows. Otherwise it returns FALSE.

1. NOT FOUND $\Rightarrow$ Its return value is TRUE if DML Statement like INSERT, DELETE and UPDATE affect no row or a SELECT INTO Statement return no rows. Otherwise it return FALSE. [opposite of found]

1. IS OPEN $\Rightarrow$ It always returns FALSE for implicit cursors, because the SQL cursor is automatically closed after executing its associated SQL Statement.

1. ROWCOUNT $\Rightarrow$ It returns the number of rows affected by DML Statement like INSERT or returned by select into Statement.

Implicit

- Closed automatically after execution
- Less efficient

Explicit

- Need to be defined explicitly by the user.
- Need to close after execution.
- More efficient

2) Explicit Cursors

- The explicit cursors are defined by the program.
- To gain more control over the context area.
- These cursors should be defined in the declaration section of the PL/SQL block. It is created on a SELECT statement which returns more than one row.

Syntax

CURSOR cursor-name IS select-statement;

Steps

- 1) Declare the cursor: for initializing the memory.
 - It defines the cursor with name and the associated SELECT statement.

Syntax → CURSOR name IS
SELECT statement;

- 2) Open the cursor for allocating the memory.
 - Used to allocate memory for the cursor.

Syntax → OPEN cursor-name;

- 3) Fetch the cursor for retrieving the data.
 - Used to access one row at a time.

Syntax → FETCH cursor-name INTO variable-list;

- 4) Close the cursor to release the data.

• Used to release the allocated memory.

• Syntax → CLOSE cursor-name;

Syntax:

```
DECLARE Variables;  
records;  
create a cursor;  
BEGIN  
OPEN cursor;  
FETCH cursor;  
process the records;  
CLOSE cursor;  
END;
```

Copyrighted by Codecap.io
All rights reserved.

Example:

ID	Name	Age	Address
1	R	23	Delhi
2	S	22	Konpur
3	M	24	Ghaziabad
4	C	25	Noida

• Create Procedure

DECLARE

C-id customers.id % type;
C-name customer.name % type;
C-addr customers.address % type;

→ to retrieve customer name and address

we return more than one row.

CURSOR C-customer is

SELECT id, name, address FROM customer;

BEGIN

OPEN C-customer;

LOOP

FETCH C-customer into C-id, C-name, C-addr;

EXIT WHEN C-customer % notfound;

dbms-output.put -> line (C-id || ' ' || C-name || ' ' ||
C-addr);

END LOOP;

CLOSE C-customers;

END;

/

Output: 1 R Delhi
= 2 S Kanpur
3 M Ghaziabad
4 C Noida.

SQL/86L procedure successfully completed.

Implicit Cursor Example

ID	Name	Age	Address	Salary
1	A	28	~	30000
2	B	29	~	32000
3	C	30	~	34000

• To update the table and increase salary of each customer by 5000.

DECLARE

total - rows number (2);

BEGIN

UPDATE customers

SET salary = salary + 5000;

IF SQL%FOUND NOT THEN

dbms_output.put_line('no customer update');

ELSE IF SQL%FOUND THEN

total-rows := SQL%ROWCOUNT;

dbms_output.put_line('total rows ' || customers updated');

END IF;

END;

SQL%FOUND
Can be used
SQL%ROWCOUNT

Output 3 Customers updated.

Select * FROM Customers;

IN	Name	Age	Address	Salary
1	A	29	%	25000
2	B	30	%	27000
3	C	30	%	29000

Triggers in PL/SQL → Used in SQL database

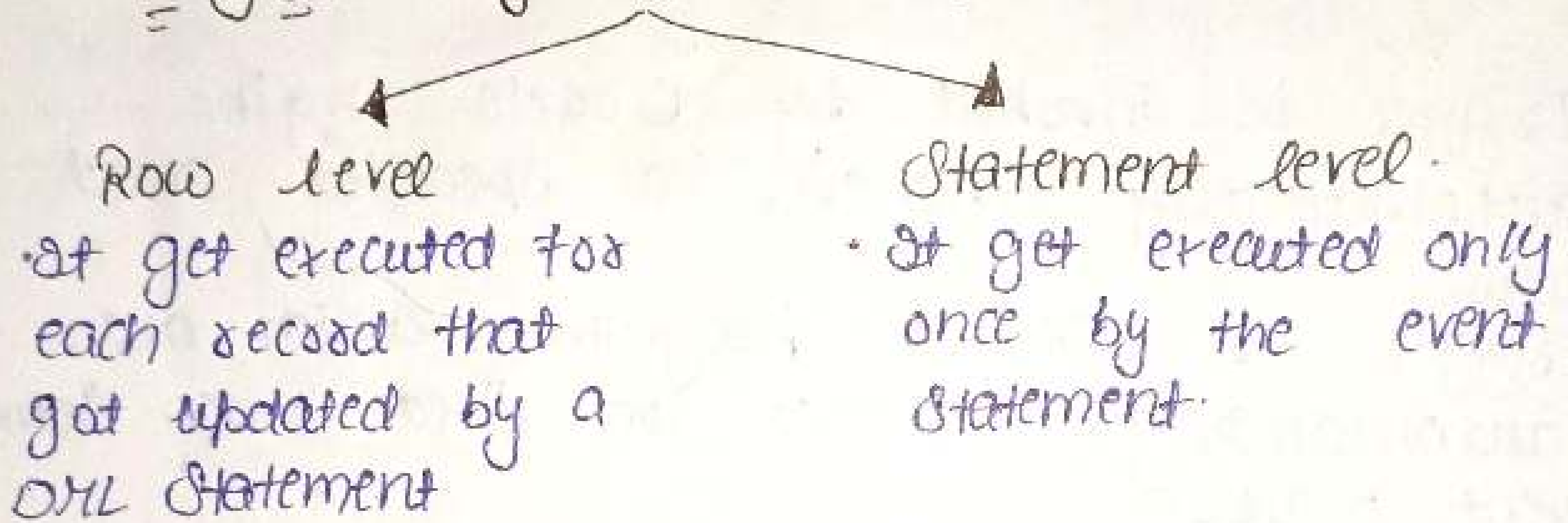
- Trigger is invoked by Oracle engine automatically whenever a specified event occurs.
- Trigger are stored programs, which are automatically executed or fired when some event occurs.
- Trigger are written to be executed in response to any of the following events.
 - a database manipulation DML Statement (INSERT, UPDATE, DELETE)
 - a Database definition DDL Statement (CREATE, ALTER, DROP)
 - a database operation (LOGON, LOGOFF, STARTUP and SHUTDOWN)
- Triggers could be defined on the table, view, schema or database with which the event is associated.

Copyrighted by Codeleap.io
All rights reserved.

Advantages

- Trigger generates some desired column values automatically.
 - Auditing
 - Enforces referential integrity
 - Event logging and storing information on table access
- Data integrity
• Performance
• Automation

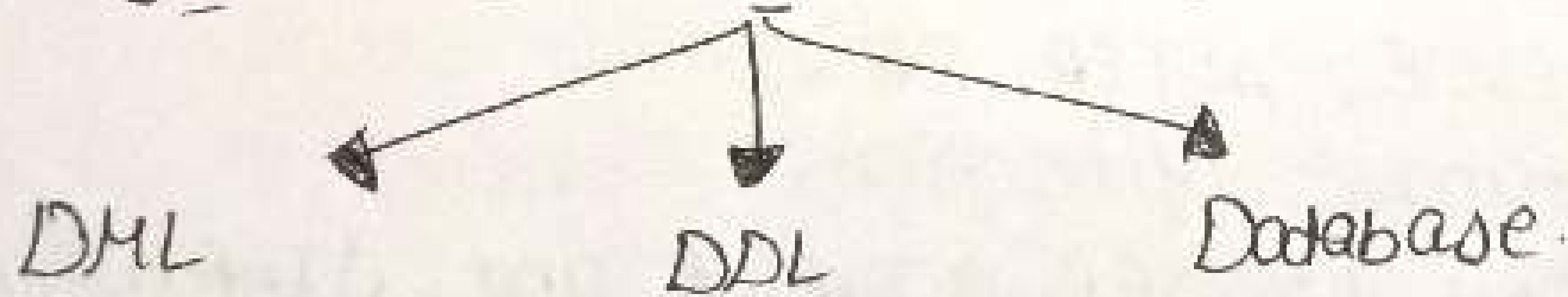
→ Categories of Trigger level



→ Categories of trigger timing



→ Categories of trigger event



→ :NEW: It holds a new value for the column of the base table/view during the trigger execution.

→ :OLD: It hold old value of the column of the base table/view during the trigger execution

→ Creating a trigger

CREATE [OR REPLACE] TRIGGER trigger-name
[BEFORE | AFTER | INSTEAD OF] → Trigger Timing
[INSERT | UPDATE | DELETE ...] → Event
[OF Col-name]
ON table-name.

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

← Row level.

WHERE [Condition]

Copyrighted by CodeLeap.io
All rights reserved.

DECLARE

Declaration - Statements

BEGIN

Executable - Statements

EXCEPTION

Exception - handling - Statements

END;

-
- Create Trigger -name: Creates an existing trigger with Trigger -name
 - Before / After: This specifies when the trigger would be executed.
 - Insert / Update / Delete: This specifies the DML operation
 - OF Col-name: This specifies the column name that would be updated.
 - ON table-name: This specifies the name of table associated with the trigger.
 - Referencing old as o NEW as n: This allow you to refer new and old values for various DML statements.

Ex:

id	Name	Age	Address	Salary
1	A	23	abc	20000
2	B	22	bcd	22000
3	C	24	edf	24000
4	D	25	ghi	26000

• CREATE OR REPLACE TRIGGER display-salary-changes
BEFORE DELETE OR INSERT OR UPDATE ON
Customer

FOR each row

WHEN (New.id > 0)

DECLARE

sal_diff number;

BEGIN

sal_diff := : New.Salary - : Old Salary;

dbms_output.put_line ('Old Salary: ' || : Old.Salary);

dbms_output.put_line ('New Salary: ' || : New.Salary);

dbms_output.put_line ('Salary diff: ' || sal_diff);

END;

/

Output:

Old Salary: 2000

New Salary: 25000

Salary difference: 5000

• • •
• • •
• • •
• • •

Copyrighted by Codeleap.io
All rights reserved.