07 Hours

Unit II : SQL and PL/SQL

DOMS | Page No.
Date / /

* Course contents :

→ SQL :
- characteristics and Advantages
- SQL Data Types and Literals.
- DDL, DML, DCL, TCL
- SQL operators

→ Tables :
- Creating, Modifying, Deleting, Updating.

→ SQL DML Queries :
- SELECT Query and clauses
- Index and sequence in SQL.

→ Views :
- Creating
- Dropping
- Updating using Indexes
- Set operations
- Predicates and Joins
- Set Membership
- Tuple variables
- Set comparison
- Ordering of Tuples
- Aggregate Functions
- SQL Functions
- Nested Queries

→ PL/SQL :
- Concept of stored Procedures and Functions
- Cursors
- Triggers
- Assertions
- Roles and Privileges.

DOMS | Page No.
Date / /

* SQL :

- SQL stands for structured Query Language. It is used for storing and managing data in relational database management system. (RDMS).
- It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
- All the RDBMS like MySQL, Informix, oracle, Ms Access and SQL Server use SQL as their standard database language.
- SQL allows users to query the database in a number of ways, using English-like statements.

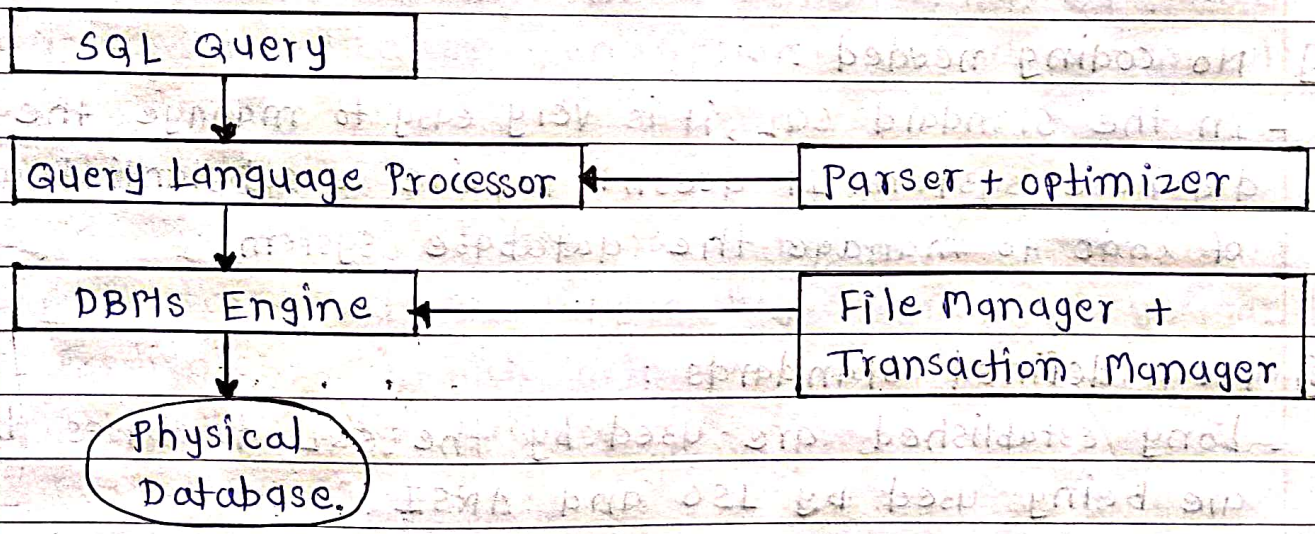
• Rules :

SQL follows the following rules :

- 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 line.
- Using the SQL statements, you can perform most of the actions in a database.
- SQL depends on tuple relational calculus and relational algebra.

• SQL Process :

- When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine determines that at how to interpret the task.
- In the process, various components are included. These components can be optimization Engine, Query engine, Query dispatcher, classic etc.
- All the non-SQL queries are handled by the classic query engine, but SQL query engine won't handle logical files.



* Characteristics of SQL :

- SQL is easy to learn.
- SQL is used to access data from relational database management systems.
- SQL can execute queries against the database.
- SQL is used to describe the data.
- SQL is used to define the data in the database and manipulate it when needed.
- SQL is used to create and drop the database & table.

Date / /

- SQL is used to create a view, stored procedure, function in a database.
- SQL allows users to set permissions on tables, procedures, and views.

* Advantages of SQL :

- There are the following advantages of SQL :

1] High speed :

- Using the SQL queries, the user can quickly and efficiently retrieve a large amount of records from a database.

2] No coding needed :

- In the standard SQL, it is very easy to manage the database system. It doesn't require a substantial amount of code to manage the database system.

3] Well defined standards :

- Long established are used by the SQL databases that are being used by ISO and ANSI.

4] Portability :

- SQL can be used in laptop, PCs, server and even some mobile phones.

5] Interactive Language :

- SQL is a domain language used to communicate with the database. It is also used to receive answers to the complex questions in seconds.

6] Multiple data view :
- Using the SQL language, the users can make different views of the database structure.

7] Scalability :
- SQL databases can handle large volumes of data and can be scaled up or down as per the requirements of the application.

8] Security :
- SQL databases have built-in security features that help protect data from unauthorized access, such as user authentication, encryption and access control.

9] Data Integrity :
- SQL databases enforce data integrity by enforcing constraints such as unique keys, primary keys, and foreign keys, which help prevent data duplication and maintain data accuracy.

10] Backup and Recovery :
- SQL databases have built-in backup and recovery tools that help recover data in case of system failure, crashes, or other disasters.

11] Data Consistency :
- SQL databases ensure consistency of data across multiple tables through the use of transactions, which ensure that changes made to one table are reflected in all related tables.

DOMS | Page No.
Date / /

* SQL Data Types :
- The data type of a column can be defined as basically what type of data format in which each cell will store the data it can be any type of integer, character, money, date and time, binary etc.

• char :
- This data type accepts character OR string type of data. It is known as fixed length data type.
- The length of the character string is specified while assigning the data type.
- char(n) where n represents the maximum size of the character string.
- If the size is not specified then the default size will be 1.
- example : char (20).

• Varchar :
- A variable-length character string with user-specified maximum length n, i.e. varchar(n).
- The full form is character varying.
- The length of the character string is specified while assigning the data type which indicates the maximum number of characters it can accept
- We can assign the length from 1 to the current table page size.
- example : Varchar (10)

Date / /

- int :

- An integer (a finite subset of the integers that is machine dependent). full form is integer.
- This data type is used to accept numeric values with a default scale as zero.
- If you assign a numeric value with a precision and scale to an integer data type, the scale portion truncates, without rounding.
- Range is 2^{-31} and $2^{31}-1$.
- Example : 167, -345, 0, 4532.98 (digits to the right of the decimal points are truncated.).

- smallint :

- The smallint data type is used to accept numeric values with default scale as zero.
- Range is 2^{-15} and $2^{15}-1$.
- Example : smallint -32768, 0, -13.7 (right digit of decimal point are truncated.)

- numeric :

- A fixed-point number with user-specified precision.
- numeric (p, d) - The number consists of p digits (plus a sign) and d of the p digits are to the right of the decimal point.
- Thus, numeric (3, 1) allows 44.5 to be stored exactly, but neither 444.5 or 0.32 can be stored exactly in a field of this type.

- decimal :
  - This data type is used to accept floating point values for which you define a precision and a scale in the data type declaration.
  - The precision is positive integer that represents the total number of digits that the number will contain (precision + scale)
  - Decimal (10,3) →
  - Example : 765.123 , 10.1234 (final digit is truncated.)

- Float :
  - float(n), a floating-point number with precision of at least n digits.
  - Example : 12345678 , 1.2 , 123.45678 , -12345678 , -1.2

- real , double precision :
  - floating point and double-precision floating-point numbers with machine-dependent precision.

- boolean :
  - This data type accepts value either TRUE or FALSE.

- date :
  - Accepts date type of value in the format YYYY-MM-DD.
  - example : DATE '1999-01-01' or DATE '2000-2-2'.

- time :
  - Accepts time value in the format HH:MM:SS.
  - example : TIME '1:10:20'.

DOMS | Page No.
Date / /

* SQL Literals :

- There are four kind of literal values supported in SQL.

1] character string :

- character string are written as a sequence of characters enveloped in single quotes.
- Ex : 'My string' , '16378' , 'I Love DBMS'.

2] Exact Numeric :

- Are written as a signed or unsigned decimal variety probably with mathematical notation.
- Ex : 8 , 80 , 80.00 , 0.8 , +88.88 , -88.88.

3] character :

- contains single character enclosed in single quotation marks.
- Ex : 'A' , '%' , 'g' , 'z' , 'c'.

4] Bit string :

- Written as a sequence of 0's and 1's enveloped in single quotes and preceded by the letter 'B'.
- Ex : B '10001011' , B '1' , B '0'.

* DDL (Data Definition Language) :

- Data Definition Language is a subset of SQL and part of DBMS.
- DDL consists of commands like create, alter, truncate, drop, rename.
- These commands are used to create or modify the tables in SQL.

1] CREATE :

- It is used to create new table in SQL. User has to give information like table name, column names and their data types.
- Syntax :

```
CREATE TABLE table_name
(
    column-1 datatype,
    column-2 datatype,
    ....
);
```

- Example :

Create a table for storing student information of a particular college.

```
CREATE TABLE student_info
(
    college_Id number (2),
    college_name varchar (30),
    Branch_varchar (10)
);
```

2] ALTER :

- It is used to add, delete or change columns in the existing table.
- The user needs to know the existing table name and can do add, delete or modify tasks easily.
- Syntax :

```
ALTER TABLE table_name
ADD column_name datatype;
```

- Example :

```
ALTER TABLE student_info
ADD CGPA number;
```

3] TRUNCATE :
- It is used to remove all rows from the table, but the structure of the table still exists.
- Syntax :
  TRUNCATE TABLE table_name ;
- Example :
  TRUNCATE TABLE student_info ;

4] DROP :
- It is used to remove an existing table along with its structure from the database.
- Syntax :
  DROP TABLE table_name ;
- Example :
  DROP TABLE student_info ;

5] RENAME :
- It is possible to change name of table with or without data in it.
- We can rename any table object at any point of time.
- Syntax :
  RENAME TABLE <Table_name> To <New Table_Name> ;
- Example :
  RENAME TABLE student_info To student_details ;

* DML (Data Manipulation Language) :
- It performs interpret-only data queries.
- It is used in a database schema to recall and manipulate the information.
- DML consists of commands select, insert, delete and update data in a database.

1] SELECT :
- It is used to get data out of the database.
- It helps users of the database to access from an operating system the significant data they need.
- Syntax :

```
SELECT * FROM <table_name>;
```

- Example :

```
SELECT * FROM students;
        OR
SELECT * FROM students where due-fees <=2000;
```

2] INSERT :
- It is used to enter the information or values into a row.
- Syntax :

```
INSERT INTO <table_name>('column_name1' <data type>,
                         'column_name2' <data type>)
  VALUES ('value 1', 'value 2');
```

- Example :

```
INSERT INTO student ('stud_Id' int, 'stud_name' varchar(20),
 'city varchar(20))
 VALUES ('1', 'Nirmit', 'Gorakhpur');
```

Date / /

3] UPDATE :

- It is used to alter existing table records within a table it modifies data from one or more records.
- This command is used to alter the data which is already present in a table.
- Syntax :

```
UPDATE <table_name>
SET <column_name = value>
WHERE condition ;
```

- Example :

```
UPDATE students
SET due_fees = 20,000
WHERE stud_name = 'mini' ;
```

4] DELETE :

- It deletes all archives from a table.
- This command is used to erase some or all of the previous tables records.
- If we do not specify the 'WHERE' condition then all the rows would be erased or deleted.
- Syntax :

```
DELETE FROM <table_name>
WHERE <condition>;
```

- Example :

```
DELETE FROM students
WHERE stud_id = '001';
```

DOMS | Page No.
Date / /

* DCL (Data control Language) :
- As the name suggests these commands are used to control privilege in the database.
- DCL command is a statement that is used to perform the work related to the rights, permissions and other control of the database system
- DCL commands are grant and revoke.

1] GRANT :
- This command is used to grant permission to the user to perform a particular operation on a particular object.
- If you are a database administrator and want to restrict user accessibility such as one who only views the data or may only update the data
- Syntax :

```
GRANT Privilege_list
ON object_name
To user_name;
```

2] REVOKE :
- This command is used to take permission / access back from the user.
- If you want to return permission from the database that you have granted to the users at that time you need to run REVOKE command.
- Syntax :

```
REVOKE priviledge_list
ON object-name
FROM user-name;
```

- Commands are granted to the user as a privilege list : EXECUTE, UPDATE, SELECT, DELETE, ALTER, ALL.

DOMS | Page No.
Date / /

* TCL (Transactional control Language) :
- These commands are used for maintaining consistency of the database and for the management of transactions made by the DML commands.
- A transaction is a set of SQL statements that are executed on the data stored in DBMS.
- TCL commands are commit, rollback, savepoint.

1] COMMIT :
- It is used to save data permanently. Whenever we perform any of the DML command like - INSERT, DELETE or UPDATE, these can be rollback if the data is not stored permanently.
- so in order to be at the safer side COMMIT command is used.
- Syntax : commit ;

2] ROLLBACK :
- It is used to get the data or restore the data to the last savepoint or last committed state.
- If due to some reasons the data inserted, deleted or updated is not correct,
- you can rollback the data to a particular savepoint or if savepoint is not done, then to the last committed state.
- Syntax :
  rollback ;

3] SAVEPOINT :

- It is used to save the data at a particular point temporarily, so that whenever needed can be rollback to that particular point.
- Syntax : savepoint A ;

- Consider the table student :

Name	Marks
John	79
Jolly	65
mini	70

```
UPDATE STUDENT
SET NAME = 'sherlock'
WHERE NAME = 'Jolly';

COMMIT ;
ROLLBACK ;
```

By using this command you can update the record and save it permanently by using COMMIT command.

- Now after COMMIT :

Name	Marks
John	79
Sherlock	65
Mini	70

- If commit was not performed then the changes made by the update command can be rollback. Now if no COMMIT performed.

```
UPDATE STUDENT
SET NAME = 'sherlock'
WHERE STUDENT_NAME = 'Jolly'
```

Date / /

- After update command the table will be :

Name	Marks
John	79
Sherlock	65
Mini	70

- Now if ROLLBACK is performed on the above table :

roll back ;

After rollback ;

Name	Marks
John	79
Jolly	65
Mini	70

- If on the above table savepoint is performed :

```
INSERT into STUDENT
VALUES ('Jack' , 95);
commit ;
UPDATE NAME
SET NAME = 'Rossie'
WHERE marks = 70 ;
SAVEPOINT A ;

INSERT INTO STUDENT
VALUES ('zack', 76);
SAVEPOINT B ;

SELECT *
FROM STUDENT ;
```

⇒

Name	marks
John	79
Jolly	65
Rossie	70
zack	76

- Now if we rollback to savepoint B and A :

Rollback to B ;

The resulting table will be

Name	marks
John	79
Jolly	65
Rossie	70
zack	76

Rollback to A ;

The resulting table will be

Name	marks
John	79
Jolly	65
Rossie	70

* SQL Operators :

- An operator is a reserved word or a character used primarily in an SQL statements WHERE clause to perform operations.

1] Arithmetic Operators :

Operator	Description	Example
+	Add	a+b
-	Subtract	a-b
*	Multiply	a*b
/	Divide	b/a
%	Modulo	b% a

2] Bitwise Operators :

operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR

3] Comparison Operators :

operator	Description	Example
=	Equal to	(a=b) is not true.
>	Greater than	(a>b) is not true
<	Less than	(a<b) is true
>=	Greater than or equal to	(a>=b) is not true
<=	Less than or equal to	(a<=b) is true.
<>	Not equal to	(a<>b) is true
!=	Not equal to	(a!=b) is true.
!<	Not less than	(a!<b) is false
!>	Not greater than	(a!>b) is true.

4] Compound Operators :

operator	Description.
+=	Add equals
-=	Subtract equals
*=	Multiply equals
/=	Divide equals
%=	Modulo equals
&=	Bitwise AND equals
^-=	Bitwise exclusive equals.
|*=	Bitwise OR equals.

Date / /

5] Logical Operators :

Operator	Description.
ALL	TRUE if all of the subquery values meet condition.
AND	TRUE if all the conditions seperated by AND is true.
ANY	TRUE if any of the subquery values meet condition.
BETWEEN	TRUE if the operand is within the range of comparisons.
EXISTS	TRUE if the subquery returns one or more records.
IN	TRUE if the operand is equal to one of a list of expressions.
LIKE	TRUE if the operand matches a pattern.
NOT	Displays a record if the condition(s) is NOT TRUE.
OR	True if any of the conditions separated by OR is True.
SOME	True if any of the subquery values meet condition.

* Tables :

- Tables are database objects that contain all the data in a database.
- In tables, data is logically organized in a row and column format similar to a spreadsheet.
- Each row represents a unique record and each column represents a field in the record.

• Creating Table :

- The CREATE TABLE statement is used to create a new table in a database.
- Syntax :

```
CREATE TABLE table_name (
    column1 datatype [size],
    column2 datatype [size],
    ....
);
```

- In this syntax, the column parameters specify the names of the columns or fields of the table.
- The datatype is the type of data which we want to store in the respective field.
- The size value indicated the maximum length of data for the field.
- If size is not given then default value depending upon the data type is assigned.
- Example :

```
CREATE TABLE Persons(
  personID int,              --PersonId column is of type int.
  LastName varchar(255),     ...columns are of type varchar
  FirstName varchar(255),    and max length for these fields
  Address varchar(255),      is 255 characters.
  city varchar(255)
);
```

The table look like :

PersonID	LastName	FirstName	Address	city.

- Create table using another table :
- A copy of an existing table can also be created using CREATE TABLE.
- The new table gets the same column definitions. All columns or specific columns can be selected.
- If you create a new table using an existing table, the new table will be filled with the existing values from the old table.

- Syntax :

```
CREATE TABLE new_table_name AS
    SELECT column1, column2, ....
    FROM existing_table_name
    WHERE ....;
```

- Example :

```
CREATE TABLE TestTable AS
SELECT customername, contactname
FROM customers;
```

} creates a new table called "TestTable" (which is a copy of the "customers" table).

- Modifying Table :

- ALTER TABLE query is used to modify structure of a table. We can add, delete or modify columns in an existing table.

→ ALTER TABLE _ ADD column.

```
ALTER TABLE table_name
ADD column_name datatype;
```

- Example :

```
ALTER TABLE Customers
ADD Email varchar(255);
```

→ ALTER TABLE - RENAME COLUMN

Syntax :

```
ALTER TABLE table_name
RENAME COLUMN old_name to new_name;
```

→ ALTER TABLE - ALTER / MODIFY Datatype.

Syntax :

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype;
```

→ Deleting all the records from table.

syntax :

TRUNCATE TABLE table-name ; } Used to delete the data inside a table, but not the table itself.

- Deleting Table :
  - DROP TABLE query is used to delete table permanently from the database.
  - DROP TABLE table-name ;
  - Eg. DROP TABLE studentdetails ;

- UPDATING Table :
  - UPDATE statement is used to modify the existing records in a table.
  - Syntax :

```
UPDATE table-name
SET column1 = value1 , column2 = value2 , ...
WHERE condition ;
```

  - Example : consider student database.

Student ID	Student-Name	City.
1	John	Pune
2	mini	Delhi
3	sana	Mumbai

- Update student

```
SET Student-Name = 'Mahi' , city = 'Nagpur'
WHERE student ID = 1 ;
```

Student ID	Student-Name	city
1	Mahi	Pune
2	mini	Delhi
3	sana	mumbai

* SQL DML Queries : SELECT Query and clayses :

- SELECT Query :

- It is used to retrieve the data from the database.
- SELECT query never make any change in the database.
- The data returned by the SELECT query is in the form of result sets.

- Syntax :

```
SELECT column1, column2, ...   -- column1, column2, .... are the fields of the table you want to select data from
FROM table_name;

SELECT * FROM table_name;   -- select all the fields in a table
```

- Example :

Consider a table student.

Roll_No.	Name	marks
101	Sara	90
102	Harry	68
103	Andrew	85

Query: SELECT Roll_No, Name from student ;

Query output:

Roll_No.	Name
101	Sara
102	Harry
103	Andrew

Query: SELECT * FROM student

Query output:

Roll_No.	Name	marks
101	Sara	90
102	Harry	68
103	Andrew	85

* Clauses :

1] WHERE Clause :

- Where clause is used to specify condition in SELECT statement while fetching records from the database.
- The where clause filters the data to be retrieved.
- Syntax :

```
SELECT column1, column2, ....
FROM table_name
WHERE condition;
```

- Example :

SELECT * from student where marks > 80;

output :

Roll_No.	Name	marks
101	sara	90
103	Andrew	85

- SELECT * from student where name = 'Harry';

output :

Roll_No.	Name	marks
102	Harry	68

2] DISTINCT clause :

- This clause is used to avoid selection of duplicate rows. consider there are duplicate values in JOB column of emp table.

ENO	Ename	Job	salary
101	Sara	clerk	12000
102	Anne	manager	18000
103	Andrew	clerk	10000
104	Harry	salesman	17000

Date / /

- Syntax :

SELECT distinct (column_name) from table_name ;

- Example :

Select distinct (job) from emp ;

- output :

Job
clerk
manager
Salesman

3] Group By clause :

- The group by clause is used in collaboration with the SELECT statement.
- It helps to arrange similar data into groups.
- It is also used with SQL functions to group the result from one or more tables.
- The group by statement is often used with aggregate functio (count() , max() , MIN() , SUM() , AVG()) to group the result-set by one or more columns.
- Syntax :

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

- Example :

consider table customers.

customer_id	first_Name	last_Name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

```
SELECT country, COUNT (*) As number
FROM customers
GROUP BY country;
```

country	number
UAE	1
UK	2
USA	2

Dye to the use of AS alias, the compiler displays the results of the count() function in the number column.

4] order By clause :

- The order by keyword is used to sort the result-set in ascending or descending order.
- This keyword sorts the records in ascending order by default.
- To sort the records in descending order, use the DESC keyword.
- syntax :

```
SELECT column1, column2,...
FROM table_name
ORDER BY column1, column2, .... ASC|DESC;
```

Date / /

- Example :

Customer ID	customer Name	contact_Name	city	postalcode	countr
1	John Doe	JD	Berlin	12209	Germar
2	Luna Robert	LR	D.F	05021	mexico
3	David Robinson	DR	D.F	05023	mexico
4	Sara Hardy	SH	London	WAI JDP	UK
5	Henry Anders	HA	Lulea	5-958-22	Sweden

Query: SELECT * FROM customers
ORDER BY country;

output :

customerID	customerName	contactName	city	Postalcode	country
1	John Doe	JD	Berlin	12209	Germary
2	Luna Robert	LR	D.F	05021	mexico
3	David Robinson	DR	D.F	05023	mexico
5	Henry Anders	HA	Lulea	5-958-22	sweden
4	Sara Hardy	SH	London	WAIJDP	UK.

→ SELECT * FROM customers
ORDER BY country DESC;
} Sorted descending by the country column.

→ SELECT * FROM customers
ORDER BY country, customer Name;
} order by several columns.

→ SELECT * FROM customers
ORDER BY country ASC, customer Name DESC;

5] Having clause :

- The having clause was added to SQL because the where keyword cannot be used with aggregate functions.

- Syntax :

```
SELECT column-name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

- Example: Customer Table

cust_id	First_Name	Last_Name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhordt	25	UK
5	Betty	Doe	28	UAE

↓

```
SELECT COUNT (cust_id), country
FROM customers
GROUP BY country
HAVING COUNT (cust_id) > 1;
```

↓

COUNT (cust_id)	country
2	UK
2	USA

* Index and sequence in SQL:

- Indexes in SQL:

- Indexes are special lookup tables that the database search engine can use to speed up data retrieval.

- Simply put, an index is a pointer to data in a table.
- An index in a database is very similar to an index in the back of a book.

1] CREATE Index command :
- Basic syntax of create index is as follows :

```
CREATE INDEX index_name ON table_name;
```

2] single - column Indexes :
- A single-column index is created based on only one table column.

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

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

```
CREATE INDEX index_name
ON table_name (column1, column2);
```

4] Unique Indexes :
- Unique indexes are used not only for performance, but also for data integrity. A unique index does not allow duplicate values to be inserted into the table.

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name);
```

5] Implicit Indexes :
- This indexes are automatically created by the database server when an object is created. Indexes are automatically created for primary key constraints and unique constraints.

6] Displaying Index :
- To display index information regarding table :

```
Show index from table_name ;
```

7] DROP Index :

```
DROP INDEX index_name ;
```

- Sequence in SQL :
- The sequences in SQL is a database object that generates a sequence of unique integer values.
- They are frequently used in databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them.
- Syntax :

```
CREATE SEQUENCE sequence_name
START WITH Initial_value.
INCREMENT BY Increment_value
MINVALUE minimum_value
MAXVALUE maximum_value
CYCLE | NOCYCLE ;
```

Here,

Sequence_name → name of sequence.

Initial_value → starting value from where the sequence should start.

Increment_value → value by which the sequence will increment by itself. This can be valued positively or negatively.

Minimum_value → Specifies minimum value of sequence.

Maximum_value → Specifies maximum value of sequence.

cycle → When the sequence reaches its maximum value, it starts again from beginning.

NoCycle → An exception will be thrown if the sequence exceeds the maximum_value.

Example : create a table 'students'

```
CREATE TABLE students (ID INT, NAME CHAR(20), AGE INT
                       NOT NULL);
```

- Insert records into tables :

```
INSERT INTO students (ID, Name, Age) values(NULL, 'Dhruv', '20
INSERT INTO students (ID, Name, Age) values (NULL, 'Arjun', '23');
INSERT INTO students (ID, Name, Age) values (NULL, 'Dev', '25');
INSERT INTO students (ID, Name, Age) values (NULL, 'Riya', '19');
```

```
SELECT * from students;
```

ID	Name	Age
NULL	Dhruv	20
NULL	Arjun	23
NULL	Dev	25
NULL	Riya	19
NULL	Arohi	24
NULL	Hsa	20
NULL	Roy	24

continue.

DOMS | Page No.
Date / /

- Create a sequence :

```
CREATE SEQUENCE my-sequence AS INT
  START WITH 1
  INCREMENT BY 1
  MINVALUE 1
  MAXVALUE 5
  CYCLE;
```

- The sequence has a max value 5 and cycles back to the starting value when it reaches the maximum value.

- once the sequence is created, it can be used to generate unique integer values. Update the data in the 'ID' column of the students table :

```
UPDATE Students set ID = Next value for my-sequence;
```

output : Commands completed successfully.

Verification : verify whether is sequence is updated in the ID column of the table or not using the following query :

```
SELECT * FROM Students;
```

ID	Name	Age
1	Dhruv	20
2	Arjun	23
3	Dev	25
4	Riya	19
5	Aarohi	24
1	Lisa	20
2	Roy	24

Page No.
Date / /

* Views :

- In SQL view is a virtual table based on the result-set of on SQL statement.
- A view contains rows and columns, just like a real tab
- The fields in a view are fields from one or more real tables in the databases.
- You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

- CREATE VIEW :

```
CREATE VIEW view_name AS
SELECT column1, column2, ....
FROM table_name
WHERE Condition;
```

Note : A view always shows up-to-date data. the database engine recreates the view, every time a user queries it.

Example :

```
create view A
As select id, name
from teacher;
```

output : view created.

```
Select * from A;
```

output :

ID	Name
1	M
2	N
3	O
4	P

- UPDATE VIEW :

- A view can be updated, using the create or replace view statement.

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ....
FROM table_name
WHERE condition;
```

Example :

```
create or replace view [Brazil customer] AS
select customerName, ContactName, City
FROM customers
WHERE Country = 'Brazil';
```

- DROP VIEW :

```
DROP VIEW view_name;
```

Example : Drop view A;

output : View dropped.

* SET Operations :

- Set operators are special type of operators which are used to combine the result of two queries.
- Different SET operators are
  1. Union 2. Union All 3. Intersect 4. Minus.
- There are certain rules which must be followed to perform operations using SET operators in SQL.
  1] The number and order of columns must be the same.
  2] Data types must be compatible.

Date / /

- Consider the following tables :

Table 1 : t - employees.

ID	Name	Department	salary	Year of experience
1	AB	Development	7200	2
2	MN	Production	45000	1
3	OP	HR	59900	3
4	LK	Accounts	57000	2
5	CD	Development	87000	3
6	HK	R&D	75000	2
7	OW	Marketting	32000	1

Table 2 : t2_employees.

ID	Name	Department	Salary	Year of experience
1	PW	R&D	49000	1
2	AP	Production	45000	1
3	GO	Development	56000	4
4	SM	Accounts	57000	2
5	RT	Production	76000	4
6	BW	R&D	75000	2
7	AS	marketting	28000	1

Table 3 : t_students.

ID	Name	Hometown	percentage	favourite_subject
1	SJ	Udaipur	89	Physics
2	MS	kanpur	92	chemistry
3	AR	Jalpur	78	History
4	PS	Nashik	88	Geography
5	RD	panipat	90	Biology
6	SK	faridabad	93	English
7	PJ	Gurugram	96	Hindi

Table 4 : t_2_Students.

ID	Name	Hometown	Percentage	favourite_subject
1	SJ	Udaipur	89	Physics
2	ID	Delhi	86	Hindi
3	AR	Jaipur	78	History
4	PA	Surat	70	sanskrit
5	RD	Panipat	90	Biology
6	JP	Pune	91	Maths
7	PJ	Gurugram	96	Hindi.

1] UNION :

- Union will be used to combine the result of two select statements.
- Duplicate rows will be eliminated from the results obtained after performing the union operation.
- Example :

Union betn the table t_employees and t_2_employees.

SELECT * FROM t_employee UNION SELECT * FROM t_2_employee.

- Consider following two tables Emp and Dept.

Table Emp :

Empno.	Ename	Job	DeptNo.	Salary.
101	Rahul	manager	10	17000
102	Vinay	clerk	20	12000
103	kunal	manager	30	18000
104	Rajesh	salesman	20	13000
105	kushal	clerk	10	11000

Table Dept :

Dept NO.	Dept Name	Loc.
10	Sales	Mumbai
20	Production	Pune
30	Accounts	Nashik
40	Research	Banglore.

- The union operator returns all distinct rows seleded by either query.
- Union will be used to combine the result of two select statements.

- Syntax : select column-name from table-1
  Union
  Select column-name from table-2;

- Example : Select DeptNo. from emp
  Union
  Select DeptNo. from dept.

- output :

DeptNO.
10
20
30
40

2] Union All :

- The union all operator returns all rows selected by either query including duplicates.

DOMS | Page No.
Date / /

- Syntax : select column_name from table_1 union all
  select column_name from table_2.
- Example : select DeptNo from emp union all
  select DeptNo from dept

- output :

DeptNo.
10
20
30
20
10
10
20
30
40

3] Intersect :

- The intersect operators returns only those rows which are common to both the queries.
- Syntax : select column_name from table_1
  intersect
  select column_name from table_2.
- Example : Select DeptNo from emp intersect
  select DeptNo from dept.

- output :

DeptNo.
10
20
30

4] Minus :
- Minus operator displays the rows which are present in the first query but absent in the second query, with no duplicates and data is arranged in ascending order by default.
- Syntax : Select column-name from table-1
minus
select column-name from table-2
- Example : select DeptNo from dept
minus
select DeptNo from emp
- output :

DeptNo.
40

* Predicates and Joins :

• Predicates :
- Predicate is an expression that evaluates to TRUE, FALSE or UNKNOWN.
- Predicates are used in the search condition of WHERE and HAVING clauses, the join conditions of FROM clauses and other constructs where value in Boolean format is expected.
- Table Emp.

EmpNo.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	King	President		11/17/1981	5000		10
7698	Blake	Manager	7839	05/01/1981	2850		30
7782	clark	Manager	7839	06/09/1981	2450		10
7566	Jones	Analyst	7839	04/02/1981	2975		20
7788	scott	Analyst	7566	12/09/1982	3000		20
7902	Ford	clerk	7566	12/03/1981	3000		20
7369	Smith	salesman	7902	12/17/1980	800		20
7499	Allen	Salesman	7698	02/20/1981	1600	300	30
7521	ward	salesman	7698	02/22/1981	1250	500	30
7654	martin	Salesman	7698	09/28/1981	1250	1400	30
7844	Turner	Salesman	7698	09/08/1981	1500	0	30
7876	Adams	clerk	7788	01/12/1983	1100		20
7900	James	clerk	7698	12/03/1981	950		30
7934	miller	clerk	7782	01/23/1982	1300		10

SQL provides various types of Predicates.

1] Comparison Predicate :

- Comparison Predicate is the combination of two expressions separated by a comparison operator.
- There are six types of comparison operators :
  =, >, <, >=, <=, <>.
- The data of NUMERIC type is compared with respect to their algebraic values.
- The data of CHARACTER STRING type is compared with respect to their alphabetic order.

• = Equal to predicate :

```
Select * from emp
where ename = 'KING'.
```

Date / /

Output :

EmpNO	ENAME	JOB	MGR	HIRE DATE	SAL	comm	DEPTN
7839	KING	President		11/17/1981	5000		10

- > Greater than Predicate :

```
select * from emp
where sql > 3000;
```

output :

EmpNo.	ENAME	JOB	MGR	HIREDATE	SAL	comm	DEPTNO
7839	KING	PRESIDENT		11/19/1981	5000		10

- < Less Than Predicate :

```
select * from emp
where sql < 3000;
```

output :

EmpNo.	ENAME	JOB	MGR	HIREDATE	SAL	comm	DEPTNO.
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7506	JONES	MANAGER	7839	04/02/1981	2973		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/28/1981	1500		30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

DOMS | Page No.
Date / /

- >= Greater Than equal to predicate:

```
select * from emp
where sal >= 3000;
```

output:

EmpNO.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO.
7839	KING	PRESIDENT		11/17/1981	5000		10
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20

- <= Less Than equal to predicate:

```
select * from emp
where sal <= 3000;
```

output:

EmpNO.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO.
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

- <> Not equal to Predicate :

```
Select * from emp
where sal <> 3000;
```

output :

EmpNo.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNo
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	30	30
7521	WARD	SALESMAN	7698	02/22/1981	1750	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

- Combination of predicates can be used with AND operator :

```
Select * from emp
where sal >= 3000 and sal <= 5000;
```

output :

EmpNo.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		11/17/1981	5000		10
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20

2] Between Predicate :

- Between Predicate is used to specify certain range of values. The AND keyword is used in this predicate.
- Syntax :

test_expression [NOT] BETWEEN begin_expression AND end_expression

- Example :

```
Select * from emp
where comm between 300 and 500 ;
```

- output :

EmpNO.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO.
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30

In 'Between' predicate the NOT keyword can also be used

```
Select * from emp.
where comm not between 300 and 500 ;
```

- Output :

EmpNO.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO.
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30

3] In Predicate :

- IN Predicate particularly determines whether the value of expression given to test matches any value in specified the list.
- Just consider that we want to display records of employees from DEPTNO 10 and 20.

-
```
Select * from emp
where deptno. in (10,20)
```

Output :

EmpNo.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTN.
7839	KING	PRESIDENT		11/17/1981	5000		10
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1982	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7934	MILLER	CLERK	7782	01/23/1982	1300		10

4] Like Predicate :

- Like operator determines whether a specific character string matches the given pattern or not.
- In the pattern we can use regular characters and wildcard characters.
- In this pattern matching, it is necessary that regular characters must exactly match the characters specified in the character string.
- However, for the wildcard characters arbitrary fragment matching by the character string is done.
- The use of wildcard characters makes the LIKE operator more flexible.
- Display records of employee whose name starts with letter 'J'.
- Query :

```
select * from emp
where ename like 'J%';
```

- Output :

EmpNo.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO.
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7900	JAMES	CLERK	7698	12/03/1981	950		30

- Display records of employee whose names ends with letter 'N'.
- Query :

```
Select * from emp
where ename like '%N';
```

- output :

EmpNO.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO.
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30

- Displays records of employee whose names contains 'L' as Second character.
- Query :

```
select * from emp
where ename like '_L%';
```

- Output :

EmpNO.	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO.
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30

- Display records of employee where names contains character 'A' anywhere.
- Query :

```
select * from emp
where ename like '%A%';
```

- Output :

EmpNO.	ENAME	JOB	MGR	HIREDATE	SAL	comm	DEPTN
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981.	1250	1400	30
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30

5] IS [NOT] NULL :

- When values for some attributes are not available then NULL value is assigned to display records having NULL value., IS NULL predicate is used.
- Display records of employees who never get any commission

```
- select * from emp
  where comm is null;
```

EmpNO.	ENAME	JOB	MGR	HIREDATE	SAL	comm	DEPTNO.
7839	KING	PRESIDENT		11/17/1981	5000		10
7698	BLAKE	MANAGER	7839	05/01/1981	2850		30
7782	CLARK	MANAGER	7839	06/09/1981	2450		10
7566	JONES	MANAGER	7839	04/02/1981	2975		20
7788	SCOTT	ANALYST	7566	12/09/1981	3000		20
7902	FORD	ANALYST	7566	12/03/1981	3000		20
7369	SMITH	CLERK	7902	12/17/1980	800		20
7876	ADAMS	CLERK	7788	01/12/1983	1100		20
7900	JAMES	CLERK	7698	12/03/1981	950		30
7934	MILLER	CLERK	7782	01/23/1982	1300		10

- The NOT keyword can be used to get values opposite to given condition.

DOMS | Page No.
Date / /

- Display records of employees who get commission.

```
select * from emp
where comm IS NOT NULL;
```

- output :

EmpNO.	ENAME	JOB	MGR	HIREDATE	SAL	comm	DEPTNO.
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30

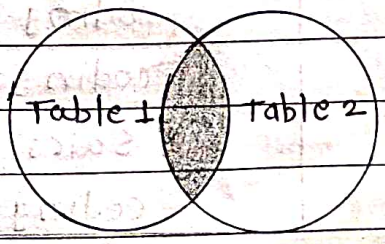
* Joins :

- A join is a means for combining columns from one (self-table) or more tables by using values common to each.
- There are following types of joins :

1] INNER
2] OUTER
3] LEFT OUTER
4] RIGHT OUTER
5] FULL OUTER
6] SELF.

1] Inner Join :

- The INNER JOIN is used to display records that have matching values in both tables.



Venn diagram : INNER JOIN.

- Syntax :

```
SELECT column_name1, column_name2, ... , columnN
FROM table_name1 INNER JOIN table-name2
ON table_name1.column_name = table-name2.column-name;
```

Table: Department

Dept_Id	Dept_Name
1001	Finance
1002	Marketing
1003	Sales
1004	coding
1005	Administration

Table: Employee_details

Emp_Id	Emp_Name	Dept_Id	Emp_salary
1	Akshay	1001	23000
2	Ram	1002	24000
3	Balram	1004	25000
4	Yatin	NULL	NULL
5	Manoj	1004	23000
6	Sheetal	1003	24000
8	Yogesh	NULL	NULL
9	Naveen	NULL	NULL
10	Tarun	1004	23000

- Select Employee_details.emp_id, Employee_details.emp_name Department.Dept_name, Employee_details.emp_salary
FROM Department INNER JOIN Employee_Details
ON
Department.Dept_ID = Employee_Details.Dept_id;

- output:

Emp_Id	Emp_Name	Dept_Name	Emp_Salary
1	Akshay	Finance	23000
2	Ram	Marketing	24000
3	Balram	coding	25000
5	Manoj	coding	23000
6	Sheetal	Sales	24000
10	Tarun	coding	23000

Date / /

2] Outer Join :

- Outer join is based on both matched and unmatched data. outer joins subdivide further into,

(i) Left outer join (ii) Right outer join (iii) full outer join.

- consider following two tables student 1 and student 2.

Table : student 1

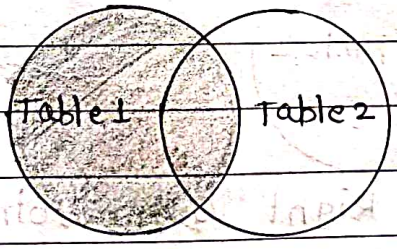
Roll NO.	Name
1	Rahul
2	kunal
3	Jay
4	Vinay
5	Preeti

Table : student 2

Roll No.	Address
1	Mumbai
2	Nashik
3	Pune
7	Banglore
8	Goa.

(i) Left Outer Join :

- The SQL LEFT JOIN returns all rows from the left table. even if there are no matches in the right table NULL values are shown at the place of right table values.



Venn diagram : Left outer join.

- SELECT column_name_list
from table_name 1
LEFT OUTER JOIN
table_name 2
ON table_1.column_name = table_2.column_name;

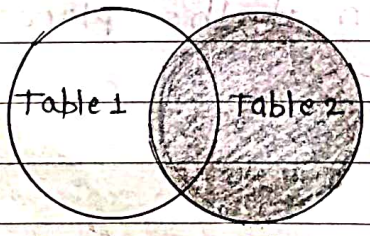
Example : SELECT * FROM student 1
LEFT OUTER JOIN student 2 ON
student 1 . rollno = student 2 . rollno ;

output :

RollNo.	Name	Roll NO.	Address
1	Rahul	1	mumbai
2	kunal	2	Nashik
3	Jay	3	Pune
4	vinay	NULL	NULL
5	Preeti	NULL	NULL

(ii) Right Outer Join :

- Returns all rows from the right table even if there are no matches in the left table. NULL values are shown at the place of left table values.



venn diagram : Right Outer Join

- Syntax :

```
Select column_name_list from table_name 1
RIGHT OUTER JOIN table_name 2
ON table_1 . column_name = table_2 . column_name ;
```

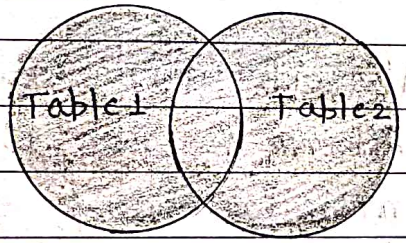
- select * from student 1
RIGHT OUTER JOIN student 2
ON (student 1 . rollno = student 2 . rollno) ;

- output :

Roll No.	Name	Roll No.	Address
1	Rahul	1	Mumbai
2	Kunal	2	Nashik
3	Jay	3	Pune
NULL	NULL	7	Banglore
NULL	NULL	8	Goa.

(iii) Full outer Join :

- The full outer join returns, a result with the matching data of both the tables and then remaining rows of both left table and then the right table.



Venn diagram : full outer Join.

Syntax : select column_name_list from table_name1
FULL OUTER JOIN table_name2
ON table_1.column_name = table_2.column_name;

Example : SELECT * FROM student 1
FULL OUTER JOIN student 2
ON (student_1.rollno = student2.rollno);

- output :

Roll No.	Name	Roll No.	Address
1	Rahul	1	Mumbai
2	Kunal	2	Nashik
3	Jay	3	Pune
4	Vinay	NULL	NULL
5	Preeti	NULL	NULL
NULL	NULL	7	Banglore
NULL	NULL	8	Goa.

3] Self Join :

- Self join is used to join a table to it-self as if the table were two tables. virtual copies of the table are considered.
- We can perform self join using table aliases.
- The table aliases allow us not to use the same table name twice with a single statement.
- If we use the same table name more than one time in a single query without table aliases, it will throw an error.
- Syntax : select S1.col_name, S2.col_name ...
  FROM table1 S1, table1 S2
  WHERE S1.common_col_name = S2.common_col_name;
- Example :

create a table "student" in a database that contains the following data.

student_id	Name	course_id	duration
1	Adam	1	3
2	Peter	2	4
1	Adam	2	4
3	Brian	3	2
2	Shane	3	5

Now, we are going to get all the result (student_id and name) from the table where student_id is equal and course_id is not equal.

Query :
```
Select s1.student_id , s1.name
FROM student AS s1 , student s2
WHERE s1.student_id = s2.student_id
AND s1.course_id <> s2.course_id ;
```

output :

student_id	Name
1	Adam
2	Shane
1	Adam
2	Peter

* SET Membership :

- It is used to check if value of expression is matching a set of values produced by a subquery or not.
- There are two keywords used for SET Membership - IN or NOT IN.
- IN is connective test for set of membership while the NOT IN is connective test for absence of SET membership.

- Table emp.

Date / /

Table - emp.

Empno.	Ename	Job	MGR	Hiredate	sal	comm	Deptno
7839	king	President		11/17/1981	5000		10
7698	Blake	Manager	7839	05/01/1981	2850		30
7782	clark	Manager	7839	06/09/1981	2450		10
7566	Jones	Manager	7839	04/02/1981	2975		20
7788	Scott	Analyst	7566	12/09/1982	3000		20
7902	Ford	Analyst	7566	12/03/1981	3000		20
7369	Smith	clerk	7902	12/17/1980	800		20
7499	Allen	Salesman	7698	02/20/1981	1600	300	30
7521	ward	salesman	7698	02/22/1981	1250	500	30
7654	Martin	salesman	7698	09/28/1981	1250	1400	30
7844	Tumer	salesman	7698	09/08/1981	1500	0	30
7876	Adams	clerk	7788	01/12/1983	1100		20
7900	James	clerk	7698	12/03/1981	950		30
7934	Miller	clerk	7782	01/23/1982	1300		10

- Example In keyword.

-Display records of employees working in SMITH's department

```
Select * from emp where deptno. in
(select deptno. from emp where ename = 'smith');
```

- output :

Empno.	Ename	Job	MGR	Hiredate	sal.	comm.	Deptno
7876	Adams	clerk	7788	01/12/1983	1100		20
7369	smith	clerk	7902	12/17/1980	800		20
7902	ford	Analyst	7566	12/03/1981	3000		20
7788	Scott	Analyst	7566	12/09/1982	3000		20
7566	Jones	manage	7839	04/02/1981	2975		20

- Example NOT IN keyword

- Display records of employees who are not working in smith's department.

select * from emp where deptno not in
(select deptno from emp where ename = 'smith');

- output :

Empno.	Ename	Job	MGR	Hiredate	Sal	comm	Deptno.
7934	Miller	clerk	7782	01/23/1982	1300		10
7782	clark	Manager	7839	06/09/1981	2450		10
7839	king	President		11/17/1981	5000		10
7900	James	clerk	7698	12/03/1981	950		30
7844	Turner	Salesman	7698	09/08/1981	1500	0	30
7654	Martin	Salesman	7698	09/28/1981	1250	1400	30
7521	Ward	Salesman	7698	02/22/1981	1250	500	30
7499	Allen	Salesman	7698	02/20/1981	1600	300	30
7698	Blake	Manager	7839	05/01/1981	2850		30

* Set Comparison :

- In nested queries, comparison operators are used with WHERE clause to specify the condition to filter the data to be display. following are the various comparison operators.

comparison operator	Description
=	Equal
< >	Not equal
>	Greater than
<	Less than
> =	Greater than or equal
< =	Less than or equal.

- Example : consider table emp (Ref. set membership e.g.)
  < operator in nested query.
  Display list of employees having salary less than Adams

- Select * from emp where sal <
  (select sal from emp where ename = 'Adams');
- output :

EmpNo.	Ename	Job	MGR	Hiredate	sal	comm	Deptr
7369	Smith	clerk	7902	12/17/1980	800		20
7900	James	clerk	7698	12/03/1981	950		30

* Tuple Variables :
- A Relation R can be listed number of times as per th requirements
- In this situation we need a way to refer to each occurrence of R.
- SQL allow us to define for each occurrence of R in the FROM clause with the help of an "alias"
- This alias is known as tuple variable. When the R is used in the FROM clause, it is followed by the keyword As which is optional and the name of the tuple variable.

- consider the table emp (Ref. set membership e.g.)

- Here the MGR number indicates the name of MANAGE of particular employee.
- for example, MGR of miller is 7782 which is empno. clark. That mean clark is manager of miller.

Date / /

- Now we want to display list of employees with their manager names. In this case we have to join this emp table to itself.
- We will consider two copies of emp table, emp A & emp B from emp A we will retrieve employee names while from emp B we will get manager names;
- Query :

```
select A.ename , B.ename from emp A, emp B where
A.MGR = B.empno;
```

- output :

Employee	Manager
Ford	Jones
scott	Jones
Allen	Blake
James	Blake
Turner	Blake
Martin	Blake
Ward	Blake
Miller	clark
Adams	scott
Blake	king
clark	king
Jones	king
Smith	ford

Here A and B are tuple variables.

* Ordering of Tuples :
- To arrange the displayed rows in ascending or descending order on given field (column), order By clause is used.
- Syntax : select * from table_name order by col1, col2.. [des

- for example :
Display the employee information as per their names in ascending order, the query will be.

Select * from emp order by Ename;

- output :

Eno.	Ename	Job	Sal.
102	Ajay	Manager	18000
104	Bharati	Manager	17000
103	Dinesh	clerk	10000
105	Prajakta	Salesman	13000
101	Susheel	clerk	12000

Now to display same information in descending order on job, the query will be.

Select * from emp order by Ename, desc;

- output :

Eno.	Ename	Job	Sal.
101	Susheel	clerk	12000
105	Prajakta	Salesman	13000
103	Dinesh	clerk	10000
104	Bharati	Manager	17000
102	Ajay	Manager	18000

- sometimes same records may available in field given for sorting criteria.
- In such case we can give names of more than one columns for sorting purpose.
- If data in first column is same, in such case the data of second column can be taken into consideration for sorting.
- consider the following table :

Eno.	Ename	Job	Sal.
101	Susheel	clerk	12000
102	Dinesh	Manager	18000
103	Dinesh	clerk	10000
104	Bharati	Manager	17000
105	Prajakta	salesman	13000

- Here names of two employees is same 'Dinesh'. Now sort the data we can mention sorting fields as ename and job.

select * from emp order by ename, job;

- output :

Eno.	Ename	Job	Sal.
104	Bharati	Manager	17000
103	Dinesh	clerk	10000
102	Dinesh	Manager	18000
105	prajakta	salesman	13000
101	susheel	clerk	12000

* Aggregate functions :

- Aggregate functions perform a calculation on a set of values & return a single value, usually these functions ignore NULL values (except for COUNT)
- There are different types of aggregate functions :
  1] Min
  2] Max
  3] Sum
  4] Avg
  5] count.
- Consider the table emp. (Ref. set membership e.g.)

1] min() :

- This function returns smallest value from specified column of the table.
- Query - select min(sal) from emp;
- output - 800.

2] max() :

- This function returns greatest value from specified column of the table.
- Query - select max(sal) from emp;
- output - 5000.

3] Sum() :

- This function returns sum of all the values of specified column of the table :
- Query - select sum(sal) from emp;
- output - 29025

Date / /

4] Avg():
- This function returns average of all the values of specified column of the table.
- Query - select avg(sal) from emp;
- output - 2073.21.

5] count():
- This function returns total no. of values of specified column of the table.
- Query - select count(ename) from emp;
- output - 14.

* SQL Functions:

• Scalar Functions:
- Are used to perform calculations on a single value and return a single result.

(i) Length():
- returns the number of characters in a string.
e.g. - select length('SQL');
output - 3

(ii) Upper():
- Converts a string to uppercase.
- e.g. - select upper(cust_name) from customers;

(iii) Lower():
- Converts a string to lowercase.
e.g. - select lower(cust_name) from customers;

(iv) concat() :
- Concatenates two or more strings together.
e.g. - select concat ('first', 'second');

(v) Round() :
- Rounds a number to a specified number of decimal places.
e.g. - select Round (total_cost, 2) from orders;

- Date and Time functions :

(i) NOW() :
- Returns the current date and time
e.g. - select NOW();

(ii) Current_date() :
- Returns the current date.
- e.g. - select current_date();

(iii) Current_time() :
- Returns current time.
e.g. - select current_time();

(iv) Year() :
- Returns the year of a date
e.g. - select year('2017/08/25') As year;

(v) Month() :
- Returns the month of a date.
e.g. - select month ('2017/08/25') As month;

(vi) Day ( ) :

- Returns the day of a date.

e.g.- select day ('2017/08/25') As dayofmonth ;

* Nested Queries :

- Writing a query inside another query is known as nested query or subquery.
- The inner query gets executed first, then the output on inner query is given as input to outer query.
- consider table emp. (Ref. set membership e.g.)

E.g.

1] To display records of employees working in smith's dept.

```
Select * from emp where deptno. =
(select deptno from emp where ename = 'smith');
```

output :

Empno.	Ename	Job	MGR	Hiredate	sal.	comm	Deptno.
7566	Jones	Manager	7839	04/02/1981	2975		20
7788	Scott	Analyst	7569	12/09/1982	3000		20
7902	ford	Analyst	7566	12/03/1981	3000		20
7369	smith	clerk	7902	12/17/1980	800		20
7876	Adams	clerk	7788	01/12/1983	1100		20

2] To display records of employees whose salary is more than the salary of ford.

```
Select * from emp where sal >
(select sal from emp where ename = 'ford');
```

output :

Empno.	Ename	Job	MGR	Hiredate	sal	comm.	Deptno.
7839	king	President		11/17/1981	5000		10

3] To display records of employees who are senior to jones

```
Select * from emp where hiredate <
(select hiredate from emp where ename = 'Jones');
```

Output :

Empno.	Ename	Job	MGR	Hiredate	Sal	comm.	Dept
7369	smith	clerk	7902	12/17/1980	800		20
7499	Allen	salesman	7698	02/20/1981	1600	300	30
7521	ward	salesman	7698	02/22/1981	1250	500	30

* PL/SQL :

- PL/SQL stands for Procedure Language / Structured Query language.
- It is the combination of SQL along with the procedural features of programming languages.
- PL/SQL includes procedural language elements such as conditions and loops.
- It allows declaration of constants and variables, procedures and functions, types and variables of those types and triggers.
- It can handle exceptions (runtime errors)
- Arrays are supported involving the use of PL/SQL collections.
- It has included features associated with object orientat
- one can create PL/SQL units such as procedures, functions, packages, types and triggers which are stored in the database for reuse the applications.

DOMS | Page No.
Date / /

* PL/SQL features :
- Allows to define and use, variables and constants.
- provides constructs to control the flow of a program :
  - Branching / conditional constructs.
  - Iterative / looping constructs.
- Allows row by row processing of data retrived from database using cursors.
- Allows to trap errors and write routines to handle predefined and user defined error situations.
- can write modular application by dividing it into subprograms.
- Packages
- Procedures and functions.
- Triggers.

* Advantages of PL/SQL :

1] Support for SQL :
- Support all the functionalities of SQL.
- Efficient database handling.

2] Improved Performance :
- SQL Processing : single statement.
- PL/SQL processing : Block of statements.
- Reduces overheads to improve performance.

3] Block structure :
- PL/SQL is a block-structured language
- Programs can be divided into logical blocks of code.

4] Higher Productivity :

- Procedural constructs.
- PL/SQL can be used with :
  - Oracle database server.
  - oracle tool like forms, reports etc.

5] Portability :

- Application written in PL/SQL are portable to
  - Any computer Hardware.
  - Any os environment where oracle RDBms is running

* Concept of stored Procedures and Functions :

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- So, if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- Syntax :

```
create procedure procedure-name
As
sql_statement
Go;
```

- Execute :

```
Exec procedure_name;
```

consider the table :

rno.	name	DOB	class
2	abc	1990-11-21	NULL
3	def	1989-10-30	NULL
4	xyz	2000-02-02	NULL
5	lmn	2000-11-17	NULL

- In parameter :

- Accepts value when procedure get called

example - create a procedure which should accept rollno as parameter and display the record.

```
create procedure display (IN r integer(3))
begin
select * from students where rno = r;
end;

CALL display(3);
```

output :

rno.	name	DOB	class
3	def	1989-10-30	NULL

- Out parameter :

- The value of an OUT parameter can be change inside the stored procedure.
- The changed value is passed back to the calling program.
- The initial value of OUT parameter cannot be accessed by the procedure.
- E.g. -

```
Create procedure display'(IN r INT, OUT nm VARCHAR(20))
    Begin
    select name into nm from students where rno = r;
    end;
```

Date / /

Now the procedure can be called as
CALL display' (3, @nm);

Here n is the OUT parameter which stores the name of student having rno. 3;
Then execute the command
select @nm;

output:

@nm
def.

@ - Remote access indicator.

* - Stored function is same as of stored procedure means it is a group of SQL statements which can be executed repeatedly.
- It allow for variable declarations, flow control and other useful programming techniques.
- Just difference is that function can return value.
- Syntax : create function function_name ([parameter (s)])
returns data type
deterministics
statements

→ create a function to return record of student having rollno. 3.
create function anual sal (sal int)
Returns int(7);
deterministic
begin
declare asal int(7);
set asal = sal * 12;
return asal;
end

Deterministic - Always returns the same results if given the same i/p values.

Then execute the command

```
select annualsal (5000) from dual;
```

- output :

annualsal (5000)
60000

* Cursors :

- Cursor is used to traverse in the database to access the records one by one.
- For example, if we want to calculate the average of marks of all the students, then we can retrieve marks of every student and add in the total_marks.
- Cursor is just like loop concept used to traverse to every row and manipulate data.
- An area of memory (context) is allocated for the processing of SQL statements.
- The context area contains information necessary to complete the processing, including the number of rows processed by the statement, a pointer to the parsed representation of the statement.
- Cursor is a handle or pointer to the context area.
- There are two type of cursors :
  (i) Implicit cursor
  (ii) Explicit cursor.

(i) Implicit Cursors :

- The implicit cursors are automatically generated by Oracle while an SQL statement is executed.
- These are created by default to process the statements when DML statements like INSERT, UPDATE, DELETE etc. are executed.

- Oracle provides some attributes known as Implicit cursor's attributes to check the status of DML opera some of them are : % FOUND, % NOTFOUND, % ROWCOUN and % ISOPEN.

S.NO.	Attribute	Description.
1.	% FOUND	Returns TRUE if an INSERT, UPDAT or DELETE statement affected on or more rows or a SELECT INTO statement returned one or more rows. otherwise, it returns FALSE.
2.	% NOTFOUND	The logical opposite of % FOUND. It retu TRUE if an INSERT, UPDATE or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. otherwise, it returns FALSE.
3.	% ISOPEN	Always returns FALSE for implicit cursors, because oracle closes the SQL cursor automatically after executing its associated SQL statemen
4.	% ROWCOUNT	Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

(ii) Explicit Cursor :

- Explicit cursors are programmer-defined cursors for gaining more control over the context area.
- An explicit cursor 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.
- The syntax for creating an explicit cursor is :

  CURSOR cursor_name IS select_statement ;
- Working with an explicit cursor includes the following steps :
  - Declaring the cursor for initializing the memory.
  - Opening the cursor for allocating the memory.
  - Fetching the cursor for retrieving the data.
  - closing the cursor to release the allocated memory.

1] Declaring the Cursor :

- Declaring the cursor defines the cursor with a name and the associated SELECT statement.
- For example,

```
CURSOR c_customers IS
  SELECT id, name, address FROM customers;
```

2] Opening the cursor :

- Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it.
- For example,

```
OPEN c_customers;
```

Date / /

3] Fetching the cursor :

- Fetching the cursor involves accessing one row at a time.
- For example,

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

4] closing the cursor :

- closing the cursor means releasing the allocated memory.
- For example,

```
CLOSE c_customers;
```

* Triggers :

- A Trigger in Structured Query Language is a set of procedural statements which are executed automatically when there is any response to certain events on th particular table in the database.
- Triggers are used to protect the data integrity in the database.
- In SQL, triggers are called only either before or after the below events :

1] INSERT Event :

This event is called when the new row is entered in the table.

2] UPDATE Event :

This event is called when the existing record is changed or modified in the table.

3] DELETE Event :

This event is called when the existing record is removed from the table.

- Types of Triggers in SQL :
- Following are the six types of triggers in SQL :

1] AFTER INSERT Trigger :
- This trigger is invoked after the insertion of data in the table.

2] AFTER UPDATE Trigger :
- This trigger is invoked in SQL after the modification of the data in the table.

3] AFTER DELETE Trigger :
- This trigger is invoked after deleting the data from the table.

4] BEFORE INSERT Trigger :
- This trigger is invoked before the inserting the record in the table.

5] BEFORE UPDATE Trigger :
- This trigger is invoked before the updating the record in the table.

6] BEFORE DELETE Trigger :
- This trigger is invoked before deleting the record from the table.

- Syntax of Trigger in SQL :

```
CREATE TRIGGER Trigger-Name
[BEFORE | AFTER] [INSERT | UPDATE | DELETE]
ON [Table_name]
[FOR EACH ROW | FOR EACH COLUMN]
AS
Set of SQL statement.
```

- Example of Trigger in SQL :

- The following query creates the student_Trigger table in the SQL database:

```
CREATE TABLE student_Trigger
(
student_RollNO INT NOT NULL PRIMARY KEY,
Student_First Name Varchar (100),
student_EnglishMarks INT,
student_PhysicsMarks INT,
Student_chemistryMarks INT,
student_MathsMarks INT,
Student_TotalMarks INT,
student_Percentage );
```

- The following query shows the structure of the student_Trigger table.

```
DESC student_Trigger.
```

- output :

Field	Type.	NULL	key	Default	Extra
student_RollNo	INT	NO	PRI	NULL	
student_FirstName	varchar(100)	YES		NULL	
student_EnglishMarks	INT	YES		NULL	
student_PhysicsMarks	INT	YES		NULL	
student_chemistryMarks	INT	YES		NULL	
student_MathsMarks	INT	YES		NULL	
student_TotalMarks	INT	YES		NULL	
student_Percentage.	INT	YES		NULL.	

- The following query fires a trigger before the insertion of the student record in the table :

```
CREATE TRIGGER student_Table_Marks
BEFORE INSERT
ON
student_Trigger
FOR EACH ROW
SET new.student_TotalMarks = new.student_EnglishMarks +
new.student_PhysicsMarks + new.student_chemistryMarks +
new.student_MathsMarks , new.
new.student_Percentage = (new.student_TotalMarks /400)*100;
```

- The following query inserts the record into student_Trigger table :

```
INSERT INTO student_Trigger( student_RollNo, student_FirstName,
student_EnglishMarks , student_PhysicsMarks , student_chemistrymarks ,
student_MathsMarks , student_TotalMarks , student_Percentage) VALUES
(201, Sonya, 88, 75, 69 , 92, 0 ,0);
```

Date / /

- To check the output of the above INSERT statement you have to type the following SELECT statement :

```
SELECT * FROM student_Trigger ;
```

- output :

student_ RollNO.	student_ FirstName	student_ Englishmarks	student_ Physicsmarks	student_ chemistrymarks	student_ Mathsmarks	student_ Totalmarks	st Perc
201	Sorya	88	75	69	92	324	81

- Advantages of Triggers in SQL :

1] SQL provides an alternate way for maintaining the data and referential integrity in the tables.

2] Triggers helps in executing the scheduled tasks because they are called automatically.

3] They catch the errors in the database layer of various businesses.

4] They allow the database users to validate values before inserting and updating.

- Disadvantages of Triggers in SQL :

1] They are not compiled.

2] It is not possible to find and debug the errors in triggers.

3] If we use the complex code in the trigger, it makes the application run slower.

4] Trigger increases the high load on the database system

* Assertions :

- When a constraint involves 2(or) more tables, the table constraint mechanism is sometimes hard and results may not come as expected.
- To cover such situation SQL supports the creation of assertions that are constraints not associated with only one table.
- And an assertion statement should ensures a certain condition will always exist in the database.
- DBMS always checks the assertion whenever modificat^{n} are done in the corresponding table.

-syntax :

```
CREATE ASSERTION [assertion_name]
  CHECK ( [condition] );
```

Example :

```
CREATE TABLE sailors (sid int, sname varchar(20),
rating int, primary key(sid),
CHECK (rating >=1 AND rating <= 10)
CHECK ((select count (s.sid) from sailors s) +
  (select count (b.bid) from boats b) < 100) );
```

* Roles and Privileges

• Roles :

- A role is a mechanism that can be used to allow authorization.
- A person or a group of people can be allowed a role or group of roles.

- By many roles, the head can manage access privileges very easily.
- The roles are provided by the database management system for easy and managed or controlled privilege management.

. Properties :

- The following are the properties of the roles which allow easy privilege management inside a database

(i) Reduced privilege administration :

- The user can grant the privilege for a group of users who are related instead of granting the same set of privileges to the users explicitly.

(ii) Dynamic privilege management :

- If the privilege of the group changes then, only the right of role needs to be changed.

(iii) Application - specific security :

- The user can also protect the use of a role by using a password.
- Applications can be created to allow a role when entering the correct and best password.
- Users are not allowed the role if they do not know about the password.

Privileges :

- The authority or permission to access a named object as advised manner, for example, permission to access a table.
- Privileges can allow permitting a particular user to connect to the database.
- In, other words privileges are the allowance to the database by the database object.

(i) Database privileges :

- A privilege is permission to execute one particular type of SQL statement or access a second persons object.
- Database privilege controls the use of computing resources.
- Database privilege does not apply to the database administrator of the database.

(ii) System privileges :

- A system privilege is the right to perform an activity on a specific type of object.
- for example, the privilege to delete rows of any table in a database is system privilege.
- There are a total of 60 different system privileges.
- System privileges allow users to CREATE, ALTER or DROP the database objects.

(iii) Object privilege :

- An object privilege is a privilege to perform a specific action on a particular table, function, or package.
- for example, the right to delete rows from a table is an object privilege.

Date / /

- For example, let us consider a row of table obj-pr that contains the name of the employee who is no longer a part of the organization, then deleting that row is considered as an object privilege.
- object privilege allows the user to INSERT, DELETE, UPDATE, or SELECT the data in the database object.