SQL UNIT-3

SQL: Basic SQL querying (select and project) using where clause, arithmetic & logical operations, SQL functions(Date and Time, Numeric, String conversion).Creating tables with relationship, implementation of key and integrity constraints, nested queries, sub queries, grouping, aggregation, ordering, implementation of different types of joins, view(updatable and non-updatable), relational set operations.

Basic SQL querying (select and project) using where clause:

SELECTION:

Selection operation can be considered as row wise filtering. We can select specific rows using condition.

1.Syntax : SELECT * FROM table_name;

Exmple:SQL>select * from student;

2.Syntax : SELECT * FROM table_name WHERE condition;

Example: SQL> Select * from student where maths=75;

PROJECTION:

The projection operation performs column wise filtering. Specific columns are selected in the projection operation.

1.Syntax:

select column_name1, column_name2, column_nameN from table_name;

EXAMPLE: SQL> Select sname, maths, total from student;

2.Syntax: Select column_name1, column_name2, ……, column_nameN from table_name where condition;

Example: Select sname, maths, cs from student where sno=1;

Arithmetic and Logical operators:

Arithmetic operators: The SQL Arithmetic operators are used to perform basic mathematical operations on numeric data types in a database. The most common arithmetic operators used in SQL are:

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Example:

Consider the emp table below

ENO ENAME CITY SALARY

---------- ---------- ---------- ----------

101 kumar hyderabad 10000

102 suma hyderabad 20000

103 anusha mumbai 30000

SQL> update emp set sal=sal+1000 where eno=103;

1 row updated;

SQL>update emp set sal=sal-1000 where eno=101;

1 row updated;

SQL>update emp set sal=sal*2 where eno=102;

1 row updated;

Logical Operators:

Logical operators are used to combine or manipulate the conditions given in a query to retrieve or manipulate data .there are some logical operators in SQL like OR, AND etc.

Types of Logical Operators in SQL

Operator	Meaning
AND	TRUE if both Boolean expressions are TRUE.
IN	TRUE if the operand is equal to one of a list of expressions.
NOT	Reverses the value of any other Boolean operator.
OR	TRUE if either Boolean expression is TRUE.
LIKE	TRUE if the operand matches a pattern.
BETWEEN	TRUE if the operand is within a range.
ALL	TRUE if all of a set of comparisons are TRUE.
ANY	TRUE if any one of a set of comparisons is TRUE.
EXISTS	TRUE if a subquery contains any rows.
SOME	TRUE if some of a set of comparisons are TRUE.

Example:

consider the emp table below

ENO	ENAME	CITY	COUNTRY
101	kumar	hyderabad	india
102	suma	hyderabad	india
103	anusha	mumbai	india
104	prathyu	mumbai	india
105	sekhar	vizag	india

AND: Logical AND compares two Booleans as expressions and returns true when both expressions are true.

```
SQL> SELECT * FROM emp WHERE city='mumbai' and  country='india';
```

ENO	ENAME	CITY	COUNTRY
103	anusha	mumbai	india
104	prathyu	mumbai	india

IN: The IN operator checks a value within a set of values separated by commas and retrieves the rows from the table that match.

```
SQL> select * from emp where city in('mumbai','vizag');
```

ENO	ENAME	CITY	COUNTRY
103	anusha	mumbai	india
104	prathyu	mumbai	india

105 sekhar vizag india

NOT: Not takes a single Boolean as an argument and change its value from false to true or from true to false.

SQL> select * from emp where city not like 'h%';

ENO ENAME CITY COUNTRY

---------- ---------- ---------- ----------

103 anusha mumbai india

104 prathyu mumbai india

105 sekhar vizag india

OR: Logical OR compares two Booleans as expressions and returns true when one of the expressions is true.

SQL> select * from emp where city='vizag' or city='mumbai';

ENO ENAME CITY COUNTRY

---------- ---------- ---------- ----------

103 anusha mumbai india

104 prathyu mumbai india

105 sekhar vizag india

LIKE: In SQL, the LIKE operator is used in the WHERE clause to search for a specified pattern in a column.
% – It is used for zero or more than one character.
_ – It is used for only one character means fixed length.

SQL> select * from emp where city like 'v%';

ENO ENAME CITY COUNTRY

---------- ---------- -------------- ----------

105 sekhar vizag india

BETWEEN: The SQL BETWEEN operator tests an expression against a range. The range consists of a beginning, followed by an AND keyword and an end expression.

SQL> select * from emp where eno between 102 and 104;

ENO ENAME CITY COUNTRY

---------- ---------- -------------- ----------

102 suma hyderabad india

103 anusha mumbai india

104 prathyu mumbai india

ALL: ALL is used to select all records of a SELECT STATEMENT. It compares a value to every value in a list of results from a query. The ALL must be preceded by the comparison operators and evaluated to TRUE if the query returns no rows.

SQL> select * from emp where eno>ALL(select eno from emp where city='hyderabad';

ENO ENAME CITY COUNTRY

---------- ---------- -------------- ----------

103 anusha mumbai india

104 prathyu mumbai india

105 sekhar vizag india

ANY: ANY compares a value to each value in a list of results from a query and evaluates to true if the result of an inner query contains at least one row.

SQL> select * from emp where eno=any(select eno from emp where city='hyderabad';

ENO	ENAME	CITY	COUNTRY
101	kumar	hyderabad	india
102	suma	hyderabad	india

EXISTS: The EXISTS checks the existence of a result of a subquery. The EXISTS subquery tests whether a subquery fetches at least one row. When no data is returned then this operator returns 'FALSE'.

SQL> select * from emp where exists(select eno from emp where city='vizag');

ENO	ENAME	CITY	COUNTRY
101	kumar	hyderabad	india
102	suma	hyderabad	india
103	anusha	mumbai	india
104	prathyu	mumbai	india
105	sekhar	vizag	india

SOME: SOME operator evaluates the condition between the outer and inner tables and evaluates to true if the final result returns any one row. If not, then it evaluates to false.

SQL> select * from emp where eno< some(select eno from emp where city='vizag');

ENO	ENAME	CITY	COUNTRY
101	kumar	hyderabad	india
102	suma	hyderabad	india
103	anusha	mumbai	india
104	prathyu	mumbai	india

SQL FUNCTIONS:

Date and Time Functions:

Function	Description
Sysdate	It returns the current date and time
Next_Day	It returns a datetime value that represents the first weekday
Last_Day	It returns the date of the last day of the month
Add_Months	It returns the added value of date for given month
Months_Between	It returns number of months between 2 dates
Round	It rounds a date to a specified precision
Trunc	It truncate a date or time to the first instance of a given date part
Greatest	It returns the greatest date from given dates
Lowest	It returns the lowest date from given dates

Examples:

1.sysdate():

SQL> select sysdate from dual;

SYSDATE

05-OCT-24

2.last_day():

SQL> select last_day('05-oct-24') from duaL;

LAST_DAY(

31-OCT-24

3.next_day():

SQL> select next_day('05-oct-24','thursday') from duaL;

NEXT_DAY(

10-OCT-24

4.add_months():

SQL> select add_months('05-oct-24',5) from duaL;

ADD_MONTH

05-MAR-25

5.months_between():

SQL> select months_between('05-oct-24','05-may-24') from duaL;

MONTHS_BETWEEN('05-OCT-24','05-MAY-24')

6.round():

SQL> select round(to_date('05-oct-24'),'year') from dual;

ROUND(TO_

01-JAN-25

7.trunc():

SQL> select trunc(to_date('05-oct-24'),'year') from dual;

TRUNC(TO_

01-JAN-24

8.greatest():

SQL> select greatest(to_date('01-jul-24'),to_date('01-sep-24'),to_date('20-oct-24')) from dual;

GREATEST(

20-OCT-24

9.least():

SQL> select least(to_date('01-jul-24'),to_date('01-sep-24'),to_date('20-oct-24')) from dual;

LEAST(TO_

01-JUL-24

10. To_char():

SQL> select to_char(sysdate,'yyyy/mm/dd') from dual;

TO_CHAR(SY

2024/10/05

11. To_date():

SQL> select to_date('2024/08/24','yyyy/mm/dd') from dual;

TO_DATE('

24-AUG-24

Numeric Functions:

Function	Description
Abs(-x)	It returns the absolute value of a number
Power(m,n)	It returns m raised to the nth power
Sqrt(n)	It returns the square root of a number
Mod(x,y)	It returns the remainder (aka. modulus) of n divided by m.
Round(x,y)	It returns a number rounded to a certain number of decimal places
Trunc(x,y)	It returns truncated to y places right of the decimal point.
Floor(x)	It returns the largest integer value that is less than or equal to a number
Ciel(x)	It returns the smallest integer value that is greater than or equal to a number
Greatest(a,b,c)	It returns the greatest value in a list of expressions
Least(a,b,c)	It returns the smallest value in a list of expressions
Exp(x)	It returns e raised to the power of a number

Examples:

1.abs():

SQL> select abs(-9) from dual;

```
  ABS(-9)
----------
         9
```

2.power():

```
SQL> select power(2,3) from dual;

POWER(2,3)
----------
         8
```

3.sqrt():

```
SQL> select sqrt(4) from dual;

  SQRT(4)
----------
         2
```

4.mod():

```
SQL> select mod(4,3) from dual;

 MOD(4,3)
----------
         1
```

5.round():

```
SQL> select round(43.65,1) from dual;

ROUND(43.65,1)
--------------
          43.7
```

6.trunc():

```
SQL> select trunc(43.65,1) from dual;
```

TRUNC(43.65,1)

43.6

7.floor():

SQL> select floor(34.5) from dual;

FLOOR(34.5)

34

8.ceil():

SQL> select ceil(35.6) from dual;

CEIL(35.6)

36

9.greatest():

SQL> select greatest(2,77,12) from dual;

GREATEST(2,77,12)

77

10.least():

SQL> select least(25,17,11) from dual;

LEAST(25,17,11)

11

11.exp():

SQL> select exp(4) from dual;

EXP(4)

54.59815

String Functions:

Function	Description
concatination	This function is used to add two strings
Lpad	This function is used to make the given string of the given size by adding the given symbol on the left side
Rpad	This function is used to make the given string of the given size by adding the given symbol on the Right side
LTRIM	This function is used tocut the given left substring from original string
RTRIM	This function is used to cut the given right substring from original string
Lower	This function converts a string to lower-case
Upper	This function converts a string to upper-case
Initcap	This function capitalize first letter of each word
Length	This function is used to find the length of a word
Substr	This function is used to find a sub string from the a string from the given position
instr	This function is used to find the occurrence of an alphabet

Examples:

1.concatenation( ):

SQL> select concat('rama','is a good boy') from dual;

CONCAT('RAMA','IS

---

ramais a good boy

2.lpad( ):

SQL> select lpad('hai',10,'$') from dual;

LPAD('HAI'

$$$$$$$hai

3.rpad( ):

SQL> select rpad('red',10,'*') from dual;

RPAD('RED'

---

red*******

4.rtrim( ):

SQL> select rtrim('hai','i') from dual;

RT

--

Ha

5.ltrim( ):

SQL> select ltrim('hyderabad','hyd') from dual;

LTRIM(

Erabad

6.lower( ):

SQL> select lower('INDIA') from dual;

LOWER

India

7.upper( ):

SQL> select upper('india') from dual;

UPPER

INDIA

8.initcap( ):

SQL> SELECT INITCAP('RAMA IS A GOOD BOY') FROM DUAL;

INITCAP('RAMAISAGO

Rama Is A Good Boy

9.length( ):

SQL> select length('rama is a good boy') from dual;

LENGTH('RAMAISAGOODBOY')

18

10.substr( ):

SQL> select substr('india',1,3) from dual;

SUB

Ind

11.instr( ):

SQL> select instr('andhrapradesh','r') from dual;

INSTR('ANDHRAPRADESH','R')

5

Creating tables with relationships:

Consider two tables student and faculty. The relation between these two tables is faculty teaches students.

Table1(Faculty)creation

Create table Faculty(fid number(3),fname varchar2(10),subject varchar2(10) primary key(fid));

Table2(student)creation

Create table Student(sid number(10),sname varchar2(10),subject varchar2(10) primary key(sid));

Relationship table

Create table Teaches(fid number(3),sid number(10),subject varchar2(5),foreign key(fid) references faculty(fid),foreign key(sid) references student(sid),Primary key(fid,sid));

Implementation of Key and Integrity Constraints:

NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK are Key and Integrity Constraints.

1. **NOT NULL Constraint:** We know that by default all columns in a table allow null values. When not null constraint is enforced, either on a column or a

set of columns in a table, it will not allow null values.

Ex: SQL>create table student1(sno number(3), sname varchar2(10) not null);

Table created.

SQL> insert into student1(sno,sname) values(101,'ramu');

1 row created.

SQL> insert into student1(sno) values(102);

ERROR at line 1:

ORA-01400: cannot insert NULL into ("SYSTEM"."STUDENT1"."SNAME")

SQL> insert into student1(sno,sname) values(101,'ramu');

1 row created.

2. **UNIQUE Constraint:** The unique constraint is used to prevent the duplication of values within the rows of specified column or a set of columns in a table. Columns defined with this constraint can also allow null values.

Ex: create table student2 (sno number(3) unique, sname varchar2(10));

SQL> insert into student2(sno,sname) values(101,'ramu');

1 row created.

SQL> insert into student2(sno,sname) values(101,'ramu');

ERROR at line 1:

ORA-00001: unique constraint (SYSTEM.SYS_C004037) violated

SQL> insert into student2(sname) values('hari');

1 row created.

3. **PRIMARY KEY:** The primary key constraint avoids both duplication of rows and null values, when enforced on a column or set of columns.

Ex: SQL> create table student3(sno number(3) primary key, sname varchar2(10));

SQL> insert into student3(sno,sname) values(01,'ramu');

1 row created.

SQL> insert into student3(sno,sname) values(01,'ramu');

ERROR at line 1:

ORA-00001: unique constraint (SYSTEM.SYS_C004038) violated

SQL> insert into student3(sname) values('hari');

ORA-01400: cannot insert NULL into ("SYSTEM"."STUDENT3"."SNO")

4. **FOREIGN KEY:** To establish a 'parent_ child' relationship between two tables having a common column, we make use of referential integrity constraints. To implement this, we should define the column in the parent table as a primary key and the same column in the child table as a foreign key referring to the corresponding parent entry.

A foreign key in one relation **references** primary key in another relation, the foreign key value must match the primary key value.

Ex : SQL> create table dept (did number(2) primary key, dname varchar2(10));

Table created.

SQL> create table emp(eno number(3) primary key, ename varchar2(10), deptno number(2) references dept(did));

Table created.

SQL> insert into emp(eno,ename,deptno) values(101,'ramu',22);

ERROR at line 1:

ORA-02291: integrity constraint (SYSTEM.SYS_C004044) violated - parent key not found

SQL> insert into dept(did,dname) values(22,'CSE');

1 row created.

SQL> insert into emp(eno,ename,deptno) values(101,'ramu',22);

1 row created.

SQL> delete from dept;

ERROR at line 1:

ORA-02292: integrity constraint (SYSTEM.SYS_C004044) violated - child record Found

SQL> delete from emp;

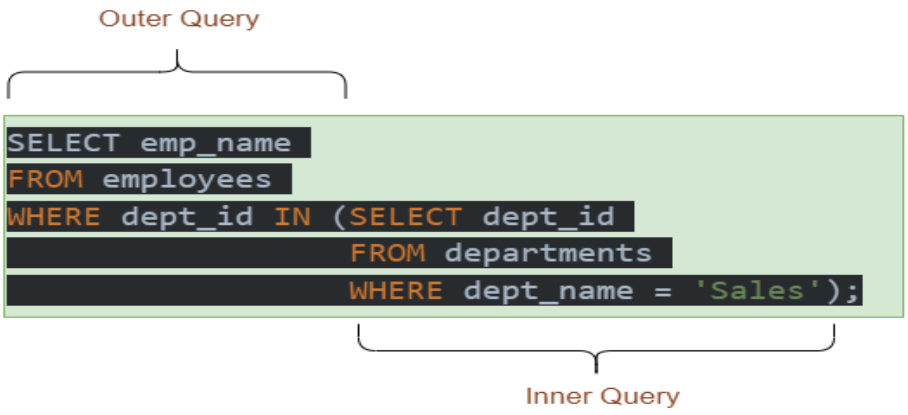
1 row deleted.

SQL> delete from dept;

1 row deleted.

NESTED QUERIES:

In SQL, a nested query involves a query that is placed within another query. Output of the inner query is used by the outer query. A nested query has two SELECT statements: one for the inner query and another for the outer query.



Syntax of Nested Queries:

The basic syntax of a nested query involves placing one query inside of another query. Inner query or subquery is executed first and returns a set of values that are then used by the outer query.

The syntax for a nested query is as follows:

SELECT column1, column2, ...

FROM table1

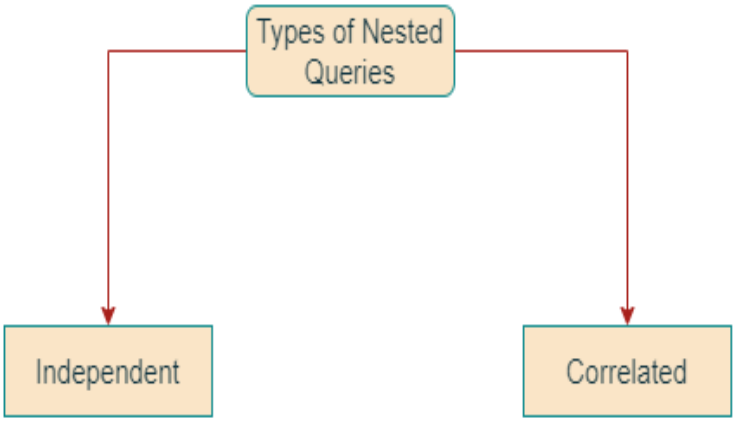
WHERE column1 IN (SELECT column1

FROM table2

WHERE condition);

Types of Nested Queries in SQL

Nested queries can be either correlated or non-correlated



Non-correlated (or Independent) Nested Queries:

Non-correlated (or Independent) subqueries are executed independently of the outer query. Their results are passed to the outer query.

Execution Order in Independent Nested Queries

In independent nested queries, the execution order is from the innermost query to the outer query. An outer query won't be executed until its inner query completes its execution. The outer query uses the result of the inner query.

Example:

IN Operator

This operator checks if a column value in the outer query's result is present in the inner query's result. The final result will have rows that satisfy the IN condition.

Select * from sailors where sid in(select sid from reserves where bid=103);

Correlated Nested Queries:

Correlated subqueries are executed once for each row of the outer query. They use values from the outer query to return result.

Execution Order in Independent Nested Queries

- First, the outer query selects the first row.
- Inner query uses the value of the selected row. It executes its query and returns a result set.
- Outer query uses the result set returned by the inner query. It determines whether the selected row should be included in the final output.
- Steps 2 and 3 are repeated for each row in the outer query's result set.

Example:

EXISTS Operator

This operator checks whether a subquery returns any row. If it returns at least one row. EXISTS operator returns true, and the outer query continues to execute. If the subquery returns no row, the EXISTS operator returns false, and the outer query stops execution.

select emp_name ,deptno from emp where exits(select dno from dept where dept.dno=em.deptno)

SUB QUERIES:

In SQL a Subquery can be simply defined as a query within another query.

- In other words we can say that a Subquery is a query that is embedded in WHERE clause of another SQL query.
- The outer query is called as main query and the inner query is called as sub query.
- The subquery generally executes first and subquery must be enclosed in parentheses.

Syntax:

Select * from table_name where condition expression operator(select column1 from table_name where condition);

Important rules for Subqueries:

1.You can place the subquery in a number of sql clauses.

i. Where clause
ii. Having clause
iii. From clause

2.The expression operator could be equality operator(or) comparision operator such as >,<,<=,>= and IN,BETWEEN,LIKE,etc.

Example:

consider a table STUDENT below.

SNO	SNAME	MARKS
101	RAMA	499
102	SITA	450
103	JANU	350
104	RANI	400
105	SAI	420

SQL>select sname from student where marks=(select max(marks) from student);

STUDENT

SNO	SNAME	DNO
101	HARI	1
102	RAMA	2
103	SAI	3

DEPT

DEPTNO	DNAME
1	CSE
2	AIML
3	IT

Eg: To find the student details who belongs to CSE department.

SQL>select * from student where dno=(select deptno from dept where dname='CSE');

GROUP BY CLAUSE:

The Group by clause in SQL is used to arrange the identical data into groups with the help of some functions (Aggregate functions).

Group by clause is used with the select statement in a Query. The group by clause is placed after the where clause and before the order by clause and having clause if used.

SYNTAX :

SQL> select col1,function_name(col2) from table_name where condition group by col1 Order by col2 ;

Here the function_name is any aggregate function .

EXAMPLE: consider the below table

ENO	ENAME	SAL	DNO
101	Hari	20000	1
102	Ramu	30000	2
103	Hari	40000	1

104	ramu	20000	2
105	sita	30000	3

SQL> select ename , sum(sal) from emp where sum(sal)> 20000 group by ename ;

O/P :

ENAME	SUM(SAL)
Hari	60000
Ramu	50000
Sita	30000

GROUP BY MULTIPLE COLUMNS :

EX:

SQL> select ename , dno , count(*) from emp group by ename , dno ;

O/P:

ENAME	DNO	COUNT(*)
Hari	1	2
Ramu	2	2
Sita	3	1

AGGREGATE FUNCTIONS:

The aggregate functions are count ,sum, avg, max, min

EX: Consider emp table given below.

ENO	ENAME	SAL
101	Deepesh	50000
102	dhanush	60000
103	Srinidhi	60000

104	Anvika	70000
105	Adwik	50000
106	Sekhar	80000

1.COUNT(*):

The count function is used to count the number of rows in a table/relation .

SYNTAX:

SQL> select count(*) from table_name;

EX:

SQL>select count(*) from emp;

O/P:

count(*)

6

The count function also count both NULL values and duplicate values.

SQL> select count(eno) from emp;

O/P:

Count(eno)

6

COUNT(DISTINCT COL_NAME) :

This aggregate function is used to count distinct column names in relation .

SYNTAX :

SQL> select count(distinct column) from table_name;

EX:

SQL> select count(distinct sal) from emp;

O/P:

Count(distinct_sal)

4

2. SUM(COL_NAME) :

The sum function is used to add the values of a particular column.

SYNTAX :

SQL> select sum(col_name) from table_name;

EX: :

SQL> select sum(sal) from emp;

O/P:

sum(sal)

370000

3.AVG(COL_NAME):

This function is used to find the average of particular column in a table/relation

SYNTAX:

SQL> select avg(col_name) from table_name;

EX:

SQL> select avg(sal) from emp;

O/P:

Avg(sal)

61666

4.MAX(COL_NAME):

This function is used to find the highest value of a particular column in table/relation.

SYNTAX:

SQL> select max(col_name) from table_name;

EX:

SQL> select max(sal) from emp;

O/P:

max(sal)

80000

4.MIN(COL_NAME):

This function is used to find the least value of a particular column in table/relation.

SYNTAX:

SQL> select min(col_name) from table_name;

EX:

SQL> select min(sal) from emp;

O/P:

min(sal)

50000

ORDER BY CLAUSE:

The Order by clause is used to sort the records in ascending or descending order .

SYNTAX :

SQL> select col1 ,col2 from table_name order by col1 asc/dec(order);

- The order by keyword sorts the record in ascending order by default .
- To sort the records in descending order we use desc keyword.

consider the below emp table

ENO	ENAME	SAL
101	Deepesh	50000
102	dhanush	60000
103	Srinidhi	60000
104	Anvika	70000
105	Adwik	50000
106	Sekhar	80000

EX-1:

SQL> select * from emp order by ename;

O/P : ENAME

---

hari

ramu

hari

ramu

sita

EX -2:

SQL> select eno, ename , dno order by desc;

ENO	ENAME	DNO
105	sita	3
104	Ramu	2
103	Ramu	2
102	hari	1
101	hari	1

JOIN:

A join is an operation that combines the rows of two or more tables based on related columns. This operation is used for retrieving the data from multiple tables simultaneously using common columns of tables. In this article, we are going to discuss every point about joins.

Join is an operation in DBMS(Database Management System) that combines the row of two or more tables based on related columns between them. The main purpose of Join is to retrieve the data from multiple tables in other words Join is used to perform multi-table queries.

Types of Joins:

There are many types of Joins in SQL. Depending on the use case, you can use different types of SQL JOIN clauses. Here are the frequently used SQL JOIN types.

1. Inner Join:
Inner Join is a join operation in DBMS that combines two or more tables based on related columns and returns only rows that have matching values among tables.

1. Conditional join/Theta join
2. Equi Join

3. Natural Join

1. Conditional Join:

Conditional join or Theta join is a type of inner join in which tables are combined based on the specified condition.
In conditional join, the join condition can include <, >, <=, >=, ≠ operators in addition to the = operator.

Example: Consider student and dept tables

student

SNO	SNAME	DEPTNO
101	SHIVA	1
102	SAI	2

dept

DID	DNAME
1	AIML
5	CSE

SQL Query:
SELECT s.sno,s.sname,s.deptno,d.did,d.dname FROM student s JOIN dept d on s.deptno<d.did.

SNO	SNAME	DEPTNO	DID	DNAME
101	SHIVA	1	5	CSE
102	SAI	2	5	CSE

Explanation: This query joins the tables student and dept and projects attributes sno , sname ,deptno ,did ,dname where condition s.deptno<d.did is satisfied.

2. Equi Join:

Equi Join is a type of Inner join in which we use equivalence('=') condition in the join condition.

Example: Consider student and dept tables

student

SNO	SNAME	DEPTNO
101	SHIVA	1
102	SAI	2

dept

DID	DNAME
1	AIML
5	CSE

SQL Query:

```
SELECT s.sno,s.sname,s.deptno,d.did,d.dname  FROM student s JOIN dept d on s.deptno=d.did;
```

SNO	SNAME	DEPTNO	DID	DNAME
101	SHIVA	1	1	AIML

Explanation: This query joins the tables student and dept and the rows are retrieved based on the equality condition.

3. Natural Join

Natural join is a type of inner join in which we do not need any comparison operators. In natural join, columns should have the same name and domain. There should be at least one common attribute between the two tables.

Example: Consider student and dept tables

student

SNO	SNAME	DEPTNO
101	SHIVA	1
102	SAI	2

dept

DEPTNO	DNAME
1	AIML
2	CSE

SQL Query:

SELECT * FROM student NATURAL JOIN dept;

SNO	SNAME	DEPTNO	DNAME
101	SHIVA	1	CSE
102	SAI	2	CSE

Explanation – Column deptno is available in both tables. Hence we write the deptno column once after combining both tables.

2. Outer Join:

Outer join is a type of join that retrieves matching as well as non-matching records from related tables. These three types of outer join

1. **Left outer join**
2. **Right outer join**
3. **Full outer join**

1. Left Outer Join:

It is also called left join. This type of outer join retrieves all records from the left table and retrieves matching records from the right table.
Example: consider two tables student and dept

student

SNO	SNAME	DEPTNO
101	SHIVA	1
102	SAI	2

dept

DID	DNAME
1	AIML
5	CSE

SQL Query:

SELECT * FROM student LEFT OUTER JOIN dept ON student.deptno=dept.did.

SNO	SNAME	DEPTNO	DID	DNAME
101	SHIVA	1	1	AIML
102	SAI	2		

Explanation: Since we know in the left outer join we take all the columns from the left table (student).In the table dept we can see that there is no match value for deptno 2. so we mark this as NULL.

2. Right Outer Join:

It is also called a right join. This type of outer join retrieves all records from the right table and retrieves matching records from the left table. And for the record which doesn't lies in Left table will be marked as NULL in result Set.

Example: consider the above tables student and dept

SQL Query:

SELECT * FROM student RIGHT OUTER JOIN dept ON student.deptno=dept.did.

SNO	SNAME	DEPTNO	DID	DNAME
101	SHIVA	1	1	AIML
			5	CSE

Explanation: Since we know in the right outer join we take all the columns from the right table (dept) In table student we can see that there is no match value for deptno 5. So we mark this as NULL.

3.Full Outer Join:

FULL JOIN creates the result set by combining the results of both LEFT JOIN and RIGHT JOIN. The result set will contain all the rows from both tables. For the rows for which there is no matching, the result set will contain NULL values.

Example: consider the above two tables student and dept

SQL Query

SELECT * FROM student FULL OUTER JOIN dept ON student.deptno=dept.did.

SNO	SNAME	DEPTNO	DID	DNAME
101	SHIVA	1	1	AIML
102	SAI	2		
			5	CSE

Explanation: Since we know in full outer join we take all the columns from both tables. In the table student and dept we can see that there is no match for deptno 2 and no match for did 5 so we mark this as NULL.

VIEW:

View is a pseudo table or virtual table .View does not store data. It just displays the data.The data is derived from one or more base tables.

Syntax:

CREATE VIEW VIEW_NAME AS SELECT attribute_list FROM tables WHERE condition;

views are of two types.

1.Updatable view

2.Non-updatable view

Example:

SQL> select * from emp;

ENO	ENAME	JOB	SALARY	DNO
501	rama	clerk	2000	201
502	geetha	analyst	5000	202
503	bhaskar	manager	20000	203
504	rama	analyst	6000	203
505	devi	developer	15000	202

1.Updatable view:

SQL> create view emp_view as select eno,ename,job from emp where salary>3000;

View created.

SQL> select * from emp_view;

ENO	ENAME	JOB

```
       502 geetha     salesman
       503 bhaskar    manager
       504 rama       salesman
       505 devi       developer
SQL> update emp_view set job='analyst' where job='salesman';

2 rows updated.
SQL> select * from emp_view;
       ENO ENAME      JOB
---------- ---------- ----------
       502 geetha     analyst
       503 bhaskar    manager
       504 rama       analyst
       505 devi       developer
SQL> select * from emp;
     ENO ENAME      JOB           SALARY        DNO
         ---------- ---------- ---------- ---------- ----------
       501 rama       clerk           2000               201
       502 geetha     analyst         5000               202
       503 bhaskar    manager        20000        203
       504 rama       analyst         6000               203
       505 devi       developer       15000          202
```

2. Non-updatable view: A view is non-updatable if it contains aggregate functions, distinct clause, group by,having,union ,certain joins and if it contains suquery in select list.

Case-1: A view is created using DISTINCT clause is usually Non-updatable.

SQL> create view emp_view1 as select distinct job from emp;

View created.

SQL> select * from emp_view1;

JOB

---

developer

clerk

analyst

manager

SQL> update emp_view1 set job='salesman' where job='analyst';

update emp_view1 set job='salesman' where job='analyst'

*

ERROR at line 1:

ORA-01732: data manipulation operation not legal on this view

DROPING VIEWS:

Syntax:

drop view view_name;

SQL> drop view emp_view;

View dropped.

SQL> select * from emp_view;

select * from emp_view

*

ERROR at line 1:

ORA-00942: table or view does not exist

SET OPERATORS:

Following operators are called as set operators.

1. Union Operator
2. Intersect Operator
3. Minus(Difference) Operator

Condition For Using Set Operators

To use set theory operators on two relations, the two relations must be union compatible.

Union compatible property means-

- Both the relations must have same number of attributes.
- The attribute domains (types of values accepted by attributes) of both the relations must be compatible.

1. Union Operator : The union operation retrieves all different rows from both tables and removes duplicate rows.

Let R and S be two relations. Then-

- R union S is the set of all tuples belonging to either R or S or both.
- In R union S, duplicates are automatically removed.
- Union operation is both commutative and associative.

Example-

Consider the following two relations R and S-

R

ID	Name	Subject
101	Ramya	English
102	Ramu	Maths
103	Sai	Science

Relation S

ID	Name	Subject
101	Ramya	English
104	Hari	French

Relation R union S

ID	Name	Subject
101	Ramya	English
102	Ramu	Maths
103	Sai	Science
104	Hari	French

2. Intersect Operator: The intersection operation retrieves only the common rows between two tables.

Let R and S be two relations. Then-

- R intersect S is the set of all tuples belonging to both R and S.
- In R intersect S, duplicates are automatically removed.
- Intersection operation is both commutative and associative.

Example-

Considering the above two relations R and S.Then, R intersect S is-

Relation R intersect S

ID	Name	Subject
101	Ramya	English

3. Minus (-) : It retrieves the rows from first table which are not present in the second table.

Let R and S be two relations. Then-

- R minus S is the set of all tuples belonging to R and not to S.
- In R minus S, duplicates are automatically removed.
- Difference operation is associative but not commutative.

Example:

Considering the above two relations R and S

Then, R minus S is-

Relation R minus S

ID	Name	Subject
102	Ramu	Maths
103	Sai	Science