Mahatma Gandhi Charitable Trust Managed

Shri Labhubhai Trivedi Institute of Engineering & Technology

CREATING CREATORS - SLTIET

Relational Database Management Systems

RDBMS - Subject Code: 4330702

Prof. Rinkal Umaraniya

CSE Department

SLTIET, Rajkot

CREATING CREATORS - SLTIET

Unit - 2

SQL In built functions and Joins

CREATING CREATORS - SLTIET

Sub Topics:

2.1 Operators
2.2 SQL Functions
2.3 Groupby, Having and Order by clause
2.4 Joins
2.5 Subqueries
2.6 SQL set operators

2.1 Operators

We can define operators as symbols that help us to perform specific mathematical and logical computations on operands.

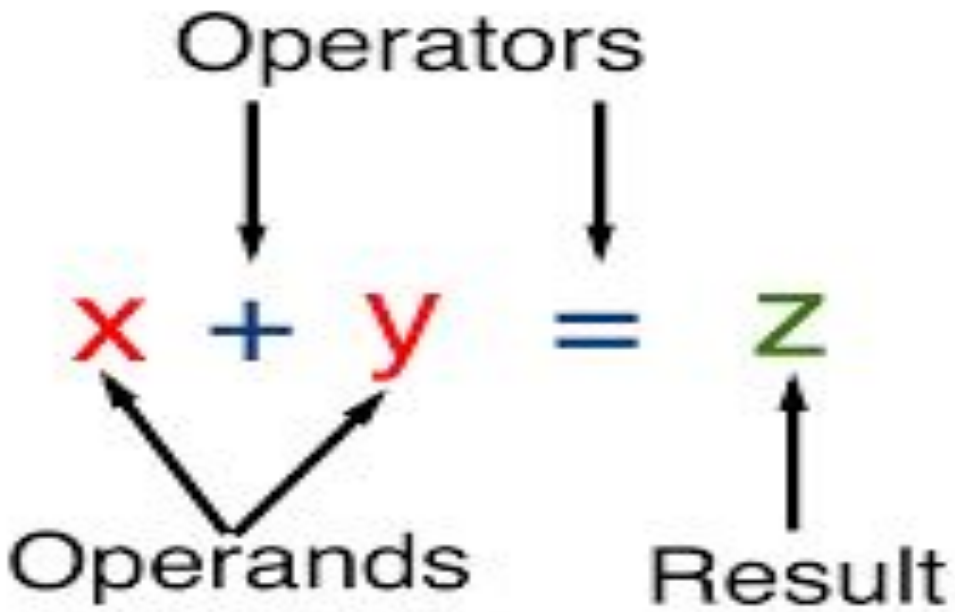


Table Name : employee

emp_id	emp_name	salary	city
101	Mr. Mehta	10000	Rajkot
102	Kavita Sharma	20000	Jamnagar
103	Dhvani Rathod	15000	Ahmedabad
104	Parmar Kevin	30000	Rajkot
105	Abeer Modi	28000	Vadodara

1) Arithmetic Operator

Operator	Operation	Description
+	Addition	The addition is used to perform an addition operation on the data values.
-	Subtraction	This operator is used for the subtraction on the data values.
/	Division	This operator is used for division on data values.
*	Multiplication	This operator is used for multiplication on data values.
%	Modulus	Modulus is used to get the remainder when data is divided by another.

Continue…

Example: SELECT emp_id, emp_name, salary "current salary", salary + 1000 "new salary" FROM employee;

Output :

emp_id	emp_name	current salary	new salary
101	Mr. Mehta	10000	11000
102	Kavita Sharma	20000	21000
103	Dhvani Rathod	15000	16000
104	Parmar Kevin	30000	31000
105	Abeer Modi	28000	29000

2) Comparison Operator

Operator	Operation	Description
=	Equal to	Checks if both operands have equal value, if yes, then returns TRUE
>	Greater than	Return TRUE if the value of the left-hand operand is greater than the right-hand operand or not
<	Less than	Returns TRUE if the value of the left-hand operand is less than the value of the right-hand operand or not
>=	Greater than or equal to	It checks if the value of the left-hand operand is greater than or equal to the value of the right-hand operand, if yes, then returns TRUE

Continue…

Operator	Operation	Description
<=	Less than or equal to	It checks if the value of the left-hand operand is less than or equal to the value of the right-hand operand, if yes, then returns TRUE
<> or !=	Not equal to	Checks if values on either side of the operator are equal or not. Returns TRUE if values are not equal
!>	Not greater than	Used to check if the left-hand operator's value is not greater than or equal to the right-hand operator's value
!<	Not less than	Used to check if the left-hand operator's value is not less than or equal to the right-hand operator's value

;

Continue…

Example: SELECT * FROM employee WHERE salary < 20000;

Output :

emp_id	emp_name	salary	city
101	Mr. Mehta	10000	Rajkot
103	Dhvani Rathod	15000	Ahmedabad

3) Logical Operator

Operator	Description
AND	Logical AND compares between two Booleans as expressions and returns true when both expressions are true.
OR	Logical OR compares between two Booleans as expressions and returns true when one of the expressions is true.
NOT	Not takes a single Boolean as an argument and changes its value from false to true or from true to false.

Continue…

Example : SELECT * FROM employee WHERE salary > 10000 AND salary < 20000;

Output :

emp_id	emp_name	salary	city
103	Dhvani Rathod	15000	Ahmedabad

4) Range Searching Operator

Operator	Description
BETWEEN	- it used to select data that belong to some particular range. - the lower & upper limit value must be linked with keyword AND.

Example : SELECT * FROM employee WHERE salary BETWEEN 20000 AND 30000;

Output :

emp_id	emp_name	salary	city
102	Kavita Sharma	20000	Jamnagar
103	Dhvani Rathod	15000	Ahmedabad
104	Parmar Kevin	30000	Rajkot
105	Abeer Modi	28000	Vadodara

Continue…

Operator	Description
NOT BETWEEN	- it used to select data that not belong to some particular range.

Example : SELECT * FROM employee WHERE salary NOT BETWEEN 19000 AND 31000;

Output :

emp_id	emp_name	salary	city
101	Mr. Mehta	10000	Rajkot
103	Dhvani Rathod	15000	Ahmedabad

5) Set Searching Operator

Operator	Description
IN	- it used to select data that belong to some particular set of value. - it can be used with numerical, character and date datatype

Example : SELECT * FROM employee WHERE city IN ('Rajkot', 'Jamnagar');

Output :

emp_id	emp_name	salary	city
101	Mr. Mehta	10000	Rajkot
102	Kavita Sharma	20000	Jamnagar
104	Parmar Kevin	30000	Rajkot

Continue…

Operator	Description
NOT IN	- it used to select data that do not belong to some particular set of value. - it can be used with numerical, character and date datatype

Example : SELECT * FROM employee WHERE city NOT IN ('Rajkot', 'Jamnagar');

Output :

emp_id	emp_name	salary	city
103	Dhvani Rathod	15000	Ahmedabad
105	Abeer Modi	28000	Vadodara

6) Character Operator

Operator	Description
LIKE	- it used to select raw contain value(string) similar to given pattern.
|| (Concatenation)	- it is used to concatenation means combines two string

Example : SELECT * FROM employee WHERE city LIKE 'R%';

Output :

emp_id	emp_name	salary	city
101	Mr. Mehta	10000	Rajkot
104	Parmar Kevin	30000	Rajkot

Continue…

Example : SELECT emp_name || ', ' || city "EmpName, City"
FROM employee ;

Output :

EmpName, city
Mr. Mehta, Rajkot
Kavita Sharma, Jamnagar
Dhvani Rathod, Ahmedabad
Parmar Kevin, Rajkot
Abeer Modi, Vadodara

Continue…

Like Operator table :

Pattern	Meaning
‘a%’	Match strings that start with ‘a’
‘%a’	Match strings with end with ‘a’
‘a%t’	Match strings that contain the start with ‘a’ and end with ‘t’.
‘%wow%’	Match strings that contain the substring ‘wow’ in them at any position.
‘_wow%’	Match strings that contain the substring ‘wow’ in them at the second position.
‘_a%’	Match strings that contain ‘a’ at the second position.
‘a_ _%’	Match strings that start with ‘a and contain at least 2 more characters.

2.2 SQL Functions:

SQL supplies a rich library of in-built functions which can be employed for various tasks.

1) Aggregate Function
2) Date Function
3) Numeric Function
4) Character Function
5) Conversion Function

1. Aggregate Functions

Function Name	Description
MAX(colName)	Return maximum value.
MIN(colName)	Return minimum value.
SUM([Distinct | All] colName)	Return sum of values. (Distinct – duplicate value not allow)
AVG([Distinct | All] colName)	Return average of values. (Distinct – duplicate value not allow)
COUNT(*)	Return number of rows.
COUNT([Distinct | All] colName)	Return number of rows not having a null value in given columnName.

Continue…

Consider below table for aggregate functions.

Table : Employee

emp_id	emp_name	salary	city
101	Kavita Sharma	20000	Jamnagar
102	Dhvani Rathod	15000	Ahmedabad
103	Parmar Kevin	30000	Rajkot
104	Mr. Mehta	15000	Rajkot
105	Abeer Modi	28000	Vadodara

Continue…

Example : Select SUM (salary) “total_salary”,

SUM (Distinct salary) ”total_distinct_salary” from employee;

Output :

total_salary	total_distinct_salary
108000	93000

2. Date Functions

i) ADD_MONTHS

- The add_months function returns a new date after adding n months.

Syntax: ADD_MONTHS (Date1, n)

- Date1 is the starting date (before the n months have been added).
- n is the number of months to add to date1.

Continue…

Example:

- SELECT ADD_MONTHS ('01-Aug-23', 3) FROM dual;
- Output: '01-Nov-23‘

- SELECT ADD_MONTHS ('01-Aug-23', -3) FROM dual;
- Output: '01-May-23'

Continue…

ii) MONTHS_BETWEEN

- Returns the months in between (separating) two given dates.

- **Syntax: MONTHS_BETWEEN (Date1, Date2)**

Date1 and Date2 are the dates used to calculate the number of months.

Example:

- SELECT MONTHS_BETWEEN (‘31-MAR-23‘, ‘31-DEC-22‘) FROM dual;
- Output: 3

- SELECT MONTHS_BETWEEN (‘2003/07/23‘, ‘2003/07/23‘) FROM dual;
- Output: 0

Continue…

iii) ROUND

- ROUND function returns a date rounded to a specific unit of measure.

Syntax: ROUND (date, [format]);

Date is the date to round and Format is unit of measure to apply for rounding.

Example:

ROUND (TO_DATE ('22-AUG-23'),'YEAR') would return '01-JAN-24'

ROUND (TO_DATE ('22-AUG-23'),'Q') would return '01-OCT-23'

ROUND (TO_DATE ('22-AUG-23'),'MONTH') would return '01-SEP-23'

ROUND (TO_DATE ('22-AUG-23'),'DDD') would return '22-AUG-23'

ROUND (TO_DATE ('22-AUG-23'),'DAY') would return '23-AUG-23'

Continue…

iv) TRUNCATE

- The TRUNC function returns a date truncated to a specific unit of measure.

Syntax: TRUNC (Date, [Format])

Date is the date to truncate and Format is the unit of measure to apply for truncating.

Example:

TRUNC (TO_DATE ('22-AUG-23'), 'YEAR') would return '01-JAN-23'

TRUNC (TO_DATE ('22-AUG-23') 'MONTH') would return '01-AUG-23'

3. Numeric Functions

Function Name	Description	Example	Output
ABS(x)	Absolute value of the number 'x'	Select ABS(-15) from dual;	15
SQRT(x)	Return square root of ‘n’	Select SQRT(64) from dual;	8
POWER(m,n)	Return m raised to nth Power	Select POWER(2,3) from dual;	8
MOD(m,n)	Return remainder of m divided by n operations	Select MOD(10,2) from dual;	0
CEIL(x)	Integer value that is Greater than or equal to the number 'x	Select CEIL(25.57) from dual;	26
FLOOR(x)	Integer value that is Less than or equal to the number 'x'	Select FLOOR(25.57) from dual;	25

Continue…

Function Name	Description	Example	Output
TRUNC (x,y)	Truncates value of number 'x' up to 'y' decimal places	- Select TRUNC(25.573,2) from dual; - Select TRUNC(25.573) from dual;	- 25.57 - 25
ROUND (x,y)	Rounded off value of the number 'x' up to the number 'y' decimal places	- Select ROUND(25.57,2) from dual; - Select ROUND(25.57) from dual;	- 25.57 - 26
EXP(n)	Return e raised to the nth power	Select EXP(2) from dual;	7.3890561
LN(n)	Return natural or base e, the logarithm of n	Select LN(10) from dual;	2.30258509
LOG(b, n)	Return Logarithm of n in the base of b	Select LOG(10.5) from dual;	0.698970004

Continue…

Function Name	Description	Example	Output
COS, SIN, TAN	Return trigonometric value of n, where is angle.	Select COS(3.1415), SIN(3.1415), TAN(3.1415) from dual;	-1 0.000092654 -0.00009565
SIGN	Return -1 if n is negative; 0 if n is zero; and 1 if n is positive;	Select SIGN(-25), SIGN(0), SIGN(25) from dual;	-1 0 1

4. Character Functions

Function Name	Example	Output
LOWER(string_value)	Select LOWER('Good Morning') from dual;	good morning
UPPER(string_value)	Select UPPER('Good Morning') from dual;	GOOD MORNING
INITCAP(string_value)	Select INITCAP('Good Morning') from dual;	Good Morning
LTRIM(string_value, trim_text)	Select LTRIM ('Good Morning', 'Good') from dual;	Morning
RTRIM (string_value, trim_text)	Select RTRIM ('Good Morning', 'Morning') from dual;	Good
TRIM (trim_text FROM string_value)	Select TRIM ('o' FROM 'Good Morning') from dual;	Gd Mrning

Continue…

Function Name	Example	Output
SUBSTR (string_value, m, n)	Select SUBSTR ('Good Morning', 5, 7)from dual;	Morning
LENGTH (string_value)	Select LENGTH ('Good Morning')from dual;	12
LPAD (string_value, n, pad_value)	Select LPAD ('Good', 6, '*')from dual;	**Good
RPAD (string_value, n, pad_value)	Select RPAD ('Good', 6, '*')from dual;	Good**
INSTR(Column|Expresssion, 'String',[m],[n])	Select INSTR('Google apps are great application','app',1,2)from duel;	23
REPLACE(Text,search_string, replacement_string)	Select REPLACE ('Data Management', 'Data', 'Database') from duel;	Database Management

5. Conversion Functions

(a) Implicit Data Type Conversion

In this type of conversion the data is converted from one type to another implicitly (by itself/automatically).

From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
DATE	VARCHAR2
NUMBER	VARCHAR2

Continue…

Example:

Table Name : employees

Emp_ID	First_Name	Last_Name	Salary	Month_Hired	City
101	Rahul	Kotak	15000	01/05/2003	Rajkot
102	Karan	Varma	16000	05/08/2001	Ahmedabad
103	Vishva	Shah	25000	15/07/2005	Jamnagar
104	Ruhi	Modi	10000	10/12/2020	Rajkot

Continue…

Example 1.

```
SELECT Emp_ID, First_Name, Salary
FROM employees
WHERE Salary > 15000;
```

Emp_ID	First_Name	Salary
102	Karan	16000
103	Vishva	25000

Example 2.

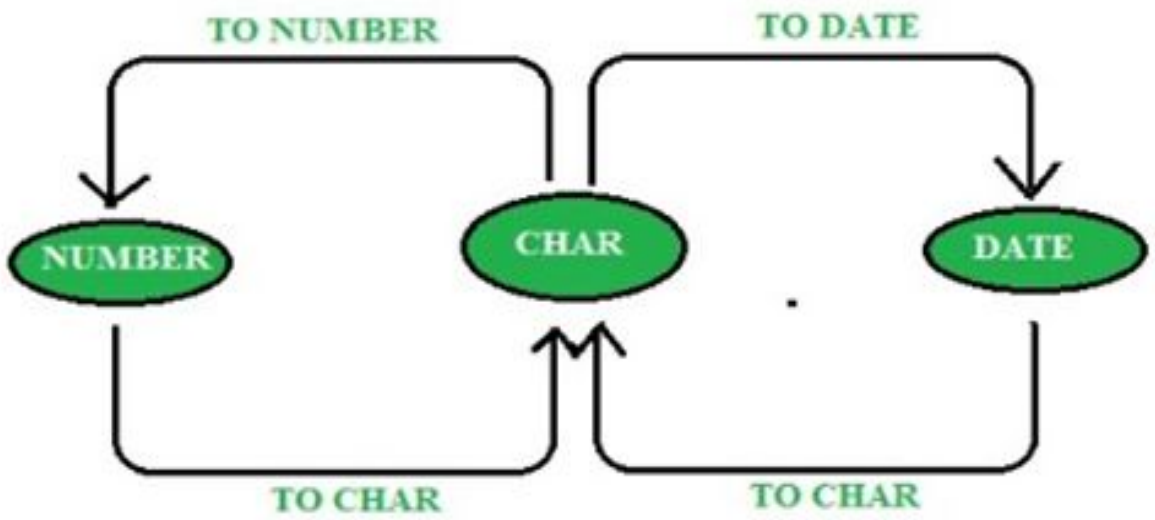
```
SELECT Emp_ID, First_Name, Salary
FROM employees
WHERE Salary > '15000';
```

Emp_ID	First_Name	Salary
102	Karan	16000
103	Vishva	25000

Here we see the output of both examples came out to be same, in the 2nd example using '15000' as text, it is automatically converted into int data type.

Continue…

(b) Explicit Data Type Conversion



Explicit Data Type Converson

Continue…

1. Using the TO_CHAR **Function with Dates :**

SYNTAX : TO_CHAR(date, 'format_model')

The format model:

- Must be enclosed in single quotation marks and is case sensitive
- Can include any valid date format element
- Is separated from the date value by a comma

Continue…

EXAMPLE :

SELECT Emp_ID, TO_CHAR(hire_date, 'MM/YY') Month_Hired

FROM employee

WHERE Last_Name = 'Shah';

Output:

Emp_ID	Month_Hired
103	07/05

Elements of the Date Format Model :

YYYY	Full year in Numbers
YEAR	Year spelled out
MM	Two digit value for month
MONTH	Full name of the month
MON	Three Letter abbreviation of the month
DY	Three letter abbreviation of the day of the week
DAY	Full Name of the Day
DD	Numeric day of the month

Continue…

2. Using the TO_CHAR Function with Numbers :

SYNTAX : TO_CHAR(number, 'format_model')

EXAMPLE :

```
SELECT TO_CHAR(Salary,  '$99,999.00') Salary
FROM employee
WHERE Last_Name = 'Varma';
```

OUTPUT :

Salary
$16000.00

Continue…

These are some of the format elements you can use with the TO_CHAR function to display a number value as a character :

9	Represent a number
0	Forces a zero to be displayed
$	places a floating dollar sign
L	Uses the floating local currency symbol
.	Print a decimal point
,	Prints a Thousand indicator

2.3 Group by, Having and Order by clause

Example:

Table Name : employees

ID	Name	Dept	Age	Salary	Location
100	Ramesh	Electrical	24	25000	Bangalore
101	Hritik	Electronics	28	35000	Bangalore
102	Harsha	Computer	28	35000	Mysore
103	Soumya	Electronics	22	20000	Bangalore
104	Priya	Info Tech	25	30000	Mangalore

1) SQL Group By clause

The SQL GROUP BY Clause is used along with the group functions to retrieve data grouped according to one or more columns.

Syntax: Select column1, column2……. columnN, Aggregate Function (argument)
from TableName Group By column1, column2….. columnN;

Example:

if you want to know the total amount of salary spent on each department, the query would be:

Continue…

SELECT Dept, **SUM** (Salary) “Total Salary” **FROM** employee **GROUP BY** Dept;

Output:

Dept	Total Salary
Electrical	25000
Electronics	55000
Computer	35000
Info Tech	30000

2) SQL Order By clause

The ORDER BY clause is used in a SELECT statement to sort results either in ascending or descending order. By default is consider ascending order of any data.

Syntax: Select column1, column2……. columnN From TableName ORDER BY column1, column2……. columnN;

Continue…

Example: **SELECT** Name, Salary **FROM** employee **ORDER BY** Salary;

Output:

Name	Salary
Soumya	20000
Ramesh	25000
Priya	30000
Hritik	35000
Harsha	35000

Continue…

Example: **SELECT** Name, Salary **FROM** employee **ORDER BY** Salary **desc**;

Output:

Name	Salary
Hritik	35000
Harsha	35000
Priya	30000
Ramesh	25000
Soumya	20000

3) SQL Having By clause

- Having clause is used to filter data based on the group functions. This is similar to WHERE condition but is used with group functions.
- Group functions cannot be used in WHERE Clause but can be used in HAVING clause.
- The HAVING clause must follow the GROUP BY clause in a query.

Syntax:

SELECT column1, column2 FROM TableName

GROUP BY column HAVING [conditions];

Continue…

Example:

SELECT Dept, **SUM** (Salary) **FROM** employee **GROUP BY** Dept **HAVING SUM** (Salary) > 25000;

Output:

Dept	Salary
Electronics	55000
Computer	35000
Info Tech	30000

2.4 Joins

- SQL Joins are used to relate information in different tables.
- A Join condition is a part of the sql query that retrieves rows from two or more tables.
- A SQL Join condition is used in the SQL WHERE Clause of select, update, delete statements.
- Let‘s use the below two tables to explain the sql join conditions.

Continue…

1) Table Name : Products

Product_ID	Product_Name	Supplier_Name	Price
100	Television	Sony	30000/-
101	Laptop	HP	65000/-
102	Mobile	Samsung	12000/-
103	Washing Machine	Bosch	20000/-
104	Refrigerator	LG	17000/-

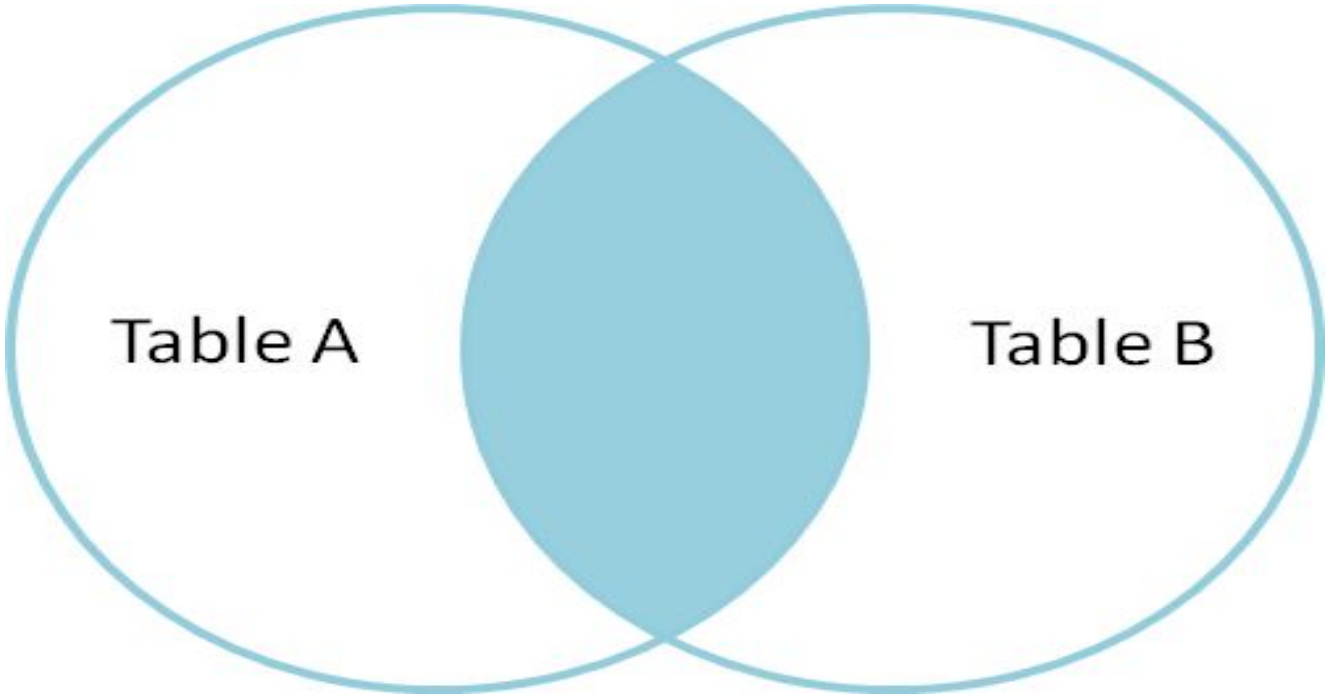
Continue…

2) Table Name : OrderItems

Order_ID	Product_ID	Total_Unit	Customer
1201	103	5	JK Electronics
1202	104	10	Shah ELectronics
1203	110	31	K.K. Mall
1204	101	50	TCS

1) Inner Join

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. i.e value of the common field will be the same.



Continue…

Syntex :

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
```

where, table1: First table

table2: Second table

matching_column:Column common to both tables.

Continue…

Example :

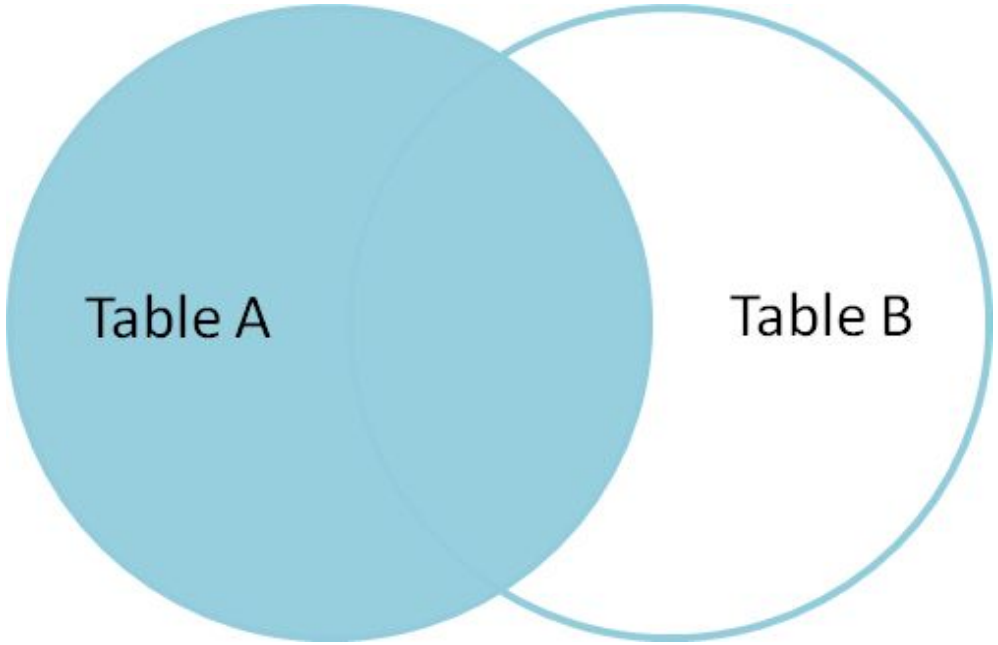
```
SELECT Products.Product_ID, Products.Product_Name, OrderItems.Customer
FROM Products
INNER JOIN OrderItems
ON Products.Product_ID = OrderItems.Product_ID;
```

Output :

Product_ID	Products_Name	Customer
103	Washing Machine	JK Electronics
104	Refrigerator	Shah ELectronics
101	Laptop	TCS

2) Left- Join

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.



Continue…

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

Continue…

Example :

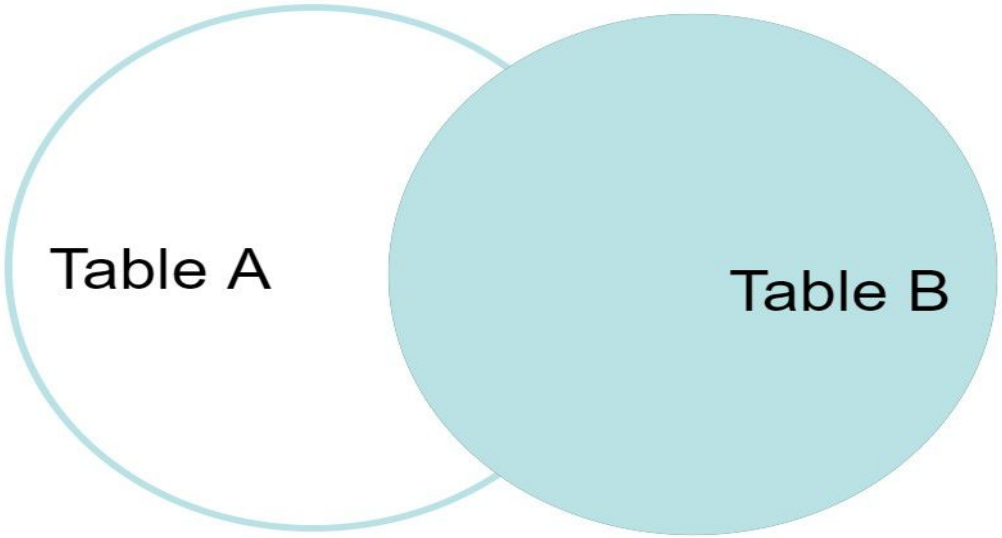
```
SELECT Products.Product_Name,OrderItems.Order_ID
FROM Products
LEFT JOIN OrderItems
ON = ON Products.Product_ID = OrderItems.Product_ID;
```

Output :

Product_Name	Order_ID
Television	NULL
Laptop	1204
Mobile	NULL
Washing Machine	1201
Refrigerator	1202

3) Right Join

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.



Continue…

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

Continue…

Example :

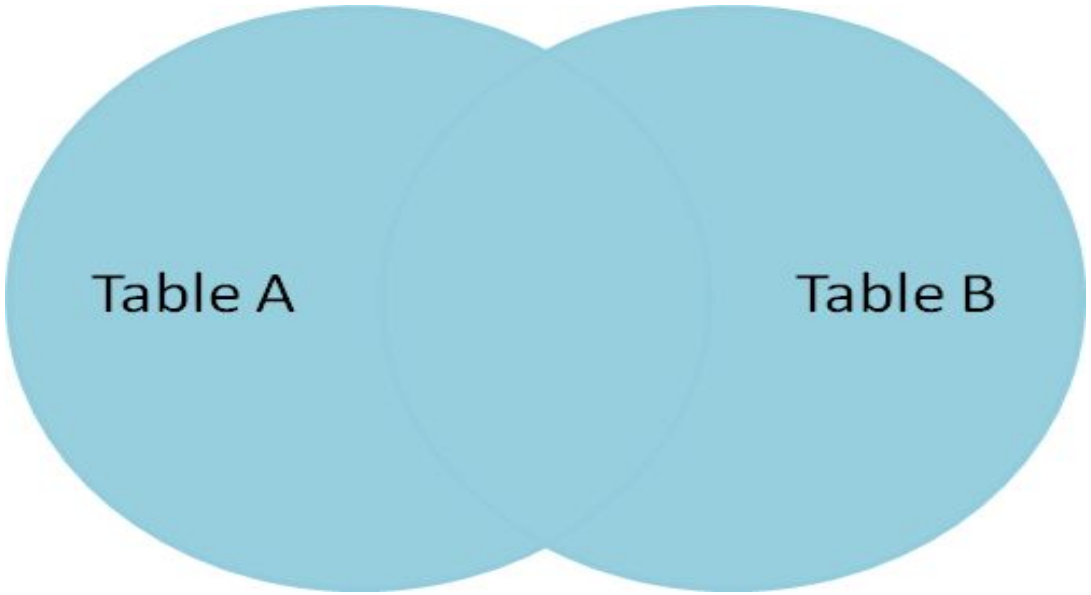
```
SELECT Products.Product_Name,OrderItems.Order_ID
FROM Products
RIGHT JOIN OrderItems
ON Products.Product_ID = OrderItems.Product_ID;
```

Output :

Product_Name	Order_ID
Washing Machine	1201
Refrigerator	1202
NULL	1203
Laptop	1204

4) Full Join

The FULL JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.FULL JOIN is also known as FULL OUTER JOIN.



Continue…

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....S
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```

Example :

```
SELECT Products.Product_Name,OrderItems.Order_ID
FROM Products
FULL JOIN OrderItems
ON Products.Product_ID = OrderItems.Product_ID;
```

Continue…

Output :

Product_Name	Order_ID
Television	NULL
Laptop	1204
Mobile	NULL
Washing Machine	1201
Refrigerator	1202
NULL	1203

5) Self Join

Sometimes, it is necessary to join a table with itself, like joining two separate tables. Such type of join is called self-join.

Self Join is similar to inner join, except that a table is joined with itself. In self-join each row of a table is combined with other rows of the same table to produce a result.

Syntax:

```
SELECT a.column1, b.column2
FROM table_name a,  table_name b
where some_condition;
```

Continue…

Example :

Table : employee

emp_id	e_name	salary	city
101	Rahul	8000	Jamnagar
102	Komal	5000	Vadodara
103	Dev	10000	Vadodara
104	Kriti	3000	Rajkot

Continue…

Input :
```
SELECT a.emp_id, b.e_name as Earn_High,

a.e_name as Earn_Less,

a.salary as Lower_Salary

FROM employee a, employee b

WHERE a.salary < b.salary;
```

Continue…

Output :

emp_id	Earn_High	Earn_Less	Lower_Salary
102	Rahul	Komal	5000
104	Rahul	Kriti	3000
104	Komal	Kriti	3000
101	Dev	Rahul	8000
102	Dev	Komal	5000
104	Dev	Kriti	3000

2.5 Subqueries

- A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.
- A Subquery is used to return data that will be used in the main query.
- Subquery can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, >=, <=, IN, BETWEEN etc.

Continue…

There are a few rules that Sub queries must follow:

- Subquery must be enclosed within parentheses.
- A Subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the Subquery to compare its selected columns.
- An ORDER BY cannot be used in a Subquery, although the main query can use an ORDER BY.
- The GROUP BY can be used to perform the same function as the ORDER BY in a Subquery.
- Subquery that return more than one row can only be used with multiple value operators, such as the IN operator.
- The BETWEEN operator cannot be used with a Subquery; however, the BETWEEN operator can be used within the Subquery.

Continue…

Example :

Customer:

cid	name
C01	Riya
C02	Jiya
C03	Diya
C04	Taral
C05	Saral

Account_Holder

cid	Aco
C01	A01
C02	A02
C03	A03
C04	A04
C05	A05
C02	A04

Account

ano	balance	bname
A01	5000	vvn
A02	6000	ksad
A03	7000	Anand
A04	8000	ksad
A05	6000	vvn

1) Single Raw Subquery

Single Raw Sub Query is used when the outer query's results are based on a singleunknown value. Although this query type is formally called "Single-Raw", the name implies that the query returns multiple columns but only one raw of results.

Example:

Select balance, bname from Account

Where balance = (Select MIN (balance) from Account);

Output :

balance	bname
5000	vvn

2) Multiple Raw Subquery

Multiple raw subquery are nested queries that can return more than one raw of result to the parent query. Multiple raw subqueries are used most commonly in WHERE and HAVING clauses. Since it returns multiple raw, it must be handled by set comparison operators (IN, NOT IN, ALL, ANY, EXISTS, NOT EXISTS).

Example:

Find out balance of account that belong to customer ‘Jiya’.

Continue…

```
SELECT balance FROM Account
WHERE ano IN ( SELECT ano FROM Account_Holder
        WHERE cid IN ( SELECT cid  FROM Customer
                WHERE name='Jiya'
                )
        );
```

Output :

balance
6000
8000

Continue…

Example :

-> SQL query can be rewritten using IN operator like below.

Select Emp_Name, Dept_ID from Employee

Where Dept_ID IN (Select Dept_ID from Employee where City = ‘Rajkot’);

Output :

Emp_Name	Dept_ID
Rakesh Sharma	2205
Sanket Joshi	2208

3) Correlated Raw Subquery

As opposed to a regular subquery, where the outer query depends on values provided by the inner query, a **correlated subquery is one where the inner query depends on values provided by the outer query**. This means that in a correlated subquery, the inner query is executed repeatedly, once for each row that might be selected by the outer query.

Correlated subqueries can produce result tables that answer complex management questions.

Example :

Find the account having the maximum balance in the branch to which it belongs. Display account number, balance and branch name for such accounts.

Continue…

```
SELECT ano, balance, bname FROM Account Acc
WHERE balance IN (
        SELECT MAX(balance) FROM Account
        WHERE bname = Acc.bname
            );
```

Output :

ano	balance	bname
A03	7000	anand
A04	8000	ksad
A05	6000	vvn

2.6 SQL Set Operations

SET operations are special type of operators which are used to combine the result of two queries.

Operators covered under 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. Rules are as follows:

1. The number and order of columns must be the same.
2. Data types must be compatible.

Continue…

Let us see each of the SET operators in more detail with the help of examples.

e1_employee

id	name	department	salary
1	Ravi	Admin	30000
2	Aakash	Civil	40000
3	Palak	H & R	20000
4	Arjun	Computer	50000

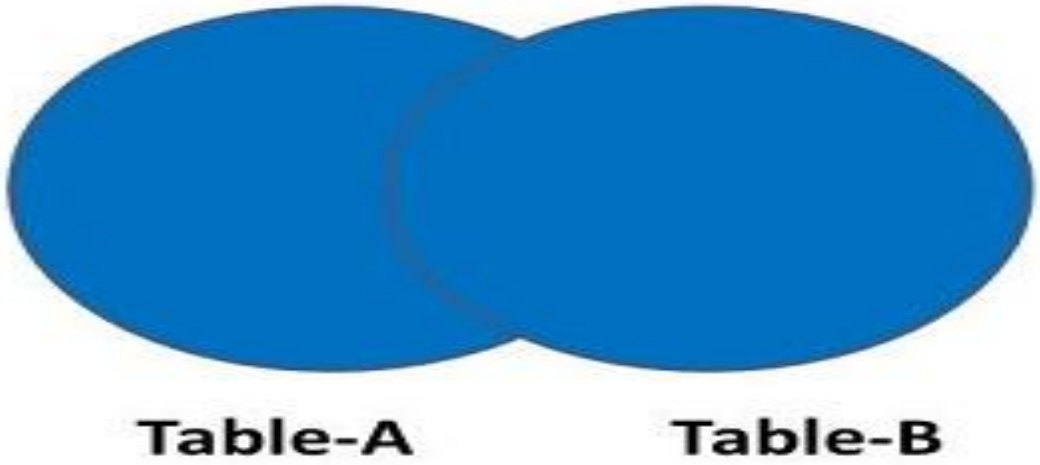
e2_employee

id	name	department	salary
1	Pooja	Account	35000
2	Aakash	Civil	40000
3	Raj	Marketing	30000
4	Arjun	Computer	50000

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.



Union Operation

Continue…

Example :

SELECT *__FROM__ e1_employee **UNION SELECT** *__FROM__ e2_employee;

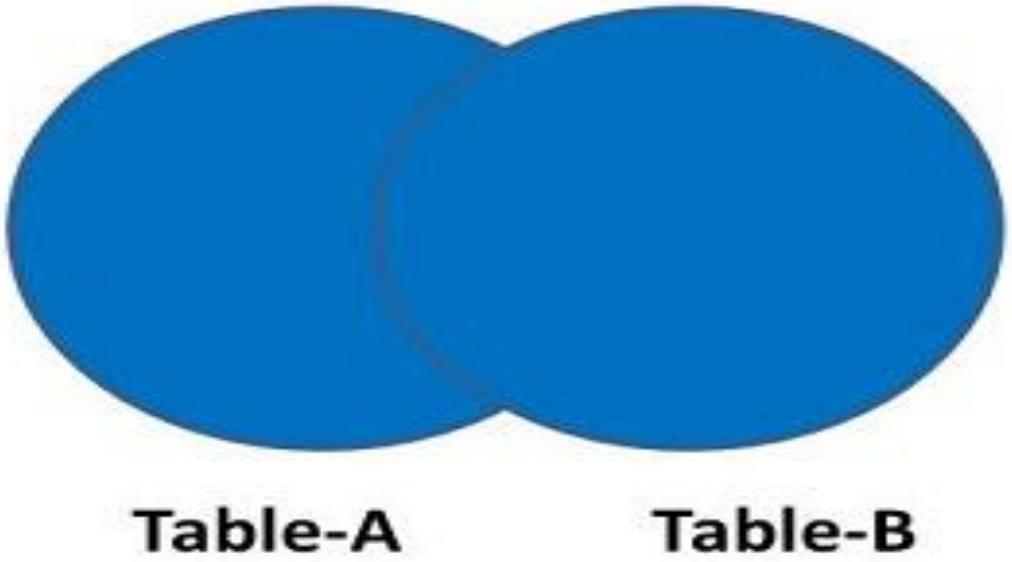
Output :

id	name	department	salary
1	Ravi	Admin	30000
2	Aakash	Civil	40000
3	Palak	H & R	20000
4	Arjun	Computer	50000
1	Pooja	Account	35000
3	Raj	Marketing	30000

2) UNION ALL

This operator combines all the records from both the queries.

Duplicate rows will be not be eliminated from the results obtained after performing the UNION ALL operation.



Continue…

Example :

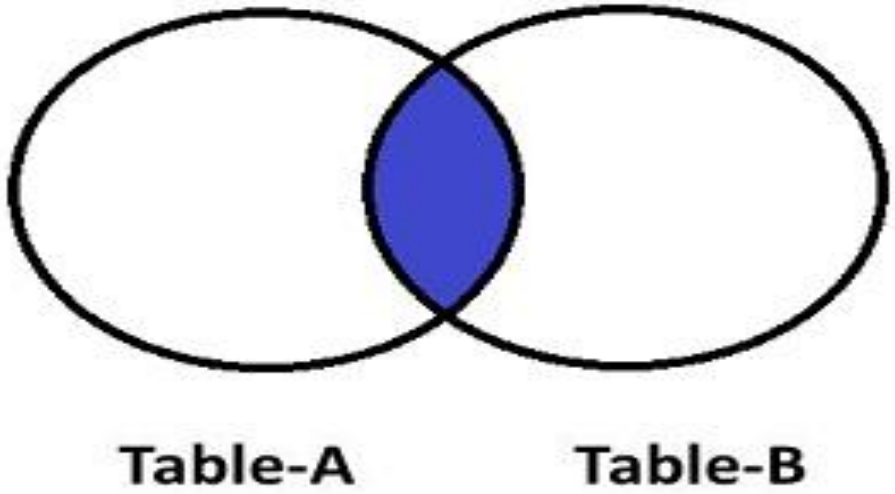
SELECT *FROM e1_employee **UNION ALL SELECT *FROM** e2_employee;

Output :

id	name	department	salary
1	Ravi	Admin	30000
2	Aakash	Civil	40000
3	Palak	H & R	20000
4	Arjun	Computer	50000
1	Pooja	Account	35000
2	Aakash	Civil	40000
3	Raj	Marketing	30000
4	Arjun	Computer	50000

3) INTERSECT

It is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements.



Intersection Operation

Continue…

Example :

SELECT ***FROM** e1_employee **INTERSECT SELECT** ***FROM** e2_employee;

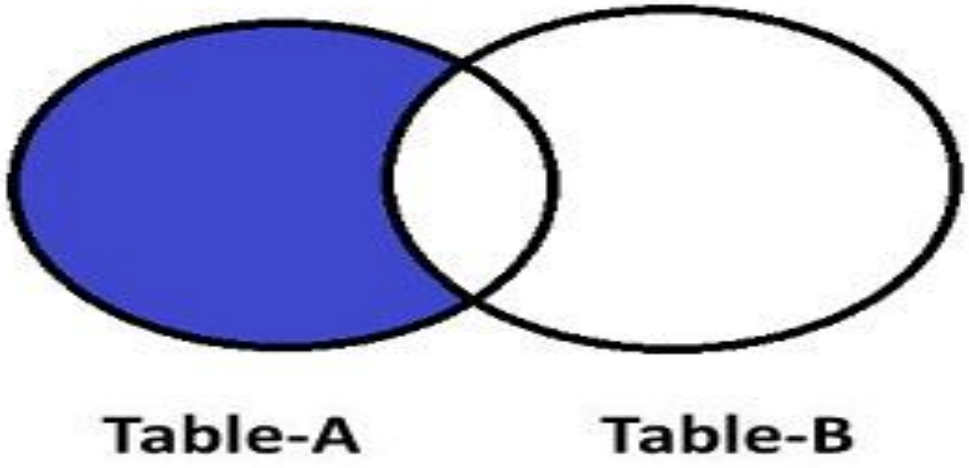
Output :

id	name	department	salary
2	Aakash	Civil	40000
4	Arjun	Computer	50000

We have performed intersect operation between both the tables, so only the common records from both the tables are displayed.

4) MINUS

It displays the rows which are present in the first query but absent in the second query with no duplicates.



Minus Operation

Continue…

Example :

SELECT *FROM e1_employee **MINUS SELECT *FROM** e2_employee;

Output :

id	name	department	salary
1	Ravi	Admin	30000
3	Palak	H & R	20000

We have performed Minus operation between both the tables, so only the unmatched records from both the tables are displayed. It display only first table data with no duplicate.

Questions

Q-1) Explain DBMS Operators: (a) Arithmetic (b) Comparison (c) Logical
(Any of one with example)

Q-2) Explain any of one SQL Function with example.

Q-3) Make a employee table as per below columns.
emp_id, emp_name, salary, dept_id, city – apply Group by, Order by and Having clause on the table.

Questions

Q-4) Make student and college tables as per below.

1) student – stu_id, stu_name, dob, branch, sem, city

2) college - col_id, stu_id, col_name, dept, seats

Apply all joins on above table and write output.

Q-5) Explain UNION, INTERSECT and MINUS with suitable example.

SLTIET
आत्मदीपोभव

Thank You

CREATING CREATORS - SLTIET