Day-6

Covered Areas :—

Inbuilt Fn → Numeric Fn
→ String
→ Date
→ Aggregate

① Numeric Function

* ABS() → Return Absolute Value → Remove -ve sign

```
SELECT ABS (8.78);
8.78
SELECT ABS (-8.78);
8.78
```

* ROUND() → Return roundoff value

```
SELECT ROUND (34.56);
35
SELECT ROUND (34.46);
34
```

>= 5 Upper Number (Ex 1)
< 5 Lower —"— (Ex 2)

ROUND(No, Place) ← It also comes with parameter.

```
SELECT ROUND (234.34334, 2);
234.34
SELECT ROUND (234.34534, 2);
234.35
```

→ Places upto which you want to roundoff. OR Precision

* 2nd value of Round() could be -ve also ROUND (No., -ve Precision)

```
SELECT ROUND (234.5, -2);
200
```

Jumps towards left we have got 2 & then 34 < 50 ∴ round off to lower Position ∴ 200

* CEIL()

Returns the smallest integer value not less than the Argument.

```
SELECT CEIL (254.53);
255
SELECT CEIL (254.2);
255
```

```
SELECT CEIL (23.03);
24
```

Here if we have something after decimal it will always return the upper number

* FLOOR()

Returns the numbers before decimal pt irrespective of value after decimal

```
SELECT FLOOR (254.54);
254
SELECT FLOOR (23.03);
23
```

* TRUNCATE() → It specify number of the decimal places.

```
TRUNCATE (NO., Position)
SELECT TRUNCATE (234.556, 2);
234.55
SELECT TRUNCATE (345.345345, 4);
345.3453
SELECT TRUNCATE (456.234, 0);
456
```

* DIV → Divide & returns Quotient without decimal.

~~DIV (No, ,~~

Number DIV Divisor

2335 DIV 34

value without decimal

4 DIV 3

1

10 DIV 3

3

2 DIV 3

0

* MOD : Modulus → Returns remainder

~~5 MOD 2~~ SELECT MOD (5, 2);

1 1

~~11 MOD 10~~ SELECT MOD (10, 3);

1

* POWER → Returns a no. raise to the power

SELECT POWER (2, 3)

8

SELECT POWER (3, 3)

27

* SQRT : Returns square root of a number.

```
SELECT SQRT(4);
   2
SELECT SQRT(3);
  1.714
```

(2) String Functions :-

* LENGTH ('STRING') → Returns length of string passed as argument

```
SELECT LENGTH ('STRING');
   6
SELECT LENGTH (' ');
   1
SELECT LENGTH ('');
   0
```

* UPPER ('STRING') → Returns the changed string to upper case.

```
SELECT UPPER ('Rinkesh');
  RINKESH
```

* LOWER ('STRING') → Returns the changed string to lower case.

```
SELECT LOWER ('RINKESH');
  rinkesh
```

SELECT UPPER (First_Name) FROM EMPLOYEES LIMIT 4;

SELECT UPPER (First_Name) FNAME, LOWER (Last_Name) LNAME
FROM Employees LIMIT 4;

FNAME	LNAME
NAME 1	Sname1
NAME 2	Sname2
NAME 3	Sname3
NAME 4	Sname4

TRUNCATE

SELECT (Sal * 1.3454343) SAL, From Employees LIMIT 2;

* CONCAT :- Returns a string after joining 2 strings.

CONCAT (n no. of strings.)

SELECT (CONCAT ('RAHUL', 'BANSAL'),

SELECT CONCAT (UPPER (First_Name), UPPER (Last_Name)) AS NAME,
FLOOR

We can use Function inside Function

UPPER (CONCAT ('String', 'String1', ...))

* SUBSTR() → Returns substring from string

SUBSTR('string', from which position, length of string)

SELECT 'SUBSTR ('RINKESH', 3, 2);

NK

SELECT CONCAT (UPPER (SUBSTR ('Rinkesh', 1, 1)),
LOWER (SUBSTR ('Rinkesh', 2, 6)) From
Employee ~~Limit~~ 4;

Rinkesh [Exact Camel Case]

SELECT CONCAT (UPPER (SUBSTR ('Rinkesh', 1, 1),
LOWER (SUBSTR ('Rinkesh', 2, LENGTH(
'Rinkesh'))) From employees ~~limit~~;

Rinkesh.

* TRIM() → Removes starting & ending spaces from string.

SELECT TRIM (' Rinkesh ');

Rinkesh

* LPAD → Left Padding

LPAD ('String', LENGTH of string, string need to be replaced for extra position)

For ex I have 'Rinkesh' of length 7 & I want to LPad it with 15

SELECT LPAD ('Rinkesh', 15, '*');

[15 - length of string]

* * * * * * * * Rinkesh

* RPAD → Right Padding

RPAD ('String', Length, String to be repked)

For ex → I have 'Rinkesh' of length 7 & I want to RPAD it with 15.

SELECT RPAD('Rinkesh', 15, '*');

Rinkesh* * * * * * * *

SELEM LPAD ('*', FLOOR (Sal/1000), '*') From employes limit 20;

Sal. to be printed * based on the

Every * = Rs. 1000

For ex → 24000

will be printed as * * * * ..... 24 times

→ 2000 → * *

→ 10000 → * * * * * * * * * *

LPAD (salary, Floor (salary/1000), '*'),

*****24000

* Replace → used to replace certain strings ~~into~~ with defined string.

Replace ('String', String to replace, String with which i need to replace)

SELECT REPLACE ('Rinkesh', 'ke', 'k');

Rinksh

* Repeat ('Rinkesh', 3)

Rinkesh Rinkesh Rinkesh

③ DATE Functions :-

* NOW() → Returns todays date & time.

SELECT NOW();

* CURDATE() → Returns todays date

SELECT CURDATE();

③. DATE_FORMAT()

It converts a date into specific format.

SELECT Date_FORMAT(CURDATE(), '%d');

28

Syntax → DATE_FORMAT(Date , FORMAT).

If i need to find a day

%d → day of the month 28/10/2020

28

%D → day with the english Suffix

28th

19th

21st

%m → month

%M → Month Name

%y → returns 2 digit from the yrs

%Y → returns years.

SELECT Date_Format(NOW(), '%D');

28th

Select Date_Format(NOW(), '%m');

10

Select Date_Format(NOW(), '%M');

~~28th~~ October

Select Date_Format(NOW(), '%D %M %Y');

28th October 2020

formates

%T → returns time

%W → returns Day

%h → 12 Hrs Format

%H → 24 Hrs format

%i → minute

%s → second

%b → returns 3 characters of month.

```
Select Date_Format(now(), '%D %M %Y')
                                'Date',
```

Date
28th October 2020

```
Seled Date_format(now(), '%W %D %M %Y')
                                'Date',
```

Date
Wednesday 28th October 2020

Arithmetic Operation on date

Oracle → Now() + 1

MySQL → Now() + Interval 1 Day

Interval Number Day / Month / year

```
Select Now() + Interval 3 Year;
```
~~Todays~~ 2023-10-28 09:06:12

```
Select Now() + Interval 3 month;
```
2024-01-28 09:06:12

```
Select Now() - Interval 6 Days;
```
2020-10-22 09:06:12

Default date format → Year - Month - Date
2020 - 10 - 28

* Date_ADD (Now() , Interval 1 Day) → Add 1 day
* Date_Sub (NOW() , Interval 3 years) → Subtract 3 years
* Year (.) → Returns years from date.
* Month ( ) → " month from date.
* Day ( ) → " Day "
* ~~DayName ( )~~
* DAYNAME ( ) → " Day name "

* STR_TO_DATE ( ) → Convert string to date.

STR_TO_DATE ('string', format);

SELECT STR_TO_DATE ('10-10-2020', '%d-%m-%y');
10-10-2020

~~SELECT STR_TO_DATE (~~

* SELECT * From employees where hire_date = str_to_date ('10/23/1994', '%m/%d/%Y');

This will take the date other than default date format.

CASE fn.

Syntax → CASE Columnname
When Condition then Return value.
When Condition the ——
⋮
ELSE
End

Example.

Divide my employees based on the salary

Sal > 15000 :: Highly Paid employee.
Sal > 8000 AND Sal < 15000 :: Avg
Sal < 8000 :: Low grade

```
Select Emp_id, Sal,
CASE
    When Sal >15000 Then 'Highly Paid'
    When Sal > 8000 AND Sal <15000 then 'Average'
    Else 'LOW'
End As 'Emp_Status'
From employees Limit 10;
```

Case could be used as Conditional loop as seen.

OR

```
SELECT Emp_ID, Sal,
    CASE
        When Sal > 15000 then 'Highly Paid'
        When Sal BETWEEN 8000 AND 15000 then 'Avg'
        ELSE 'Low'
    END As 'Emp_Status'
    From employees Limit 10;
```

IFNULL() → Used for comparison with null.

IfNULL (Columns, 0) → this could be anything but for null we consider it as 0 ∴ we use 0.

If 1st para is null returns 0
Else return Column value.

Aggregate f^n → Always returns single value.

* Count() →

COUNT(*) :- returns no. of rows.

COUNT (COLUMN NAME) → returns ~~cont~~ Count of rows, excluding Null in that col.

Select Count(*) from employees;-
107

Select Count (commission_PCT) from employees;
35 (as rest are Null)

* Sum() → returns the Sum

Sum (Columns) :-

Select Sum (Salary) from Employees ;

will print sum of Salary Col.

* Max() → returns max

Max (column's

Select Max (Salary) from employees ;

→ Min() → returns min

→ Avg() → returns Avg

Select Count(*) 'No_of_employees', Max (Salary) 'MaxSal',
Min (Salary) 'MinSal', Avg(Salary) As AvgSal
from employees

Group By fn

① Used to create groups base on criteria

② Group Col. & Perform Aggregate fn / Group fn.

③ Display the result.

Asked → Provide me Sal. provided to each dept.

Select Sum(salary) from employee where dept_id = 10, 20, 30, 90;

Using Group by

```
Select dept_id, Sum(salary)
From
    Employees
    Group By dept_id;
```

101	Rinkesh	20	5000
102	Asmi	30	2000
104	Rahul	40	1000
105	Rishi	10	8000
106	Raju	20	2000
108	BabuRao	30	7000
109	Shyam	20	8000

Group by executed with 3 steps :-

1) Brings Col. required in the temp mem

20	5000
30	2000
40	1000
10	8000
20	2000
30	7000
20	8000

2) Sorts the data acc. to Group by clause i.e. deptid in Ascending Order (By default)

10 8000
20 5000
20 2000
20 8000
30 7000
30 2000
40 10000

3) Creates Group Based on Col. passed in the group by Clause.

10 8000 Group 1

20 5000 Group 2
20 2000
20 8000

30 7000 Group 3
30 2000

40 10000 Group 4

4) Execute the group Fn On the group : : Sum (salary)

10 8000
20 15000
30 9000
40 10000

Rules for using Group By :-

* Cannot use Column else
* All the col. using in select u have to include all those col. in group by.
* All the col. in Group By clause may or may not be a part of Select

Select dept_id, Job_id, sum(sal) From employees Group By Dept_id, Job_id, Fname;

101	Raju	10	Manager	20000
102	Shyam	10	President	10000
103	BabuRao	20	Developer	5000
104	Tiwari	30	Clerk	35000
105	Babu Bhai	30	Quality	10000
106	Asmi	20	Developer	150000
107	Rinkesh	20	Quality	20000

101	Raju	10	Manager	20000	Group 1
102	Shyam	10	President	10000	
		12000			
103	BabuRao	20	Developer	5000	Group 2
106	Asmi	20	Developer	150000	
		200000			
107	Rinkesh	20	Quality	20000	
		20000			
104	Tiwari	30	Clerk	35000	Group 3
105	Babu Bhai	30	Quality	1000	
		45000			

We have to take same col. in select as GroupBy because if we don't then SQL will gets confused when it get 2 kinds of values

For ex in 2nd Group we have developers & Quality as job id then SQL will print only Top most Job id ie developers.

Select deptid, job id from employees group by dept id;

101
102
103 Babu Rao 20 developer 50000
104 Lotta Tiwari 30 Clerk 35000 which is wrong

Filteration

Select Dept_id, job_id Sum (Sal)
From employees
Group by dept_id, job_id Having Sum(Salary) >= 10000;

Where → Alway filter data from rows in table

Having → Filter on the output

Select (mandatory)
From (mandatory)
Where (Optional)
Group by (optional)
Having (Optional)
Orders By (optional)