SQL Structure Query language

Curriculum

Topice	Pages
1) Intro, features, uses, Types	1-3
2) Data Type	4
3) Primary, foregin, Constraints key	5-7
4) Create, Delete, Drope	8
5) No. of database, Create, table, Insert Value	9
6) Show table data, Update, Delete	10
7) ALTER Query	11-12
8) Select Statement, where clause	13-14
9) Operation in SQL	14-15
10) NOT, Limit and order by Query	16-17
11) How To Import CSV file.	17-19
12) String function	20-22
13) Agregation function	23-24
14) Group By having Clause	25-28
15) Time stamp and Extract function	29-31
16) Joins (left, Right, Full, inside)	32-39
17) Self Join	40-41
18) union and union All	42-43
19) window function	44-55
20) Case Expression / Statement	56-58
21) CTE	- Not complete
22) CTE Recursive	- Not complete
23) Sub Query IN / Exist -	- Not complete

The END

SQL

Structured Query Language

SQL :- SQL is a Structured Query language is a Standard Programming language used to manage and manipulate relational databases. It allows users to Create, retrieve, update and delete data from a database.

Key features of SQL.

1) Data Querying

Use SELECT to retrieve Specific data

2) Data manipulation

Use INSERT, UPDATE, DELETE to modify data.

3) Data definition

Use CREATE, ALTER, DROP to define and manage db.

4) Data control -

Use GRANT, REVOKE to control access to data

other definition of SQL -

Structured Query Language (SQL) is a Programming language used to interact with database.

What is Database -

Database is a system that allow users to Store and organise data

Two Types of Database

1) Relational Database
2) Non-Relational Database

Database Use :-

-> Large amount of data.
-> Store Real time data from website / App.
-> Easily Combine with different datasets.
-> Automate Steps and can Re-use
-> Easy & safe access.
-> Data Integrity
-> Deep Search Capabilities.

- Use Various tools -

1) MySQL
2) Microsoft SQL server
3) Snowflake
4) Amazon RedShift
5) SQL Lite
6) ORACLE PL/SQL

Data Types.

- Data Type of a Column defines what value the Column can store in table

-> Defined while Creating table in database.

-> Data types mainly classified into three Categories most used

- String : char, varchar etc.
- Numeric : int, float, boolean etc.
- Date and time : date, datetime etc

Data type

Commonly used data types in SQL :-

int - used for the integer value

float - used to specify a decimal point number.

Bool - used to specify Boolean value true and false

char - fixed length string that can contain numbers, letters, and special characters.

Varchar - Variable length string that can contain numbers, letters and special characters.

Date - date formate YYYY-MM-DD

Datetime - date & time combination, formate is YYYY-MM-DD hh:mm:ss

Primary and foreign keys:-

Primary key (PK)

- A Primary key is a unique column we set in a table to easily identify and locate data in queries.

- A table can have only one Primary key, which should be unique and Not NULL.

Foreign key (fk)-

- A foreign key is a column used to link two or more tables together

- A table can have any number of foreign keys, can contain duplicated and NULL value.

Constraints -

- Constraints are used to specify rules for data in a table.

- This ensures the accuracy and reliability of the data in the table.

- Constraints can be specified when the table is created with the Create table Statement. Or.

- after the table is created with the ALTER TABLE Statement

Syntax

```
CREATE TABLE table_name (
  Column 1 datatype Constraint,
  Column 2 datatype Constraint,
  Column 3 datatype Constraint ....
)
```

Constraints

Commonly used Constraints in SQL:-

1) NOT NULL- Ensure that a column cannot have a NULL value

2) UNIQUE- Ensure that all values in a column are different.

3) Primary key- A combination of a NOT NULL and unique

4) foreign key- Prevents actions that would destroy links b/w tables (used to link multiple tables together.

5) Check -

Ensure that the value in a Column Statisfies a Specific Condition

6) DEFAULT -

Sets a default Value for a Column if no Value is Specified.

7) Create index -

Used to Create and retrieve data from the database Very quickly.

We Install 2 tool / S/w

1) Postgresql
2) PGadmin4.

To Create DataBase (Query)

CREATE DATABASE Test;

Delete database (Query)

DROP DATABASE DBName.

Delete ~~all~~ database

~~Truncate~~

Drop & Truncate Table

The Drop table Command delete a table in the database.

DROP TABLE table_name;

The Truncate table Command delete the data inside a table, but not the table itself

TRUNCATE TABLE table_name;

No. of database

SHOW DATABASE; – Work on workbench
Select * from Pg_database; Work on Postgel
Nt – Post.

How to Create table.

CREATE TABLE Mytable (Id int, Name Varchar (20), age int, city char (20));

How To insert data in the table

one way –

Insert INTO Customer (Id, Name, age, city)
Values
(1, 'Ram', 25, 'Indore');
(2, 'Sita', 20, 'Ujjain'),
(3, 'Gita', 30, 'Dewas');

Two way –

Insert INTO Customer Values
(1, 'Ram', 25, 'Indore'),
(2, 'Ram', 20, 'Dewas');

Show table data

Select * from mytable; - sql workbench
× ~~Select * from pg_database;~~ - ~~Post sql~~
× ~~" " "~~ - ~~Post sql~~

Particular Table ke Column ko dekhne k liye

Replace kardenge * ko Column name se

Select Id from mytable;

Select city from mytable;

Select Id, City from mytable;

Update Value in Table

Update mytable
SET Name = 'Laxita', age = 32
where Id = 3;

Delete Value in Table

DELETE from mytable
where Id = 2;

ALTER Table

The ALTER TABLE Statement is used to add, delete or modify Column in an existing table.

A.

=> For adding new column in the table

ALTER TABLE [Table-Name]
ADD Column [Column-name datatype]

=> for deleting existing Column in the Table

ALTER TABLE [Table-Name]
DROP Column [Column-name];

=> for Renaming the column name in the Table

ALTER TABLE [Table-Name]
RENAME Column [OldName] To
[NewName].

=> for changing Column Datatype

ALTER TABLE [Table_Name]
ALTER/MODIFY Column [Column_Name] [Modify Datatype]

Select Statement

1) SELECT Column_Name from table_Name;
   Show selected Column data
2) SELECT * from table_Name;
   Show all data
3) SELECT DISTINCT Column_Name from table_Name

To select distinct / unique fields available in the table.
for example - remove duplicate data and show only unique value in Specific Column.

WHERE CLAUSE :-

The Where Clause is used to filter records

It is used to extract only those records that fulfill a specified condition

SELECT column_Name from table_Name
Where Condition;

Example - 1) SELECT name from classroom
WHERE grade = 'A';

2) SELECT * from table_name where grade = 'A';

3) SELECT * from table_name where marks > 70;

Operations In SQL

The SQL reserved words and characters are called operators, which are used with a WHERE clause in a SQL query

MOST used operators -

1) Arithmetic operators:- Arithmetic operators on numeric values

Ex- +, -, x, /, and %.

2) Comparison operators - Compare two different data of SQL Table

Ex:- Equal (=), Not equal (!=)

Greater Than (>), Greater Than equals (>=)

3) logical Operators - (Perform the Boolean operations.

Ex - ALL, IN, BETWEEN, LIKE, AND, OR, NOT, ANY.

4) Bitwise Operators - (Perform bitwise Operation on the Integer Values

Ex. Bitwise AND (&), Bitwise OR (1)

Query

1) SELECT * from classroom
WHERE grade = 'A' AND roll > 3.

2) SELECT * from ~~classroo~~ table-Name
WHERE marks > 60 AND result = 'Pass';

3) SELECT * from table-Name
WHERE Marks > 60 OR result = 'Pass';

NOT operator

```
SELECT * from table_Name
where Not Result = 'fail';
```

LIMIT Clause

The LIMIT Clause is used to set an upper limit on the number of tuples returned by SQL.

Ex - below code will return 5 row of data.

```
SELECT column_Name from table_Name
~~where~~ LIMIT 5;
```

ORDER BY.

The ORDER By is used to sort the result set in ascending (ASC) or descending order (DESC)

By defaurt ORDER By ascending
Rnta h.

=> Select * from table_Name
ORDER By Name; (Column name)
↳ by defaut

=> Select * from table_Name
ORDER by Name Asc;
↳ Ascending

=> SELECT * from table_Name
ORDER By Name DESC;

How To Import Excel (CSV) file in SQL

There are two way

first Code
Second directly

Step one

1) Create Table

```
CREATE TABLE Customer
(Custome Id int8 Primary key,
last_name varchar (50),
first_Name varchar (50),
email varchar (100));
```

Step 2

```
Copy Customer ( Customer_id ,first_name,
                last_name, email)
from 'f:/Customer.csv'
 DELIMITER ','
 CSV HEADER;
```

Customer table name h
than Column name btane he
from me location or baki same code

Way 2

1) Create Table
2) Click ~~on~~ Right ~~butte~~ on the Table
   Choose Import/ Export option.
3) fill all details.
4) after click on ok. button.

NOTE- What is delimiter in this code

A delimiter is a Character used to Separate individual data element or field within a record or String. In the case of a CSV file, the delimiter is a comma (,)

There are two way to Create CSV file

1) Method 1) Using a Spread Sheet program (e.g, Microsoft Excel, Google Sheet).

- Save file with .csv extension

2) Method 2 - Using a Text Editor (e.g Notepad, TextEditor)

1) Open a new text file
2) Enter you data. Each field with a Comma.
3) Save the file with a .csv extension

Example -

Name, age, Country
John, 28, USA
Jane, 30, UK.

STRING FUNCTIONS / Aggregate function.

Function in SQL

function in SQL are the database objects that contains a set of SQL statements to perform a specific task.

A function accepts input parameters, perform actions, and then return the result.

Types of function -

1) System defined function - these are built-in-functions.
Ex - rand(), round(), upper(), lower(), count(), sum(); min(), max()

2) User-defined function - once you define a function and call it in the same way as a built in - function.

Most used String function

String function are used to perform an operation on I/P String and ~~resul~~ return an o/p String.

1) Upper() Converts the value of a field to Uppercase

2) LOWER() Converts the Value of a field to lowercase

3) LENGTH() return the length of the value in a text field.

4) SUBSTRING() extracts a Substring from a String.

5) Now() return the Current system date and time

6) formate() used to set the formate of a field

7) CONCAT() add two or more string together

8) Replace() Replace all occurences of a

Substring with in a string, with a new substring

g) TRIM() remove leading and trailing space from a string.

and more function.

Query — Query

=> SELECT UPPER (name) from table-name

~~SELECT UPPER (~~

=> SELECT LOWER (name) from table-name

=> SELECT LENGTH (name), name from table-name.

=> SELECT SUBSTRING (name, 1, 3), name from tab-name

=> SELECT CONCAT (name, sername), name, sername from tab-name.

=> SELECT REPLACE (name, 'Rahul', 'Ronak') name from table-name

=> SELECT

AGGREGATE function

Most used Aggregate functions.

Aggregate function Performs a Calculation on multiple Values and return a Single Value.

And Aggregate function are often used with Group by & Select Statement

1) Count () - return number of values

2) Sum () - return Sum of all Values

3) AVG () - Return average Value

4) MAX () - return maximum Value

5) MIN () - return minimum Value

6) Round () - Round a number to a Specified number of decimal Places.

Id	amount	mode	Pay-date

Column

Query :- Query —

Column-name

=> SELECT COUNT (~~m~~.amount) from Payment.

Table-Name

=> SELECT SUM (column-Name) from table-Name

=> SELECT MAX (Column-Name) from table-Name

=> SELECT MIN (Column-Name) from table-Name

=> SELECT AVG (Column-name) from -/-

=> SELECT Round (AVG (Column-name), 2) from -/-

SELECT Round (AVG (~~A~~Column-name), 2) from -/-

function function decimal k bd kitne no. chaye wo btana h. apne hisab se de skte h.

and So more function are available. in website.

Group By & Having Clause

- The Group By Statement group rows that have the same values into summary rows.

- It is often used with aggregate function (Count(), Max(), Min(), sum() Avg()) to group the result-set by one or more columns

1) Syntax -

```
SELECT column_name
from table_name
Group By Column_name;
```

Ex -

2) Select mode, Sum (amount) As total
from Payment
Group By mode;

04:00

Mode, Id, Amount, Pay-date

Query -

```
SELECT mode, SUM (amount) AS Total
from Payment
Group By mode
```

Column name

column

prefix

Select mode , sum (amount) As Total

Agegregate function

from Payment

table-name

optional

Group By mode

Column name

ORDER By total Asc

Ascending

Group By having Clause

The having clause is used to apply a filter on the result of Group By based on the Specified Condition.

Q The where clause places conditions on the selected columns, whereas the Having clause places conditions on groups created by the Group By clause.

Syntax -

```
SELECT Column-name()
from table_name
WHERE Condition (s)
GROUP BY Column_name(s)
Having Condition (s).
```

Example -

```
SELECT Mode , Count (amount) As total
from Payment
Group By mode
Having Count (amount) >=3
ORDER BY total DESC
```

Querry :-

```
SELECT mode, COUNT (amount) As total
FROM payment
GROUP BY mde
HAVING COUNT (amount) >= 2
    AND AND Count ( amount) < 4
ORDER BY total DESC;
```

Multiple Condition

60, 11, 20, 14

Condition >= 14

ea an >=14 And

And

TimeSTAMPs And EXTRACT function

The Timestamp data type is used for values that contain both date and time parts.

1) Time - Contain only time, formate HH:MM:SS

2) Date - Contains on data, formate YY-mm-DD

3) Year - Contain on year, formate YYYY or YY

4) TimeSTAMP - Contain data and time, formate YYY-MM-DD HH:MI:SS

5) TimeSTAMP2 - Contains data, time and time zone.

Timestamp function | Operation

Below are the Timestamp functions and operators in SQL:-

1) SHOW Time zone => zone show karta
2) SELECT NOW() => recent Time and date
3) SELECT TimeOfDAY() => Day Time Month
4) SELECT CURRENT_TIME -> only Time
5) SELECT CURRENT_DATE - only date

EXTRACT function

Column name

SELECT EXTRACT (MONTH from Pay-date) As Payment_month ; Payment_date

Column name

New Column Name

data ko save Store karne ko

from Payment

table_Name

EXTRACT function

The EXTRACT () function extracts a parts from a given date value

Syntax - SELECT EXTRACT (MONTH from date-field) from Table

- Year
- Quarter - 3 Mon 1 Quater
- Month
- WEEK
- DAY
- HOUR
- MINUTE
- DOW - Day of week
- DOY - Day of year

Query -

SELECT EXTRACT (MONTH from Payment-date) AS Payment - month, Payment date from payment

SQL Join

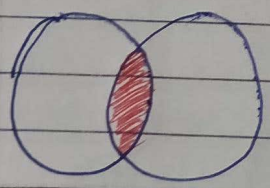
→ **JOIN** - Join means to Combine Something.

→ A Join Clause is used to Combine data from two or more tables, based on a Related Column b/w them.

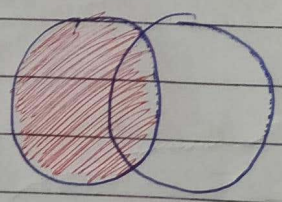
Typer of Joins

Inner Join

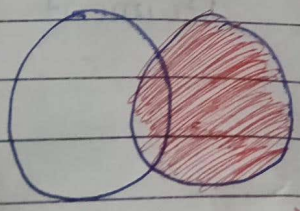
Outer Join
- left Join
- Right Join
- full Join



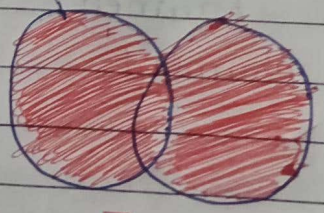
Inner Join



left Join



Right Join



full Join

Inner Join

Return records that have matching values in both tables.

for example

Table - A
student table

Id	Name
1	Priya
2	Riya
3	Swati

Table - B
course table

Id	~~Name~~ Course
1	C++
2	Java
4	Python
5	HTML

OUTPUT :-

Id	Name	Course
1	Priya	C++
2	Riya	Java.

Syntax :-

```
SELECT  Column_name (s)
  from  Table A
    Inner Join  TableB
      ON  TableA.col-name = TableB.
                            colname
```

Id f.n . l-n address-id

Id amont mode Payment-date 34

Query -

Alias
temporary
name deta h
Column ko

Select * from Table name (A) As c
Inner Join Table_name (B) As p
~~ON C. Customer-id~~
ON Table nameA. Columnnome = tablenameB. Customer-id

Code

Select * from Customer As c
Inner Join Payment As p
ON C. Id = P. Id ;

OR Select c. first name , P. amount, p. mode
from Customer As c
Inner Join payment As p
ON C. Id = P. Id;

left Join

- Returns all records from the left table, and the match records from the right table

Table A

Student

Id	Name
101	Rahul
102	Ram
103	Charan

Table B

Course

Id	Course
101	C++
102	Java
103	C
106	Java

output -

Id	Name	Course
101	Rahul	C++
102	Ram	Java
103	charan	C

Syntax -

```
SELECT ~~colu~~ * from
  Table A
  left Join Table B
    ON  Table A. Column name =
        Table B. Column name;
```

Query -

```
Select * from
  Customer AS c
  left Join Payment As p
  on c.Id = p.Id;
```

Right Join

Return all records ~~from~~ from the right table, and the matched records from the left table

Syntax -

```
SELECT * from
  Table A  Right Join  Table B
  ON  Table A. Columnname = Table B. Column name
```

Id	Name
1	Sneha
2	Ankit
4	Rekha
5	

Id	Name
1	Aniket
2	Vansh
3	Rahul
4	Priti

out- 1, 2, 3, 4

Query :-

```
SELECT *
  from table A  AS -
    Right Join  Table B  AS -
      on table A. Column Name = table B.
                                Column name;
```

full Join

Return all records when there is a match in either left or right table

table A

1	x_1
2	x_2
3	x_3

table B

1	y_1
2	y_2
4	y_4

o/p :-

1	x_1	y_1
2	x_2	y_2
3	x_3	null
4		y_4

Syntax -

```
SELECT *
from TableA
full OUTER Join TableB
ON TableA.Col_Name = TableB.
```

Query -

```
Select * from
Customer AS C
Full OUTER Join payment AS P
ON C.Id = P.Id;
```

Which Join to use

1) Inner Join - Return records that have matching values in both tables

2) LEFT Join - Return all records from the left table, and the matched records from the right table.

3) Right Join - Return all records from the right table, and the matched records from the left table

4) full Join - Return all records when there is a match in either left or Right table.

3. Lo Name

Self Join

→ A Self join is a regular join which a table is joined to itself

→ Self Join are powerful for comparing value in a column of rows with the Same table

- Syntax :-

```
SELECT column_name(s)
FROM Table AS T1
Join Table AS T2
ON T2.Col-name = T1.Col-name
```

emp-Id	empname	manager-id
1	Agni	3
2	Akash	4
3	Dharti	2
4	Vayu	3

O/P

Querry -

emp-ID	empname	manager-name
1	Agni	Dharti

```
Select * from employe AS T1
Join employe AS T2
ON T2.~~empl~~ manager- Id = T1.magage-id
```

Name show karna h

Code query

Select T1. ~~employeeme~~ As employee_name,
, T2. employeeme As manager_name

from emp As T1
Join emp As T2
ON T2. empid = T1. manager_id;

Inside process — work on Same/Single Table

1 => Phle hum Select * From employee As e1
Join employee As e2
ON e2. empid = e1. manager Id;

Hum hamare table 1 ek copy generate karenge Joinse
than second table ki Id ko 1st table
k manage_id se change karenge.

2 => than hum e1.id — Id 1st table ki
Select A, e1. emp_name — name 1st table ka , e2. emp_name
as manage_name

2nd ka emp_name
Jo change ho gya
Allocate to manager_id
change ho gya
To

us data ka ~~hum~~ Column name
As ya k manager_name
o/p Aa gya

UNION -

The SQL UNION clause/ operator is used to combine / concatenate the results of two or more select statements without returning any duplicate rows and keeps unique records.

→ To use this union clause, each select statement must have

- The same number of column selected and expressions.
- The same data type and
- Have them in the same order.

Syntax -

```
Select column_name from Table
UNION
Select column_name from TableB
```

Table - Customer A

Cust_name	Cust_amount
Madan Mohan	2100
Gopi Nath	1200
Govind Dev	5000

Table - CustomerB

Cust_name	Customer
Gopal	1500
Madan Mohan	2100

Query -

```
Select cust_name, cust_amount
from customerA
UNION
Select cust_name, cust_amount
from customerB;
```

O/P.

Cust_name	Cust_amount
Madan Mohan	2100
Gopi Nath	1200
Govind Dev	5000
Gopal	1500

No duplicate Value

UNION All -

In union all everything is same as union, it combines, concatenate two or more table but keeps all records, including duplicates.

Syntax -

```
SELECT Column_name from tableA
UNION All
SELECT Column_name from TableB
```

Query -

Ex - Select cust_name, cust_amount from customer A
UNION All
Select Cust_name, cust_amount from customer B.

O/P -

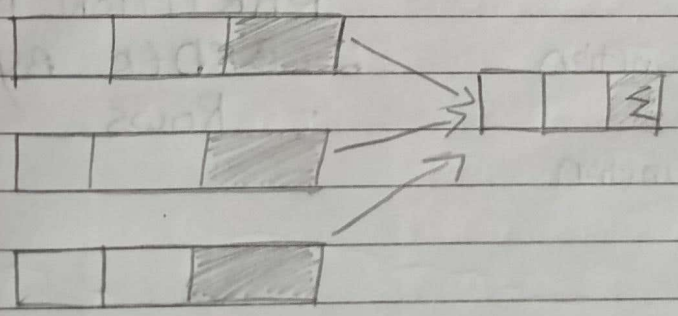
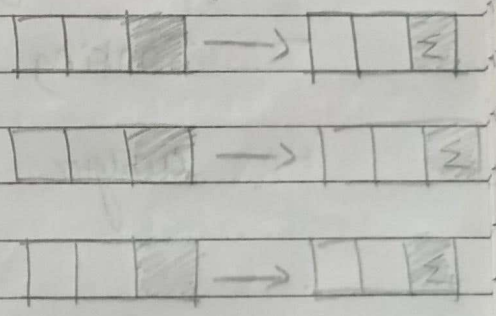
Cust_name	Cust_amount
Madan Mohan	2100
Gopi Nath	1200
Gopind Dev	5000
Gopal Bath	1500
Madan Mohan	2100

Window Function -

- **Window function -**

window functions applies aggregate, ranking and analytic functions over a particular window (set of rows).

- And OVER Clause is used with window functions to define that window.

Aggregate function (Sum Avg etc)	window functions
	
Give output one row per aggregation	The rows maintain their separate identities

Syntax:-

SELECT column name(),

fun() OVER ([< PARTITION By Clause]
[ORDER By Clause]
[< Row or Range Clause]

from table - Name;

Select a function
- Agegreate function
- Ranking function
- Analytic function

Define a window
- PARTITION By
- ORDER By
- Rows.

6) Rows-

Rows can be used if we want to further limit the rows within our Partition. This is Optional and usually not used.

Window function Terms-

let's look at some definitions -

1) window function-

applies aggregate, ranking function and analytic functions over a particular window; for ex:- sum, avg, or row number.

2) Expression -

Expression is the name of the column that we want the window function operated on. This may not be necessary depending on what window function is used.

3) Over -

Over is just to signify that this is a window function.

4) PARTITION BY -

divided the row into partitions so we specify which rows to used to compute the window function.

5) ORDER BY - is used so that we can order the rows within each partition. This is optional and does not have to be specified.

Window Function Types.

→ There is no official division of the SQL window function into Categories. but high level we can divide into Three types.

Window function

Aggregate	Ranking	Value / Analytic
- Sum	- Row-Number	- LEAD
- Count	- RANK	- LAG
- AVG	- DENSE RANK	- first-Value
- MIN	- PERCENT-RANK	- last-Value
- MAX		
- Round		

and so on but it used on most cases.

Window function using with aggregate function

48

Query –

= SELECT newid, new_cat,
sum(new_id) OVER (PARTITION By new_cat ORDER By new_id) As "Total",

~~SELECT~~
= AVG (new_id) OVER (PARTITION By new_cat ORDER By ~~new~~ new_id) As "Average",

= COUNT (new_id) OVER (PARTITION By new_cat ORDER By new_id) As 'Count',

= MIN (new_id) OVER (PARTITION By new_cat ORDER By new_id) As 'MIN',

= MAX (new_id) OVER (PARTITION By new_cat ORDER By new_id) As 'MAXIMUM'.

from table_Name;

Sari query sath me h Alg Alg likh sakte h sikhne k liye sb ko sath me run karo he

I/P I/P

new_id	new_cat
100	Agni
~~200~~	~~Agni~~
700	Dharti
200	Vayu
200	Agni
500	Dharti
300	Vayu
500	Vayu
~~20~~	

→ O/P

New_id	New_cat	Total	Avg	Count	Min	Max
100	Agni	300	150	2	100	200
200	Agni	300	150	2	100	200
500	Dharti	1200	600	2	500	700
700	Dharti	1200	600	2	500	700
300	Vayu	1000	333.3333	3	200	500
200	Vayu	1000	333.3333	3	200	500
500	Vayu	1000	333.333	3	200	500

Aggregate function using window function

Using with order By —

SELECT new_id, new_cat,

⇒ Sum (new_id) OVER (ORDER By new_id Rows Blw unbounded and unbounded following) As "Total",

⇒ AVG (new_id) OVER (ORDER By new_id Row Blw unbounded and unbounded following) As "Avg",

⇒ MIN (new_id) OVER (ORDER By new_id Row Blw unbounded and unbounded following) As 'MIN',

⇒ Count (new_id) OVER (ORDER By new_id Row Blw unbounded and unbounded following) As 'Count',

⇒ MAX (new_id) OVER (ORDER By new_id Row Blw unbounded and unbounded following) As 'Max'

from table_name;

Max
200
200
700
700
500
500
500

I/p

new_id	new_cat
100	Agni
200	Vayu
300	Vayu
200	Agni
500	Dharti
500	~~Agni~~ Vayu
700	Dharti

O/p -

new_id	new_cat	Total	Avg	Min	Max	Count
100	Agni	2500	~~2500~~	100	700	
200	Agni	2500	~~2500~~	100	700	
200	Vayu	2500	357.142	100	700	
300	Vayu	2500	-/-	100	700	
500	Vayu	2500	-/-	100	700	
500	Dharti	2500	-/-	100	700	
700	Dharti	2500	-/-	100	700	
		~~2500~~				

Window function used Ranking function

S2

Querry -

```
=> SELECT new_id,
   Row Number() OVER (ORDER By new_id)
   AS "Row Number";

=> RANK() OVER (ORDER By new_id) AS 'RANK',

=> DENSE () OVER (ORDER BY new_id) AS
   'DENSE RANK',

=> Percent RANK () OVER (ORDER By new_id
   AS 'Percent_Rank'.

from table_name;
```

I/P →

new_id
100
300
200
200
300
700
300

new_id	Row_No.	Rank	DANSE	Percent
100	1	1	1	0
200	2	2	2	0.166
200	3	2	2	0.166
300	4	4	3	0.5
500	5	5	4	0.666
500	6	5	4	0.666
700	7	7	5	1

=> Using Analytic function.

=> SELECT new_id,

=> first_value (new_id) OVER (ORDER BY new_id
AS 'first_value',

=> last_value (new_id) OVER (ORDER BY
new_id) AS '~~first~~ last_value',

=> last_value (new_id) OVER (ORDER BY
new_id Row B/W unbounded
and unbound following) AS 'last_value',

=> LEAD (new_id) OVER (ORDER BY new_id) AS 'LEAD',

=> LAG (new_id) OVER (ORDER BY new_id) AS 'lag'.

from table_name;

Note :-

- last Name ager Normal Vala karte h To same value aygi
- ager Row B/w vali line lgange To last_value aygi

- lead - 1 value skip karke chlta h
- lag - value k bni phle ki value skip kar ke chlta h

=> O/P

new_id	first_value	last_value		LEAD	LAG
100	100	100	700	200	null
200	100	200	700	300	100
200	100	200	700	300	200
300	100	300	700	500	200
500	100	500	700	500	300
500	100	500	700	700	500
700	100	700	700	NULL	500

Q. Offset the lead and lag values by 2 in the o/p column.

I/P
100
200
200
300
500
500
700

O/P		
new-id	lead	lag
100	0	NULL
200	300	null
200	500	100
300	500	200
500	700	200
500	null	300
700	null	500

Query — By default lead and lag 1 value skip karta h.

=> Select new-id,

~~LEAD () OVER (ORDER~~

=> lead (new-id, 2) OVER (order By new-id) as 'lead by 2',

=> lag (new-id, 2) over (order By new-id) as 'lag by 2'

from table - name ,

=> lead me

1
2
3
4
5

Skip hoga first and baki scroll honge to

2
3
4
5
null

yha value khtm ho jygi column khali to null

lag me

1
2
3
4

null
1
2
3

1 null khega baki down scroll hoga to

CASE Statement / CASE Expression

- The Case Statement and Expression goes through Condition and return a value.
  When the first Condition is met (like if - then - else). If no Condition are true, it return the value in the else clause.

- If there is no else part and no Condition are true, it return null,

Syntax

Case statement

→ General Case Syntax

```
Select Column_name,
Case
    WHEN Condition1 Then result1
    WHEN Condition2 Then result2
    WHEN Condition N Then result N
    ELSE other_result

END
from table_name;
```

Example -

```
Select id, amount,
  CASE
      WHEN amount > 100 THEN 'Expensive product'
      WHEN amount = 100 THEN 'Moderate pro'
    ELSE 'Inexpensive product'
  End As ProductStatus
```

→ temporary o/p Column Name

```
  from payment
```

Syntax

Case ~~Statement~~ Expression

→ General case Syntax

```
Select column_name,
  Case Expression
      WHEN Value1 THEN result1
      WHEN Value2 THEN result2
        ELSE other_result
  END
      from table_Name;
```

→ Column_name

Example - Case Expression

```
=> Select customer-id,
   CASE amount → Column-name  Issue date, mid-month
      WHEN 500 THEN 'Prime Customer'
      WHEN 100 THEN 'Plus Customer'
      Else 'Regular Customer'
   END AS CustomerStatus
   from Payment.
```

Complete.