SQL NOTES !!!

SELECT * FROM tname

shows all columns from the table

SELECT coulmnnames FROM tname

shows selected columns from the table

SELECT COUNT(colname) FROM users HAVING COUNT(colname) > 1

DUPLICATE RECORDS

SELECT DISTINCT colname/s FROM tname

shows unique values from col name/s mentioned

SELECT COUNT (DISTINCT) colname/s FROM tname

counts those unique values

SELECT ROUND(colname, -2) FROM tname

Rounds to 100, positive means after decimal place

SELECT FLOOR(AVG(colname)) FROM tname

SELECT LENGTH(colname) FROM tname

Length of each record in column, can use char_length as well

SELECT UCASE(colname) FROM tname

SELECT * FROM tname WHERE LCASE (colname) = " "

Return all values in column in upper case. LCASE for lower case

In this way if we unsure in what case text is , we turn all to lower case and search

Psql mai LOWER and UPPER

SELECT INITCAP(' STRING')

Makes first letter of each word in the string capital)

SELECT LEFT(colname,num) FROM tanme

SELECT RIGHT(colname,num) FROM tanme

SELECT SUBSTRING(colname,start position, length) from tname

Middle letters

Number of characters from left, right, middle u want to see

SELECT POSITION('string' in 'whole text/colname)

SELECT colname/s/* FROM tname WHERE (NOT) condition (AND/OR/NOT/XOR) condition/s

shows records as per condition

We can combine And, Or, Not, XOR in any ways that we want.

Not something and not something

Something and (something or something)

Etc

SELECT colname/s/* FROM tname WHERE colname IS / IS NOT NULL

Finds out null/not null records in a particular column

SELECT colname/s/* FROM tname ORDER BY colname(or col number) ASC, colname DESC

sorts the columns as given.

If u do not mention asc/desc, default is asc

SELECT colname/s FROM tname WHERE condition/s LIMIT number

Shows a particular number of records (for MySQL, others have other syntax)

SELECT MIN/MAX/COUNT/AVG/SUM colname AS heading, colname/s FROM tname WHERE condition

min/max/count/average/sum of a column, it will be displayed with heading

SELECT colname/s FROM tname WHERE condition HAVING condition

HAVING COUNT(colname) > val

HAVING SUM(colname) > val

basically having was introduced because we could not use where for putting conditions on aggregate functions , and having works with group by only

SELECT colname/s FROM tname WHERE colname LIKE '_a%'

Use ILIKE instead of like when u don't care if text you mentioned is upper case/lower case

You can use NOT ahead of these to have all other records

https://www.geeksforgeeks.org/mysql-regular-expressions-regexp/#:~:text=MySQL%20supports%20another%20type%20of,performing%20regular%20expression%20pattern%20matches.

SELECT colname + ' ' +colname AS newcolname FROM tname

Concatenate

SELECT colname/s FROM tname WHERE colname IN (… ,… ,…)

SELECT colname/s FROM tname WHERE colname IN (SELECT …..)

Allows us to specify multiple values

Instead of or or or we can do this

SELECT colname/s FROM tname WHERE colname BETWEEN val1 AND val 2

Range can be specified

SELECT colname AS newname FROM tname

SELECT colname/s FROM tnam AS newname

Change name temporarily only for this query

SELECT colname/s {in format tname.colname}

FROM t1name

INNER JOIN/LEFT JOIN/RIGHT JOIN/FULL OUTER JOIN t2name

ON t1name.colname = t2name.colname

WHERE condition/s

ORDER BY tname

joins that can be done

CROSS JOINS bhi hota hai. It does not have an ON clause

SELF JOIN samjhaaa

SELECT colname/s, agg function(colname) over (partition by colname order by colname ROWS between 1 preceding and 1 following)

FROM tname

Over clause, partition by

select col name,

row_number() over (order by colname),

rank() over (order by colname),

dense_rank() over (order by colname),

percent_rank() over (order by colname),

cume_dist() over (order by colname)

FROM colname

Row number all unique

Rank is 12335

Dense rank is 12334

Percent rank is 0 to 1

Cume dist is 0 to 1 excluding 0

SELECT colname/s FROM t1name

UNION / UNION ALL (takes duplicates also)

SELECT colname/s FROM t2name

used to combine results, number of columns should be same, dtype should be same, columns should be proper order

SELECT colname/s FROM tname WHERE condition/s GROUP BY colname

Groups the result in the column specified

SELECT colname/s FROM tname WHERE colname =/>/</>=/!= ANY/ALL (SELECT …)

Any operator returns true if subquery values meet condition

All operator returns true if all subquery values meet condition

SELECT colname/s INTO newtablname FROM tname WHERE condition

Creates new with columns given and condition satisfied

Use 1 hash (#) before newtablename for local temporary table

Use 2 hash for global temporary table

2nd method to make temporary table

CREATE TABLE #tname (

Col name dtype,

Col name dtype

)

INSERT INTO #tname

SELECT colname/s FROM tname WHERE condition/s

SELECT colname/s

CASE

WHEN condition THEN “”

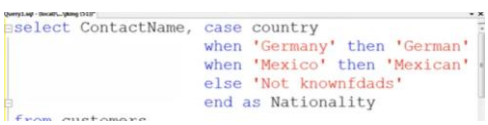
WHEN condition THEN “”

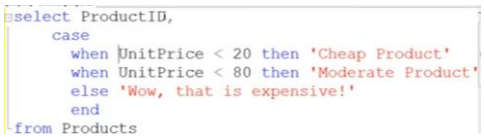
ELSE “”

END AS (name the column)

FROM tname

can test a number of cases and give o/p accordingly.





SELECT IFNULL (val1,val2)

Selects first null value

SELECT COALESCE (val1, val2, val3,…)

Selects first non null value

HANDLE NULL VALUES

SELECT ISNULL(colname," replacement text") FROM tname WHERE condition/s

SELECT colname/s CASE WHEN colname IS NULL THEN "" ELSE colname END FROM tname

SELECT COALESCE (colname, colname, colname, " replacement text") FROM tname

INSERT INTO tname (colname 1,…) VALUES (val1,…),(val1,…),(val1,…)

Inserts values into columns.

If all columns, don't need to mention col names.

INSERT INTO newtable SELECT colname/s FROM tname WHERE condition/s

Datatype should match, existing records are unaffected

CREATE PROCEDURE proc_name @colname nvarchar(30), @colname2 navarchar(30),…

AS

Sql_statement (select * from customer where colname=@colname and colname2=@colname2

GO

To execute,

EXEC proc_name @colname="", @colname2= ""

create procedure or function to execute again and again

UPDATE tname SET colname/s=value WHERE condition/s

Change or update value/s in table

DELETE FROM tname WHERE condition/s

Deletes records which satisfy condition

--mvsjrikv jisvmrsk

commentttt

SELECT VARIANCE(col) FROM tname

SELECT stddev(col) FROM tname

Calculate variance, standard deviation

CREATE DATABASE dname

DROP DATABASE dname

BACKUP DATABASE dname

TO DISK= ‘path’

(WITH DIFFERENTIAL)

Backs up database, with diff means only parts of db where changes made are backed up

CREATE TABLE tname (

    Col1 dtype,

    Col2 dtype,

    …

)

SELECT colname/s INTO newtablname FROM tname WHERE condition

Creates new table with columns given and condition satisfied

CREATE TABLE tname (

    Col1 dtype constraint,

    Col2 dtype constraint,

…

Constraint

)

Constraint can be

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
  Can only be written in end, alag se constraint.
  FOREIGN KEY (colname) REFERENCES t2name (colname)
- CHECK
  Can only be written in end, alag se constraint.
  CONSTRAINT colname CHECK (colname>20 AND colname=”yo”)
- DEFAULT
  DEFAULT “”
- AUTO_INCREMENT= startvalue (default start is 0)

CREATE INDEX index_name

ON tname

CREATE VIEW [vname] AS

SELECT colname/s

FROM tname

WHERE condition/s

SELECT * FROM [vname]

Create and display a view

DROP TABLE tname

ALTER TABLE tname

ADD COLUMN colname dtype

DROP COLUMN colname

MODIFY COLUMN colname dtype

any of the three can be used

1) Use column names instead of *
2) HAVING clause is used to filter the rows after all the rows are selected. It is just like a filter.
3) Try to have minimum subqueries as possible

DWD LEARNING

No count(distinct *) in psql, gotta create subquery select count from select distinct

The above is only for all records

You can do 1st part if specific column ka u want unique

We can use the MAX function to return the largest number from a numeric column, be careful when using this on non-numeric columns as it will actually return you the last value in alphabetical order like "Zebra" is considered a higher value than "Alligator"

Can also use MIN MAX for timestamp, max means latestf

Order by 1/2/3 can be used as per the column we want to order by. The reference is from select columns part

Use as name for any calculation don't leave default

SELECT price::INTEGER to convert data type
Can be float as well
Keep data type in mind while doing operations

Be very careful when using the `WHERE` clause with the `GROUP BY` - you will always filter out the records first before any aggregations are calculated.
This is in direct contrast to the `HAVING` clause which is used to filter off records based off the aggregated results!

Remember this as it will be touched upon very often in practice!

Regular expression deets - https://www.postgresql.org/docs/current/functions-matching.html#REGULAR-EXPRESSION-DETAILS

the important takeaway is that `NULL` values are not counted in aggregate functions!