

SQL for Data Analysis

October 10, 2023

Prepared By- Murtaza Kamal Pasha Khan



Murtaza Kamal Pasha Khan has more than 2 years of experience as a data professional. He has worked mainly in online marketplaces with clients from varied sectors such as edtech, corporate trainers, and healthcare professionals. He enjoys working with data and aspires to be a champion Data Scientist. In 2019, he completed his MSc from the University of Minnesota. You can follow him on [LinkedIn](#) to find out more about him.

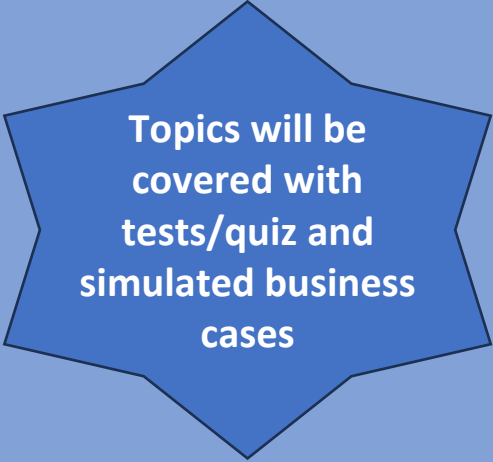
Acknowledgement

This course is built with the help of the course created by Dr. Enayetur Raheem. More about it can be found from his website

dataskool.org

Table of Contents

- Six Keywords (SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY)
- JOIN and UNION clause
- Temporary table, subquery and CTE
- Calculated/Derived columns
- Window Functions
- Handling Date time data types
- Data cleaning/ string functions
- Updating records, tables, columns
- Primary and Foreign keys
- Stored Procedure



Topics will be covered with tests/quiz and simulated business cases

Tools

- Postgresql
- DBeaver

Six Important Keywords

- **SELECT**
 - Selects the columns or variables of the resulting data
- **FROM**
 - The table to query the data from
- **WHERE**
 - Filters the row based on given condition(s) (car = “Ferrari”)
- **GROUP BY**
 - Groups data by a certain variable(s), Used for aggregate functions
- **HAVING**
 - Comes after Group By. Filters the aggregated columns
- **ORDER BY**
 - Sorts the data by given columns in ascending (default) or descending (DESC) order.



The Where Clause

- Between
- IN ()
- NOT IN ()
- <>/!=
- IS NULL
- IS NOT NULL
- >= / <=
- AND, OR

Data Types

Type	Name	Description
String	CHAR, VARCHAR	Holds strings, CHAR- fixed length VARCHAR – length varies (e.g. 256 chars)
	TEXT	Holds longer string that do not fit into VARCHAR
Numeric	INT	Holds integers or whole numbers (no fraction)
	FLOAT	Holds decimal numbers (e.g. 1.5)
Logical	Boolean	Can be TRUE or FALSE
Date	Date Time Timestamp Timestamp with time zone	YYYY-MM-DD 14:43 YYYY-MM-DD 14:43 YYYY-MM-DD 14:43 +6

Aggregate Functions

- MIN
- MAX
- COUNT()
- SUM()

GROUP BY

ORDER BY

- To sort values
- DESC, to sort in descending order

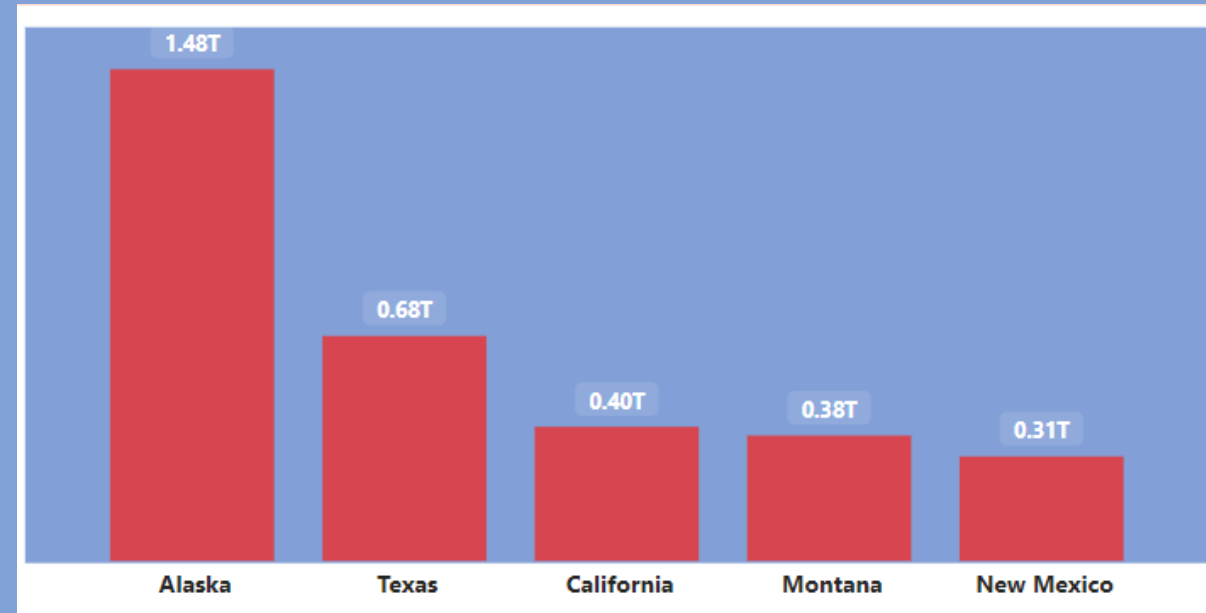
LIMIT Clause

Limits the number of records to be returned.

Business Case -1

Write a code to return top 5 states with largest land area?

```
SELECT state_name, sum(area_land) AS  
total_land_area  
FROM us_counties_population  
GROUP BY 1  
ORDER BY 2 DESC  
LIMIT 5;
```



Intuition Behind JOIN

- Side-by-side addition
- Based on a common key

Intuition Behind UNION

- Stacking two tables
- The two tables should have same number of columns
- The data types of the two columns should be matched
- UNION – stacking without duplicates
- UNION ALL- stacking with duplicates

Types of JOINS

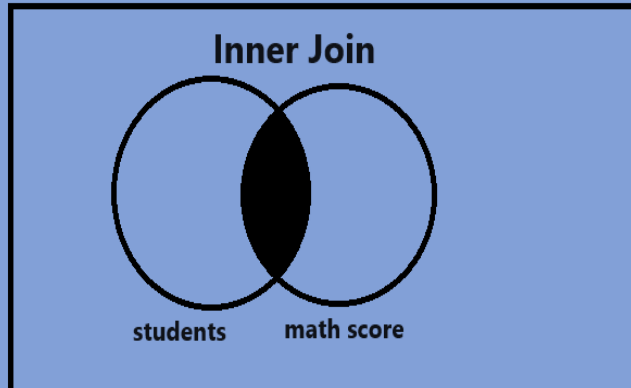
- INNER JOIN
 - INNER JOIN
- OUTER JOIN
 - LEFT JOIN
 - RIGHT JOIN
 - FULL OUTER JOIN

INNER JOIN

id	First name	Last name	Grade
101	Abdul	Latif	9
102	Mohammad	Ali	10
103	Zakia	Sultana	8
104	Samir	Kumar	9
105	Suman	Sarkar	10

Section id	Student number	Math score
A	101	89
A	102	78
A	103	88
B	201	79
B	202	90
B	203	60
C	301	98
C	302	77
C	303	87
C	304	92

id	First name	Last name	grade	Math score
101	Abdul	Latif	9	89
102	Mohammad	Ali	10	78
103	Zakia	Sultana	8	88



```
SELECT s.*, ms.math_score
FROM math_score ms
INNER JOIN students AS s
ON ms.student_number = s.id ;
```

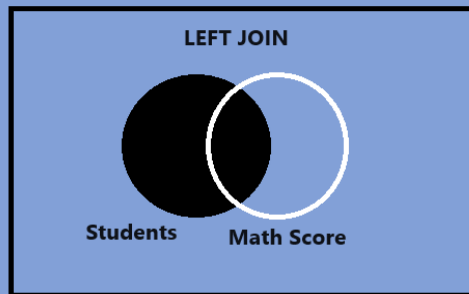
LEFT JOIN

priority

id	First name	Last name	Grade
101	Abdul	Latif	9
102	Mohammad	Ali	10
103	Zakia	Sultana	8
104	Samir	Kumar	9
105	Suman	Sarkar	10

Section id	Student number	Math score
A	101	89
A	102	78
A	103	88
B	201	79
B	202	90
B	203	60
C	301	98
C	302	77
C	303	87
C	304	92

id	First name	Last name	Grade	Math score
101	Abdul	Latif	9	89
102	Mohammad	Ali	10	78
103	Zakia	Sultana	8	88
104	Samir	Kumar	9	[NULL]
105	Suman	Sarkar	10	[NULL]



```
SELECT s.*, ms.math_score
FROM students AS s
LEFT JOIN math_score ms
ON ms.student_number = s.id;
```

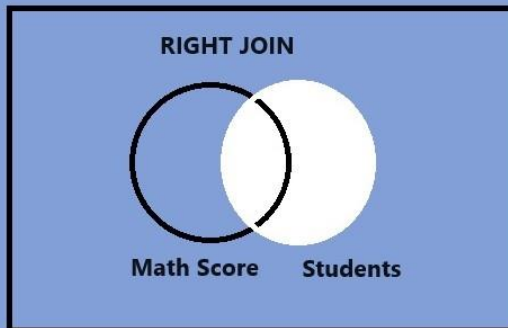
RIGHT JOIN

priority

id	First name	Last name	Grade
101	Abdul	Latif	9
102	Mohammad	Ali	10
103	Zakia	Sultana	8
104	Samir	Kumar	9
105	Suman	Sarkar	10

Section id	Student number	Math score
A	101	89
A	102	78
A	103	88
B	201	79
B	202	90
B	203	60
C	301	98
C	302	77
C	303	87
C	304	92

id	First name	Last name	Grade	Math score
101	Abdul	Latif	9	89
102	Mohammad	Ali	10	78
103	Zakia	Sultana	8	88
104	Samir	Kumar	9	[NULL]
105	Suman	Sarkar	10	[NULL]



```
SELECT s.*, ms.math_score
FROM math_score AS ms
RIGHT JOIN students AS s
ON ms.student_number = s.id;
```

FULL JOIN

students

id	First name	Last name	Grade
101	Abdul	Latif	9
102	Mohammad	Ali	10
103	Zakia	Sultana	8
104	Samir	Kumar	9
105	Suman	Sarkar	10

This table has the priority

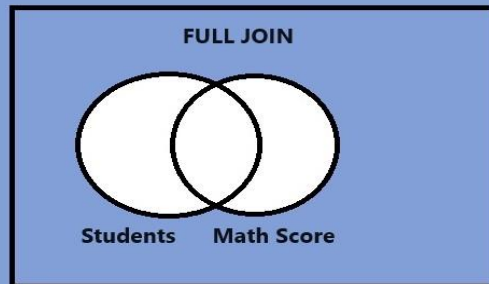
math scores

Section id	Student number	Math score
A	101	89
A	102	78
A	103	88
B	201	79
B	202	90
B	203	60
C	301	98
C	302	77
C	303	87
C	304	92

This too also has the priority

output

id	First name	Last name	Grade	Math score
101	Abdul	Latif	9	89
102	Mohammad	Ali	10	78
103	Zakia	Sultana	8	88
				79
				90
				60
				98
				77
				87
				92
104	Samir	Kumar	9	
105	Suman	Sarkar	10	



```
SELECT s.*, ms.math_score
FROM math_score AS ms
FULL JOIN students AS s
ON ms.student_number = s.id;
```

UNION

students

id	First name	Last name	Grade
101	Abdul	Latif	9
102	Mohammad	Ali	10
103	Zakia	Sultana	8
104	Samir	Kumar	9
105	Suman	Sarkar	10

Both table has the same number of columns; and has identical data types in the same order, therefore, all the unique records from the both table will be added in the resultant table

students2

id	First name	Last name	Grade
106	Momen	Mondol	7
107	Mohammad	Zakaria	9
108	Sushanto	Chakrabarti	10
109	Zerin	Tasneem	10
110	Yasmin	Tazreen	8
101	Abdul	Latif	9

```
SELECT * FROM students
UNION
SELECT * FROM students2
ORDER BY id;
```

output

id	First name	Last name	Grade
101	Abdul	Latif	9
102	Mohammad	Ali	10
103	Zakia	Sultana	8
104	Samir	Kumar	9
105	Suman	Sarkar	10
106	Momen	Mondol	7
107	Mohammad	Zakaria	9
108	Sushanto	Chakrabarti	10
109	Zerin	Tasneem	10
110	Yasmin	Tazreen	8

Notes

- In a Full Join, both of the tables you are joining have equal priority. Consequently, the resultant table will contain queried data from all the tables and fill in with null values when there is no match.
- However, In an UNION operation, both of the tables should have equal number of columns and have identical data type in the same order.

UNION ALL

id	First name	Last name	Grade
101	Abdul	Latif	9
102	Mohammad	Ali	10
103	Zakia	Sultana	8
104	Samir	Kumar	9
105	Suman	Sarkar	10

id	First name	Last name	Grade
106	Momen	Mondol	7
107	Mohammad	Zakaria	9
108	Sushanto	Chakrabarti	10
109	Zerin	Tasneem	10
110	Yasmin	Tazreen	8
101	Abdul	Latif	9

```
SELECT * FROM students
UNION ALL
SELECT * FROM students2
ORDER BY id;
```

id	First name	Last name	Grade
101	Abdul	Latif	9
101	Abdul	Latif	9
102	Mohammad	Ali	10
103	Zakia	Sultana	8
104	Samir	Kumar	9
105	Suman	Sarkar	10
106	Momen	Mondol	7
107	Mohammad	Zakaria	9
108	Sushanto	Chakrabarti	10
109	Zerin	Tasneem	10
110	Yasmin	Tazreen	8

Temporary Tables

- Temporary Tables
- Subquery
- Common Table Expressions (CTE)

```
DROP TABLE IF EXISTS  
students_combined;  
CREATE TEMPORARY TABLE  
students_combined AS  
SELECT * FROM students  
UNION  
SELECT * FROM students2;
```

Temporary
Table

```
SELECT count(*) FROM  
(  
  SELECT * FROM students  
  UNION  
  SELECT * FROM students2  
) AS a;
```

Subquery

```
WITH a AS (  
  SELECT * FROM students  
  UNION ALL  
  SELECT * FROM students2  
)  
SELECT count(*) FROM a;
```

CTE

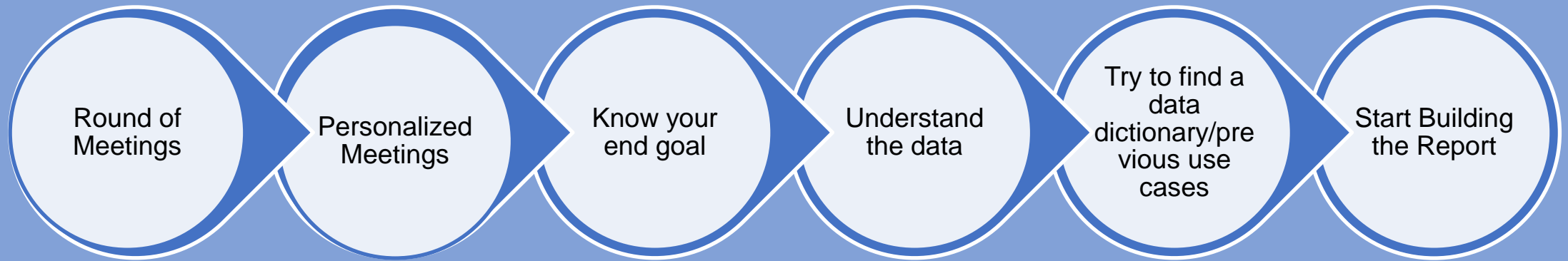
Calculated / Derived Columns

- Creating New columns
 - With any mathematical or statistical operations (division, addition, subtraction, std dev etc)
- Knowing more information about the data
- Data type and their impact
- Data type conversion

Numeric Types

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Steps to Prepare A Report



Explore US Census Data

- How many columns are there?
- How many records are there?
- What is the data type of each

LIKE and ILIKE

```
SELECT * FROM ucpe  
WHERE state_name ILIKE '%rado%';
```

Basic Math Operations

+

-

/

*

^

CAST

CASE statements

Create a 'region_name' column with the following conditions

region 1 = Northeast
region 2 = Midwest
region 3 = South
region 4 = West

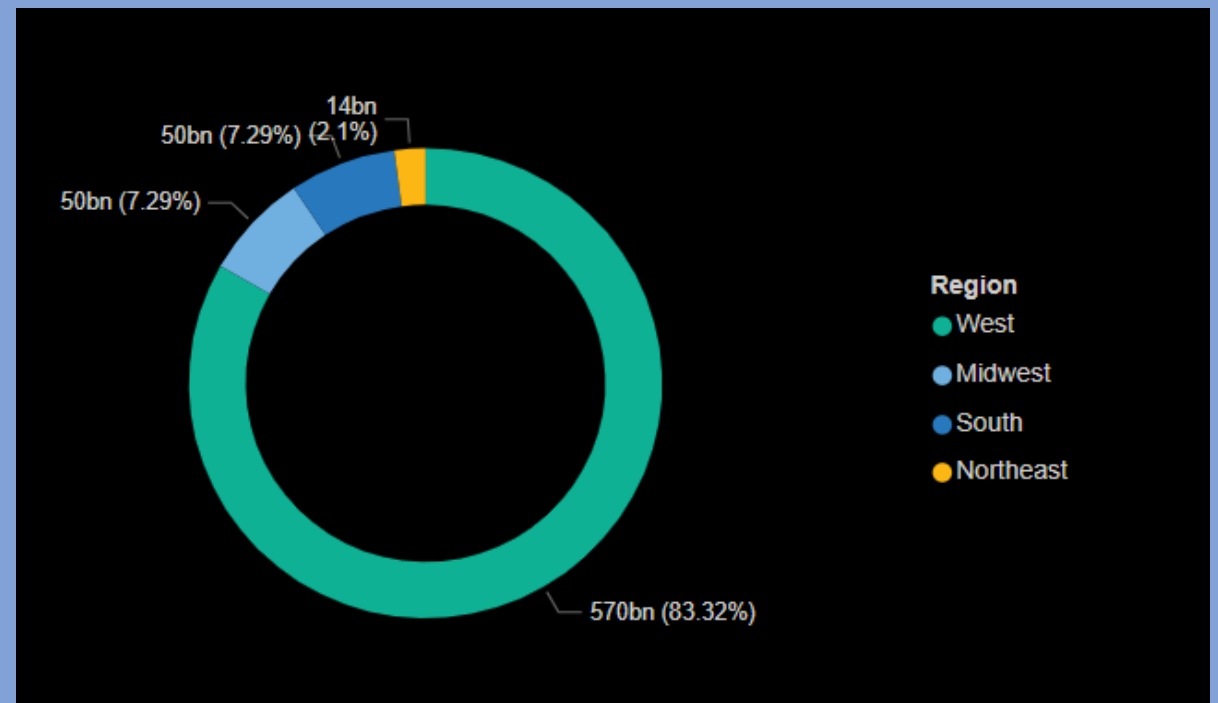
```
SELECT
*,
CASE
WHEN region = 1 THEN 'Northeast'
WHEN region = 2 THEN 'Midwest'
WHEN region = 3 THEN 'South'
ELSE 'West'
END AS region_name
FROM
us_counties_population;
```

Business Case -2

Now that you have the region in place, can you estimate the total water area by these four regions?

Steps: Use a CTE for creating the region column. Then, use an aggregate function

```
WITH water_by_region AS(
SELECT
  *,
  CASE
    WHEN region = 1 THEN 'Northeast'
    WHEN region = 2 THEN 'Midwest'
    WHEN region = 3 THEN 'South'
    ELSE 'West'
  END AS region_name
FROM
  us_counties_population
)
SELECT
  region_name AS region,
  SUM(area_water) AS total_water_area
FROM
  water_by_region
GROUP BY
  region_name
ORDER BY
  total_water_area DESC;
```



Business Case -3

Can you find the top ten states that have fastest growing population?

Steps: Use a CTE to create a column that calculates the difference of the two year population. Then, find the top ten states.

```
WITH cte AS (  
  SELECT DISTINCT (state_name) AS state, SUM(pop_est_2018) AS pop18,  
    SUM(pop_est_2019) AS pop19, (SUM(pop_est_2019) - SUM(pop_est_2018)) AS diff  
  FROM us_counties_population ucp  
  WHERE state_name NOT LIKE 'Dis%'  
  GROUP BY 1  
  ORDER BY 2 DESC  
)  
  
SELECT state, pop18, pop19, ROUND (diff::NUMERIC / pop18 *100 , 2) AS perc_change  
FROM cte  
ORDER BY perc_change DESC  
LIMIT 10;
```

