†

– SQL BootCamp

- Databases. — systems that allow users to store and organize the data.

• Spread sheets	• Databases.
→ one time analysis	– Great for data integrity
→ Quickly need to start something out	– Can handle massive amount of data
→ Reasonable data size	– Quickly combine different datase
→ Ability for untrained people To work with date	– Automate steps for re use
	– Can support data for website & applications.

- SQL → programming language used to communicate with our data base

Post gre SQL – sql engine that stores data and reads queries and returns info

Pg Admin – Graphical user interface connecting with Post gre SQL.

– SQL – PW – cranth007 kranthi007
Port no – ~~5432~~ 5433

→ About the challenge - me are SQL consultant for DVD store

- **SELECT** — used to retrive info from table

```
SELECT col_name FROM table_name
```

- **DISTINCT** - to return only distict value in column

```
SELECT DISTINCT (column) FROM table
```

- **COUNT** - n of input rows that match a specific conditions

```
SELECT COUNT (name) FROM table ;
        count (*)
```

- **WHERE** - allows us to specify conditions on columns for the rows to be returned.

```
SELECT column
FROM   table
WHERE  conditions ;
```

- **ORDER BY** .- to sort rows based on column value in (Asciding or deeciding)

```
SELECT col1, col2
FROM table
ORDER BY col1 ASC/DESC
                  DESC
```

- **LIMIT** - Allows us to limit the numbes of rows returned for a query.

```
→ limit 5 ;
```

→ $\boxed{\text{BETWEEN}}$ - can be used to match the value againg a range of value

WHERE Amoun BETWEEN X AND Y ;

↓

'2007-02-01' AND '2007-02-03'

$\boxed{\text{IN}}$ -

WHERE color IN ('red', 'blue')

case sensitive ← LIKE and ILIKE → case-insensitive → what if u want match against a pattern.

LIKE → Allows us to perform pattern matching against string data with the use of wildcard characters

- percent %
  - ↳ matches any sequence of char
- underscore _
  - ↳ matchs any single char

ex WHERE name LIKE 'A%'

↳ names that begin with A

'%a'

↳ end with a

Like '_her%'

LIKE 'version#__'

%er%

Vrinth

→ GROUP BY → allow us to aggregate data and apply functions to better understand how data is distributed per category.

↳ Aggregate functions → main idea is to take multiple inputs and return a single output.

→ Calls happen only in SELECT clause or HAVING clause

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

ROUND( )

↳ In Group BY

we need to choose categorical column to GROUP BY

non-continoues

```
SELECT category-col, AVG(data_col)
FROM table
GROUP BY category-col
```

→ DATE( )

→ HAVING clause → allows us to use the aggregate result as a filter along with a GROUP BY.

→ allows us to filter after an aggregation has already taken place

→ We cannot use where to filter based off of aggregate results, because those happen after a where is executed.

JOINS — allows us to combine information from multiple tables.

- AS clause → allows us to create an "alias" for a column or result.

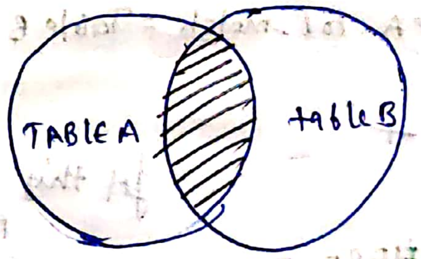
```
SELECT column AS new_name
FROM table.
```

→ AS operator gets executed at the very end of a query, we can't use the ALIAS inside a WHERE operator

→ INNER JOINS

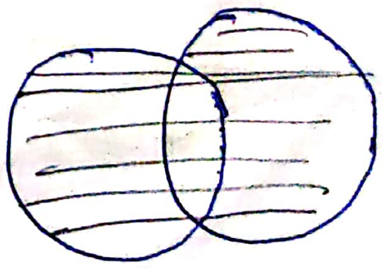
```
SELECT * FROM TABLE A
INNER JOIN TABLEB
ON TABLEA.col_match = TABLEB.col_match
```

P.Yoxnth kumy



→ OUTER JOINS → allows us to specify how to deal with the values only present in one of the tables being joined.

→ FULL OUTER JOIN



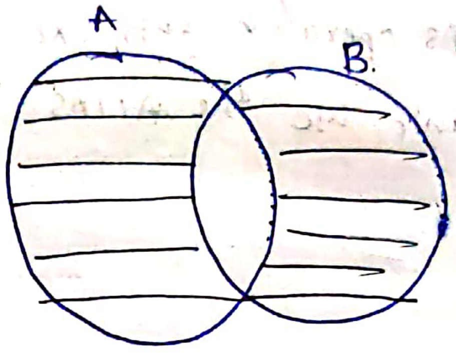
```
SELECT * FROM Table A
FULL OUTER JOIN Table B
ON Table A. col_match = TableB.col_match
```

Scanned with CamScanner

FULL OUTER JOIN with WHERE

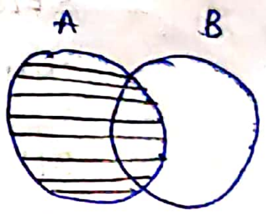
↳ get rows unique to either table.

```
SELECT * FROM Table A
FULL OUTER JOIN Table B
ON Table A. Col-match = Table B. Col-match
WHERE Table A. id IS null OR Table B. id is null
```



LEFT OUTER JOIN

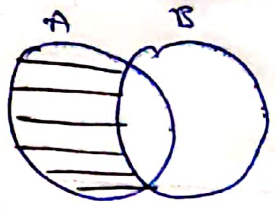
```
SELECT * FROM Table A
LEFT OUTER JOIN Table B
ON Table A. Col-match = Table B. Col-match
```



+

```
WHERE Table B.id ~~is~~ IS null
```

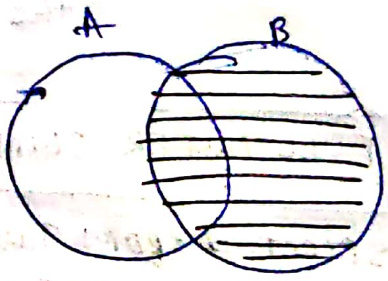
for this ←

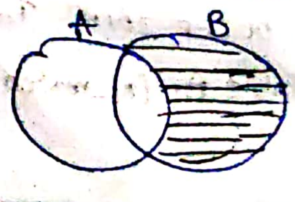


RIGHT JOIN

→ essentially same as left Joins except the tables are switched.

Same





UNION → This operator is used to combine the result-set of two or more SELECT statements

```
SELECT col_name(s) FROM table 1
UNION
SELECT col_name(s) FROM table 2
```

ADVANCED SQL TOPICS

- Time Stamps and Extract

Refer documentation

TIME - only time
DATE - only date
TIMESTAMP - date and time
TIMESTAMPTZ - date, time & time zone.

- SHOW ALL
- SHOW TIMEZONE
- SELECT NOW()
- SELECT TIMEOFDAY( )
- " CURRENT_TIME
- " - DATE

→ EXTRACT( ) — YEAR, MONTH, WEEK, DAY, QUARTER

AGE ( )

TO_CHAR( )

↓

Converts date types to text

↓

TO_CHAR(date_col, 'mm-dd-yyyy')

- EXTRACT (YEAR FROM date-col)

$\boxed{\text{dow}}$ query

- Mathematical operations - (documentation) ✓

- string function & operations - (documentation)

- SubQuery -

  - allows us to construct complex queries, essentially performing a query on the result of the another query. (2 select statements)

  - EXISTS operator is used to test for existance of rows in a subquery.

Self JOIN - Query in which a table is joined to itself

  - useful for comparing values in a column of rows within the same table.

```
SELECT  tableA.col, tableB.col
FROM  table AS tableA
JOIN  table AS tableB ON
tableA.some_col  = tableB.other_col
```

Creating databases and tables

Data types

- Boolean (True or false)
- characters (char, variable and text)
- Common
- Numeric (int, floating - point numbers)
- Temporal (date, time, time stamp, interval
- UUID (universally unique identifiers)
- Array (stores arrays of strings, nums etc)
- JSON
- Hstore key - value pair
- special types such as Network adress and geometric data

Primary key - column or a group of columns used to indentify a row uniquely in a table

Foreign key - field or group of fields in a table that uniquely identifies a row in another table

- table that Contain foreign key is called (Referencing table or parent table)

- The table to which the foreign key refer ceeds is called referenced table or parent table.

$$\begin{tikzcd} & \text{Column Constraints} \\ \underline{\text{Constraints}} \arrow[ur] \arrow[dr] & \\ & \text{Table Constraints} \end{tikzcd}$$

- these are rules enforced on data columns on table
- used to prevent invalid data from being entered into the data base

- This ensures the accuracy and reliability of data in database

<u>Common Constraints</u>

- NOT NULL
- UNIQUE
- PRIMARY key
- FOREIGN key
- CHECK
- EXCLUSION
- REFERENCES

CREATE

```
CREATE TABLE table_name (

    Column_name TYPE column_constraint,
    Column_name TYPE column_constraint,
    table_constraint table_constraint )
    INHERITS existing_table_name;
```

→ Common simple syntax

SERIAL - a sequence is a special kind of database object that generates a sequence of integers

- A sequence is often used as primary key column in a table.

⇒ **INSERT** - allows to add rows to a table.

Ins.

```
INSERT INTO table (col1, col2....)
VALUES
    (value1, value2...),
    (value 1, value 2...), ...;
```

(or)

```
INSERT INTO table (col1, col2...)
SELECT col1, col2...
FROM another_table
WHERE condition;
```

another table values.

- UPDATE - allows for the changing of values of the columns in a table.

→
```
UPDATE table
SET column1 = Val1,
        col2 = Val2, ....
WHERE
    condition ;
```

Another tables values
→
like joining

```
UPDATE table A
SET Original_col = Table B. new_col
FROM table B
WHERE table A.id = Table B. id
```

- Returning

    - RETURNING x, y. .

DELETE - clause used to remove rows from the table

```
DELETE from table
WHERE row_id = 1
```

(or)

```
DELETE FROM table A
USING table B
WHERE table A . id = table B . id .
```

ALTER - allows for changes to an existing table structure

∀ - Adding, droping or renaming
- data type changing
- seting ~~def~~ DEFAULT values
- Add CHECK constraints
- Renaming table etc.

~~ALTER table~~

```
ALTER TABLE table_name
```

DROP - allows for the complete removal of column in a table.

But
(It will not remove columns used in views, triggers or stored procedures without the additional CASCADE clause

```
ALTER TABLE table_name
DROP COLUMN col_name
```

- To Remove all dependencies

```
ALTER TABLE tablename
DROP COLUMN col_name CASCADE
```

IF EXISTS

$\boxed{\text{CHECK}}$ → allows us to create more customized
constraint constrains that adhere to a certain condition.

```
CREATE TABLE examp (
    ex_id SERIAL PRIMARY KEY,
     age SMALLINT CHECK (age>21),
     Parent_age SMALLINT CHECK (Parent_age >age)
     );
```

Scanned with CamScanner

Conditional Expressions and Procedures.

① CASE < general CASE / CASE expression

- we cane use Case statement to only execute SQL code when certain conditions are met (Just like IF/else in C, C++ PY).

- General CASE

```
CASE
    WHEN condition1 THEN result1
     :
     ELSE other result
 END
```

- CASE expression

```
o CASE expression
    WHEN Val1 THEN result1
     :
    ELSE other
 END as name
```

- **COALESCE** → function which accepts an unlimited no of arguments. It returns the first argument that is not null. If all arguments are null this function will return null.

```
| COALESCE (arg-1, arg-2 .. arg-n)
```

→ This becomes helpful when querying a table that contains null values and substituting it with another value

- **CAST** → lets you convert from one datatype to other

```
| SELECT CAST ('5' AS INTEGER)
```

(or)

in PostgreSQL

```
| SELECT '5' ::INTEGER
```

- **NULLIF** → It takes in 2 inputs and returns NULL if both are equal, otherwise it returns the first argument passed.

```
| NULLIF (arg1, arg2)
```

- **VIEW** → to quickly see query with simple code

→ A view is a database obj that is of stored query

Virtual table

Scanned with CamScanner