THE iSCALE

SQL COURSE FOR BEGiNNERS

Basic to Advance SQL By Nishant Dhote Sir || The iScale

- Website: www.theiscale.com
- YouTube: https://www.youtube.com/@theiScale
- Instagram: https://www.instagram.com/theiscale

@theiscale.founders

THE iSCALE

SQL Syllabus :

Part 1

Unit – 01 Introduction to SQL-What Is SQL & Database
Unit – 02 Data Types, Primary-Foreign Keys & Constraints
Unit – 03 Create Table In SQL & Create Database
Unit – 04 INSERT UPDATE, DELETE & ALTER Table
Unit – 05 SELECT Statement & WHERE Clause
Unit – 06 How To Import Excel File (CSV) to SQL
Unit – 07 Functions in SQL & String Function
Unit – 08 Aggregate Functions –Types & Syntax
Unit – 09 Assignment Level 1 for SQL
Unit – 10 Practice & Test Questions Part 1

Part 2

Unit – 11 Group By and Having Clause
Unit – 12 Time Stamp and Extract Function, Date Time Function
Unit – 13 SQL JOINS –Types & Syntax
Unit – 14 SELF JOIN, UNION & UNION ALL
Unit – 15 Subquery
Unit – 16 Window Function –Types & Syntax
Unit – 17 Case Statement/Expression with examples
Unit – 18 CTE-Common Table Expression with examples
Unit – 19 Assignment Level 2 for SQL
Unit – 20 Practice & Test Questions Part 2

Unit - 1

INTRODUCTION TO SQL-WHAT IS SQL & DATABASE-INTRODUCTION

Introduction to SQL

- What is Database
- Excel v/s Database in SQL
- What is SQL
- It’s applications
- SQL v/s NoSQL
- Types of SQL Commands

What is Database?

Database is a system that allow users to store and organize data.

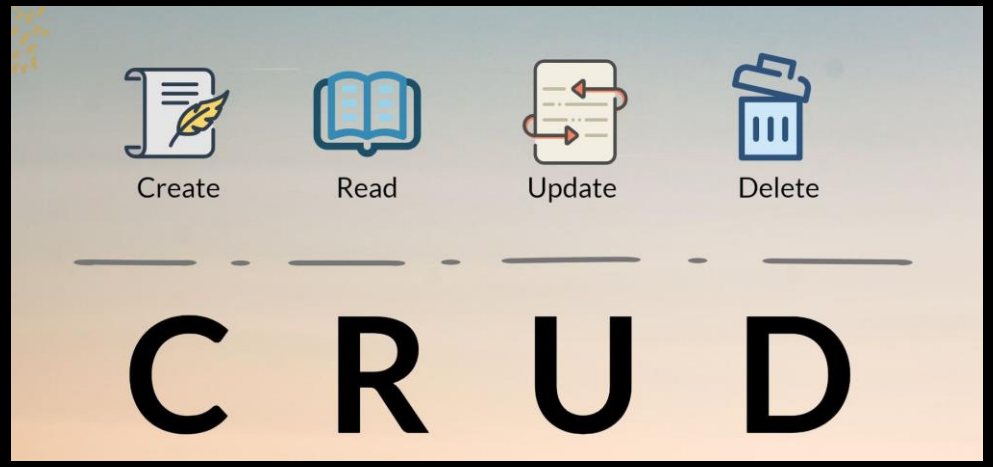
Excel vs Database

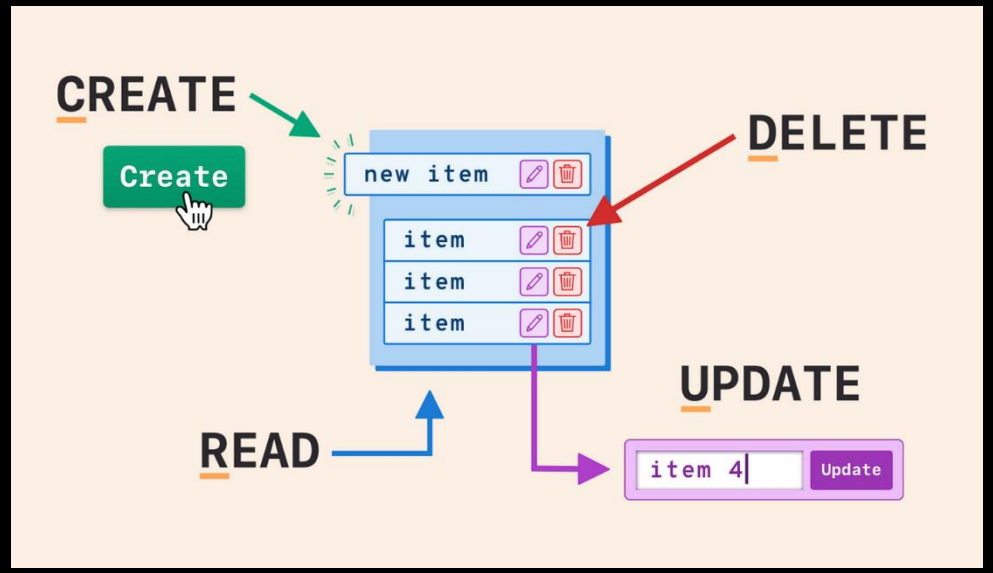
Excel	Database
Easy to use- untrained person can work	Trained person can work
Data stored less data	Stores large amount of data
Good for one time analysis, quick charts	Can automate tasks
No data integrity due to manual operation	High data integrity
Low search/filter capabilities	High search/filter capabilities

What is SQL?

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

What is SQL Application?





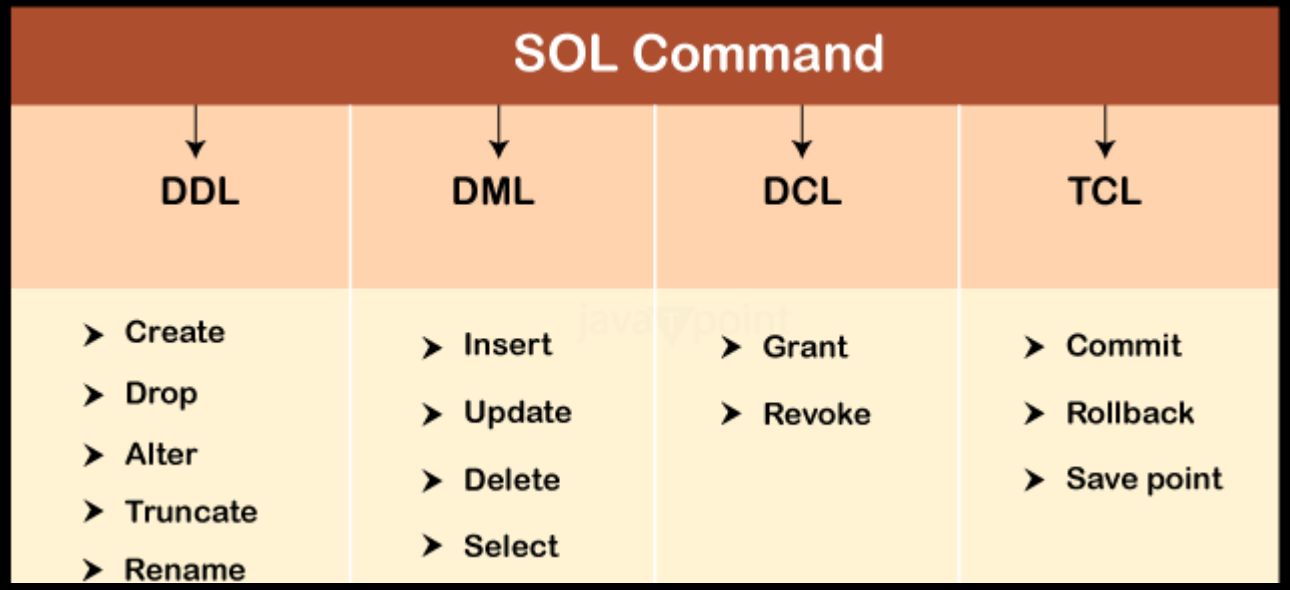
CRUD is an acronym for CREATE, READ(SELECT), UPDATE, and DELETE statements in SQL

SQL vs NoSQL

- What is Relational and Non Relational Database ?

Relational Database	Non-Relational Database
SQL database	NoSQL database
Data stored in tables	Data stored are either key-value pairs, document-based, graph databases or wide-column stores
These databases have fixed or static or predefined schema	They have dynamic schema
Low performance with huge volumes of data	Easily work with huge volumes of data
Eg: PostgreSQL, MySQL, MS SQL Server	Eg: MongoDB, Cassandra, Hbase

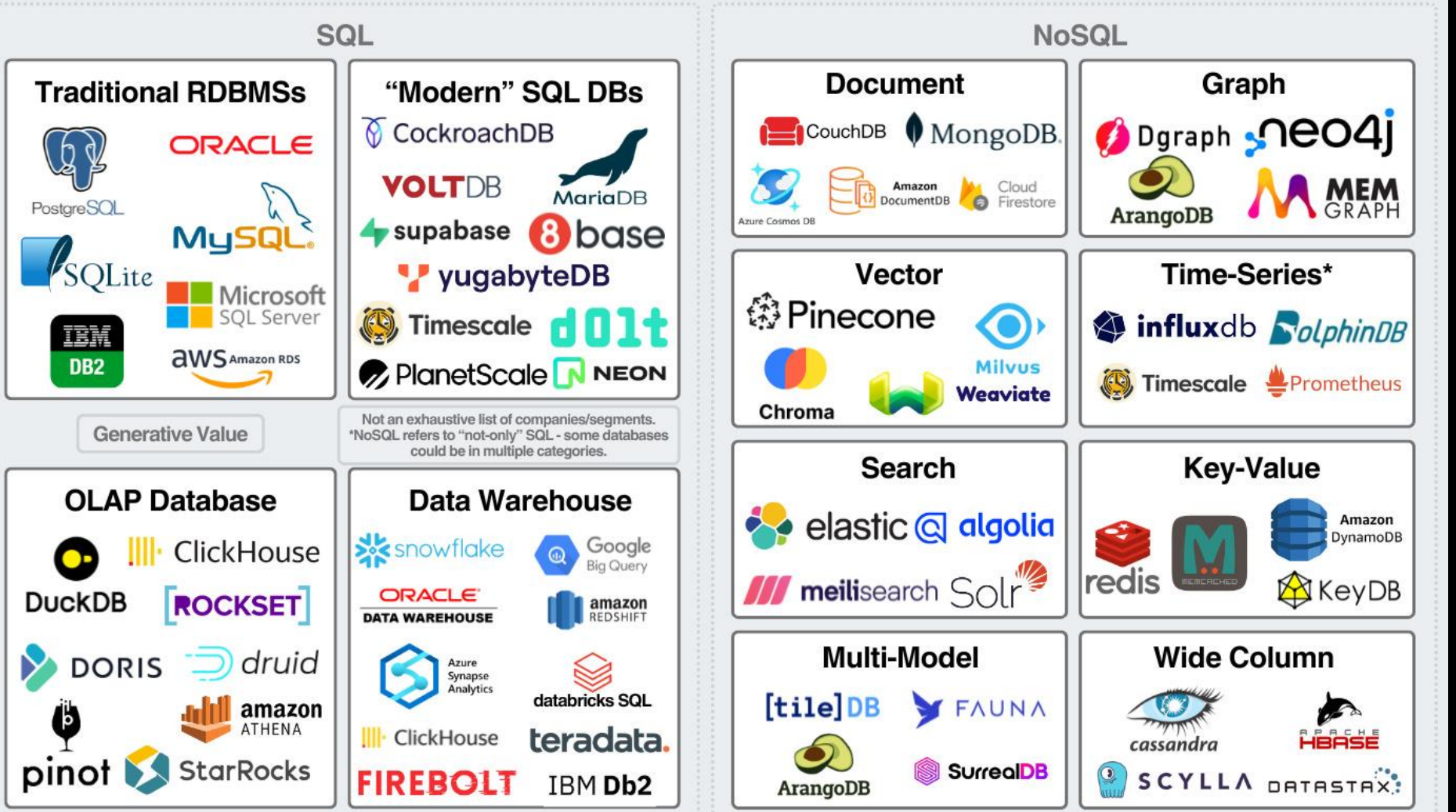
SQL Commands



DDL – Data Definition Language.
DML – Data Manipulation Language.
DCL – Data Control Language.
TCL – Transaction Control Language.

SQL Databases

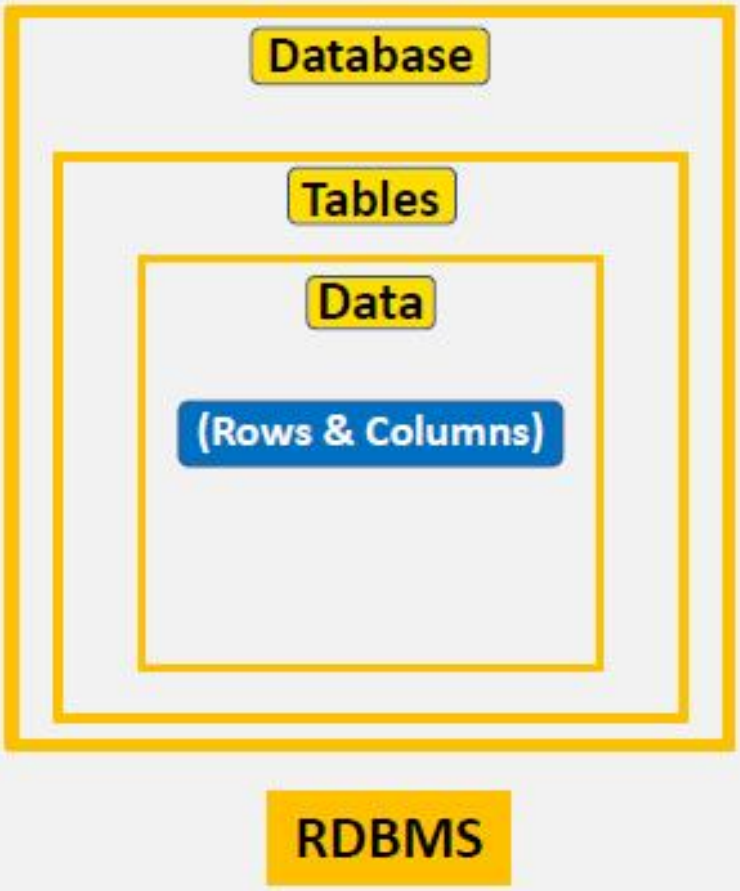
Database Ecosystem



Unit - 2

DATA TYPES, PRIMARY & FOREIGN KEYS, CONSTRAINTS

SQL Structure



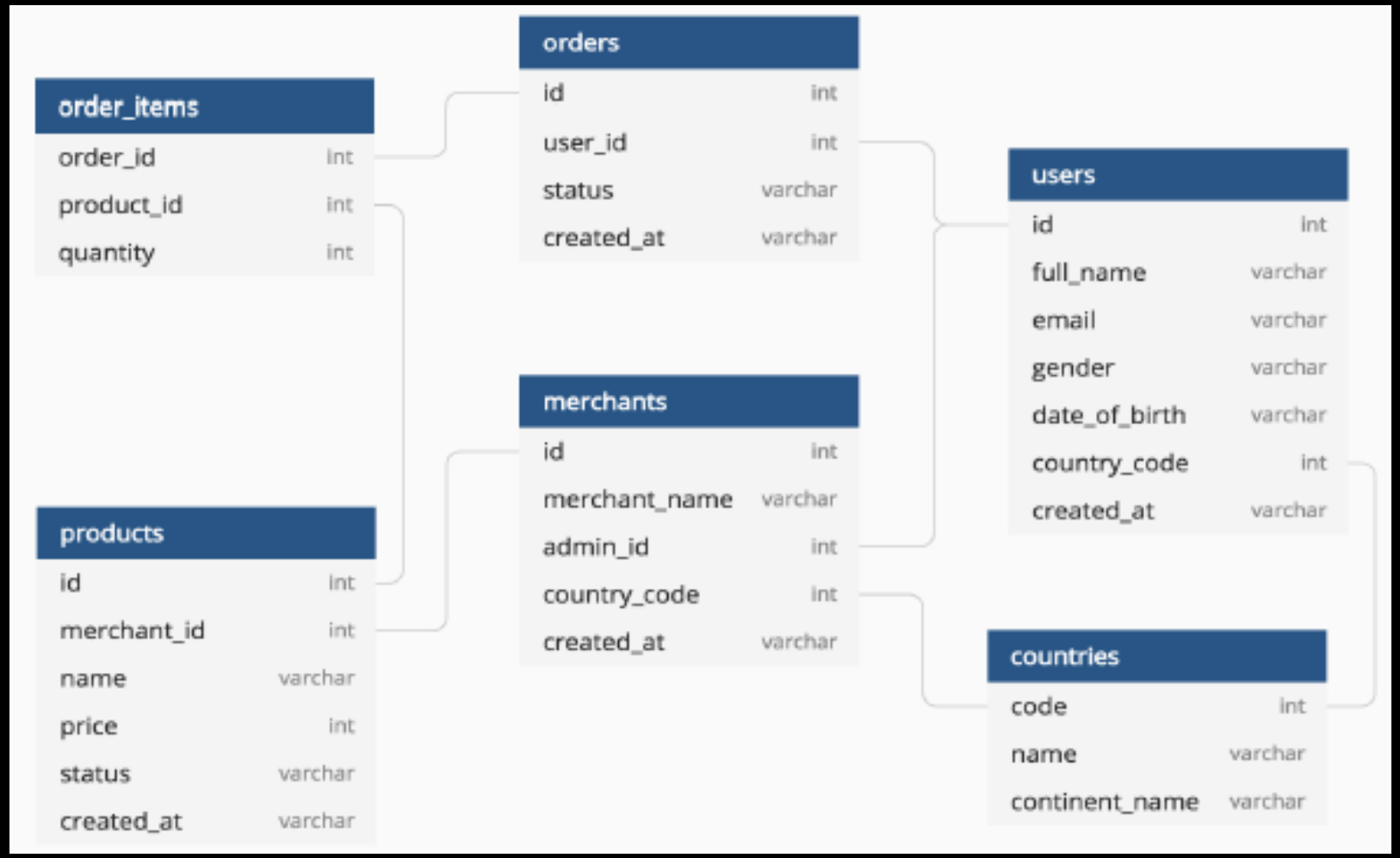
Columns

Rows

Example

```
mysql> SELECT* FROM employee;
+--------+----------+-------------------+---------+---------+
| emp_id | emp_name | emp_dept          | emp_age | emp_sex |
+--------+----------+-------------------+---------+---------+
| E00001 | JHONNY   | BACKEND DEVELOPER |      26 | male    |
| E00002 | DARSHI   | NULL              |      27 | male    |
| E00003 | JASMINE  | NULL              |      37 | female  |
| E00004 | LILLY    | NULL              |      47 | female  |
| E00005 | RONALD   | UI DEVELOPER      |      26 | male    |
+--------+----------+-------------------+---------+---------+
5 rows in set (0.01 sec)
```

Database Diagram



Creating Database & Table

- Data types
- Primary & Foreign keys
- Constraints
- SQL Commands

✓ CREATE
✓ INSERT
✓ UPDATE
✓ BACKUP
✓ DELETE
✓ ALTER
✓ DROP, TRUNCATE

Data Types

- Data type of a column defines what value the column can store in table
- Defined while creating tables in database
- Data types mainly classified into three most use categories

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

Data Types

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 values 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 format YYYY MM DD
datetime: date & time combination,
format 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 keys (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 duplicate and NULL values

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 (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
   );
```

Constraints:

Commonly used constraints in SQL:

NOT NULL -Ensures that a column cannot have a NULL value
UNIQUE -Ensures that all values in a column are different
PRIMARY KEY -A combination of a NOT NULL and UNIQUE
FOREIGN KEY -Prevents actions that would destroy links between tables (used to link multiple tables together)
CHECK -Ensures that the values in a column satisfies a specific condition
DEFAULT -Sets a default value for a column if no value is specified
CREATE INDEX -Used to create and retrieve data from the database very quickly

Unit - 3

CREATING DATABASE & TABLES IN SQL

Install PostgreSQL

Check the description box or comment section for the latest PostgreSQL installation video

CREATE TABLE:

The CREATE TABLE statement is used to create a new table in a database

Syntax

```
CREATE TABLE table_name
(
column_name1 datatypeconstraint,
column_name2 datatypeconstraint,
column_name3 datatypeconstraint,
);
```

Example

```
CREATE TABLE customer
(
CustID int8  PRIMARY KEY,
CustName varchar(50) NOT NULL,
Age  int  NOT NULL,
City  char(50),
Salary  numeric
);
```

Unit - 4

INSERT, UPDATE, DELETE VALUES IN TABLE WITH ALTER, DROP & TRUNCATE TABLE

INSERT VALUES IN TABLE:

The INSERT INTO statement is used to insert new records in a table

Syntax

```
INSERT INTO TABLE_NAME
(column1, column2, column3,...columnN)
VALUES
(value1, value2, value3,...valueN);
```

Example

```
INSERT INTO customer
(CustID, CustName, Age, City, Salary)
VALUES
(1, 'Nishant', 26, 'Delhi', 19000),
(2, 'Swati', 22, 'Bangalore', 11000),
(3, 'Ragini', 19, 'Goa', 24000),
(4, 'Saurabh', 32, 'Pune', 10000);
```

UPDATE VALUES IN TABLE:

The UPDATE command is used to update existing rows in a table

Syntax

```
UPDATE TABLE_NAME
SET “Column_name1” = ‘value1’, “Column_name2” = ‘value2’
WHERE “ID” = ‘value’
```

Example

```
UPDATE customer
SET CustName= 'Prashant' , Age= 32
WHERE CustID= 1;
```

DELETE VALUES IN TABLE

The DELETE statement is used to delete existing records in a table

Syntax

DELETE FROM table_nameWHERE condition;

Example

DELETE FROM customer
WHERE CustID= 3;

ALTER TABLE:

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table

- **ALTER TABLE -ADD Column Syntax**

  ALTER TABLE table_name
  ADD COLUMN column_name;

- **ALTER TABLE -DROP COLUMN Syntax**

  ALTER TABLE table_name
  DROP COLUMN column_name;

- **ALTER TABLE -ALTER/MODIFY COLUMN Syntax**

  ALTER TABLE table_name
  ALTER COLUMN column_namedatatype;

ALTER TABLE EXAMPLE

- **ADD Column Syntax:** Adding new ‘Gender’ column to customer table

```
ALTER TABLE customer
ADD COLUMN Gender varchar(10);
```

- **ALTER/MODIFY COLUMN Syntax:** changing Gender column data type from varchar(10) to char(10)

```
ALTER TABLE customer
ALTER COLUMN Gender char(10);
```

- **DROP COLUMN Syntax:** Deleting Gender column from customer table

```
ALTER TABLE customer
DROP COLUMN Gender;
```

DROP & TRUNCATE TABLE

The DROP TABLE command deletes a table in the database

- **Syntax**

  DROP TABLE table_name;

  The TRUNCATE TABLE command deletes the data inside a table, but not the table itself

Syntax

TRUNCATE TABLE table_name;

Unit - 5

SELECT STATEMENT & WHERE CLAUSE
OPERATORS, LIMIT , ORDER BY CLAUSE

CREATING DATASET

Creating a Classroom Dataset for Practice

```
CREATE TABLE classroom
(
rollno int8  PRIMARY KEY,
name  varchar(50) NOT NULL,
house char(12) NOT NULL,
grade  char(1)
);

INSERT INTO classroom
(rollno, name, house, grade)
VALUES
(1, 'Nishant', 'Amber', 'B'),
(2, 'Aditya', 'Agni', 'A'),
(3, 'Tanya', 'Jal', 'B'),
(4, 'Ragini', 'Agni', 'A'),
(5, 'Aditya', 'Yayu', 'B');
```

SELECT STATEMENT

The SELECT statement is used to select data from a database.

- **Syntax**

  SELECT column_name FROM table_name;

To select all the fields available in the table

- **Syntax**

  SELECT * FROM table_name;

To select distinct/unique fields available in the table

- **Syntax**

  SELECT DISTINCT Column_name FROM table_name;

WHERE CLAUSE

The WHERE clause is used to filter records.
It is used to extract only those records that fulfill a specified condition

- **Syntax**

SELECT column_name FROM table_name
WHERE conditions;

- **Example**

SELECT name FROM classroom
WHERE grade='A';

OPERATORS IN SQL

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

1. **Arithmetic operators :**arithmetic operations on numeric values

Example: Addition (+), Subtraction (-), Multiplication (*), Division (/), Modulus (%)

2. Comparison operators: compare two different data of SQL table

Example: Equal (=), Not Equal (!=), Greater Than (>), Greater Than Equals to (>=)

3. Logical operators: perform the Boolean operations

Example: ALL, IN, BETWEEN, LIKE, AND, OR, NOT, ANY

4. Bitwise operators: perform the bit operations on the Integer values

Example: Bitwise AND (&), Bitwise OR(|)

LIMIT CLAUSE

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

Example: below code will return 5 rows of data

```
SELECT column_name FROM table_name
LIMIT 5;
```

ORDER BY CLAUSE

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

Example: below code will sort the output data by column name in ascending order

```
SELECT column_name FROM table_name

SELECT * FROM classroom
ORDER BY name ASC;
```

Unit - 6

HOW TO IMPORT EXCEL FILE (CSV) TO SQL

CSV FILES

THE iSCALE

👉Download customer csv file: https://github.com/TheiScale/YouTube-Video-Notes/blob/main/customer.csv

```
CREATE TABLE customer
(
customer_id int8 PRIMARY KEY,
first_name varchar (50),
last_name varchar(50),
email varchar(100),
address_id int8
                )

 COPY
 customer(customer_id,first_name,last_name,email,address_id)
 FROM 'E:\customer.csv'
 DELIMITER ','
 CSV HEADER;
```

Unit - 7

FUNCTIONS IN SQL & STRING FUNCTION

FUNCTIONS IN SQL

Functions 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
   - Example: rand(), round(), upper(), lower(), count(), sum(), avg(), max(), etc

2. User-Defined Function : Once you define a function, you can call it in the same way as the built-in functions

MOST USED STRING FUNCTIONS

String functions are used to perform an operation on input string and return an output string

- **UPPER()** converts the value of a field to uppercase
- **LOWER()** converts the value of a field to lowercase
- **LENGTH()** returns the length of the value in a text field
- **SUBSTRING()** extracts a substring from a string
- **NOW()** returns the current system date and time
- **FORMAT()** used to set the format of a field
- **CONCAT()** adds two or more strings together
- **REPLACE()** Replaces all occurrences of a substring within a string, with a new substring
- **TRIM()** removes leading and trailing spaces (or other specified characters) from a string

Unit - 8

AGGREGATE FUNCTIONS – TYPES & SYNTAX

MOST USED AGGREGATE FUNCTIONS

Aggregate function performs a calculation on multiple values and returns a single value.
And Aggregate function are often used with GROUP BY & SELECT statement

COUNT() returns number of values
SUM() returns sum of all values
AVG() returns average value
MAX() returns maximum value
MIN() returns minimum value
ROUND() Rounds a number to a specified number of decimal places

Practice CSV FILES Import Dataset

THE iSCALE

👉Download account csv file: https://github.com/TheiScale/YouTube-Video-Notes/blob/main/accounts.csv

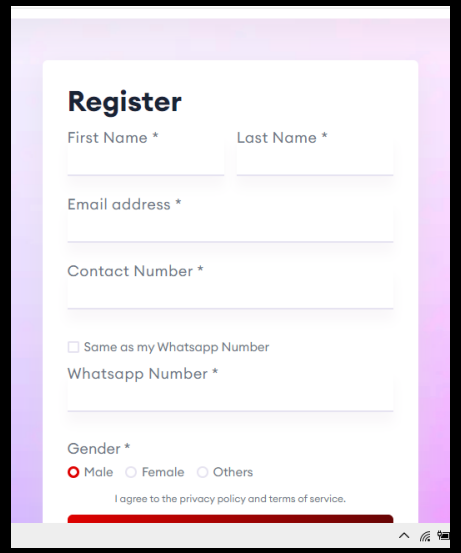
```
CREATE TABLE accounts
(
customer_id int8 PRIMARY KEY,
amount int8,
mode varchar(50),
payment_date date


COPY accounts(customer_id,amount,mode,payment_date)
FROM 'E:\accounts.csv'
DELIMITER ','
CSV HEADER;
```

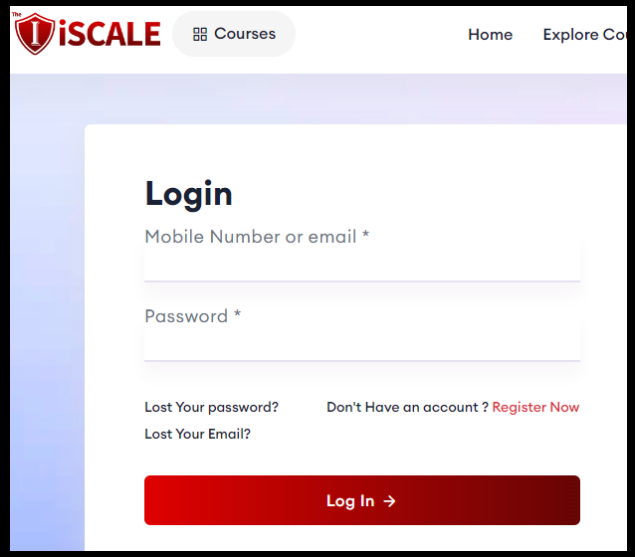
Unit - 09

ASSIGNMENT LEVEL 1 FOR SQL

Step 1 Register Or Login : www.theiscale.com



OR



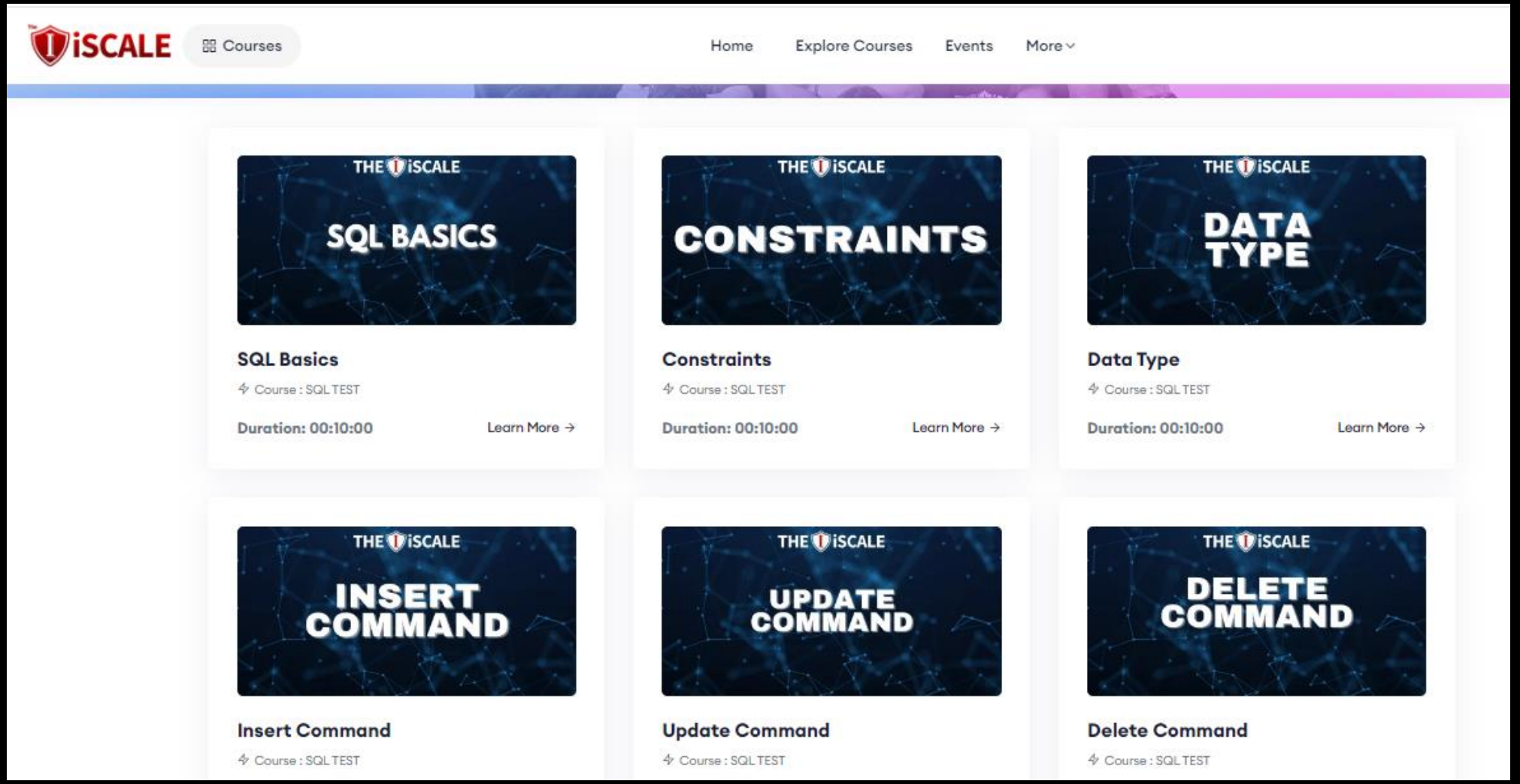
Step 2 Enroll Free Data Analytics Course

Step 3 : Click Latest Practice Test and Download Assignment PDF

Unit - 10

PRACTICE & TEST QUESTIONS PART 1

Step 3 : Click Latest Practice Test



THE iSCALE

SQL Syllabus :

Part 1

Unit – 01 Introduction to SQL-What Is SQL & Database
Unit – 02 Data Types, Primary-Foreign Keys & Constraints
Unit – 03 Create Table In SQL & Create Database
Unit – 04 INSERT UPDATE, DELETE & ALTER Table
Unit – 05 SELECT Statement & WHERE Clause
Unit – 06 How To Import Excel File (CSV) to SQL
Unit – 07 Functions in SQL & String Function
Unit – 08 Aggregate Functions –Types & Syntax
Unit – 09 Assignment Level 1 for SQL
Unit – 10 Practice & Test Questions Part 1

Part 2

Unit – 11 Group By and Having Clause
Unit – 12 Time Stamp and Extract Function, Date Time Function
Unit – 13 SQL JOINS –Types & Syntax
Unit – 14 SELF JOIN, UNION & UNION ALL
Unit – 15 Subquery
Unit – 16 Window Function –Types & Syntax
Unit – 17 Case Statement/Expression with examples
Unit – 18 CTE-Common Table Expression with examples
Unit – 19 Assignment Level 2 for SQL
Unit – 20 Practice & Test Questions Part 2

Unit - 11

GROUP BY & HAVING CLAUSE

GROUP BY STATEMENT

The GROUP BY statement group rows that have the same values into summary rows.
It is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns

- Syntax

```
SELECT column_name(s)
FROM table_name
GROUP BY column_name(s);
```

- Example

```
SELECT mode, SUM(amount) AS Total_Amount
FROM accounts
GROUP BY mode
ORDER BY Total_Amount ASC
```

HAVING CLAUSE

The **HAVING** clause is used to apply a filter on the result of **GROUP BY** based on the specified condition. 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(s)
FROM table_name
WHERE condition(s)
GROUP BY column_name(s)
HAVING condition(s)
```

Example

```
SELECT mode, COUNT(amount) AS total_amount
FROM accounts
GROUP BY mode
HAVING COUNT(amount) >= 3
ORDER BY total DESC
```

Unit - 12

TIME STAMP AND EXTRACT FUNCTION

DATE - TIME FUNCTION

TIMESTAMP

The **TIMESTAMP** data type is used for values that contain both date and time parts

- **TIME** contains only time, format HH:MI:SS
- **DATE** contains on date, format YYYY-MM-DD
- **YEAR** contains on year, format YYYY or YY
- **TIMESTAMP** contains date and time, format YYYY-MM-DD HH:MI:SS
- **TIMESTAMPTZ** contains date, time and time zone

TIMESTAMP FUNCTIONS / OPERATORS

Below are the TIMESTAMP functions and operators in SQL:

- **SHOW TIMEZONE**
- **SELECT NOW()**
- **SELECT TIMEOFDAY()**
- **SELECT CURRENT_TIME**
- **SELECT CURRENT_DATE**

EXTRACT FUNCTION

The **EXTRACT()** function extracts a part from a given date value.

Syntax: SELECT **EXTRACT**(MONTH FROM date_field) FROM Table

- **YEAR**
- **QUARTER**
- **MONTH**
- **WEEK**
- **DAY**
- **HOUR**
- **MINUTE**
- **DOW**–day of week
- **DOY**–day of year

Unit - 13

SQL JOINS – TYPES SYNTAX FUNCTION

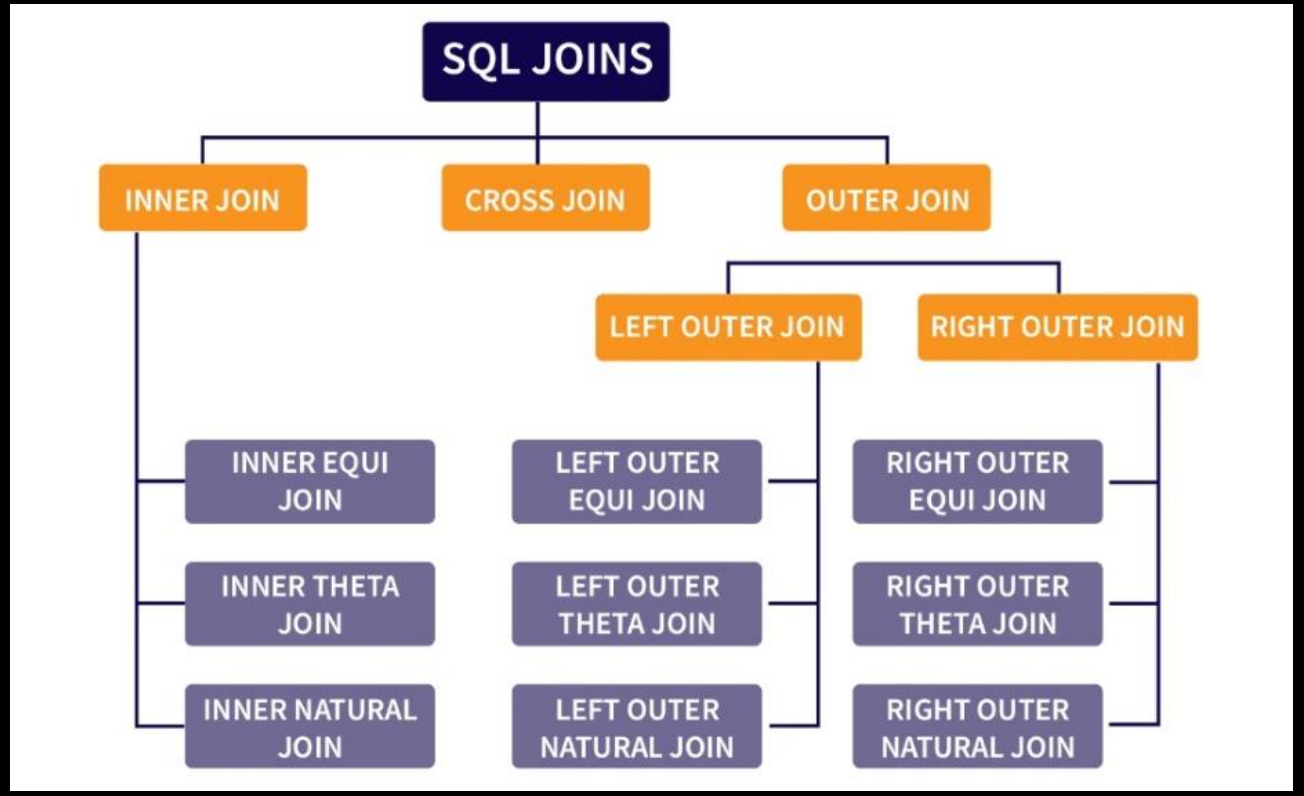
TOPICS IN JOIN

- WHAT IS JOIN?
- JOIN TYPES
- WHICH JOIN TO USE
- JOIN SYNTAX
- EXAMPLES IN SQL

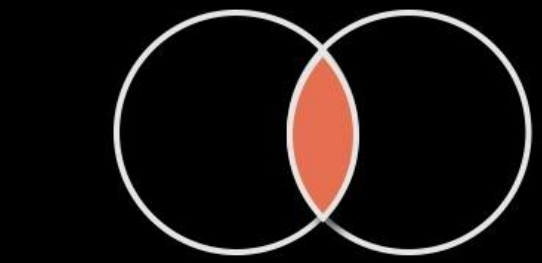
- **JOIN** means to combine something.
- A **JOIN** clause is used to combine data from two or more tables, based on a related column between them

TYPES OF JOIN

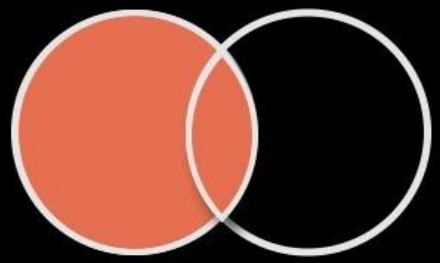
- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN



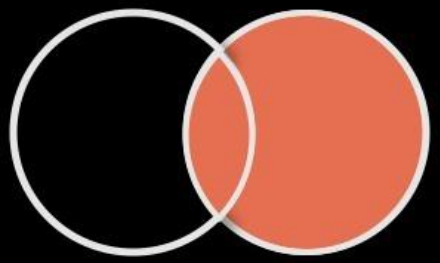
SQL Joins



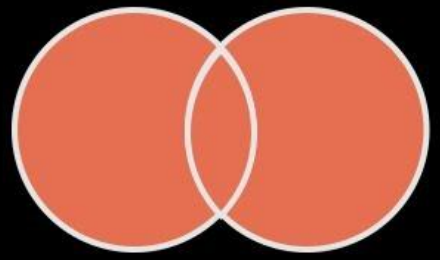
Inner Join



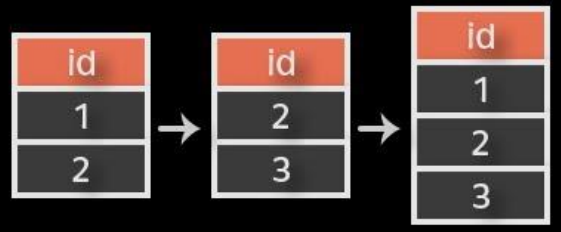
Left Join



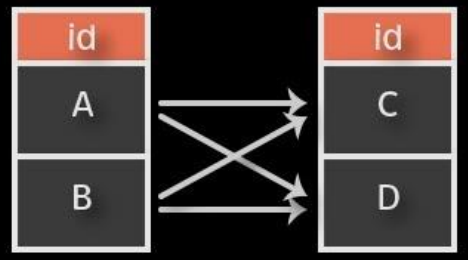
Right Join



Full Join

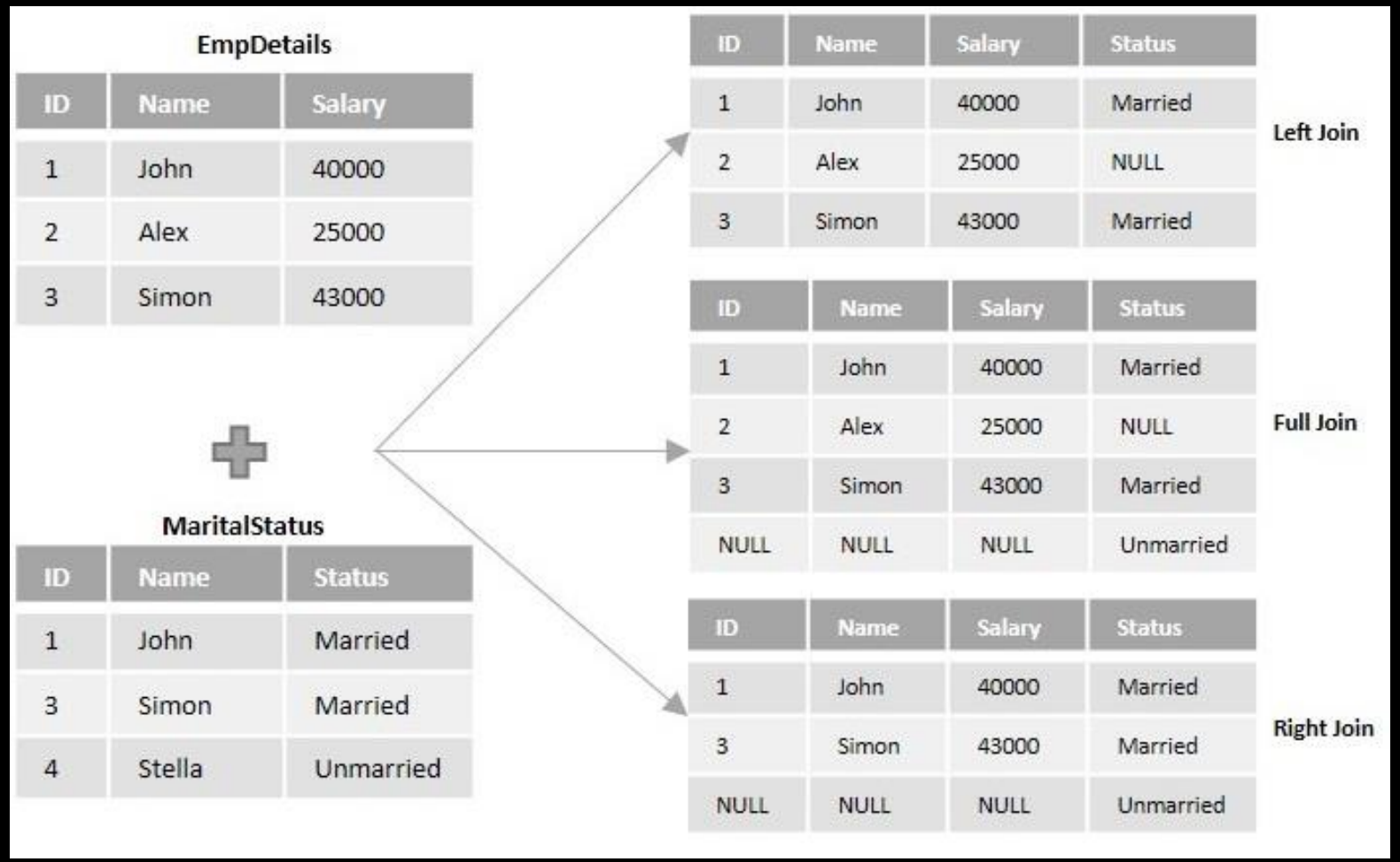


Union



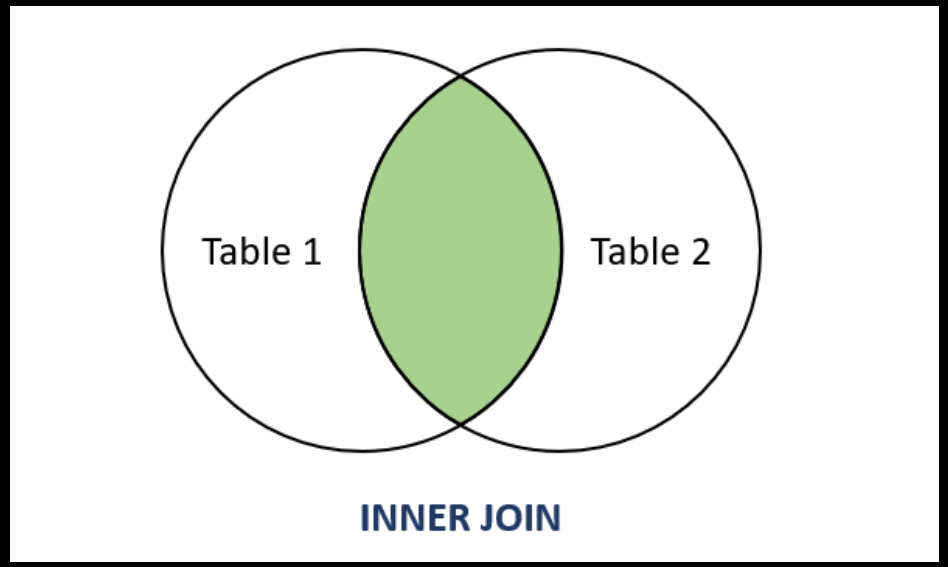
Cross

JOIN EXAMPLE



INNER JOIN

Returns records that have matching values in both tables.



INNER JOIN

Returns records that have matching values in both tables.

- **Syntax**

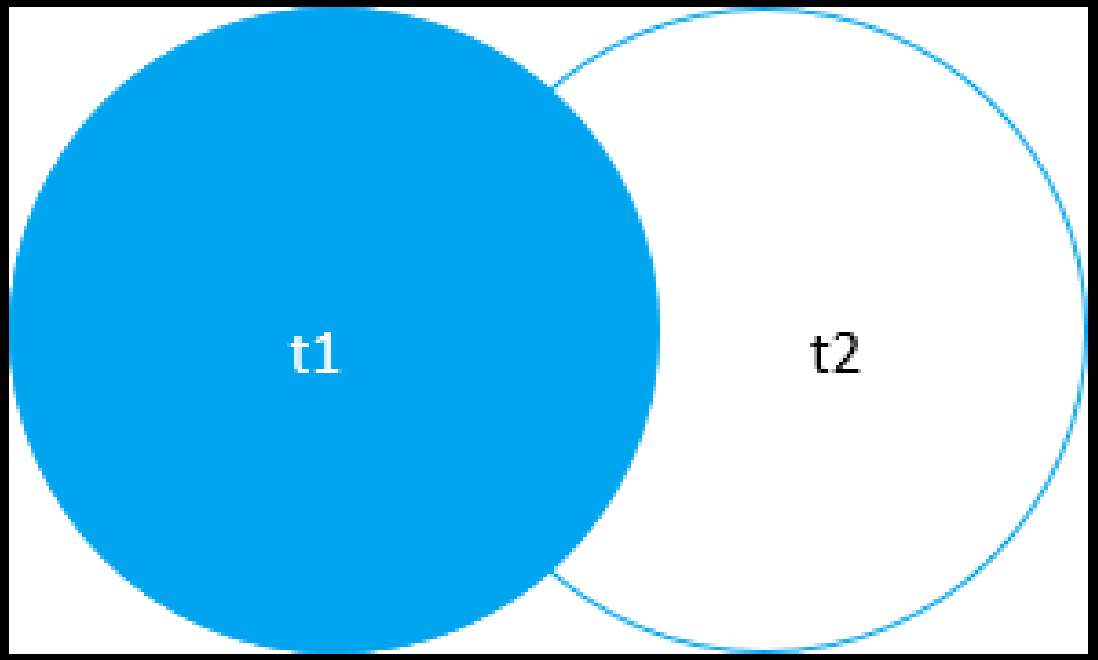
```
SELECT column_name(s)
FROM TableA
INNER JOIN TableB
ON TableA.col_name= TableB.col_name
```

- **Example**

```
SELECT *
FROM customer AS c
INNER JOIN accounts AS a
ON c.customer_id= a.customer_id
```

LEFT JOIN

Returns all records from the left table, and the matched records from the right table



LEFT JOIN

Syntax

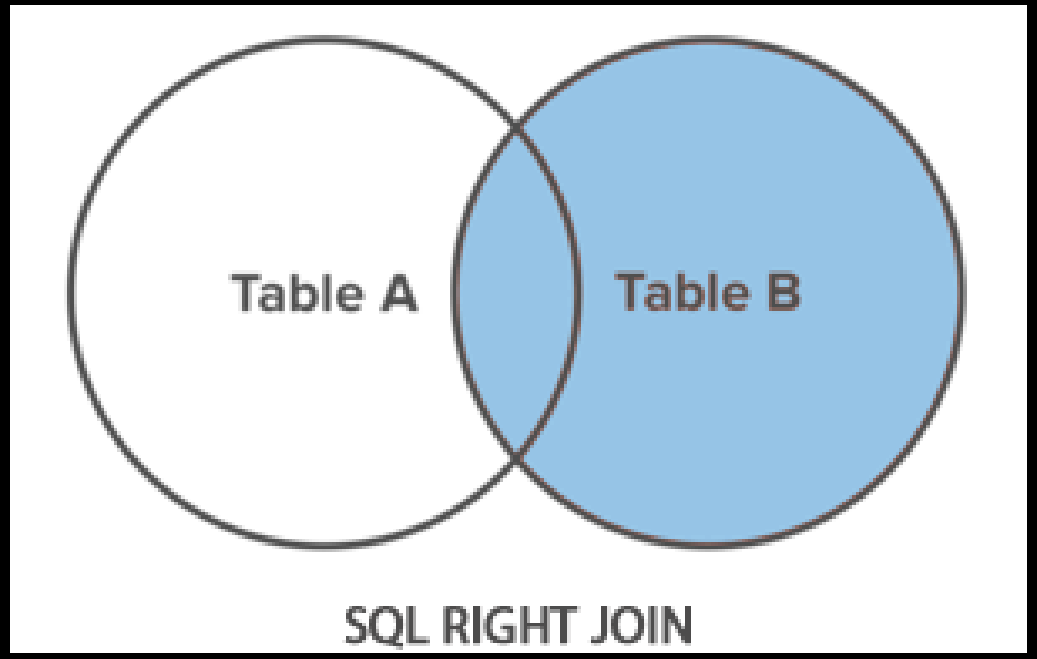
SELECT column_name(s)
FROM **TableA**
LEFT JOIN **TableB**
ON **TableA**.col_name= **TableB**.col_name

Example

```
SELECT *
FROM customer AS c
LEFT JOIN accounts AS a
ON c.customer_id= a.customer_id
```

RIGHT JOIN

Returns all records from the right table, and the matched records from the left table



RIGHT JOIN

Syntax

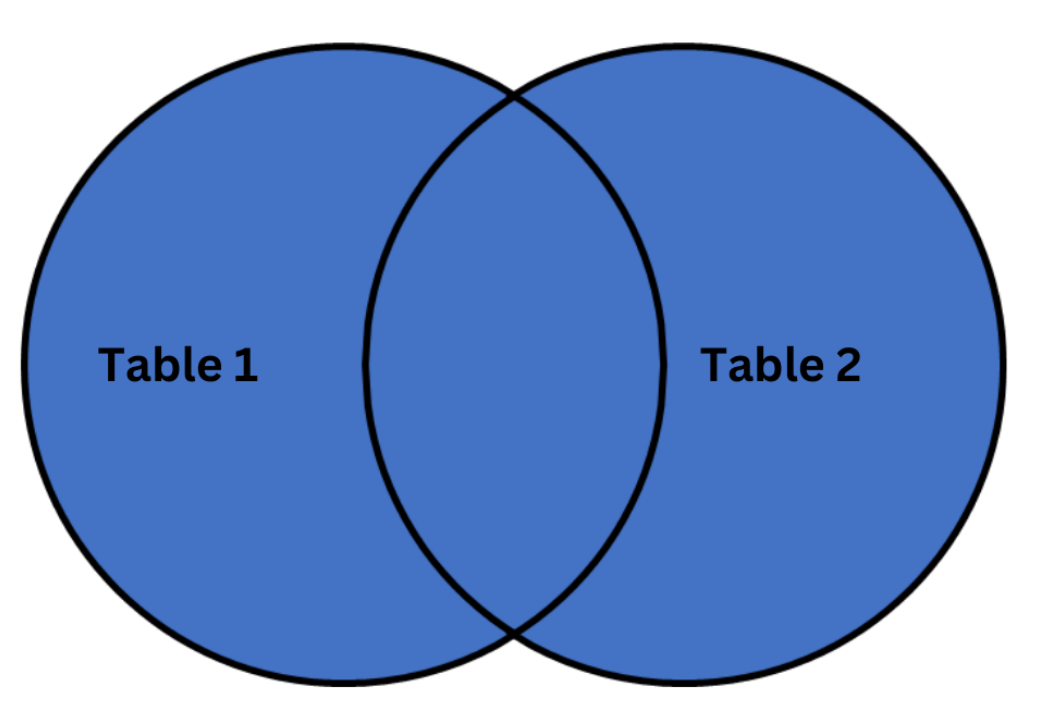
SELECT column_name(s)
FROM **TableA**
RIGHT JOIN **TableB**
ON **TableA**.col_name= **TableB**.col_name

Example

```
SELECT *
FROM customer AS c
RIGHT JOIN accounts AS a
ON c.customer_id= a.customer_id
```

FULL JOIN

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



FULL OUTER JOIN

Syntax

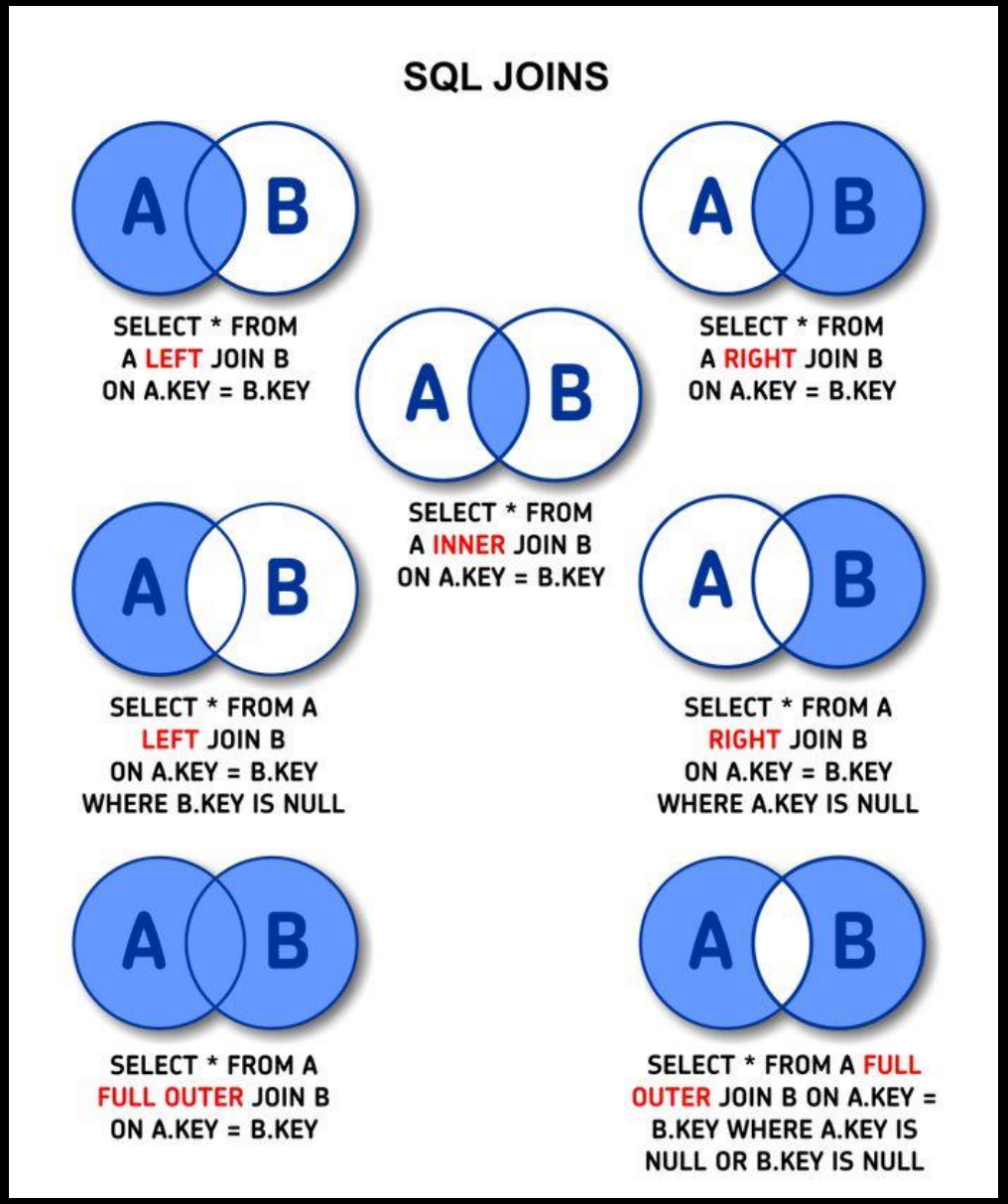
SELECT column_name(s)
FROM **TableA**
FULL OUTER JOIN **TableB**
ON TableA.col_name= **TableB**.col_name

Example

```
SELECT *
FROM customer AS c
FULL JOIN accounts AS a
ON c.customer_id= a.customer_id
```

WHICH JOIN TO USE

- **INNER JOIN:** Returns records that have matching values in both tables
- **LEFT JOIN:** Returns all records from the left table, and the matched records from the right table
- **RIGHT JOIN:** Returns all records from the right table, and the matched records from the left table
- **FULL JOIN:** Returns all records when there is a match in either left or right table


SQL JOINS
A B
SELECT * FROM
A LEFT JOIN B
ON A.KEY = B.KEY
A B
SELECT * FROM
A RIGHT JOIN B
ON A.KEY = B.KEY
A B
SELECT * FROM
A INNER JOIN B
ON A.KEY = B.KEY
A B
SELECT * FROM A
LEFT JOIN B
ON A.KEY = B.KEY
WHERE B.KEY IS NULL
A B
SELECT * FROM A
RIGHT JOIN B
ON A.KEY = B.KEY
WHERE A.KEY IS NULL
A B
SELECT * FROM A
FULL OUTER JOIN B
ON A.KEY = B.KEY
A B
SELECT * FROM A FULL
OUTER JOIN B ON A.KEY =
B.KEY WHERE A.KEY IS
NULL OR B.KEY IS NULL

Unit - 14

SELF JOIN, UNION & UNION ALL

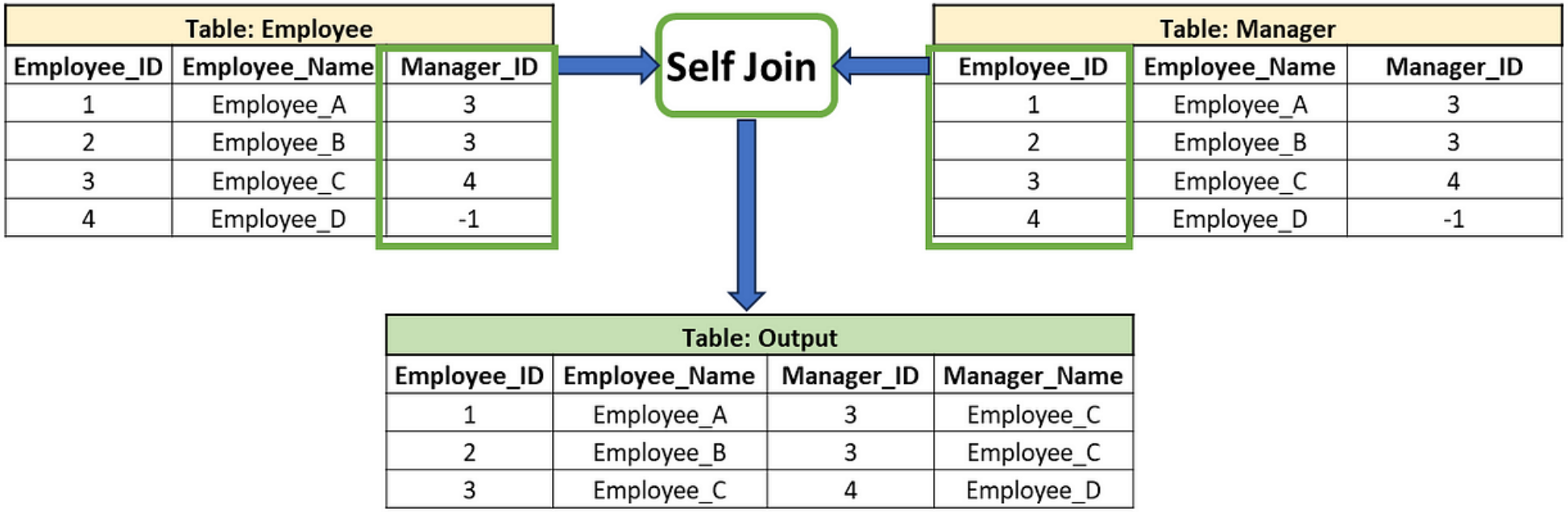
SELF JOIN

- A **self join** is a regular join in which a table is joined to itself
- **SELF Joins** are powerful for comparing values in a column of rows with the same table

Syntax

```
SELECT *
FROM classroom AS T1
JOIN classroom AS T2
ON T2.rollno = T1.st_id
```

SELF JOIN EXAMPLE



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 columns selected and expressions
- The same data type and
- Have them in the same order

Syntax

SELECT column_name(s) FROM **TableA**
UNION
SELECT column_name(s) FROM **TableB**

Example

```
SELECT customer_id
FROM accounts
UNION
SELECT customer_id
FROM customer
```

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(s) FROM **TableA**
UNION ALL
SELECT column_name(s) FROM **TableB**

Example

```
SELECT customer_id
FROM accounts
UNION ALL
SELECT customer_id
FROM customer
```

Unit - 15

SUB QUERY SQL

SUB QUERY

A **Subquery** or Inner query or a Nested query allows us to create complex query on the output of another query

- Sub query syntax involves two SELECT statements

Syntax

SELECT column_name(s)
FROM table_name
WHERE column_name***operator***
(**SELECT column_nameFROM table_nameWHERE ...**);

SUB QUERY EXAMPLE

Question: Find the details of customers, whose payment amount is more than the average of total amount paid by all customers

SUB QUERY Solution

```
---Find the average value
SELECT avg(amount) from accounts

---Filter the customer details  (grater then or >) avg value

SELECT *
FROM accounts
WHERE amount > 57

SELECT * FROM accounts

---Dynamic Solution Use Sub Query

SELECT *
FROM accounts
WHERE amount > (SELECT avg(amount) from accounts)
```

Unit - 16

WINDOWS 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.

WINDOW FUNCTION SYNTAX

```
SELECT column_name1,
 window_function (column_name2)
 OVER(
        [PARTITION BY column_name1]
        [ORDER BY column_name3])
        [ROWS or <Range Clause>]
        FROM table_name;
```

Select a function

- Aggregate functions
- Ranking functions
- Analytic functions

Define a Window

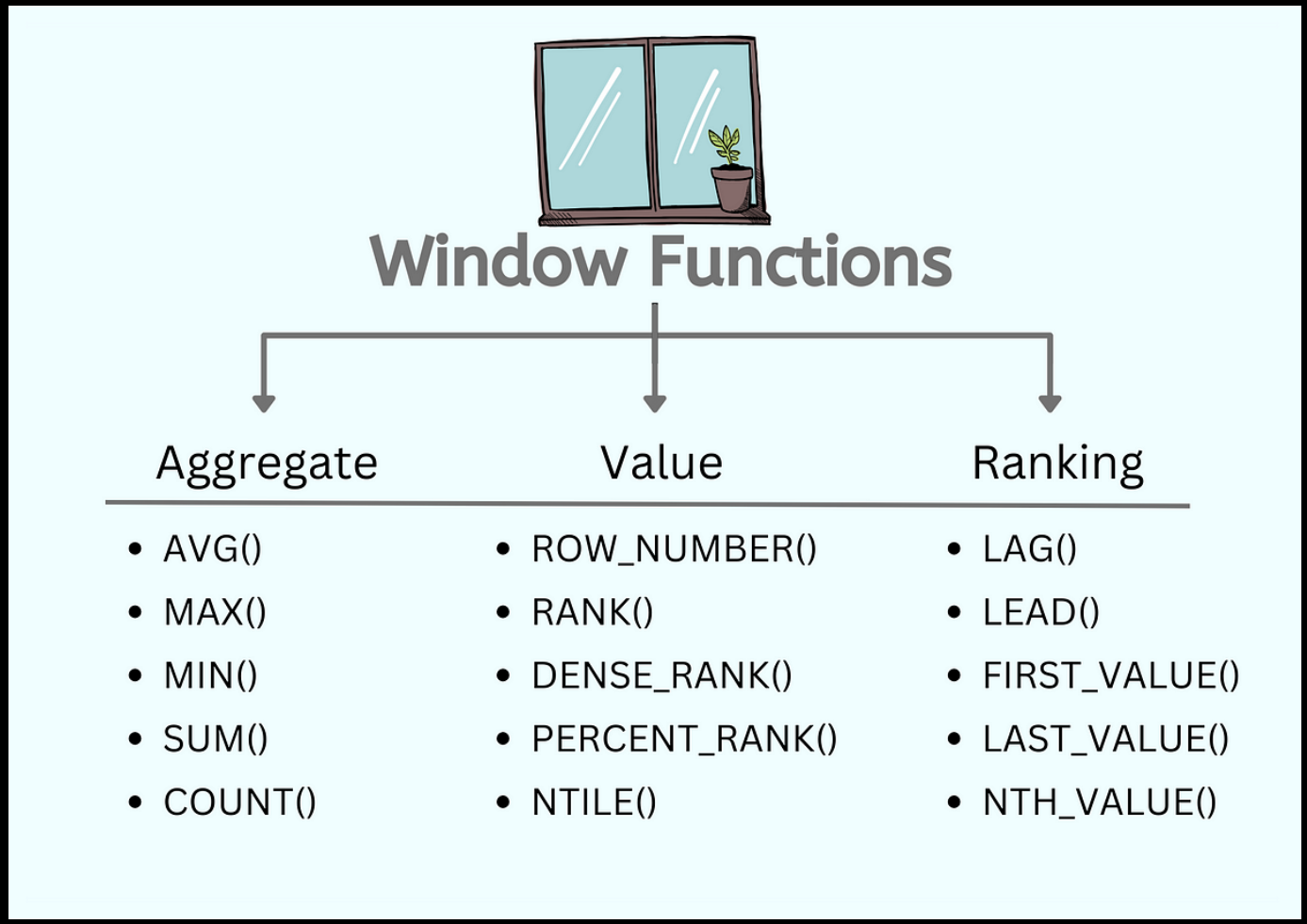
- PARTITION BY
- ORDER BY
- ROWS

WINDOW FUNCTION TERMS

Let's look at some definitions:

- **WINDOW FUNCTION** applies aggregate, ranking and analytic functions over a particular window; for example, sum, avg, or row_number
- **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
- **OVER** is just to signify that this is a window function
- **PARTITION BY** divides the rows into partitions so we can specify which rows to use to compute the window function
- **ORDER BY** is used so that we can order the rows within each partition. This is optional and does not have to be specified
- **ROW S**can be used if we want to further limit the rows within our partition. This is optional and usually not used

WINDOW FUNCTION TYPES



Practice CSV FILES Import Dataset

☞Download student csv file: https://github.com/TheiScale/YouTube-Video-Notes/blob/main/student.csv

```
SELECT Marks, st_subject,

SUM(Marks) OVER (PARTITION BY st_subject ORDER BY Marks) AS "Total",
AVG(Marks) OVER (PARTITION BY st_subject ORDER BY Marks) AS "Average",
COUNT(Marks) OVER (PARTITION BY st_subject ORDER BY Marks) AS "Count",
MIN(Marks) OVER (PARTITION BY st_subject ORDER BY Marks) AS "Minimum",
MAX(Marks) OVER (PARTITION BY st_subject ORDER BY Marks) AS "Maximum"
FROM student
```

Example Aggregate Function

```
SELECT Marks, st_subject,
SUM(Marks) OVER ( ORDER BY Marks ROWS BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) AS "Total",
AVG(Marks) OVER ( ORDER BY Marks ROWS BETWEEN UNBOUNDED PRECEDING AND
UNBOUNDED FOLLOWING) AS "Average",
Count(Marks) OVER ( ORDER BY Marks ROWS BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) AS "Count",
MAX(Marks) OVER ( ORDER BY Marks ROWS BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) AS "Maximum",
MIN(Marks) OVER ( ORDER BY Marks ROWS BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING) AS "Minimum"
FROM student
```

Example Ranking Function

```
SELECT Marks ,
ROW_NUMBER() OVER (ORDER BY Marks ) AS "ROW_NUMBER",
RANK() OVER (ORDER BY Marks ) AS "RANK",
DENSE_RANK() OVER (ORDER BY Marks ) AS "DENSE_RANK",
PERCENT_RANK() OVER (ORDER BY Marks ) AS "PERCENT_RANK"
FROM student
```

Example Analytic Function

```
SELECT Marks ,
FIRST_VALUE (Marks ) OVER ( ORDER BY Marks ) AS "FIRST_VALUE",
LAST_VALUE (Marks ) OVER ( ORDER BY Marks ) AS "LAST_VALUE",
LEAD (Marks ) OVER ( ORDER BY Marks ) AS "LEAD",
LAG (Marks ) OVER ( ORDER BY Marks ) AS "LAG"
FROM student
```

Unit - 17

CASE EXPRESSION / STATEMENT

CASE EXPRESSION

- The CASE expression goes through conditions and returns a value when the first condition is met (like if-then-else statement). If no conditions are true, it returns the value in the ELSE clause.

- If there is no ELSE part and no conditions are true, it returns NULL.

- Sometimes Case Expression also called Case Statement

CASE STATEMENT

THE iSCALE

Syntax:

```
CASE
WHEN condition1 THEN result1
WHEN condition2 THEN result2
WHEN condition NTHEN resultN
ELSE other_result
END;
```

Example:

```
SELECT customer_id , amount,
CASE
WHEN amount > 55 THEN 'Expensive'
WHEN amount = 55 THEN 'Less Expensive'
ELSE 'No Expensive'
END AS Show_Status
FROM accounts
```

CASE EXPRESSION

THE iSCALE

Syntax:

CASE EXPRESSION

```
WHEN value1 THEN result1
WHEN value2 THEN result2
WHEN valueN THEN resultN
ELSE other_result
END;
```

Example:

```
SELECT customer_id, amount,
CASE amount
WHEN 55 THEN 'Prime Customer'
WHEN 45 THEN 'Plus Customer'
ELSE 'Regular Customer'
END AS CustomerStatus
FROM accounts
```

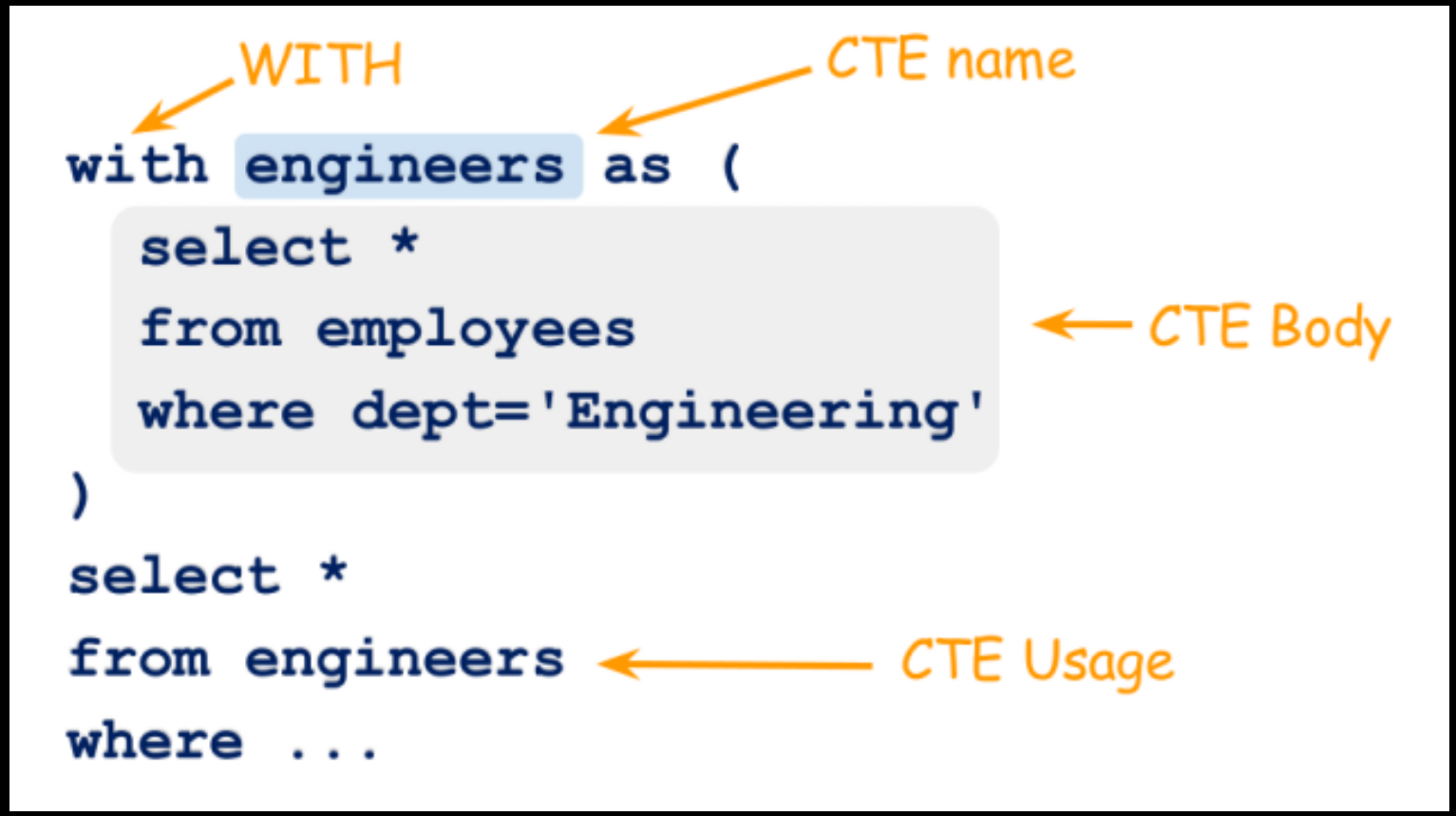
Unit - 18

COMMON TABLE EXPRESSION

COMMON TABLE EXPRESSION (CTE)

- A common table expression, or CTE, is a temporary named result set created from a simple SELECT statement that can be used in a subsequent SELECT statement
- We can define CTEs by adding a WITH clause directly before SELECT, INSERT, UPDATE, DELETE, or MERGE statement
- The WITH clause can include one or more CTEs separated by commas

CTE Syntax:


WITH
CTE name
with engineers as (
select *
from employees
where dept='Engineering'
)
CTE Body
select *
from engineers
CTE Usage
where ...

EXAMPLE Use CTE Windows:

```
WITH
cte_table AS (SELECT *, AVG(amount) OVER(ORDER BY
        p.customer_id ) AS "Average_Price",
        COUNT(address_id ) OVER(ORDER BY c.customer_id ) AS
"Count"
   FROM accounts as p
        INNER JOIN customer AS c
ON p.customer_id = c.customer_id
)
SELECT first_name , last_name, amount
FROM cte_table
```

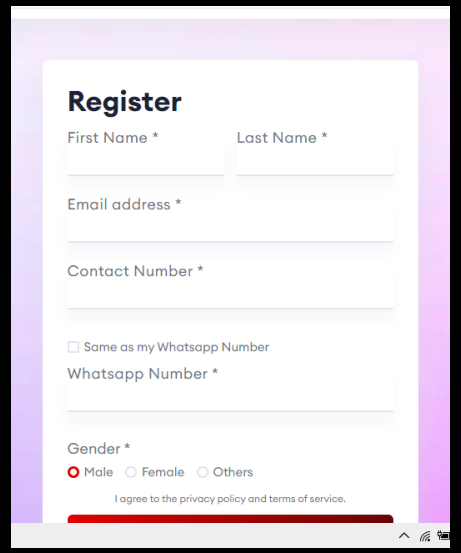
EXAMPLE JOIN Use CTE:

```
WITH
cte_table AS ( SELECT mode, MAX(amount) AS highest_price , SUM(amount)
AS total_price
                                        FROM accounts
                                        GROUP BY mode
                                )
SELECT accounts.*, my.highest_price , my.total_price
FROM accounts
JOIN cte_table my
ON accounts.mode = my.mode
```

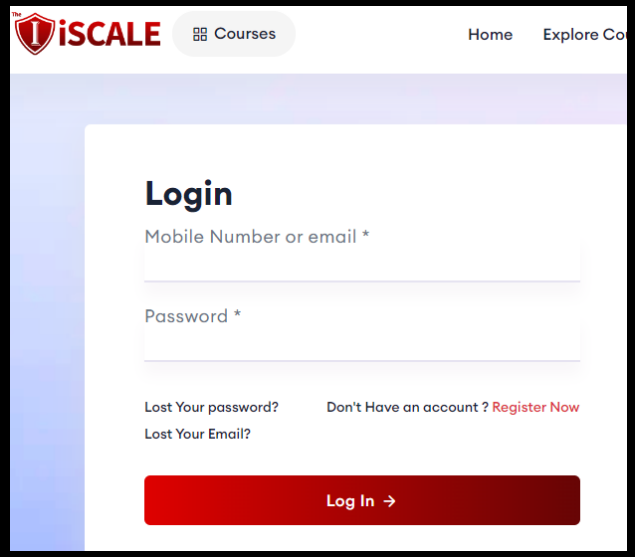
Unit - 19

ASSIGNMENT LEVEL 2 FOR SQL

Step 1 Register Or Login : www.theiscale.com



OR



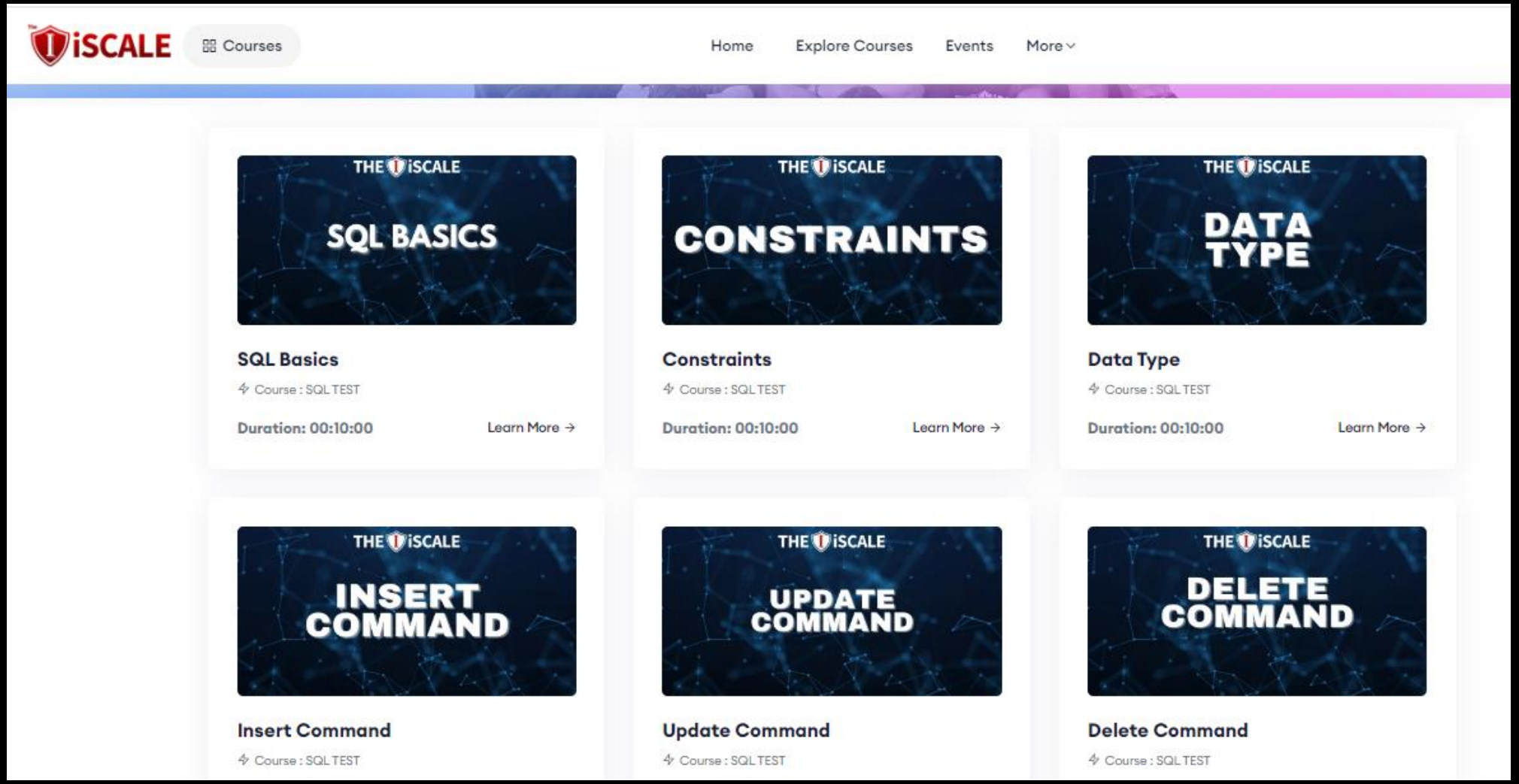
Step 2 Enroll Free Data Analytics Course

Step 3 : Click Latest Practice Test and Download Assignment PDF

Unit - 20

PRACTICE & TEST QUESTIONS PART 2

Step 3 : Click Latest Practice Test and Assignment



SQL Syllabus :

Part 1

Unit – 01 Introduction to SQL-What Is SQL & Database
Unit – 02 Data Types, Primary-Foreign Keys & Constraints
Unit – 03 Create Table In SQL & Create Database
Unit – 04 INSERT UPDATE, DELETE & ALTER Table
Unit – 05 SELECT Statement & WHERE Clause
Unit – 06 How To Import Excel File (CSV) to SQL
Unit – 07 Functions in SQL & String Function
Unit – 08 Aggregate Functions –Types & Syntax
Unit – 09 Assignment Level 1 for SQL
Unit – 10 Practice & Test Questions Part 1

Part 2

Unit – 11 Group By and Having Clause
Unit – 12 Time Stamp and Extract Function, Date Time Function
Unit – 13 SQL JOINS –Types & Syntax
Unit – 14 SELF JOIN, UNION & UNION ALL
Unit – 15 Subquery
Unit – 16 Window Function –Types & Syntax
Unit – 17 Case Statement/Expression with examples
Unit – 18 CTE-Common Table Expression with examples
Unit – 19 Assignment Level 2 for SQL
Unit – 20 Practice & Test Questions Part 2

THE iSCALE

Thankyou

- Website: www.theiscale.com
- YouTube: https://www.youtube.com/@theiScale
- Instagram: https://www.instagram.com/theiscale

@theiscale.founders