**HEXAWARE TRAINING**

**DATA ENGINEERING**

**DAY 1 – 17-07-2025**

**SQL**

**Domain Driven Design:**

We keep names and create tables and other attributes of it by thinking according to the domain.

EX:

Educational Application

           Student, Teacher, Course, Department, Subject

Banking Application

           Account, Customer, Fund Transfer …

Retail Application

           Product, Order, Cart, Payment

EXAMPLES : 🡺 **CRUD OPERATIONS:**

create table students (

student\_id int primary key,

name varchar(50),

course varchar(50),

join\_date date

);

insert into students

values (1,'HARITHA','DATA ENG','2025-07-17'),

(2,'ANBU','DATA SCN','2025-08-17'),

(3,'KAMALI','DATA ENG','2025-08-17');

select \* from students

select name,join\_date from students

**WHERE:**

select \* from students where student\_id = 1

select name from students where join\_date > '2025-06-01'

select \* from students where course ='DATA ENG' and join\_date > '2025-06-01'

select \* from students where course in ('DATA ENG','DATA SCN')

select \* from students where join\_date between '2025-06-01' and '2025-08-17'

**LIKE:**

select \* from students where name like 'h%'

select \* from students where course like '%data%'

select \* from students where name like '%a'

**UPDATE:**

update students set name = 'HARITHA H ' where name = 'HARITHA'

update students set course = 'FULL STACK ' where student\_id = 1

**DELETE:**

delete from students where join\_date in( '2025-08-17','2025-10-17')

delete from students where join\_date = '2025-08-17' or student\_id = 2

**SUBQUERY:**

CREATE DATABASE simple\_sql;

USE simple\_sql;

CREATE TABLE employees (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100),

department VARCHAR(50),

salary INT,

age INT

);

INSERT INTO employees VALUES

(1, 'Amit', 'HR', 30000, 25),

(2, 'Neha', 'IT', 45000, 28),

(3, 'Rahul', 'IT', 50000, 30),

(4, 'Divya', 'Sales', 40000, 26),

(5, 'Kiran', 'Sales', 35000, 24),

(6, 'Meena', 'HR', 32000, 29);

**INLINE QUERY:**

Query inside a query.

select \* from employees

where salary >(

select avg(salary) from employees

)

select dept\_avg.department,dept\_avg.avg\_salary

from (

select department,avg(salary) as avg\_salary

from employees

group by department

) as dept\_avg

select emp\_name,department,salary,

rank() over (order by salary desc) as salary\_rank

from employees

**Joins:**

create database analytics\_practice;

use analytics\_practice;

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(100),

city VARCHAR(50)

);

INSERT INTO customers VALUES

(1, 'Amit Sharma', 'Delhi'),

(2, 'Neha Reddy', 'Hyderabad'),

(3, 'Rahul Iyer', 'Mumbai'),

(4, 'Divya Mehta', 'Chennai');

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

product\_name VARCHAR(100),

order\_amount INT,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

INSERT INTO orders VALUES

(101, 1, 'Laptop', 55000),

(102, 2, 'Mouse', 500),

(103, 1, 'Keyboard', 1500),

(104, 3, 'Monitor', 7000),

(105, 2, 'Printer', 8500);

**INNER JOIN:**

select customers.customer\_name,orders.product\_name,orders.order\_amount

from customers

inner join orders on customers.customer\_id = orders.customer\_id;

**LEFT JOIN :**

select customers.customer\_name, orders.product\_name

from customers

left join orders on orders.customer\_id = customers.customer\_id;

**RIGHT JOIN:**

select orders.product\_name, customers.customer\_name

from customers

right join orders on orders.customer\_id = customers.customer\_id;

**ALIAS:**

select o.order\_id,c.customer\_name,c.city,o.product\_name,o.order\_amount

from orders o

join customers c on c.customer\_id = o.customer\_id;

select c.customer\_name, count(o.order\_id) as total\_orders

from customers c

join orders o on o.customer\_id = c.customer\_id

group by c.customer\_name

having total\_orders = 2;

select c.customer\_name , sum(o.order\_amount) as total

from customers c

join orders o on o.customer\_id = c.customer\_id

group by customer\_name;

select c.customer\_name

from customers c

left join orders o on o.customer\_id = c.customer\_id

where o.order\_id is NULL;