# TASK-1

***-- Database Creation***

CREATE TABLE StudentInfo (

STU\_ID INT PRIMARY KEY,

STU\_NAME VARCHAR(50),

DOB DATE,

PHONE\_NO VARCHAR(15),

EMAIL\_ID VARCHAR(100),

ADDRESS VARCHAR(200)

);

CREATE TABLE CoursesInfo (

COURSE\_ID INT PRIMARY KEY,

COURSE\_NAME VARCHAR(50),

COURSE\_INSTRUCTOR\_NAME VARCHAR(50)

);

CREATE TABLE Enrollmentinfo (

ENROLLMENT\_ID INT PRIMARY KEY,

STU\_ID INT,

COURSE\_ID INT,

ENROLL\_STATUS VARCHAR(20),

FOREIGN KEY (STU\_ID) REFERENCES StudentInfo(STU\_ID),

FOREIGN KEY (COURSE\_ID) REFERENCES CoursesInfo(COURSE\_ID)

);

***-- Data Creation***

*-- Sample data for StudentInfo*

INSERT INTO StudentInfo (STU\_ID, STU\_NAME, DOB, PHONE\_NO, EMAIL\_ID, ADDRESS)

VALUES

(1, 'Miit Sajnani', '1999-12-27', '123-456-7890', 'miit@email.com', 'ABC Chennai road'),

(2, 'Bhawya Puri', '2002-06-25', '987-654-3210', 'Bhawya@email.com', 'DEF Pondycherry road'),

(3, 'Ganesh', '1996-10-25', '980-604-3010', 'Ganesh@email.com', 'PQR Bangalore road');

*-- Sample data for CoursesInfo*

INSERT INTO CoursesInfo (COURSE\_ID, COURSE\_NAME, COURSE\_INSTRUCTOR\_NAME)

VALUES

(101, 'Math 101', 'Prof. Krish'),

(102, 'Science 101', 'Prof. Keerthana');

*-- Sample data for Enrollmentinfo*

INSERT INTO Enrollmentinfo (ENROLLMENT\_ID, STU\_ID, COURSE\_ID, ENROLL\_STATUS)

VALUES

(1, 1, 101, 'Enrolled'),

(2, 1, 102, 'Enrolled'),

(3, 2, 101, 'Enrolled');

**-- Retrieve Student Information**

-- a) Retrieve student details, including name, contact information, and enrollment status

SELECT SI.STU\_NAME, SI.PHONE\_NO, SI.EMAIL\_ID, EI.ENROLL\_STATUS

FROM StudentInfo SI

INNER JOIN Enrollmentinfo EI ON SI.STU\_ID = EI.STU\_ID;

**-- b) Retrieve a list of courses in which a specific student is enrolled**

SELECT CI.COURSE\_NAME

FROM CoursesInfo CI

INNER JOIN Enrollmentinfo EI ON CI.COURSE\_ID = EI.COURSE\_ID

WHERE EI.STU\_ID = 1;

**-- c) Retrieve course information, including course name and instructor information**

SELECT CI.COURSE\_NAME, CI.COURSE\_INSTRUCTOR\_NAME

FROM CoursesInfo CI;

**-- d) Retrieve course information for a specific course**

SELECT CI.COURSE\_NAME, CI.COURSE\_INSTRUCTOR\_NAME

FROM CoursesInfo CI

WHERE CI.COURSE\_ID = 101;

**-- e) Retrieve course information for multiple courses**

SELECT CI.COURSE\_NAME, CI.COURSE\_INSTRUCTOR\_NAME

FROM CoursesInfo CI

WHERE CI.COURSE\_ID IN (101, 102);

**-- f) Test the queries to ensure accurate retrieval of student information**

-- Reporting and Analytics (Using Joining Queries)

**-- a) Retrieve the number of students enrolled in each course**

SELECT CI.COURSE\_NAME, COUNT(EI.STU\_ID) AS ENROLLED\_STUDENTS

FROM CoursesInfo CI

LEFT JOIN Enrollmentinfo EI ON CI.COURSE\_ID = EI.COURSE\_ID

GROUP BY CI.COURSE\_NAME;

-- b) Retrieve the list of students enrolled in a specific course

SELECT SI.STU\_NAME

FROM StudentInfo SI

INNER JOIN Enrollmentinfo EI ON SI.STU\_ID = EI.STU\_ID

WHERE EI.COURSE\_ID = 101;

-- c) Retrieve the count of enrolled students for each instructor

SELECT CI.COURSE\_INSTRUCTOR\_NAME, COUNT(EI.STU\_ID) AS ENROLLED\_STUDENTS

FROM CoursesInfo CI

LEFT JOIN Enrollmentinfo EI ON CI.COURSE\_ID = EI.COURSE\_ID

GROUP BY CI.COURSE\_INSTRUCTOR\_NAME;

-- d) Retrieve the list of students who are enrolled in multiple courses

SELECT SI.STU\_NAME

FROM StudentInfo SI

INNER JOIN Enrollmentinfo EI ON SI.STU\_ID = EI.STU\_ID

GROUP BY SI.STU\_ID

HAVING COUNT(EI.ENROLLMENT\_ID) > 1;

-- e) Retrieve the courses that have the highest number of enrolled students

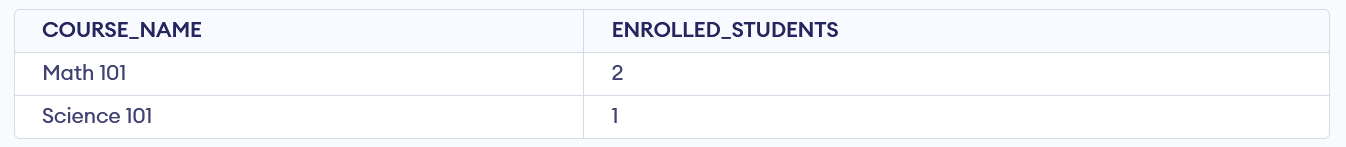
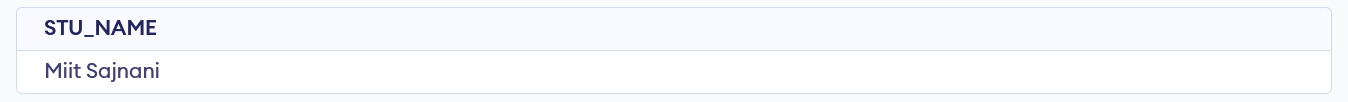
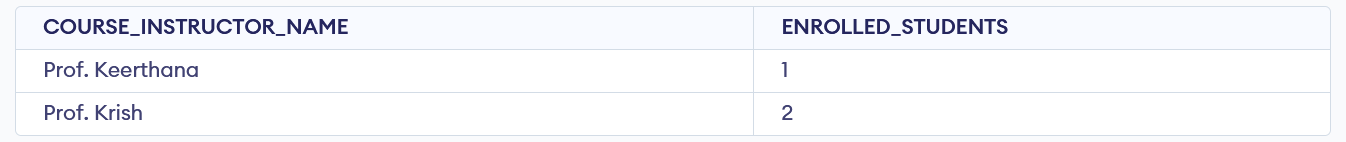
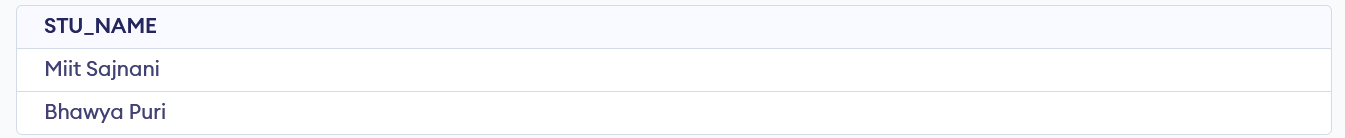
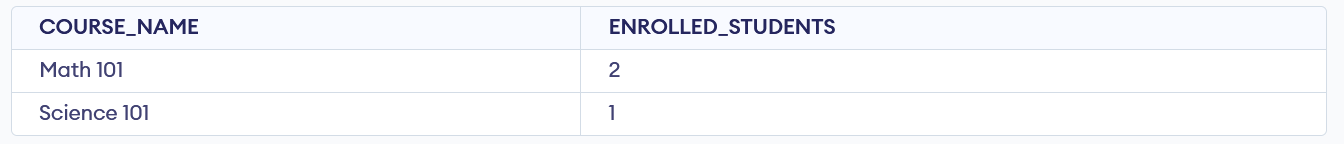
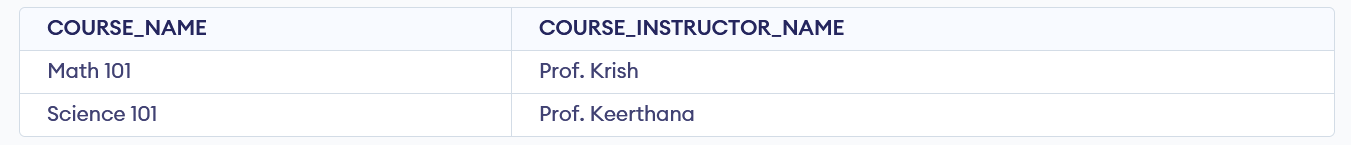
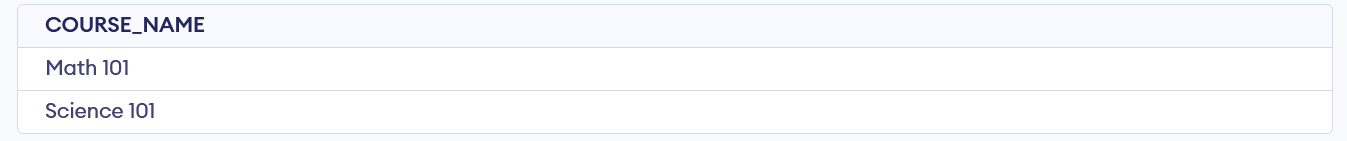
SELECT CI.COURSE\_NAME, COUNT(EI.STU\_ID) AS ENROLLED\_STUDENTS

FROM CoursesInfo CI

LEFT JOIN Enrollmentinfo EI ON CI.COURSE\_ID = EI.COURSE\_ID

GROUP BY CI.COURSE\_NAME

ORDER BY ENROLLED\_STUDENTS DESC;



# Task 2

**-- 1. Database Setup**

CREATE DATABASE student\_database;

\c student\_database;

**-- 2. Create student\_table**

CREATE TABLE student\_table (

Student\_id serial PRIMARY KEY,

Stu\_name text,

Department text,

email\_id text,

Phone\_no numeric,

Address text,

Date\_of\_birth date,

Gender text,

Major text,

GPA numeric,

Grade text

);

**-- 3. Data Entry**

INSERT INTO student\_table (Stu\_name, Department, email\_id, Phone\_no, Address, Date\_of\_birth, Gender, Major, GPA, Grade)

VALUES

('Miit Doe', 'Computer Science', 'Miit@example.com', 1234567890, '123 Main St', '1999-05-15', 'Male', 'Computer Science', 3.8, 'B'),

('Bhawya Smith', 'Mathematics', 'Bhawya@example.com', 9876543210, '456 Elm St', '2000-08-20', 'Female', 'Math', 4.2, 'A'),

('John Johnson', 'Physics', 'John@example.com', 5555555555, '789 Oak St', '1998-03-10', 'Male', 'Physics', 3.5, 'C'),

('Balaji Lee', 'Biology', 'Balaji@example.com', 3333333333, '567 Pine St', '2001-01-05', 'Female', 'Biology', 4.5, 'A'),

('David Brown', 'Chemistry', 'david@example.com', 4444444444, '345 Birch St', '1997-11-25', 'Male', 'Chemistry', 3.9, 'B'),

('Emily White', 'History', 'emily@example.com', 6666666666, '234 Cedar St', '1999-09-15', 'Female', 'History', 4.0, 'B'),

('Keerti Davis', 'Economics', 'Keerti@example.com', 7777777777, '432 Maple St', '2000-07-30', 'Male', 'Economics', 3.7, 'C'),

('Olivia Taylor', 'Psychology', 'olivia@example.com', 9999999999, '543 Walnut St', '1998-12-05', 'Female', 'Psychology', 4.3, 'A'),

('Krish Wilson', 'Political Science', 'krish@example.com', 8888888888, '876 Redwood St', '1996-04-18', 'Male', 'Political Science', 3.4, 'C'),

('Monica Anderson', 'English', 'Monica@example.com', 2222222222, '321 Cedar St', '2000-06-20', 'Female', 'English', 4.1, 'B');

**-- 3. Student Information Retrieval**

SELECT \*

FROM student\_table

ORDER BY Grade DESC;

**-- 4. Query for Male Students**

SELECT \*

FROM student\_table

WHERE Gender = 'Male';

**-- 5. Query for Students with GPA less than 5.0**

SELECT \*

FROM student\_table

WHERE GPA < 5.0;

**-- 6. Update Student Email and Grade**

UPDATE student\_table

SET email\_id = 'newemail@example.com', Grade = 'A'

WHERE Student\_id = 1; -- Update the student with the specific ID

**-- 7. Query for Students with Grade "B"**

SELECT Stu\_name, EXTRACT(YEAR FROM age(Date\_of\_birth)) AS Age

FROM student\_table

WHERE Grade = 'B';

**-- 8. Grouping and Calculation**

SELECT Department, Gender, AVG(GPA) AS Average\_GPA

FROM student\_table

GROUP BY Department, Gender;

**-- 9. Table Renaming**

ALTER TABLE student\_table RENAME TO student\_info;

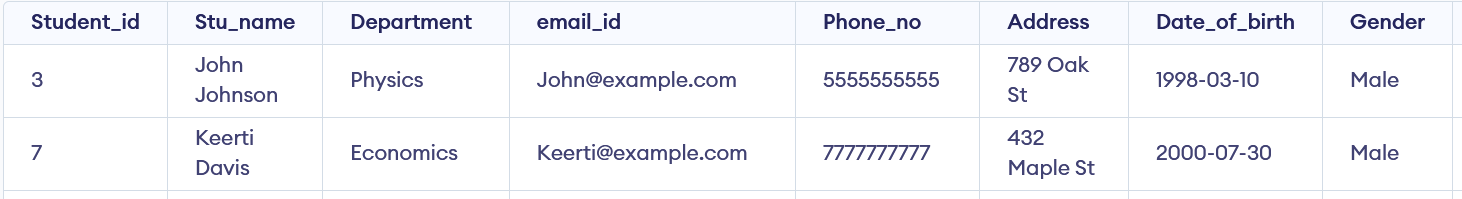
**-- 10. Retrieve Student with Highest GPA**

SELECT Stu\_name

FROM student\_info

ORDER BY GPA DESC

LIMIT 1;

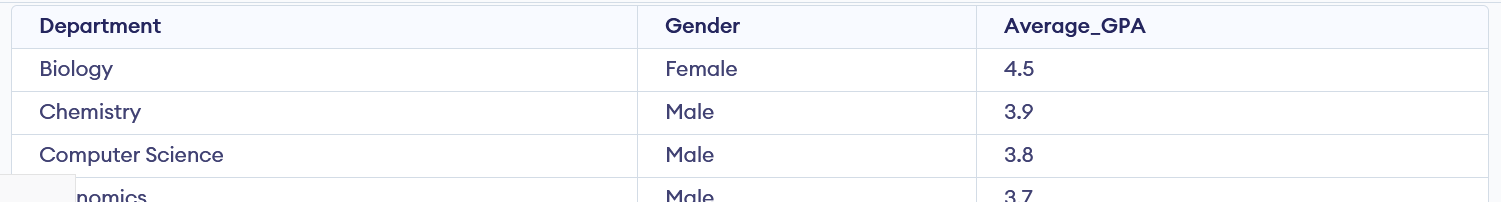
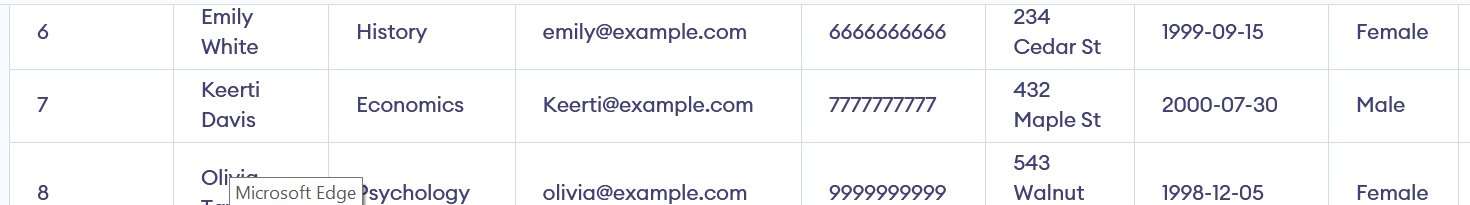


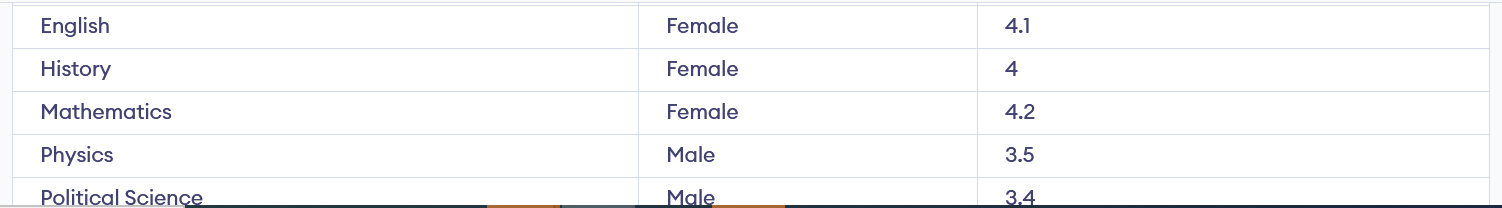


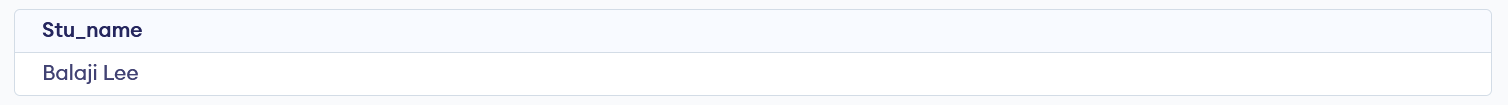












# Task 3

**-- 1. Database Creation**

--CREATE DATABASE EventsManagement;

--\c EventsManagement;

**-- 1. Create Events table**

CREATE TABLE Events (

Event\_Id serial PRIMARY KEY,

Event\_Name text,

Event\_Date date,

Event\_Location text,

Event\_Description text

);

**-- 2. Create Attendees table**

CREATE TABLE Attendees (

Attendee\_Id serial PRIMARY KEY,

Attendee\_Name text,

Attendee\_Phone text,

Attendee\_Email text,

Attendee\_City text

);

**-- 3. Create Registrations table with foreign key constraints**

CREATE TABLE Registrations (

Registration\_Id serial PRIMARY KEY,

Event\_Id INT,

Attendee\_Id INT,

Registration\_Date date,

Registration\_Amount numeric,

FOREIGN KEY (Event\_Id) REFERENCES Events(Event\_Id),

FOREIGN KEY (Attendee\_Id) REFERENCES Attendees(Attendee\_Id)

);

**-- 2. Data Creation**

-- Sample data for Events

INSERT INTO Events (Event\_id,Event\_Name, Event\_Date, Event\_Location, Event\_Description)

VALUES

(1,'Conference', '2023-12-15', 'Convention Center', 'Annual tech conference'),

(2,'Workshop', '2023-10-20', 'Training Center', 'Web development workshop'),

(3,'Seminar', '2023-11-05', 'University Auditorium', 'Marketing trends seminar');

**-- Sample data for Attendees**

INSERT INTO Attendees (Attendee\_id,Attendee\_Name, Attendee\_Phone, Attendee\_Email, Attendee\_City)

VALUES

(1,'John Doe', '123-456-7890', 'john@example.com', 'New York'),

(2,'Jane Smith', '987-654-3210', 'jane@example.com', 'Los Angeles'),

(3,'Mike Johnson', '555-555-5555', 'mike@example.com', 'Chicago');

**-- Sample data for Registrations**

INSERT INTO Registrations (Event\_Id, Attendee\_Id, Registration\_Date, Registration\_Amount)

VALUES

(1, 1, '2023-12-01', 50.00),

(1, 2, '2023-11-25', 50.00),

(2, 1, '2023-10-15', 75.00);

**-- 3. Manage Event Details**

-- a) Inserting a new event

INSERT INTO Events (Event\_Name, Event\_Date, Event\_Location, Event\_Description)

VALUES ('Exhibition', '2024-02-10', 'Art Gallery', 'Art exhibition');

**-- b) Updating an event's information**

UPDATE Events

SET Event\_Location = 'Museum'

WHERE Event\_Id = 4; -- Update the event with a specific ID

**-- c) Deleting an event**

DELETE FROM Events

WHERE Event\_Id = 3; -- Delete the event with a specific ID

**-- 4. Manage Track Attendees & Handle Events**

-- **a) Inserting a new attendee**

INSERT INTO Attendees (Attendee\_Name, Attendee\_Phone, Attendee\_Email, Attendee\_City)

VALUES ('Sarah Lee', '333-333-3333', 'sarah@example.com', 'San Francisco');

-- **b) Registering an attendee for an event**

INSERT INTO Registrations (Event\_Id, Attendee\_Id, Registration\_Date, Registration\_Amount)

VALUES (1, 3, '2023-12-05', 50.00); -- Register Sarah Lee for the conference

**-- 5. Develop queries**

**-- Retrieve event information**

SELECT \* FROM Events;

**-- Generate attendee lists for a specific event**

SELECT A.Attendee\_Name

FROM Attendees A

INNER JOIN Registrations R ON A.Attendee\_Id = R.Attendee\_Id

WHERE R.Event\_Id = 1; -- Replace 1 with the event ID you want to retrieve attendees for

**-- Calculate event attendance statistics (count of attendees for each event)**

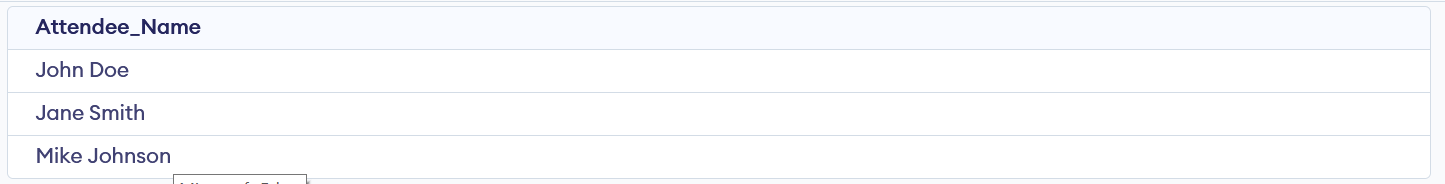
SELECT E.Event\_Name, COUNT(R.Attendee\_Id) AS Attendee\_Count

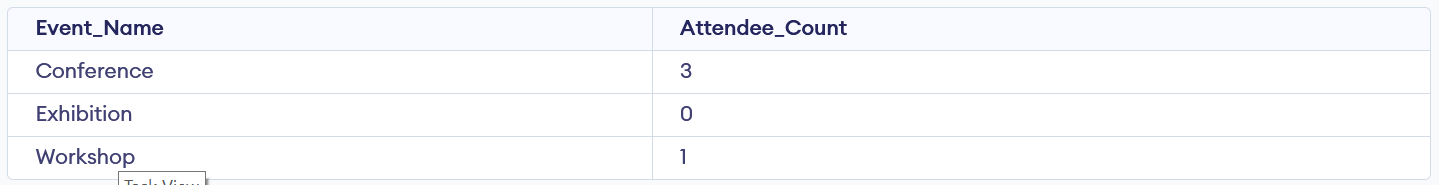
FROM Events E

LEFT JOIN Registrations R ON E.Event\_Id = R.Event\_Id

GROUP BY E.Event\_Name;







# Task 4

-- Database: Sales Data

-- DROP DATABASE IF EXISTS "Sales Data ";

CREATE DATABASE "Sales Data "

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'English\_United States.1252'

LC\_CTYPE = 'English\_United States.1252'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

--1) Database creation

Create table Sales\_sample (Product\_Id Int, Region Varchar(50), On\_date Date,

Sales\_Amount Numeric);

--2) Data Creation

Insert into Sales\_sample (Product\_Id, Region, On\_date, Sales\_Amount) values

('1', 'East', '2023-11-10', '20000'),

('2', 'West', '2023-08-19', '50000'),

('2', 'East', '2023-11-21', '40000'),

('3', 'North', '2023-07-20', '15000'),

('4', 'North', '2023-08-07', '45000'),

('2', 'South', '2023-08-24', '45000'),

('5', 'North', '2023-11-22', '20000'),

('5', 'West', '2023-11-11', '60000'),

('3', 'East', '2023-09-19', '50000'),

('1', 'West', '2023-09-29', '70000')

;

Select \* from Sales\_Sample;

--3) OLAP operations

--a) Drill down

Select Region, Product\_Id, Sum(Sales\_Amount) as Sales\_Amount

From Sales\_Sample

Group By 1,2

Order By Region, Product\_Id, Sales\_Amount;

--b) Roll Up

Select Region, Product\_Id, Sum(Sales\_Amount) as Sales\_Amount

From Sales\_Sample

Group By Rollup (1,2)

Order By Region;

--c) Cube

Select Region, Product\_Id, On\_Date, Sum(Sales\_Amount) as Sales\_Amount

From Sales\_Sample

Group By Cube (1,2,3)

Order By Region, Product\_Id, On\_Date, Sales\_Amount;

--d) Slice

Select Region, Product\_Id, On\_Date, Sum(Sales\_Amount) as Sales\_Amount

From Sales\_Sample

Where Region in('North', 'South') OR On\_Date between To\_date('2023-08-20','YYYY-MM-DD') And To\_Date('2023-10-20','YYYY-MM-DD')

Group By 1,2,3

Order By Region, Product\_Id, On\_Date, Sales\_Amount;

--e) Dice

Select Region, Product\_Id, On\_Date, Sum(Sales\_Amount) as Sales\_Amount

From Sales\_Sample

Where Region in('North', 'South') AND Product\_Id IN (1,2) AND On\_Date between To\_date('2023-08-20','YYYY-MM-DD') And To\_Date('2023-10-20','YYYY-MM-DD')

Group By 1,2,3

Order By Region, Product\_Id, On\_Date, Sales\_Amount;

