DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI

Name: Sandip Kala

Student Code: BWU/BTA/23/173

Course Code: PCC-CSM591

Sl. No	Topic Name	Date of Experiment	Signature with Date	Remarks
1.	Experiment-01	26-07-25		
2.	Experiment-02	2-8-25	28/8/25	
3.	Experiment-03	9-8-25		
4	Experiment-04	9-8-25	23/08/25	
5.	Experiment-05	23-08-25		
6.	Experiment-06	23-08-25	30/08/25	
7.	Experiment-07	30-08-25	4/9/25	
8.	Experiment-08	06-09-25	13/09/25	
9.	Experiment-09	13-09-25	13/09/25	
10.	Experiment-10	18-10-25	18/10/25	
11	Experiment-11	25-10-25	25/10/25	
12	Experiment-12	01-11-25		
13	Experiment-13	08-11-25	08/11/25	

Converting an "E-R" diagram into a relational database table.

Objective: Learn how to convert an "E-R diagram" into a relational database table.

Theory:

i) E-R diagram gives visual representation of entities, attributes and relationships among entities.

ii) Relational table represents entities with column as 'attribute' and rows as 'records'.

Steps:

Step 01: Convert entities into table

rule: Each entity in the E-R diagram is represented as a table in the relational database.

Explanation: The attributes of the entity becomes the columns of the table. The primary key of the entity is used as the primary key of the table.

Step 02: Convert relationships to foreign keys.

rule: Relationship b/w entities are represented by foreign key.

Explanation: The primary key of one entity is included as a foreign key in the table of another entity to establish the relationship.

Step 03: Handle multi-valued attributes

Rule: Multi-valued attributes are managed by creating separate table.

Explanation: A new table is created for each multi-valued attribute with a foreign key reference to the original table and a column for the multi-valued attribute.

Step 04: Convert Weak Entities

Rule: Weak entities are represented using composite primary key

Explanation: The weak entity table includes the primary key of the related strong entity as part of it composite primary key to identify the weak entity.

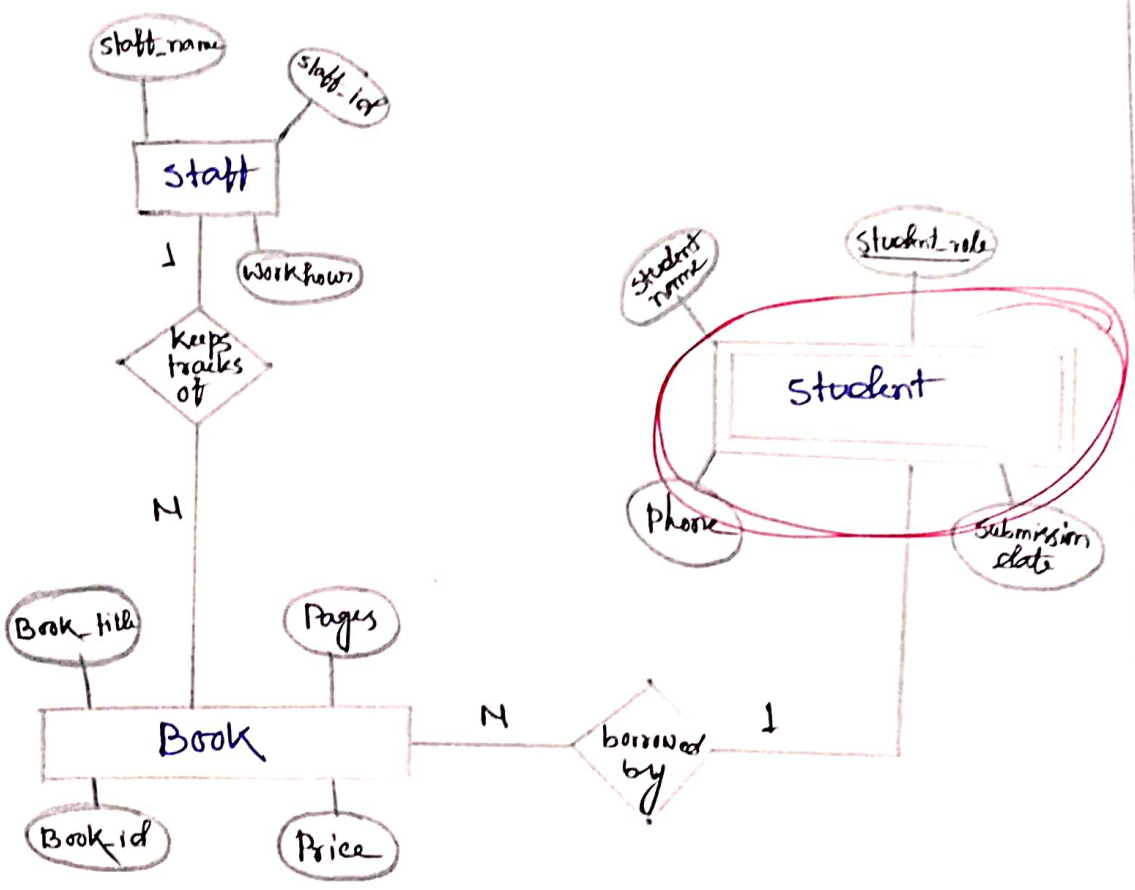
Step 05: Address Generalization and specialization

Rule: It handled by creating separate table for the generalized and specialized entities.

Explanation: A table for the generalize entities is created along with separate tables for each specialized entity with foreign keys link in back to the general entities.

- Draw the "E-R" diagram that involves three main entity named 'Student', 'Book', & 'Staff' and each of them are assigned with some specific attributes. Also mention the cardinality & primary key. The "E-R" diagram is simply showing a simple library-Management system.

Diagram:



Query: Now, the above "E-R diagram" is converted to a relational database table and insert 8 values in each. And also find the two query.

query no. 01>

Display the names of those students whose student_id is 3 and 4.

query no. 02>

Display Book_id and Book_name from the table "Book".

Solution:

code:

```
CREATE TABLE staff(
    staff_name CHAR(20),
    staff_id NUMBER(3) PRIMARY KEY,
    work_hour NUMBER(2)
);

CREATE TABLE Book(
    Book_title CHAR(30),
    Pages NUMBER(4),
    Book_id NUMBER(5) PRIMARY KEY,
    Price NUMBER(4)
);

CREATE TABLE student(
    student_name CHAR(30),
    Phone NUMBER(10),
    student_role NUMBER(2),
    Submission_date DATE
);
```

```
INSERT INTO staff (staff_name, staff_id, work_hour)
        VALUES ('Sanjib Roy', 123, 5);
INSERT INTO staff (staff_name, staff_id, work_hour)
        VALUES ('Raj Jana', 124, 6);
INSERT INTO staff (staff_name, staff_id, work_hour)
        VALUES ('Ram Roy', 125, 7);
INSERT INTO staff (staff_name, staff_id, work_hour)
        VALUES ('Sanjay Das', 126, 5);
INSERT INTO staff (staff_name, staff_id, work_hour)
        VALUES ('Samir Das', 127, 8);
INSERT INTO staff (staff_name, staff_id, work_hour)
        VALUES ('Suresh Jana', 128, 9);
INSERT INTO staff (staff_name, staff_id, work_hour)
        VALUES ('Roma Roy', 130, 7);
INSERT INTO staff (staff_name, staff_id, work_hour)
        VALUES ('Rohit Jana', 129, 6);

SELECT * FROM staff;

INSERT INTO student (student_name, phone, student_role,
 submission_date) VALUES ('Sumita Pattanayak', 9564486753,
 3, '03-AUG-25');

INSERT INTO student (student_name, phone, student_role,
 submission_date) VALUES ('Bidipta chatterjee', 9678321448, 4,
 '02-AUG-25');

INSERT INTO student (student_name, phone, student_role,
 submission_date) VALUES ('Purba Mandal', 8367912459, 1,
 '04-AUG-25');
```

```
INSERT INTO student (student_name, phone, student_role,
Submission_date) VALUES ('Supriya Jana', 8391264301, 2,
'05-AUG-25');
INSERT INTO student (student_name phone, student_role,
Submission_date) VALUES ('Somelita Maity', 7319025095, 6,
'08-AUG-25');
INSERT INTO student (student_name, phone, student_role, submission_
date) VALUES ("Sma Mandal', 7361432509, 7, '05-AUG-25');
INSERT INTO student (student_name, phone, student_role, submission_
date) VALUES ('Somdip Kala', 8391849250, 3, '06-AUG-25');
INSERT INTO student (student_name, phone, student_role, submission_
date) VALUES ('Partha Ghose', 7390164230, 9, '07-AUG-25');
SELECT * FROM student;

INSERT INTO Book (Book_title, Pages, Book_id, Price)
    VALUES ('DEVDAS', 200, 13, 500);
INSERT INTO Book (Book_title, Pages, Book_id, Price) VALUES
('RAJNEETA', 100, 15, 400);
INSERT INTO Book (Book_title, Pages, Book_id, Price) VALUES
('PATHER PANCHALI', 150, 14, 600);
INSERT INTO Book (Book_title, Pages, Book_id, Price) VALUES
('GORA', 160, 18, 300);
INSERT INTO Book (Book_title, Pages, Book_id, Price) VALUES
('GITANGALI', 200, 20, 400);
INSERT INTO Book (Book_title, Pages, Book_id, Price) VALUES
('KABULIWALA', 250, 19, 380);
```

```
INSERT INTO Book (Book_title, Pages, Book_id, Price) VALUES
('CHOKHER BALI', 280, 22, 450);
INSERT INTO Book (Book_title, Pages, Book_id, Price) VALUES
('MACBETH', 300, 25, 390);
SELECT * FROM Book;
SELECT * FROM student where student_id = 3 or student_id = 4;
SELECT Book_id, Book_name FROM Book;
```

Output:

Book:

Book_title	Pages	Book_id	Price
DEVDAS	200	13	500
PARINEETA	100	15	400
PATHERPANCHALI	150	14	600
GORA	160	18	300
GITANJALI	200	20	400
KABULIWALA	280	19	350
CHOKHER BALI	280	22	450
MACBETH	300	25	390

Student:

Student_name	Phone	Student_role	submission_date
Sumita Pattanayak	9564486950	3	03-AUG-25
Bidipta chatterjee	9678321448	4	02-AUG-25
Purba Mandal	8367912459	1	04-AUG-25
Supriya Jana	8391264301	2	05-AUG-25
Smchita Maity	731902509	6	08-AUG-25
Sma Mandal	8361432509	7	05-AUG-25
Sandip Kala	8391849250	8	06-AUG-25
Partha Ghosh	7390164230	9	07-AUG-25

Staff:

Staff_name	staff_id	work_hour
Sanjib Roy	123	5
Raj Jana	124	6
Ram Roy	125	7
Sanjay Das	126	5
Samir Das	127	8
Suresh Jana	128	9
Roma Roy	130	7
Rohit Jana	129	6

Student:

Student_name	phone	student_role	submission_date
Sumita Pattanayak	9564486950	3	03-AUG-25
Bidipta Chatterjee	7678321448	4	02-AUG-25

Book

Book_title	Book_id
DEVDAS	13
PARINEETA	15
PATHER PANCHALI	14
GORA	18
GITANJALI	20
KABULIWALA	19
CHOKHER BALI	22
MACBETH	25

Experiment - 02

Creating a database and tables and specifies datatypes:

Theory: A database is a structured collection of data that can be accessed and managed efficiently. In SQL, a database contains table, each define by rows (records) and columns (fields), where each column has a specific datatype that determines the kind of data it can hold. Common datatypes include →

i) **Integer:** For storing whole numbers.

ii) **VARCHAR:** Variable length string used to store text data.

iii) **DATE:** For storing dates in standard format (YYYY-MM-DD)

Tables must have also primary key (used as a identifier for each row) and foreign keys (to establish relationships between table). Defining proper datatypes ensures data integrity and efficient database operations.

Procedure:

1. Create a new database
2. Use the database
3. Define and create tables with appropiate datatypes.

CODE:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;
CREATE TABLE students(
StudentID INT AUTO_INCREMENT PRIMARY KEY,
FirstName VARCHAR(50);
LastName VARCHAR(50),
BirthDate DATE,
EnrollmentDate DATE);

CREATE TABLE Course(
CourseID INT AUTO_INCREMENT PRIMARY KEY,
CourseName VARCHAR(100),
Credits INT);

CREATE TABLE Enrollments(
EnrollmentID INT AUTO_INCREMENT PRIMARY KEY,
studentID INT,
CourseID INT,
FOREIGN KEY(StudentID) REFERENCES students
(studentID),
FOREIGN KEY(CourseID) REFERENCES Course(courseID));

DESC students;
DESC Course;
DESC Enrollments;
```

Output:

Field	Type	Null	Key	Default	Extra
StudentID	int(11)	NO	PRI	NULL	auto increment
FirstName	varchar(50)	YES		NULL	
LastName	varchar(50)	YES		NULL	
BirthDate	date	YES		NULL	
EnrollmentDate	date	YES		NULL	

Field	Type	Null	Key	Default	Extra
CourseID	int(11)	NO	PRI	NULL	auto increment
CourseName	varchar(100)	YES		NULL	
Credits	int(11)	YES		NULL	

Field	Type	NULL	Key	Default	Extra
EnrollmentID	int(11)	NO	PRI	NULL	auto increment
StudentID	int(11)	YES	MUL	NULL	
CourseID	int(11)	YES	MUL	NULL	

Experiment-3

Specify constraint and creating indices:

Objective: Learn how to specify constraints (e.g: Primary key, foreign key, Unique, Not null) and create indices for tables.

Theory: Constraints are rules applied two tables columns to enforce data integrity and reliability within a database. common constraints are include —

i> **Primary key:** Ensure that each record in a table is unique and not null.

ii> **Foreign key:** Establishes a link between two tables ensuring refferential integrity.

iii> **Unique:** Ensure that all values in a column are different

iv> **Not Null:** Prevents null values from being inserted into a column.

v> **chuk:** Ensure that all the values are satisfy specific condition.

Indices improve the speed of data retrieval operation of a table at the cost of additional storage. Indexes can be created one or more columns to make searching faster.

Procedure:

1. Add constraints to table (Primary key, Foreign key, Unique, not null)
2. Create Indices on specific columns for fast retrieval.
3. Verify the constraints and indices by inspecting the schema.

Query:

ALTER TABLE add constraint uc_studentname
unique(FirstName, LastName);

StudentID in the enrollments1 table as unique.
Coursename in the courses1 table as ~~unique~~ not null.

Code:

```
CREATE TABLE students (
    Student_id.INT PRIMARY KEY AUTO_INCREAMENT,
    First_name VARCHAR(50),
    Last_name VARCHAR (50),
    Date of Birth DATE,
);
CREATE TABLE Courses(
    Course_id INT PRIMARY KEY AUTO_INCREAMENT,
    Course_name VARCHAR(50),
    Credits INT
);
CREATE TABLE Enrollment(
    Enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    Course_id INT,
    FOREIGN KEY(student_id) REFERENCES students
                                (student_id),
    FOREIGN KEY (Course_id) REFERENCES Courses
                                (courses_id),
);
--Make First_name and Last_name combinition unique
```

```
ALTER TABLE students
ADD CONSTRAINT unique_student_name UNIQUE (First_
                                    name, Last_name);

--Make student_id in Enrollments unique
ALTER TABLE Enrollments
ADD CONSTRAINT unique_student_in_enrollment UNIQUE
                                    (student_id);
--Make course_name in Courses NOT NULL
ALTER TABLE Courses;
MODIFY Course_name VARCHAR(100) NOTNULL;

DESC students;
DESC Courses;
DESC Enrollments;
```

Output:

Field	Type	NULL	Key	Default	Extra
Student_id	int	NO	PRI	NULL	auto increament
First_name	varchar(50)	YES	MUL	NULL	
Last_name	varchar(50)	YES		NULL	
Date of Birth	date	YES		NULL	
Enrollment_date	date	YES		NULL	

Field	Type	NULL	Key	Default	Extra
course_id	int	NO	PRI	NULL	auto increament
course_name	varchar(100)	NO		NULL	
Credits	int	YES		NULL	

Field	Type	NULL	Key	Default	Extra
Enrollment_id	int	NO	PRI	NULL	auto increament
student_id	int	YES	UNI	NULL	
course_id	int	YES	MUL	NULL	

Assignment-04

Problem statement: Table & Record Handling

Objective: Learn how to use the insert statement to add records to a table.

Theory: In relational databases, the insert statement is used to add new rows of data into a table. It allows the user to specify values for each column ensuring data integrity by checking constraints like primary key, foreign key etc is crucial during the insertion process. Records are the actual data stored in the tables.

Procedure:

1. Insert records into the previously created table (students, courses, Enrollments).
2. Ensures the integrity of the inserted records by checking the data.
3. View the data to confirm that the records were successfully inserted.

Example:

```
INSERT into students (First_name, Last_name, DOB, Enrollment_
date) VALUES ('John', 'Doe', '2000-20-5', '01-09-2023');
```

Some specific Commands: (Insert & Fetch data)

1. All rows & columns => Select * From Table_name
2. Selet rows & all columns => select From Tablename where {condition}
3. All rows & selected columns => selet col1, col2, & col3 From Tablename.
4. Selected columns & selected rows => select col1, col2 from Tablename where {condition}

Problem statement: Create a database "University" and insert 8 values in each table then perform some quiries →

1. Selet student_id, First_name, Last_name of all student.
2. Display the courses whose credit is greater than or equal to 3.
3. Display the students who has enrolled in the year 2024 or 2025.
4.

Solutions:

code:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;
CREATE TABLE students (
    student_id INT PRIMARY KEY AUTO_INCREAMENT,
    First_name VARCHAR (50),
    Last_name VARCHAR (50),
    DateofBirth DATE,
    Enrollment_date DATE.
);
CREATE TABLE Courses (
    Course_id INT PRIMARY KEY AUTO_INCREMENT,
    Course_name VARCHAR (100),
    Credits INT
);
CREATE TABLE Enrollments (
    Enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    course_id INT,
    FOREIGN KEY (student_id) REFERENCES students (student_id)
```

```
FOREIGN KEY(course_id) REFERENCES Courses (course_id),
INSERT INTO students (First_name, Last_name, Date of Birth,
Enrollment_date) VALUES
('Sumita', 'Pattanayak', '2005-01-19', '2023-08-16'),
('Purba', 'Mondal', '2005-07-25', '2023-07-20'),
('Bikas', 'Kotal', '2004-08-16', '2022-06-14'),
('Sandip', 'Kala', '2005-02-13', '2022-08-15'),
('Sanchita', 'Maity', '2005-07-11', '2023-05-18'),
('Sanket', 'Maity', '2004-12-15', '2022-08-01'),
('Partha', 'Ghosh', '2006-08-09', '2023-05-18'),
('Deeptanshu', 'Paul', '2005-08-24', '2023-04-28');

INSERT INTO Courses (Course_name, Credits) VALUES
('Database Management System', 4),
('Operating system', 4),
('Computer Organization & Architecture', 4),
('Artificial Intelligence', 3),
('Machine Learning', 3),
('Cloud computing', 2),
('Web Development', 2),
('Computer Networks', 4);

INSERT INTO Enrollments (students_id, Course_id) VALUES
(1, 4),
(2, 3),
(3, 6),
(4, 8),
```

```
(5,1),
(6,2),
(7,7),
(8,5);
SELECT *FROM Students;
SELECT * FROM Courses;
SELECT * FROM Enrollments;
-- Selet students_id, First_name, Last_name of all students.
SELECT student_id, First_name, Last_name FROM Students;
-- Display the courses whose credits >= 3
SELECT Course_id, Course_name FROM Course WHERE
Credits >= 3;
--Display the students who enrollment in the year 2023 or 2024
SELECT student_id, First_name, Last_name, Enrollment_date
FROM students
WHERE YEAR(Enrollment_date) IN (2023, 2024);
```

Output:

1. 'Students' TABLE

Student_id	First_name	Last_name	Date of Birth	Enrollment_date
1.	Sumita	Pattanayak	2005-01-19	2023-08-16
2.	Purba	Mondal	2005-07-25	2023-07-20
3.	Bikas	Kotal	2004-08-16	2022-06-14
4.	Sandip	Kala	2005-02-13	2022-08-22
5.	Sanchita	Maity	2005-07-11	2023-05-18
6.	Sanket	Maity	2004-12-15	2022-08-01
7.	Partha	Ghosh	2006-08-09	2023-05-18
8.	Deeptanshu	Paul	2005-08-24	2023-04-28

2. 'Courses' Table.

Course_id	Course_name	credits
1	Database Management system	4
2	Operating system	4
3	Computer Organisation & Architecture	4
4	Artificial Intelligence	3
5	Machine Learning	3
6	cloud Computing	2
7	Web Development	2
8	Computer Networks	4
9		

3. 'Enrollments' Table.

Enrollment_id	Student_id	Course_id
1	1	4
2	2	3
3	3	6
4	4	8
5	5	1
6	6	2
7	7	7
8	8	5

4. Query 1

Student_id	First_name	Last_name
1	Sumita	Pattanayak
2	Puriba	Mondal
3	Bikas	Kotal
4	Sandip	Kala
5	Sanchita	Maity
6	Sanket	Maity
7	Partha	Ghosh
8	Deeptanshu	Paul

5. Query 2

Course_id	Course_name	Credits
1	Database Management System	4
2	Operating Systems	4
3	Computer Organisation & Architecture	4
4	Artificial Intelligence	3
5	Machine Learning	3
8	Computer Networks	4

6. Query 3

Student_id	First_name	Last_name	Enrollment_date
1	Sumita	Pattanayak	2023-08-16
2	Puriba	Mondal	2023-07-20
5	Sanchita	Maity	2023-05-18
7	Partha	Ghosh	2023-05-18
8	Deeptanshu	Paul	2023-04-28

Experiment-5

Problem statement: Quering data using select and delete

Objective: Learn how to retrieve data using select and remove data using delete statement.

Theory: The select statement is most commonly used in SQL commands to query and retrieve data from tables. The delete statement is used to remove the records from a table. Both these commands are fundamental to manipulating the data stored in a Database

Procedure:

1> Use the select statement to retrieve specific data from the tables.

2> Use the delete statement to remove records from the table.

3> To varify the result by viewing the modified tables.

Query:

1> Select all records from the Student table

2> Select specific fields from the Courses tables.

3> Delete records from Enrollment table where studentID is 1.

Solution:

Code:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;
CREATE TABLE students(
StudentID INT AUTO_INCREMENT PRIMARY KEY,
FirstName VARCHAR(50),
LastName VARCHAR(50),
BirthDate DATE DATE,
EnrollmentDate DATE);

CREATE TABLE Course(
CourseID INT AUTO-INCREMENT PRIMARY KEY,
CourseName VARCHAR(100),
Credits INT);

CREATE TABLE Enrollments(
EnrollmentID INT AUTO-INCREMENT PRIMARY KEY,
StudentID INT,
CourseID INT,
FOREIGN KEY(StudentID) REFERENCES students(StudentID)
FOREIGN KEY(CourseID) REFRENCES Course(CourseID));

INSERT INTO Course(CourseName, Credits) VALUES
('Artificial Intelligence', 4),
('Operating System', 3),
('Web Development', 3);
```

```
INSERT INTO students (FirstName, LastName, BirthDate,
    EnrollmentDate) VALUES
('Sandip', 'Kala', '2005-02-13', '2023-08-16'),
('Sumita', 'Pattanayak', '2005-01-19', '2023-08-17'),
('Partha', 'Ghosh', '2005-11-12', '2023-09-10');

INSERT INTO Enrollments (StudentID, CourseID) VALUES
(1,1),
(2,3),
(3,2);

SELECT * FROM students;
SELECT CourseName FROM Course;
DELETE FROM Enrollments
WHERE StudentID = 1;
SELECT * FROM Enrollments
~~WHERE StudentID~~
```

output:

1. students Table

StudentID	FirstName	LastName	BirthDate	EnrollmentDate
1	Sandip	Kala	2005-02-13	2023-08-16
2	Sumita	Pattanayak	2005-01-19	2023-08-17
3	Partha	Ghosh	2005-11-12	2023-09-10

2> Course Table

CourseName
Artificial Intelligence
Operating systems
Web Development

3> Enrollment Table

EnrollmentID	StudentID	CourseID
2	2	3
3	3	2

23/08/25

Experiment-06

Problem statement: Using update, truncate, drop and alter statement

Objective: Learn how to update, truncate, drop and alter tables in DB.

Theory:

- **Update:** This command is used to modify existing records in a table.
- **Truncate:** It removes all records from a table but it keeps the table structure.
- **Drop:** This command is used to completely remove a table from the database.
- **Alter:** It is used to modify the structure of a table (Add or remove columns).

Procedure:

1. Use the Update statement to modified data in the table.
2. Use the Truncate statement to clear all records from the table.
3. Use the Drop statement to delete a table.
4. Use the Alter statement to modify the structure of the table.

Query:

1. Update the course's table SET credits = 5 where CourseName = Database system.
2. Truncate the table Enrollment
3. Alter the students table to add a new column Email.

4. Update the Course table SET CourseName = RDBMS where CourseName = Database system.

5. Alter the students table change the size of the field StudentName to 80.

Solution:

code:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;

CREATE TABLE students(
StudentID INT PRIMARY KEY AUTO_INCREMET,
StudentName VARCHAR(50),
DateofBirth DATE,
Enrollment_date DATE
);

CREATE TABLE Course(
CourseID INT PRIMARY KEY AUTO_INCREMENT,
CourseName VARCHAR(100),
Credits INT
);

CREATE TABLE Enrollments(
EnrollmentID INT PRIMARY KEY AUTO_INCREMENT,
StudentID INT,
CourseID INT,
FOREIGN KEY (StudentID) REFERENCES students(StudentID),
FOREIGN KEY (CourseID) REFERENCES Course(CourseID));
```

```
INSERT INTO Students (studentName, Date_of_Birth, Enrollment_date)
VALUES
('Sandip', '2005-02-13', '2025-08-16'),
('Sumita', '2005-01-19', '2025-07-17'),
('Partha', '2005-02-10', '2024-08-18'),
('Sanchita', '2005-07-11', '2023-07-10'),
('Sanket', '2004-12-15', '2025-06-20'),
('Sana', '2005-03-18', '2024-09-01'),
('Bikas', '2005-07-30', '2024-07-01');

INSERT INTO Course (courseName, Credits) VALUES
('Artificial Intelligence', 4),
('Database Systems', 3),
('Web Development', 4),
('Data Structure and Algorithms', 3),
('Operating Systems', 3),
('Computer Networks', 3),
('Machine Learning', 4);

INSERT INTO Enrollments (studentID, CourseID) VALUES
(1, 4),
(2, 3),
(3, 6),
(4, 2),
(5, 1),
(6, 5),
(7, 7);
```

```
SELECT * FROM Students;
SELECT * FROM Course;
SELECT * Enrollments;
-- Update the Course table set Credits = 5 where CourseName
  = Database Systems.
 Update Course
 SET credits = 5
 WHERE courseName = 'Database systems';
-- Truncate the table Enrollments
 TRUNCATE TABLE Enrollments;
-- Alter the students table to add a new column Email.
  ALTER TABLE students.
  ADD Email VARCHAR (100);
-- Update the Course table set CourseName = RDMS Where
  CourseName = Database Systems.
  Update Course
  SET CourseName = 'RDMS'
  WHERE courseName = 'Database systems';
-- Alter the students table change the size of the field
   studentName to 80
  ALTER TABLE students
  MODIFY studentName VARCHAR(80);
```

Output:

1. Students table

StudentID	StudentName	DateofBirth	Enrollment_date
1	Smolip	2005-02-13	2025-08-16
2	Sumita	2005-01-19	2025-07-17
3	Partha	2005-02-10	2024-08-18
4	Samchita	2005-07-11	2023-07-10
5	Samket	2004-12-15	2025-06-20
6	Sma	2005-03-18	2024-09-01
7	Bikas	2005-07-30	2024-07-01

2. Course table

CourseID	CourseName	Credits
1	Artificial Intelligence	4
2	Database Systems	3
3	Web Development	4
4	Data structure and Algorithms	3
5	Operating Systems	3
6	Computer Networks	3
7	Machine Learning	4

3. Enrollments table

StudentID	CourseID
5	1
4	2
2	3
1	4
6	5
3	6
7	7

4) Query 1

Course Table

CourseID	CourseName	Credits
1	Artificial Intelligence	4
2	Database Systems	5
3	Web Development	4
4	Data Structure and Algorithms	3
5	Operating Systems	3
6	Computer Networks	3
7	Machine Learning	4

5) Query 2

Enrollments Table

StudentID	CourseID

6) Query 3

Students Table

StudentID	StudentName	DateofBirth	Enrollment-date	Email
1	Sandip	2005-02-13	2025-08-16	NULL
2	Sumita	2005-01-19	2025-07-17	NULL
3	Partha	2005-02-10	2024-08-18	NULL
4	Sanchita	2005-07-11	2023-07-10	NULL
5	Sanket	2004-12-15	2025-06-20	NULL
6	Sana	2005-03-18	2024-09-01	NULL
7	Bikas	2005-07-30	2024-07-01	NULL

7> Query 4

Course Table

CourseID	CourseName	Credits
1	Artificial Intelligence	4
2	RDMS	5
3	Web Development	4
4	Data Structure and Algorithms	3
5	Operating Systems	3
6	Computer Networks	3
7	Machine Learning	4

8> Query 5

Students Table

StudentID	StudentName	DateofBirth	Enrollment_date	Email
1	Somdip	2005-02-13	2025-08-16	NULL
2	Sumita	2005-01-19	2025-07-17	NULL
3	Partha	2005-02-10	2024-08-18	NULL
4	Sanchita	2005-07-11	2023-07-10	NULL
5	Sanket	2004-12-15	2025-06-20	NULL
6	Soma	2005-03-18	2024-09-01	NULL
7	Bikas	2005-07-30	2024-07-01	NULL

Experiment-07

Problem statement:

Objective: Learn how to retrieve data using the select statement with WHER clause.

Theory: The WHERE clause in SQL is used to filter records based on specific conditions. It is commonly use with the SELECT statement to retrieve only those records that meet the define criteria.

Procedure:

1) Use the SELECT statement with where clause to retrieve specific records.
2) Perform various queries using different conditions in the WHERE clause.

Query:

1) Select the students whose first name is "John".
2) Display the studentName and studentID who enroll after 20
3) Display the studentName and studentID who taken the subject DSA or Database systems.
4) Display the studentName, studentID and EmailID of those students whose name start with letter 's'.
5) Display the studentName and studentID who didn't enroll for the course image processing or Robotics, cyber security.

6) Display the StudentName, studentID and Email of the students in descending order of their ID.

7) Display the studentName, studentID of students who have enrolleighter in 2023 or 2024

Solution:

Code:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;

CREAT TABLE Students (
    StudentId INT PRIMARY KEY AUTO_INCREMENT,
    StudentName VARCHAR (50),
    Date_of_Birth DATE,
    Enrollment_date DATE,
    EmailId VARCHAR (100)
);

CREATE TABLE Course (
    courseid INT PRIMARY KEY AUTO_INCREMENT,
    courseName VARCHAR (100),
    Credits INT
);

CREATE TABLE Enrollments (
    StudentId INT,
    courseid INT,
    PRIMARY KEY (studentid, courseid),
    FOREIGN KEY (studentid) REFERENCES Students (studentid),
    FOREIGN KEY (courseid) REFERENCES Course (courseid)
);
```

```
INSERT INTO Students (studentNameid, studentName, Date_of_Birth,
Enrollment_date, EmailId) VALUES
(1, 'Sandip', '2005-05-14', '2025-08-16', 'sandip@example.com'),
(2, 'Sumita', 2005-01-19', '2025-07-17', 'sumita@example.com'),
(3, 'Partha', '2005-02-10', '2024-08-18', 'partha@example.com'),
(4, 'Sanchita', '2005-07-11', '2023-07-10', 'sanchita@example.com'),
(5, 'Sanket', '2004-12-15', '2025-06-20', 'sanket@example.com'),
(6, 'Sana', '2005-03-18', '2024-09-01', 'sana@example.com'),
(7, 'Bikash', '2005-07-30', '2024-07-01', 'bikas@example.com'),
(8, 'John', '2005-07-30', '2024-07-01', 'John@example.com');

INSERT INTO Course (courseName, Credits) VALUES
('Artificial Intelligence', 4),
('Database systems', 3),
('Web Development', 4),
('Data structures and Algorithms', 3),
('Cyber Security', 3),
('Image Processing', 3),
('Machine Learning', 4),
('Robotics', 5);

INSERT INTO Enrollments (studentid, courseid) VALUES
(1,4),
(2,3),
(3,6),
(4,2),
```

```
(5,1),
(6,5),
(7,7),
(8,1);

-- Select the students whose first name is "John".
SELECT * FROM students WHERE studentName = 'John';

-- Display the student name and ID who enroll after 2024
SELECT studentid, studentName
FROM students
WHERE Enrollment_date > '2024-12-31';

-- Display the student name, studentid and Email id of those
students whose name start with letter 's'.
SELECT studentid, studentName, EmailId
FROM students
WHERE studentName LIKE 'S%';

-- Display the student name, id, and Email id of the students
in descending order of their ID.
SELECT studentid, studentName, EmailId
FROM students
ORDER BY studentid DESC;

-- Display the student name, id of students who have enroll in
2023 or 2024.
SELECT studentid, studentName FROM students
WHERE YEAR (Enrollment_date) IN (2023, 2024);
```

Output

1) Query-1

Students Table

StudentId	StudentName	Date_of_Birth	Enrollment_date	EmailId
8	John	2005-07-30	2024-07-01	john@exampl.com
NULL	NULL	NULL	NULL	NULL

2) Query-2

Students Table

StudentId	StudentName
1	Sandip
2	Sumita
5	Sanket
NULL	NULL

3) Query-4

Students Table

StudentId	StudentName	EmailId
1	Sandip	sandip@example.com
2	Sumita	sumita@exampl.com
4	Sanchita	sanchita@example.com
5	sanket	Sanketa@example.com
6	Sana	Sana@example.com
NULL	NULL	NULL

4> Query - 6

Students Table

StudentId	StudentName	EmailId
8	John	John@example.com
7	Bikas	bikas@example
6	Sarna	sarna@example.com
5	Sanket	sanket@example.com
4	Sanchita	sanchita@example.com
3	Partha	partha@example.com
2	Sumita	sumita@example.com
1	Sandip	sandip@example.com

5> Query - 7

Students Table

StudentId	StudentName
3	Partha
4	Sanchita
6	Sarna
7	Bikas
8	John

Experiment-08

Problem statement: Retrieving data using IN, BETWEEN, LIKE

Objective: Learn how to retrieve data using IN, BETWEEN, LIKE and other filtering condition in SQL.

Theory:

IN: Allows you to specify multiple values in a WHERE clause.

BETWEEN: Select values within a given range.

LIKE: Used for pattern matching with wild cards.
For example, persons for any number of characters.

Query:

1. Retrieve those students whose studentid is either 2, 5 or 7
2. Retrieve those students whose enrollment date in between 1st august 2023 and 30th august 2023
3. Retrieve those students whose name start with 's'.
4. Retrieve details of those students whose studentid is not 1, 4 or 9.
5. Display the students in Group who have enroll either in Data structure and Algorithm and Database systems.

Solution:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;

CREATE TABLE Students (
 student_id INT PRIMARY KEY AUTO_INCREMENT,
 StudentName VARCHAR(100),
 date_of_birth DATE,
 Email VARCHAR(100),
  enrollment_date DATE
);

 CREATE TABLE Courses (
 course_id INT PRIMARY KEY AUTO_INCREMENT,
 Course_name VARCHAR(100),
  course_code VARCHAR(20),
  Credits INT
 );

 CREATE TABLE Enrollment(
        student_id INT,
         course_id INT,
         FOREIGN KEY (student_id) REFERENCES (student_id),
         FOREIGN KEY (course_id) REFERENCES (course_id)
        );

 INSERT INTO Students (student_id, StudentName, date_of_birth,
  Enrollment_date, Email) VALUES
 (1,'Sandip','2005-02-13','2025-08-16','sandip@example.com'),
 (2,'Sumita','2005-01-19','2023-08-17','sumita@example.com'),
 (3,'Partha','2005-02-10','2024-08-18','partha@example.com'),
```

(4, 'Smchita', '2005-07-11', '2023-08-10', 'smchita@example.com'),
(5, 'Smket', '2004-12-15', '2025-06-20', 'smket@example.com'),
(6, 'Sma', '2005-03-18', '2024-09-01', 'sma@example.com'),
(7, 'Bikas', '2005-07-30', '2024-07-01', 'bikas@example.com'),
(8, 'John', '2005-07-26', '2024-07-13', 'john@example.com'),
(9, 'Purba', '2005-05-25', '2023-08-22', 'purba@example.com');

```
INSERT INTO Course (Course_name, credits) VALUES
('Artificial Intelligence', 2),
('Database systems', 3),
('Web Development', 4),
('Data structures and Algorithms', 3),
('Cyber Security', 3),
('Image Processing', 3),
('Machine Learning', 4),
('Robotics', 5),
('Data Mining', 4);
```

```
INSERT INTO Enrollments (student_id, course_id) VALUES
(1,4),
(2,3),
(3,6),
(4,2),
(5,8),
(6,5),
(7,7),
(8,1),
(9,9);
```

- Retrieve those students whose student-id is eigther 2, 5 or 7.

```
SELECT student-id, studentName, date-of-birth, enrollment-date, Email
FROM students
WHERE student-id IN (2,5,7);
```

-- Retrieve those students whose enrollment-data in between 1st august and 30th august 2023

```
SELECT student-id, studentName, date-of-birth, enrollment-date, Email
FROM students
WHERE enrollment-date BETWEEN '2023-08-01' AND '2023-08-30';
```

-- Retrieve those students whose name start with 's'.

```
SELECT student-id, studentName, date-of-birth, enrollment-date, Email
FROM students
WHERE studentName LIKE 'S%';
```

-- Retrieve details of those students whose student-id is not 1, 4 or 9

```
SELECT student-id, studentName, date-of-birth, enrollment-date, Email
FROM students
WHERE student-id NOT IN (1,4,9);
```

-- Display the students in Group who have enroll eigther in Data structure algorithm and Database systems.

Output:

1> Query-1

Student_id	StudentName	date_of_birth	enrollment_date	Email
2	Sumita	2005-01-19	2023-08-17	sumita@example.com
5	Sanket	2004-12-15	2025-06-20	sanket@example.com
7	Bikas	2005-07-30	2024-07-01	bikas@example.com

2> Query-2

Student_id	StudentName	date_of_birth	enrollment_date	Email
2	Sumita	2005-01-19	2023-08-17	sumita@example.com
4	Sanchita	2005-07-11	2023-08-10	sanchita@example.com
9	Purba	2005-05-25	2023-08-22	purba@example.com

3> Query-3

Student_id	StudentName	date_of_birth	enrollment_date	Email
1	Sandip	2005-02-13	2025-08-16	sandip@example.com
2	Sumita	2005-01-19	2023-08-17	sumita@example.com
4	Sanchita	2005-07-11	2023-08-10	sanchita@example.com
5	Sanket	2004-12-15	2025-06-20	sanket@example.com
6	Sana	2005-03-18	2024-09-01	sana@example.com

4> Query-4

Student_id	StudentName	date_of_birth	enrollment_date	Email
2	Sumita	2005-01-19	2023-08-17	sumita@example.com
3	Partha	2005-02-10	2024-08-18	partha@example.com
5	Sanket	2004-12-15	2025-06-20	sanket@example.com
6	Sana	2005-03-18	2024-09-01	sana@example.com
7	Bikas	2005-07-30	2024-07-01	bikas@example.com
8	John	2005-07-26	2024-07-13	john@example.com

Experiment-09

Problem statement: Using aggregate function

Objective: Learn how to use sql aggregate function to summerize data.

Theory: Aggregate functions in sql performs calculations on multiple rows of data and return a single data.

Common aggregate functions include:

i> COUNT(): Returns the number of rows.

ii> SUM(): Returns the sum of numeric value.

iii> AVG(): Returns the average of numeric values.

iv> MIN() & MAX(): Return the minimum and maximum value.

Query:

i> Count the total no. of students.

ii> Find the average course credits.

iii> Find the maximum and minimum enrollment dates.

iv> Find the courses with minimum and maximum credits.

v> Display the count on number of the students that have ~~enrollment~~ enrolled for DBMS.

Solution:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;
CREATE TABLE students(
 Studentid INT PRIMARY KEY AUTO_INCREAMENT,
 StudentName VARCHAR(50),
 Date_of_Birth DATE,
 Enrollment_date DATE
 );
 CREATE TABLE Course(
 Courseid INT PRIMARY KEY AUTO_INCREAMENT,
 CourseName VARCHAR(100),
 Credits INT
 );
  CREATE TABLE Enrollment(
  Studentid INT,
  Courseid INT,
  FOREIGN KEY (Studentid) REFERENCES students (Studentid),
  FOREIGN KEY (Courseid) REFERENCES Course (Courseid));
 INSERT INTO students (Studentid, StudentName, Date_of_Birth,
  Enrollment_date) VALUES
 (1, 'Somdip', '2005-02-13', '2023-08-01'),
 (2, 'Sumita', '2005-01-19', '2023-08-19'),
 (3, 'Partha', '2005-02-10', '2024-07-16');
 INSERT INTO Course (CourseName, Credits) VALUES
 ('Artificial Intelligence', 2),
```

```
('DBMS', 4),
('Web Development', 3)
);

INSERT INTO Enrollments (studentid, courseid) VALUES
(1, 2),
(2, 3),
(3, 1));

-- count the total number of students
SELECT COUNT(*) AS TOTAL_students from students;
-- Find the average course
SELECT ANY(Credits) AS AVERAG_Credits from course;
-- Find the maximum and minimum enrollment dates.
SELECT MAX(Enrollment_date) AS MAX_Enrollment_date, MIN
(Enrollment_date) AS MIN_Enrollment_date from students;

SELECT CourseName, Credits from course where Credits =
(select min(credits) from course) or Credits = (select max
(credits) from course);

select count(*) as DBMS_students from Enrollment join course
```

Output:

1)

Total Student
3

2)

MAX_Enrollment_date	Min_Enrollment_date
2023-08-19	2023-08-01

3)

Average credits
3.0000

4)

CourseName	Credits
Artificial Intelligence	2
DBMS	4

5)

DBMS_Students
1

13/09/25

Experiment-10

Problem statement: Combine table using join and sub queries.

Objective: Learn how to combine data from multiple tables using join and sub queries.

Theory:

Joins: Used to retrieve data from two or more tables based on a related columns.

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.

Sub-queries: A query nested inside another query, used to retrieve data based on the result of the outer query.

Procedure:

- Perform an inner join to combine data from two tables.
- Used left and right join to retrieve data from related table.
- Write sub queries to fetched data from one table based on condition from another.

Queries:

1. Perform an inner join to get students and their enrolled courses.
2. Use a left join to retrieve all students and their enrollments.
3. Write a sub queries to find students enroll in 'Data Base systems'.

Solution:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;

CREATE TABLE students (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    StudentName VARCHAR(100),
    date_of_birth DATE,
    enrollment_date DATE,
);

CREATE TABLE Courses (
    course_id INT PRIMARY KEY AUTO_INCREMENT,
    course_name VARCHAR(100),
    course_code VARCHAR(20),
    credits INT
);

CREATE TABLE Enrollment (
    enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    course_id INT,
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id).
);

INSERT INTO students (student_id, studentName, date of birth,
  enrollment_date) VALUES
(1, 'Sandip', '2005-02-13', '2023-08-01'),
(2, 'Sumita', '2005-01-19', '2023-08-19'),
(3, 'Partha', '205-09-24', '2019-07-15');
```

```
INSERT INTO Courses (course_name, course_code, credits) VALUES
('Artificial Intelligence', 'CSE303', 4),
('DBMS', 'CSE301', 4),
('Operating Systems', 'CSE302', 3);

INSERT INTO Enrollment (student_id, course_id) VALUES
(1,1),
(2,3),
(3,2);

-- Perform an inner join to get students and their enrolled courses.
SELECT
    s.student_id,
    s.studentName,
    e.course_id,
    e.course_name,
    e.course_code,
    e.credits
FROM
    Students AS s
INNER JOIN
    Enrollment AS e ON s.student_id = e.student_id
INNER JOIN
    Courses AS c on e.course_id = c.courses_id;

-- Use a left join to retrieve all students and their enrollments.
SELECT
    s.student_id,
    s.studentName,
    e.course_id,
```

```
e.course_name,
e.course_code,
e.credits
FROM
   students AS s
LEFT JOIN
   Enrollment AS e ON s.student_id = e.student_id
LEFT JOIN
   Courses AS c ON e.course_id = c.course_id;

-- Write a sub queries to find students enroll in DBMS
 SELECT studentName
 FROM students .-
 WHERE student_id IN(
      SELECT student_id
      FROM Enrollment
      WHERE course_id = (
           SELECT course_id FROM Courses course_name = 'DBMS'
      )
 );
```

output:

query-1:

student_id	studentName	course_id	course_name	course_code	credits
1	Sandip	1	Artificial Intelligence	CSE303	4
2	Sumita	3	Operating Systems	CSE302	3
3	Partha	2	DBMS	CSE301	4

query-2

student_id	StudentName	course_id	course_name	course_code	credits
1	Sandip	1	Artificial Intelligence	CSE303	4
2	Sumita	3	operating systems	CSE302	3
3	Partha	2	DBMS	CSE301	4

query-3

	StudentName
	Partha

Assignment-11

Problem statement: Creating and managing views

Objective: Learn how to create views and perform operation on them.

Theory: A view is the virtual table based on the result set of an sql query. View allow user to simplify complex query and provide a level of abstraction. of an use of restrict data access.

Procedure:

1) Create a view that simplify the complex query.
2) Perform operation like select on the created view
3) Drop the view when it is no longer needed.

Query:

1) Create view 'Student_courseView'
2) Select * From studentcourseview.
3) Display those student-id with course-id 1 and 3.
4) Drop view studentcourseview.

Code:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;

CREATE TABLE Students (
    student_id INT PRIMARY KEY AUTO_INCREAMENT,
    student_firstName VARCHAR (100),
    student_lastName VARCHAR (100),
    enrollment_date DATE
);

CREATE TABLE Courses (
    course_id INT PRIMARY KEY AUTO_INCREMENT,
    course_name VARCHAR (100),
    course_code VARCHAR (20)
);

CREATE TABLE Enrollment (
    enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    course_id INT,
    FOREIGN KEY (student_id) REFERENCES Students (student_id),
    FOREIGN KEY (course_id) REFERENCES Courses (course_id)
);

INSERT INTO Students (student_firstName, student_lastName,
enrollment_date) VALUES
('Sandip', 'Kala', '2023-08-01'),
```

('Sumita', 'Pattanayak', '2023-08-19'),
('Partha', 'Ghosh', '2019-07-15');

```
INSERT INTO courses (course_name, course_code) VALUES
('Artificial Intelligence', 'CSE303'),
('DBMS', 'CSE301'),
('Operating systems', 'CSE302');
INSERT INTO Enrollment (student_id, course_id) VALUES
(1,1),
(2, 3),
(3, 2);

-- Create view 'studentcourseview'
CREATE VIEW studentcourseview AS
SELECT
    s.student_id,
    CONCAT(s.student_firstname, ' ', s.student_lastName) AS full_name,
    s.enrollment_date,
    c.course_id,
    c.course_name,
    c.course_code
FROM
    students s
JOIN
    Enrollment e ON s.student_id = e.student_id
JOIN
    Courses c ON e.course_id = c.course_id;
```

Assignment 10

```
-- Select * FROM studentcourseview.
SELECT * FROM studentcourseview;
-- Display those student_id with course id 1 & 3.
SELECT *
FROM studentcourseview
WHERE course_id IN (1,3);
-- Drop view studentcourseview.
DROP VIEW studentcourseview;
```

output:

query-2:

student_id	full_name	enrollment_date	course_id	Course_name	course_code
1	Sandip Kala	2023-08-01	1	Artificial Intelligence	CSE303
2	Sumita Pattanayak	2023-08-19	3	operating systems	CSE302
3	Partha Ghosh	2017-07-15	2	DBMS	CSE301

query-3:

student_id	full_name	enrollment_date	Course_id	Course_name	Course_code
1	Sandip Kala	2023-08-01	1	Artificial Intelligence	CSE303
2	Sumita Pattanayak	2023-08-19	3	operating systems	CSE302

View' Example-2

Problem statement: Create Two tables 'Branch-master' Table and 'Addr-Details' Table. In Brmch-Master' the attributes are 'Branch_no' (Primary key) and 'Name' whereas in 'Addr_details' table the attributes are 'Addr_no', 'code_no', 'Addr_type', 'Add1', 'Add2', 'city', 'state', 'Pincode'.

Queries:

1. Create a view Name 'Branch_Details'
2. Display the city, state, pincode of the branch NO. 2 & 5.
3. Display Name of all headoffices.
4. Display the Add_no, city, Add1, Add2 & pincode of all branches.

Code:

```
CREATE DATABASE Brmch;
USE Brmch;
CREATE TABLE Branch_Master (
        Branch_no INT(6) PRIMARY KEY,
        Name VARCHAR(10)
);
CREATE TABLE Addr_Details (
      Addr_no INT PRIMARY KEY,
      Code_no INT,
      Addr_type VARCHAR(10),
      Add1 VARCHAR(10),
      Add2 VARCHAR(100),
      City VARCHAR(50),
      state VARCHAR(50),
      Pincode VARCHAR(10),
      FOREIGN KEY (code_no) REFERENCES Brmch_Master (Branch_no));
```

```
INSERT INTO Branch_Master VALUES
    (1, 'Central Branch'),
    (2, 'North Branch'),
    (3, 'South Branch'),
    (4, 'East Branch'),
    (5, 'West Branch'),

INSERT INTO Add_Details VALUES
(101, 1, 'H', '123 Main st', 'Suite 500', 'New York,' 'NY', '10001'),
(102, 2, 'B', '456 North Ave', 'Floor 2', 'chicago', 'IL', '60611'),
(103, 3, 'B', '789 South Blvd', 'Building B', 'Houston,' 'TX', '77002'),
(104, 4, 'B', '321 East Rd', 'Room 12', 'Miami', 'FL,' 33101'),
(105, 5, 'B', '654 West st', 'Suite 20', 'Los Angeles', 'CA', '90001');

-- Create 'Branch' view
CREATE VIEW Branch AS
SELECT
    brn.Branch_no,
    brn.Name AS Branch_Name,
    ad.Addr_no,
    ad.Addr_type,
    ad.Add1
    ad.Add2
    ad.city,
    ad.state,
    ad.Pincode
FROM Branch_Master brn
JOIN Add_Details ad
ON brn.Branch_no = ad.Code_no;
-- Display the city, state, pincode of the branch_no 2 & 5.
SELECT Branch_no, City, State, Pincode FROM Branch
    WHERE Branch_no IN (2,5);
```

Assignment-12

--Display name of all head offices.

```
SELECT Branch_Name FROM Branch
    WHERE Addr_type = 'H';
```

-- Display the Addr_no, city, Addr1, Addr2 & pincode of all branches.

```
SELECT Addr_no, city, Addr1, Addr2, Pincode FROM Branch.
    WHERE Addr_type = 'B';
```

Output:

1.

Branch_no	Branch_Name	Addr_NO	Addr_type	Addr1	Addr2	city	state	Pincode
1	Central Branch	101	H	123 Mainst	Suite 500	New York	NY	10001
2	North Branch	102	B	456 North Ave	Floor2	chicago	JL	60611
3	South Branch	103	B	789 South Blvd	Building B	Houseton	TX	77002
4	East Branch	104	B	321 East Rd	Room 12	Miami	FL	33101
5	West Branch	105	B	654 Westst	Suite 20	Los Angeles	CA	90001

2.

Branch_NO	city	state	Pincode
2	Chicago	JL	60611
5	Los Angeles	CA	90001

3.

Branch_Name
Central Branch

4.

Addr_NO	city	Addr1	Addr2	Pincode
102	chicago	456 North Ave	Floor 2	60611
103	Houston	789 South Blvd	Building B	77002
104	Miami	321 East Rd	Room 12	33101
105	Los Angeles	654 West st	Suite 20	90001

Assignment-12

Problem Statement: PL SQL stored procedures and Functions.

Objective: To create and execute stored procedure and Function in PL SQL.

Theory:

i) A stored procedure is reusable of block of code that perform a specific task and can be invoked with parameters.

ii) A Function is similar to procedure but returns a value.

Procedure:

i) Write a stored procedure to insert a new student in students table.

ii) Create a Function to calculate the total no. of courses a student is enrolled in.

Query:

```
Create procedure AddStudents (
    IN firstName VARCHAR (100),
    IN lastName VARCHAR (100),
    IN birthDate DATE,
    IN enrollmentDate DATE
)
Begin
    Insert into students (student_firstName, student_lastName,
    birth_date, enrollment_date) VALUES
    (firstName, lastName, birthDate, enrollmentDate);
End;

Call Add student('Anita', 'sharma', '2006-05-21', '2024-07-01');
```

Code:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;

CREATE TABLE Students (
    Student_id INT PRIMARY KEY AUTO_INCREMENT,
    Student_firstName VARCHAR (100),
    Student_lastName VARCHAR (100),
    birth_date DATE,
    enrollment_date DATE
);

CREATE TABLE Courses (
    Course_id INT PRIMARY KEY AUTO_INCREMENT,
    Course_name VARCHAR (100),
    Course_code VARCHAR (20),
);

CREATE TABLE Enrollment (
    enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    course_id INT,
    FOREIGN KEY (student_id) REFERENCES students (student_id),
    FOREIGN KEY (course_id) REFERENCES Courses (course_id)
);

INSERT INTO students (student_firstName, student_lastName,
birth_date, enrollment_date) VALUES
('Sandip', 'Kala', '2005-02-13', '2023-08-01'),
('Sumita', 'Pattnayak', '2005-01-19', '2023-08-19'),
('Partha', 'Ghosh', '2005-11-19', '2019-07-15');
```

```
INSERT INTO courses (course_name, course_code) VALUES
('Artificial Intelligence', 'CSE303'),
('DBMS', 'CSE301'),
('Operating systems', 'CSE302');

INSERT INTO Enrollment (student_id, course_id) VALUES
(1,1);
(2,3),
(3,2);

DELIMETER $$
CREATE PROCEDURE AddStudent (
    IN firstName VARCHAR (10),
    IN lastName VARCHAR (10),
    birthDate DATE;
    enrollmentDATE DATE
)
BEGIN
INSERT INTO students (student_firstName, student_lastName,
birth_date, enrollment_date) VALUES
(firstName, lastName, birthDate, enrollmentDate);

 SELECT LAST_INSERT_ID () as new_student_id;
END $$
DELIMITER;
CALL AddStudent ('Anita', 'sharma', '2006-05-21', '2024-07-01');

SELECT * FROM students;
```

output:

student_id	student_firstName	student_lastName	birth_date	enrollment_date
1	Sandip	Kala	2005-02-13	2023-08-01
2	Sumita	Pattanayak	2005-01-19	2023-08-19
3	Partha	Gosh	2005-11-19	2019-07-15
4	Anita	Sharma	2006-05-21	2024-07-01

08/09/25

Assignment-13

Problem statement: Cursor

Objective: Understand how to work with cursor, triggers and various tables operation.

Theory:

- **Cursor:** Used to retrieve row from a result set one at a time.
- **Triggers:** Automatically perform action based on certain events on table.

Procedure:

1. Create a cursor to fetch data
2. Define a trigger to automatically update and insert based on certain conditions.
3. Perform table operation like updating, truncating, dropping, altering tables.

Code:

```
CREATE DATABASE UNIVERSITYdb;
USE UNIVERSITYdb;
GO

CREATE TABLE students(
StudentID INT IDENTITY (1,1) PRIMARY KEY,
FirstName NVARCHAR(50),
LastName NVARCHAR(50),
Email NVARCHAR(100),
DateofBirth DATE,
EnrollmentDATE DATE
);
GO
```

```
INSERT INTO students (FirstName, LastName, Email, DateofBirth,
EnrollmentDate) VALUES
('Alice', 'Johnson', 'alice.johnsm@exmple.com', '2008-05-15', '2024-01-15'),
('Bob', 'Smith', 'bob@example.com', '1999-08-21', '2024-01-16'),
('Carol', 'White', 'carol@example.com', '2001-12-11', '2024-01-17'),
('David', 'Brown', 'david@example.com', '2001-03-30' '2024-01-18'),
('Eva', 'Green', eva@example.cm', '1998-11-09', '2024-01-19'),
('Frank', 'Nové', 'frnk@example.com', '1999-07-22', '2024-01-20'),
('Grace', 'Taylor', 'grace@example.com', '2001-02-17', '2024-01-21'),
('Henry', 'Wilsm', 'henry@example.com', '2000-10-05', '2024-01-22');
GO

DECLARE @StudentID INT, @FirstName NVARCHAR(50),
@LastName NVARCHAR(50);

DECLARE Student_Cursor CURSOR FOR
SELECT StudentID, FirstName, LastName
FROM Students
WHERE EnrollmentDate >= '2024-01-19';
OPEN Student_cursor;
FETCH NEXTFROM Student_ cursor INTO @ StudentID, @ FirstName,
@LastName;

WHILE @@FETCH_STATUS = 0
BEGIN
    PRINT 'StudentID: ' + CAST (@StudentID AS NVARCHAR(10)) +
        ', Name: ' + @FirstName + ' ' + @LastName;
    FETCH NEXT FROM student_ cursor INTO @ StudentID, @FirstName,
                @LastName;
```

```
END
CLOSE student_cursor;
DEALLOCATE student_cursor;
GO
CREATE TRIGGER trg_AfterInsertStudent
ON Students
AFTER INSERT
AS
BEGIN
 DECLARE @StudentID INT, @FirstName NVARCHAR(50), @LastName
  NVARCHAR(50);
 SELECT @StudentID = StudentID,
        @FirstName = FirstName,
        @LastName = LastName
 FROM INSERTED;
   PRINT 'New student added: StudentID = ' + CAST(@StudentID
   AS NVARCHAR(10)) + ', Name: ' + @FirstName + ' ' + @LastName;
   END;
```

output:

StudentID: 5, Name: Eva Green
StudentID: 6, Name: Frank Norre
StudentID: 7, Name: Grace Taylor
StudentID: 8, Name: Henry Wilson.

08/11/25.