Assignment - 1

Experiment - 1 Converting a ER Diagram to relational database table.

Objective. To learn how to convert an ER Diagram into a relational database table.

Theory. ER Diagrams gives visual representation of entities, their attributes and relationships among entities. Relational table represent entities with column's attributes and rows as records.

Steps involved:

Step 1: Convert entities to table:

Rule - Each entity in the ER diagram is represented by a table in the relational database.

Step 2: Convert relationships to foreign keys:

Rule - Relationships b/w entities are represented by foreign keys.

Explanation - A new table is created for each multivalued attributes with a foreign key reference to the original table and a column for the multivalued attributes.

Step 3: Convert weak entities:

Rule - Weak entities are represented by using a composite primary key,

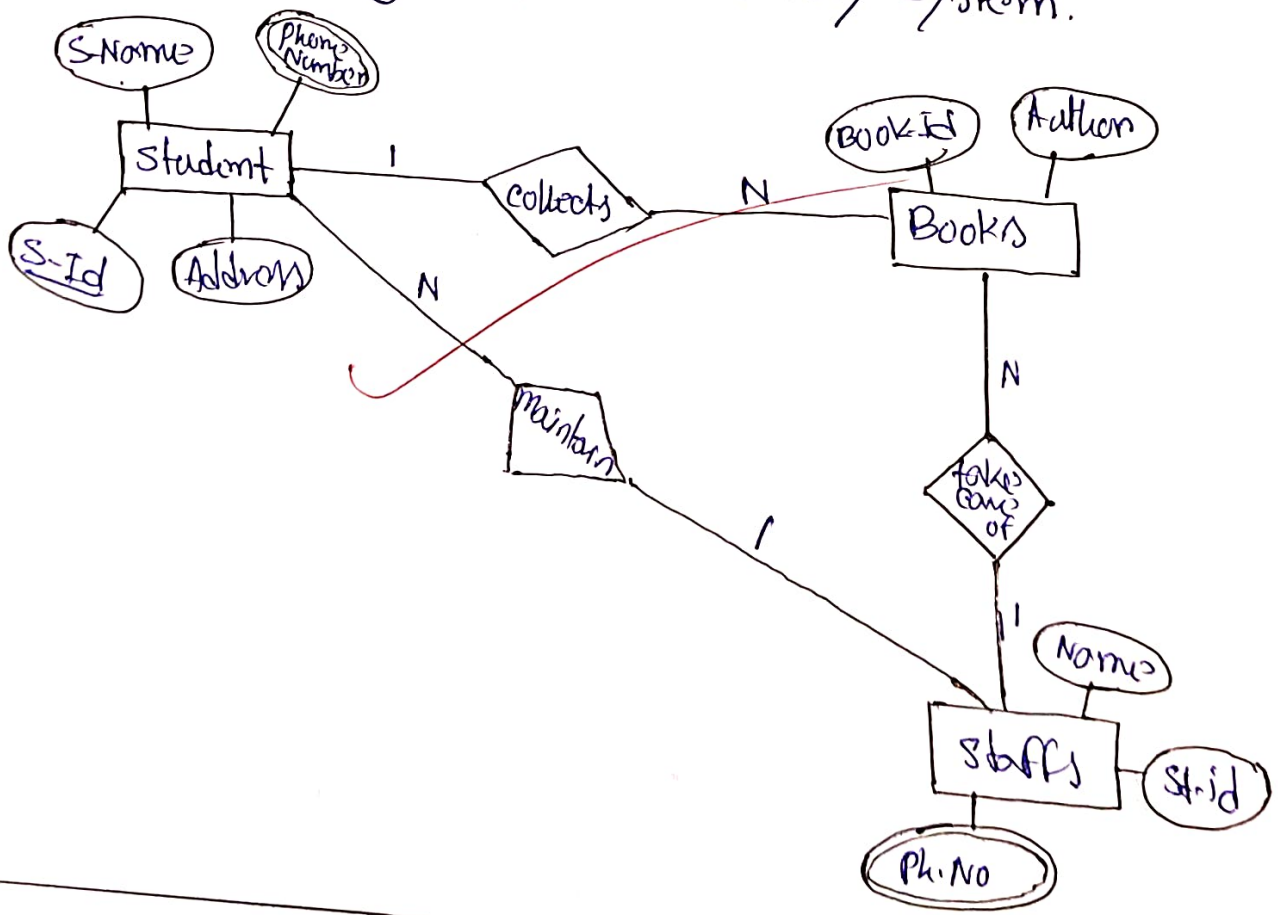
Explanation - The weak entity table includes the primary key of the related strong entity as part of its composite primary key to uniquely identify the weak entity.

Step 5: Address generation on Specialization

Rule - Generalization and specialization are handled by creating separate tables for the generalized and specialized entities.

Explanation - A table for the generalized entity is created along with separate tables for each specialized entity with foreign keys linking back to the general entity.

Example - ER Diagram of BMV Library System.



Transferring ER Diagram into Relational Database Table-

1. For students: Task: Display the names of those students whose S-ID is either 163 or 137

Code-

```
CREATE TABLE STUDENT(
    Name CHAR(50),
    S-Id INT(6). PRIMARY key
    Address VARCHAR(200),
    phone-Number INT(10));

INSERT INTO STUDENT (Name. S-Id, Address, phone-Number) VALUES ('RK', 139, 'HOWRAH', 859102523),
    ('SM', 163, 'P.medipur', 87139967),
    ('AP', 130, 'Kolkata', 98205 7193),
    ('NJ', 331, 'kolkata', 9023579165),
    ('JB', 137, 'knpur', 902 70123);
    ('SB', 012, 'Birbhum', 9857 20);

SELECT * FROM STUDENT;

SELECT * FROM STUDENT WHERE S-Id IN (163,137);
```

Output

Name	S-Id	Address	phone-Number.
RK	139	Howroh	859102523
SM	163	P.Medinipur	87139296 6
AP	130	kolkata	982057193
NJ	331	Kolkata	902357 165
JB	137	Kapur	90279123

Name	S-Id	Address	phone-Number
SM	163	P.medinipur	85 79 102 B23
JB	137	Kanpur	902 3549165

2. For Books : Task: Display those Book-id and Book-name from the table Book.

Code -

```
CREATE TABLE Book(
        Book-Name CHAR(50),
        Author CHAR (50),
        Book-Id INT(10) PRIMARY KEY
        publish-year INT(5),
        price INT(5) ;
INSERT INTO Book(Book-Name, Book-Id, publish-year,
publish-year, price) VALUES ('Animal Farm', G.orwell',
7100192, 1945, 375),
('Some Tree', 'J.Ashbery'. 7100176, 1956, 250),
('The pearl', 'J. Steinbeck', 7100 329. 1940, 500),
('The time Machine', 'H.G. Wells', 7100702, 1968, 900);
  SELECT * FROM Book;
  SELECT Book-Id, Book-Name. FROM Book;
```

Output -

Book-Name	Author	Book Id	publish year	price
Animal Farm	G. Orwell	7100192	1945	375
Some Tree	J. Ashbery	7100176	1056	250
The pearl	J steinbeck	7100329	1961	300
Coraline	N. Graiman	7100625	1940	500
The time machine	H.G. wells	7100702	1968	700

Book-Id	Book. Name
7100192	Animal Farm
7100176	Some Tree
7100329	The pearl
7100625	Coraline
7100792	The time machine.

3. For STUFFS.

Code -

```
CREATE TABLE STUFF(
        Name CHAR(50),
        st_id INT(6) PRIMARY KEY
        phone_Number INT(10));
INSERT INTO STUFF (Name, st_id, phone Number). VALUES
('S. Jana', 1107, 920646719),
('S. Dey', 1108, 9341092342,
('J. Biswas', 1110, 8571234160),
('N. Roy', 1197, 62310346),
('M. Dey', 1100, 96172057);
SELECT * from STUFF;
```

02/08/25.

Output:

Name	st_id	Phone_Number
S-Jana	1170	920656179
S. Dey	1108	934109223
J. Biswas	1110	85213160
N. Roy	1197	62310346
M. Dey	1100	92675497

Experiment - 02 Creating a database and tables and specify datatypes.

Objectives: To learn how to create a database and tables and specify datatypes.

Theory: A database is a structured collection of data that can be accessed and manage efficiently. In SQL a database contains tables, 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 data types include:

i) INTEGER: For storing whole numbers.

ii) VARCHAR: variable length, string used to store text data.

ii) DATE: For storing dates in standard format (DD-MON-YY)

Tables must also have primary key (that used as a identifier of each row) and foreign key (to establish relationship b/w tables).

Defining proper datatypes ensure data integrity and efficient database operations.

Steps involved:

Step 1: Create a new database by using

CREATE DATABASE Database Name;

Step 2: Switch to the newly created database by using 'USE' keyword.

Step 3: Create a 'students' table

```
CREATE TABLE students(
Student ID INT AUTO-INCEMETT PRIMARY KEY,
FirstName VARCHAR(50),
LastName VARCHAR(50),
BirthDate DATE,
EnrollmentDate DATE);
```

Step 4: create a 'courses' table

```
CRETE TABLE Courses(
CourseID INT AUTO-INCREMENT PRIMARY KEY,
CourseName VARCHAR(100),
Credits INT);
```

Step 5: Create an 'Enrollments' table to link students and courses.

```
CREATE TABLE 'Enrollments(
Enrollment ID INT AUTO-INCREMENT PRIMARY KEY,
StudentID INT,
CourseID INT,
FOREIGN Key (studentID) REFERENCES students(StudentID),
FOREIGN KEY (CourseID) REFERENCES Courses(courseID);
```

Code:

```
CREATE DATABASE UniversityDB;
USE UniversityDB;
CREATE TABLE Students(
    student_Id INT PRIMARY KEY,
    Name VARCHAR(50),
    Birth_Date DATE,
    Enrollment_Date DATE);

CREATE TABLE Courses(
    Courses_Id INT AUTO_INCEMENT PRIMARY KEY,
    course_Name VARCHAR(100),
    Credits INT(5));

CREATE TABLE Enrollments(
    Enrollment_Id AUTO_INCREMENT. PRIMARY KEY,
    student_Id INT(6),
    course_Id INT(6),
    FOREIGN KEY (student_Id) REFERENCES students
                (student_Id),
    FOREIGN KEY (course_Id) REFFERENCES course
                (course_Id));

DESC Students;
DESC Courses;
DESC Enrollments;
```

Output:

Students -

Field	Type	Null	Key	Default	Extra
Student-Id	int(11)	NO	PRI	NULL	Auto Increment
Name	varchar(50)	YES	"	"	"
Birth-Date	date	YES	"	"	"
Enrollment-Date	date	YES	"	"	"

Courses -

Field	Type	Null	Key	Default	Extra
Course-Id	INT(11)	NO	PRI	NULL	auto increment
Cause-Name	VARCHAR(100)	YES	"	"	"
Credits	INT(11)	"	"	"	"

Enrollments -

Field	Type	Null	Key	Default	Extra
Enrollment Id	INT(11)	NO	PRI	NULL	auto increment
Student-Id	INT(11)	YES	MUL	NULL	
Course-Id	INT(11)	YES	MUL	NULL	

23/04/25

Assignment 3

Problem statement - Specifying constraints & creating indices.

Objective - Learn how to specify constraints (Example- primary key, foreign key, unique - Not Null) and create indices for tables.

Theory - Constraints are rules applied to table columns to enforce data integrity and reliability within a database. Common constraints include-

i) **primary key (PK)** - Ensures that each record in a table is unique and not null.

ii) **Foreign key (FK)** - Establish a link between two tables ensuring referencial 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) **check** - Ensure that all values in a column satisfy a specific condition.

Indices include the speed of data retrival operations on a table at the cost of additional storage. Indexes can be created on one or more column to make searching faster.

Procedure -

(1) Add constraints to table (primary key, Foreign key, unique and Not Null).

② Create indices on specific columns for fast retrival.
③ Verify the constraints and indices by inspecting the schema.

Query- Student-Id in the Enrollment table as unique and course name in the courses name in the courses table as Not Null.

Solution

```
CREATE DATABASE University DB;
USE UniversityDB;
CREATE TABLE students(
        Student-Id INT PRIMARY KEY,
        Name VARCHAR(50),
        Birth_Date DATE,
        Enrollment_Date DATE);
CREATE TABLE Courses(
        Courses-Id INT AUTO-INCREMENT PRIMARY KEY,
        Course-Name VARCHAR(100) NOT NULL,
        credits INT(5));
CREATE TABLE Enrollments(
        Enrollment Id INT AUTO-INCREMENT
                PRIMARY KEY,
        Student-Id INT(6),
        Course-Id INT(6),
        FOREIGN KEY (Student Id) REFERENCES
                students(Student-Id),
```

FOREIGN KEY (course-Id) REFERENCES
courses (course-Id));

ALTER TABLE Enrollments ADD CONSTAAINT
UC - Student UNIQUE (student-Id);

DESC students;

DESC courses;

DESC Enrollments;

Output.

students.

Field	Type	Null	Key	Default	Extra
Student-Id	Int(11)	NO	PRI	NULL	
Name	varchar(50)	YES		NULL	
Birth Date	date	YES		NULL	
Enrollment Date	date	YES		NULL	

Courses:

Field	Type	Null	Key	Default	Extra
course-Id	Int(11)	NO	PRI	NULL	Auto Increment
Course-Name	varchar(100)	NO		NULL	
Credits	int(5)	YES		NULL	

Enrollments:

Field	Type	Null	Key	Default	Extra
Enrollments Id	Int(11)	NO	PRI	NULL	Auto Increment
Student-Id	Int(6)	YES	UNI	NULL	
course-Id	Int(6)	YES	MUL	NULL	

22/08/25

Assignment - 04

problem statement - Table and Record Handling.

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

Theory - In relational Databases the insert statement used to as new rows of the data into a table. It allows the users to specify values for each columns ensuring data integrity by checking constraints like primary key, foreign key etc is crucial during the insertion process. Records are the actual data stores in the tables.

Procedure -

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

Queries -

1. Display the student-Id, First and last name of all the students.
2. Display the courses whose credit is greater than equal to 3.

③ Display the student whose course-Id is 2 and student-Id is 4.

④ Display the students who has enrolled in the year 2023.

Solution -

```
CREATE DATABASE UniversityDB;
USE UniversityDB;
CREATE TABLE students(
        Student-Id INT PRIMARY KEY,
        Name VARCHAR(50),
        Birth-Date DATE,
        Enrollment-Date DATE);
INSERT INTO students (Student-Id, Name, Birth-Date,
Enrollment-Date) VALUES (1, 'Rohan Karani', '2005-3-13',
'2023-07-05'),
(2, 'Sana Mondal', '2005-7-12', '2023-08-07');
CREATE TABLE Courses(
        Course-Id INT PRIMARY KEY,
        Course_Name VARCHAR(100),
        Credits INT(5));
INSERT INTO Courses (courses-Id, Course-Name, Credits)
VALUES (1, 'DBMS', 4), (2, 'OS', 2);
```

```
CREATE TABLE Enrollments(
    Enrollment_Id INT AUTO_INCREMENT PRIMARY KEY,
    Student_Id INT(6),
    Course_Id INT(6),
    FOREIGN KEY(Student_Id) REFERENCES
        Courses(Course_Id));

INSERT INTO Enrollments (Student_Id, Course_Id)
VALUES (1,1) .
(2,2)
SELECT Student_Id, Name FROM students;
SELECT Course_Id, Course_Name, Credits FROM
Courses WHERE Credits >= 3;
SELECT S.Student_Id, S.Name, C.Course_Name FROM
Students S JOIN 'Enrollments E ON S.Student_Id =
E.Student_Id JOIN Courses C ON E.Course_Id =
C.Course_Id WHERE S.Student_Id = 2 AND C.Course_Id=2;

SELECT Student_Id, Name, Enrollment_Date FROM
Students WHERE YEAR(Enrollment_Date) = 2023;
```

output:

(1)

Student_Id	Name
1	Rohan Kumar
2	Sara Mondal

2)

Course-Id	Course-Name	Credits
1	DBMS	4

3)

Student-Id	Name	Course-Name
2	Sana mandal	05

)

Student-Id	Name	Enrollment-Date
2	Sana mandal	2023-07-08

23/05/25

Assignment 5

Problem Statement: Query data using select & Delete.

Objective - Learn how to retrive data using select and remove data using the delete statement.

Theory - The select statement is most commonly used SQL command to query and retrive data from tables.

The delete statement is used to remove records from a table. Both these commands are fundamental to manipulating the data store in a database.

Procedure:

1. Use the select statement to retrive specific data from the tables.
2. Use the delete statement to remove records from the tables.
3. Verify the results by viewing the modified tables.

Queries:

1. Select all the records from the students table.
2. Select specific fields from the courses table.
3. Delete the records from the Enrollment table where Student-Id = 1.

Solution:

Code:

```
create database collegeDB;
use collegeDB;
create table Student(
```

```
S_Id    int    primary key    auto_increment;
First_Name  varchar(50),
Last_Name . varchar(50),
DOB date,
Enrollment_date date);

insert into student (First_Name, Last_Name, DOB, Enrollment_date)
values ('Rohan', 'Karumi', '2005-12-07', '2023-07-08'),
('Abhna', 'punkaj', '2004-09-17', ('2024-08-09');

create table course(
C_Id   int  auto_increment  primary key,
C_Name  . varchar(100),
credits int);
insert into course (C_Name, credits int) values ('OS',2);
insert into (.'DBMS', 1);
create table Enrollment(
E_Id   int  auto_increment  primary key,
S_Id  int,
C_Id  int,
foreign key (S_Id ) references student (S_Id),
foreign key (C_Id) references course (C_Id));
insert into Enrollment (S_Id, C_Id) values (1,1); (2,2);
```

(i) Select * from Student;

(ii) Select C_Name from Course;

(iii) delect from Enrollment where S_Id = 1;
Select * from Enrollment;

output

(1)

S-Id	First Name	Last-Name	DoB	Enrollment Date
1	Rohan	Karemi	2005-[illegible]-07	2023-07-08
2	Aditya	purkait	2001-09-17	2021-08-09

(2)

C-Name
OS
DBMS

(3)

E-Id	S-Id	e-Id
2	2	2
Null	Null	Null

3/08/25

Assignment-06.

problem statement: Querying data using update, truncate, drop alter.

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

Theory-

update - This command used to modify existing records in a table.

Truncate - It removes all records from a table but keeps the table structure.

Drop - This command is used to completely remove 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 modify data in tables.

② Use the Truncate statement to clear all records from a table.

③ Use the Drop statement to delete a table.

④ Use the Drop state to ~~delete a table~~ modify the structure of a table.

Queries.

① Update the course table, set credits = 5 where course name: DBMS.

⑪ Truncate the table Enrollments.

(iii) Alter the student table to add a new column email.

(iv) Update the courses table set course name = RDBMS where C-Name = DBMS

(v) Alter the students table, change the size of the field S-Name to 80.

Code:

```
create database collegeDB;
use collegeDB;
create table student(
S-Id int primary key auto-increament,
First-Name varchar(50),
Last-Name varchar(50),
DOB date,
Enrollment-date date);
insert into student (First-Name, Last-Name, DOB, Enrollment
date) values ("Rohan', 'Karimi', '2005-12-07', '2023-07-08'),
('Abhora', 'Pankaj', '2005-07-11', '2023-07-13');
create table course(
C-Id int auto-increament primary key,
C-Name varchar(100),
credits int);
insert into course(C-Name, credits) values ('OS', 2),
('DBMS', 1);
```

```
Create table Enrollment(
  E-Id int auto-increment primary key,
  S-Id int,
  C-Id int,
  foreign key (S-Id) references student (S-Id),
  foreign key (C-Id) references course (C-Id));
  insert into Enrollment (S-Id, C-Id) values (1,1), (2,2);
  Select * from student;
  Select * from course;
  Select * from Enrollment;
```

(I) update course set credits = 5 where C-Name = 'DBMS';

(II) Truncate table Enrollment;

(III) alter table student add (email varchar(100));

(IV) update course set C-Name = 'RDBMS' where C-Name = 'DBMS';

(V) alter table student modify first varchar(80);
set sql-state-updates = 0;

Output

(I) & (IV)

C-Id	C-Name	Credits
1	OS	2
2	DBMS	4
NULL	NULL	NULL

(II)

E-Id	S-Id	C-Id
NULL	NULL	NULL

(III) & (V)

S-Id	First-Name	Last Name	DOB	Enrollment Date	email
1	Rohan	Kaur	2005-12-07	2023-07-08	NULL
2	Abhay	pankaj	2005-07-11	2023-07-13	NULL
NULL	NULL	NULL	NULL	NULL	NULL

30/08/25

Assignment - 07

Problem Statement - Retrieving data with WHERE Clause.

Objective - Learn how to retrieve data using the select statement, with where clause.

Theory - The where clause in sql is used to filter records based on specific conditions. It is commonly used with the select statement to retrieve only those records that meet the defined criteria.

procedure-

(i) Use the select statement with the where clause to retrive specific records.

(ii) preforms various queries using diffrent conditions in the where clause.

Queries:

(i) Select the students whose first name is 'Rohan'.

(ii) Display the student name and Id who are enrolled after 2022.

(iii) Display the S-Name, S-Id & phone no, who has taken course either DSA or DBMS

(iv) Display the name, Id and email-Id of those students whose name starts with the letter S.

(v) Display the name and Id of those students who didn't enroll for the course Image processig, Cyber security & Robotics.

(vi) Display the Id, Name and email of students in

descending order of there ids.

(VII) Display the Name, Id of students who have enroll either in 2022 on in 2023

Code:

```
create database MahavidyalayaDB2;
use MahavidyalayaDB2;
Create table students(
S-Id int primary key auto-increement,
first_Name varchar(50),
Last_Name varchar(50),
Enrollment_Date date,
Email .varchar(50),
phone_Number bjgint);
insert into students(first_Name, Last_Name, Enrollment_Date,
Email, phone_number) values ('Rohan', 'Karani', '2023-07-08',
'Rohan@gmail.com', 9876954831),
'Sana', 'mondd', '2018_07-09', 'Sanan76856@gmail.com',
                                        9874576851),
'Amy', 'Ghosh', '2019-05-08', 'sweet00@gmail.com',
                                        9877495782),
'Yashraj', 'Gupta', '2025-07-08', '4734@gmail.com', 987569832),
'Rinki', 'Roy', '2022-07-08', 'rnki34@gmail.com', 987495832);
create table Enrollments(
E-Id int auto-increment primary key,
SId int,
C-Id int,
```

foreign key (S-ID) references students (S-ID),
foreign key (c-ID) references courses (c-ID));
insert into Enrollments (S-ID, c-ID) values (1,1),
(2,2),
(4,4),
(5,5),

(i) Select * from students where first_Name = 'Rohan';

(ii) select S-Id, first_Name, Last_Name .from students where Enrollment_Date > '2022-12-31';

(iii) Select S-ID, first_Name, Last_Name from students.

(iv) Select S-ID, first_Name, Last_Name, Email from students. where first_Name Like 'S%';

(v) Select S-ID, Sfirst-Name, S.Last_Name, from students S join Enrollments e on S.ID = e.S.ID join courses c on e.c-ID = c.c-ID .where c.c_Name . not in ('cyber Security', 'Image processing', 'Robotics');

(vi) [Select - S-ID, Concat (first_Name, ' ', 'Last-Name) as full-Name, Email- from students order by desc;]

(vii) Select S-ID, concat(first Name, ' ', Last Name) as full Name from students where year (Enrollment-date) in (2022,2023);

Output:

(i)

First-Name	Last-Name	Enrollment Date	Email	Phone-Number
Rohan	Kanani	2023-01-08	Rohan@gmail.com	9879513822
Null	Null	Null	Null	Null

(II)

S-ID	First-Name	Last-Name
1	Rohan	Karuni
4	Yashraj	Gupta
Null	Null	Null

(IV)

SID	First-Name	Last Name	Email
1	Rohan	Karuni	Rohan@gmail.com
Null	Null	Null	Null

(V)

S-ID	First-Name	Last Name
1	Rohan	Karuni
2	Sana	Mondal
3	Rinki	Roy
Null	Null	Null

(III)

SID	First-Name	Last-Name	Phone Number
1	Rohan	Karuni	9867541
2	Sana	Mondal	6681475 67
Null	Null	Null	Null

(VI)

S-ID	Full-Name	Email
5	Rinki Roy	rinki34@gmail.com
4	Yashraj Gupta	YG@gmail.com
3	Amy Ghosh	Shree900@gmail.com
2	Sana Mondal	Sana@gmail.com
1	Rohan Karuni	Rohan@gmail.com
Null	Null	Null

(VII)

SID	Full-Name
1	RohanKarni"
5	Rinki Roy .
Null	Null

Assignment-08

problem statement: Retriving data using in, between and like.

Objective - Learn how to retrive data using in between and like and other filtering conditions in sql.

Theory.

(i) In - Allows you to specify multiple values in a where clause.

(ii) Between - Selects values within a given range.

(iii) Like - Used for pattern matching with wild cards.

Eg: percent for any number of characters.

Queries.

(i) Retrive those students whose S-ID in either 2, 5 or 7.

(ii) Retrive those students whose Enrollment date between 1st August 2023 & 30th August 2023.

(iii) Retrive those students whose name start with S.

(iv) Retrive the details of those students whose S-ID is not 1, 4 or 9.

(v) Display the students who have enrolled DBMS and DSA.

Code:

```
create database colleges;
Use colleges;
create table pupil(
S-ID int primary key,
Full_Name varchar(50),
```

```
Enrollment_Date date,
Address varchar(100),
phone_Number bigint);
insert into pupil (S_ID, Full_Name, Enrollment_Date,
Address, phone_Number) values (1, 'Rohan Karuni', '2023-03-02',
'Kolkata', 9087563423),
(4, 'Rita maity', '2023-08-03', 'Bihar', 876342098),
(2, 'Arjan jana', '2023-07-12', 'Asansol', 899958698),
(7, 'Riya Roy', '2023-09-05', 'Pune', 9858961295),
(9, 'Riva Arora', '2023-03-07', 'Arambag', 7886430912),
(5, 'Yash Debgan', '2023-05-14', 'Mumbai', 9085768412);

Create table Course(
C_ID int auto_increment primary key,
C_Name varchar(80));
insert into course(C_Name) values ('DBMS'),
('DSA'),
('cyber Security'),
('Robotics'),
('Image processing'),
('OS');
Create table Enrollment(
E_Id int auto_increment primary key,
S_Id int,
C_Id int,
foreign key(S_Id) references pupil(S_Id),
```

foreign key (c-Id) references course (c-Id));

insert into Enrollment (S-Id, c-Id) values (1,1), (1,2),
(2,3),
(7,1),
(9,5),
(5,6);

(i) Select * from pupil where s-Id in (2,5,7);

(ii) Select * from pupil where Enrollment-Date between '2023-08-01' and '2023-08-30';

(iii) Select * from pupil where full-name like 'S.';

(iv) Select * from pupil where s-Id not in (1,4,9);

(v) Select p.S-Id, p. full-Name from pupil p join Enrollment e on p.S-Id = e.S.Id join course c on e.course-Id = c.Course-Id where e.course-Name in ('DSA', 'DBMS') group by p.S-Id, p.full-Name having count (distinct e.Course-Name) = 2;

Output:

(i)

S-Id	Full-Name	Enrollment-Date	Address	Phone-Number
2	Anjan Jana	2023-07-12	Asansol	8954586098
5	Yash Debgan	2023-05-14	Mumbai	9087566312
7	Riya Roy	2023-09-05	Pune	9758961295
Null	Null	Null	Null	Null

(ii)

S-Id	Full-Name	Enrollment-Date	Address	Phone-Number
4	Riya Maity	2023-08-03	Bihar	87 56342098
Null	Null	Null	Null	Null

(iii)

S-Id	Full-Name	Enrollment-Date	Address	Phone-Number
1	Rohan Karan	2023-03-02	Kolkata	9081563423
Null	Null	Null	Null	Null

(iv)

S-ID	Full-Name	Enrollment Date	Address	Phone-Number
2	Anjan Jana	2023-07-12	Asansol	8954589985
5	Yash Debgan	2023-05-14	mumbai	9087563412
7	Rita Maity	2023-09-05	pune	9758961295
Null	Null	Null	Null	Null

13/09/25

S-Id	Full-Name
1	Rohan Karmi
7	Rita Maity
Null	Null

Assignment-9

problem statement- Using aggregate function.

objective- Learn how to use sql aggregate function to summerize data.

Theory: Aggregate functions in SQL perform 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 the numeric value.

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 or number of the student that have enrolled for DBMS.

Solution:

```
CREATE DATABASE University DB;
USE University DB;

CREATE TABLE students(
 Studentid INT PRIMARY KEY AUTO-INCREMENT,
 StudentName VARCHAR(50),
 Date of Birth DATE,
 Enrollment-date DATE
```

```
);
CREATE TABLE Course(
CourseId INT PRIMARY KEY AUTO-INCREMENT,
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, DateofBirth,
Enrollment_date) VALUES
('1', 'RK', '2005-02-13', '2023-08-01')
('2', 'SN', '2005-01-19', '2023-08-19')
('3', 'PG', '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 couse

SELECT ANY(credits) AS AVERAG. credits from couse.

Find maximum and minimum enrollment dates.

SELECT MAX (Enrollment_date) AS MAX Enrollment date, MIN (Enrolment-date) as MIN- Enrollment- date from students;

SELECT CourseNome, credits from course where credits = (Select max(Credits) from couse) or credits = (Select max (credits) from couse);

Select count(*) as DBms- Student from Enrollments join course.

output.

1.

Total Student
3

2.

Max- Enrollment label	min Enrollment date
2023-08-19	2023-08-01

3.

Average credits
3.0000

4.

CourseNome	credits
Artificial Intellige	2
DBms	4

5.

DBms- students
1

Assignment -10

Problem Statement - Combine table using join of sub

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

Theory -

Joins - Used to retrive data from two or more tables based on a related clumn.

Inner join - Returns records that have matching values in both tables.

- Left join - Returns all records from the left table and the the matched records from the right.
- Right join - Returns all records from the right table and the matched records from the left table.

Sub queries: A query nested inside mother query, used to retrive data based on the result of the outer query.

procedure:

- perform an inner join to combine data from two table.
- Used sub quries to fetched data from one table based on condition from mother.

quries:

- perform an inner join to get students and their.
- Use a left join to retrieve all students and their enrollments.
- write a such queries to find student erorr in

Solution -

```
CREATE DATABASE UniversityDB,
  USE UniversityDB;
CREATE TABLE Students (
        Student_id - INT PRIMARY KEY AUTO-INCEMENT,
        Student Name VARCHAR(100)
        date of birth DATE,
        enrollment-date . DATE,
);
CREATE TABLE Courses(
      Course_id INT PRIMARY KEY AUTO-INCEMENT,
      Course_name VARCHAR (100),
      Course_code VARCHAR(20),
       credits INT
  );
CREATE TABLE Enrollment (
      enrollment_id INT PRIMARY KEY AUTO-INCENMENT,
      student_id INT,
      FOREIGN KEY (student_id) REFERENCES students
                                    (student_id)
      FOREIGN KEY (course_id) REFFRENCES .
    );                              course( course_id);
   Insert into students (Student_id, Student_Name, date_of_birth,
                         enrollment_date) VALUES
  (1, 'Rohan', '2004-03-20', '2023-08-01'),
  (2, 'Sumita', '2005-01-19', '2023-08-19'),
  (3, 'Partha', '2005-09-24', '2023-07-15'),
Insert Into Courses (Course_name, Course_code, Credits) VALUES
( 'Artificial Intelligence', 'CSE303', 4);
( 'DBMS', 'CSE301', 4),
( 'OS', '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,
      C.Course_id,
      C.Course_Name,
      C.Course_Code,
      C.Credits.
FROM
    Students As S
INNER JOIN
      Enrollment As e or S.student_id = e.student_id
INNER JOIN
      Courses As c one . course_id = C.course_id;
```

USE a left join to retrieve all students and their enrollments

```
SELECT
      S.Student_id,
      S.Student_Name,
      C.Course_id,
      C.Course_Name,
      C.Course_Code,
      C.Credits
FROM
    Student As S
LEFT JOIN
      Enrollment As C or S.student_id = e.student_id;
```

write a sub queries to find students enroll in DBMS

```
SELECT Student Name
FROM Students
WHERE Student_id INT(
      'SELECT Student_id
      FROM Enrollment
      WHERE Course_id = C
            SELECT Course_id FROM COURSES Course Name = 'DBM
      )
```

Output :-

Query : 1 →

St-id	Student_Name	Course_id	Course_Name	Course_Code	Credits
1	Rohan	1	A-I	CSE303	4
2	Sumita	3	O.S	CSE302	3
3	Partha	2	DBMS	CSE301	4

Query : 2 →

St-id	Student_Name	Course_id	Course_Name	Course_Code	Credits
1	Rohan	1	A-I	CSE303	4
2	Sumita	3	O.S	CSE503	3
3	Partha	2	DBMS	CSE301	4

Query : 3 →

	Student_Name
	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 simply 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 course view'.
2. select * from studentviewco...
3. Display those student_id with couse_id 1 and 3.
4. Drop view studentcourseview.

Code -

```
CREATE DATATABLE universityDB;
Use universityDB;

CREATE TABLE students(
    student_id INT PRIMARY KEY Auto-INCREMENT,
    student_firstName VARCHAR (100),
    student_LastName VARCHAR (100),
    enrollment_date DATE
);

CREATE TABLE courses(
    course_id INT PRIMARY KEY AUTO-INCEMENT,
```

```
    Course_name  VARCHAR (100),
    Course_code  VARCHAR (20)
);
CREATE  TABLE  Enrollment (
    enrollment_id INT PRIMARY KEY AUTO_INCEMENT,
    Student_id INT,
    course_id INT,
    FOREIGN KEY (Student_id) REFERENCES Courses
                                (course_id)
);
INSERT  INTO  Students (student_firstName, student_Lastname
  enrollment_date) VALUES
("RK", '2023-08-01');
("SM", '2023-08-19');
("SN", '2019-07-15');
INSERT  INTO  Courses (course_name, Course_code) VALUES
('Artificial Intelligence', 'CSE303'), ('DBMS', 'CSE301'),
("Operating System", '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
    Student S
JOIN
    Enrollment ON S.Student_id = e.Student_id
JOIN
    Enrollment ON e.coyse_id = c.Coure_id,
```

```
SELECT * FROM studentcourseview,
SELECT * FROM StudentCourse view;
```

Display those student_id with course id 1 & 3.

```
SELECT *
FROM studentcourseview WHERE Coure 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	RK	2023-08-01	1	Artificial Intelligence	CSE305
2	SM	2023-08-19	3	operating system	CSE302
3	SN	2019-07-15	2	DBMS	CSE301

query - 3

Studen-t-id	Full-name	enrollment date	course-id	course_name	course_code
1	RK	2023-08-01	1	Artifical intelligence	CSE305
2	SM	2023-08-19	3	operating system	CSE302

Assignment-12

problem statement: PL/SQL Stored, procedure & functions.

Objective: To create and execute stored procedure & functions in PL/SQL.

Theory:

1. A stored procedure is a reusable piece/block of code that performs a specific task & can be invoked with parameters.
2. A function is similar to a procedure but returns a value.

1. Create procedure add_student (

```
FirstName VARCHAR (30),
LastName VARCHAR (50)
BirthDate DATE,
EnrollmentDate DATE
```

),

As begin

INSERT INTO students (

FirstName, LastName, BirthDate, EnrollmentDate.

)

VALUES (FirstName, LastName, BirthDate, EnrollmentDate);

END;

CALL ADD_student ('John', 'Doe', '2000-01-01', '2023-09-15');

Code:

```
CREATE DATABASE Universitydb;
USE Universitydb;
CREATE TABLE STUDENTS(
    StudentID INT AUTO-INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100),
    DateofBirth DATE
);
INSERT INTO STUDENTS VALUES
(1, 'Alice', 'Johnson', 'alice.johnson@gmail.com', '2000-05-5',
'2024-01-15'),
(2, 'Bob', 'Smith', 'bob@gmail.com', '1999-08-21', '2024-01-16'),
(3, 'Carol', 'White', 'carol@gmail.com', '2001-12-11', '2024-01-17'),
(4, 'David', 'Brown', 'david@gmail.com', '2000-03-30', '2024-01-18')
- create stored functions.
DELIMETER //
CREATE PROCEDURE ADD Student(
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100),
    Date of Birth DATE,
    Enrollment Date DATE
)
BEGIN
INSERT INTO STUDENTS (FirstName, LastName, Email,
    DataofBirth, Enrollment Date) VALUES
```

(FirstName, LastName, Email, DateofBirth, EnrollmentDate);

END //

DELIMITER;

CALL ADD-student ('Rohan', '[illegible]', 'abc@gmail.com',
'2000-01-01', '2023-09-15');

SELECT * FROM STUDENTS;

output:

Student ID	First Name	Last Name	Email	Dataof Birth	Enrollment Date
1	Alice	Johnson	alice@gmail.com	2000-05-15	2024-01-15
2	Bob	Smith	bob@gmail.com	1999-08-21	2024-01-16
3	Carol	White	carol@gmail.com	2001-12-11	2024-01-17
4	David	Brown	david@gmail.com	2000-03-30	2024-01-18
5	Rohan	Kanani	abc@gmail.com	2000-01-01	2023-09-15