e). We can identify the following functional dependencies:

Admission table:

Determinant: Admission ID

Dependent Fields: Admission Date, Admission Status, Phone

Attendance table:

Determinant: Attendance Date, Student ID

Dependent Fields: Student Present, Student Absent

Payment table:

Determinant: Receipt Number

Dependent Fields: Student Name, Amount Paid, Student Phone

Student table:

Determinant: Student ID

Dependent Fields: Student Name, Student Class, Student Phone

Based on the functional dependencies, following entity sets or tables can be identified:

Admission (with Admission ID as the primary key)

Attendance (with a composite primary key of Attendance Date and Student ID)

Payment (with Receipt Number as the primary key)

Student (with Student ID as the primary key)

The relationships among the entity sets can be identified as follows:

Admission and Student have a one-to-one relationship (one admissions can belong to one student)

Attendance and Student have a many-to-one relationship (many attendance records can belong to one student)

Payment and Student have a many-to-one relationship (many payments can be made by one student)

For each relationship, we can identify the connectivity and participation as follows:

Admission to Student: one-to-one, mandatory (one admission must belong to a student)

Attendance to Student: many-to-one, mandatory (every attendance record must belong to a student)

Payment to Student: many-to-one, mandatory (every payment must be made by a student)

f).

As following entities are be identified:

Admission

Admission\_ID (Primary Key, INT)

Admission\_Date (DATE)

Admission\_Status (VARCHAR)

Phone (VARCHAR)

Student\_ID (INT)

Attendance

Attendance\_ID (Primary Key, INT)

Attendance\_Date (DATE)

Student\_ID (Foreign Key, INT)

Student\_Present (VARCHAR)

Student\_Absent (VARCHAR)

Payment

Receipt\_Number (Primary Key, INT)

Student\_ID (Foreign Key, INT)

Amount\_Paid (DECIMAL)

Payment\_Date (DATE)

Student

Student\_ID (Primary Key, INT)

Student\_Name (VARCHAR)

Student\_Class (VARCHAR)

Student\_Phone (VARCHAR)

To ensure that the tables are in 2nd Normal Form, we have ensured that each table has a single primary key attribute. All other attributes are dependent on this primary key attribute. Therefore, each table is in 2nd Normal Form.

To ensure that the tables are in 3rd Normal Form, we need to check that there are no transitive dependencies. We can see that the Student table has a transitive dependency between Student\_ID, Student\_Name, Student\_Class, and Student\_Phone. To remove this transitive dependency, we can create a new table called Student\_Details as follows:

Student\_Details

Student\_ID (Primary Key, INT)

Student\_Name (VARCHAR)

Student\_Phone (VARCHAR)

Student

Student\_ID (Primary Key, INT)

Student\_Class (VARCHAR)

Student\_Details\_ID (Foreign Key, INT)

Now, Student\_Details has Student\_ID as its primary key and Student\_Name and Student\_Phone as dependent attributes. Student table has Student\_ID as its primary key, Student\_Class as a dependent attribute, and a new foreign key Student\_Details\_ID. This ensures that each table is in 3rd Normal Form.

To ensure that the tables are in Boyce-Codd Normal Form (BCNF), we need to ensure that every determinant is a candidate key. We can see that all tables already have a single primary key attribute, which is a candidate key. Therefore, all tables are in BCNF.

We also need to introduce foreign keys to connect related tables. We have already added foreign keys to Attendance, Payment, and Student tables.

Admission\_ID (Foreign Key, INT)

Attendance\_ID (Foreign Key, INT)

To make 8 entities we can add course, enrolment and professor table.

Course

Course\_ID (Primary Key, INT)

Course\_Name (VARCHAR)

Enrolment

Course\_ID(Foreign Key, INT)

Student\_ID(Foreign Key, INT)

Professor

Prof\_ID (Primary Key, INT)

Prof\_Name (VARCHAR)

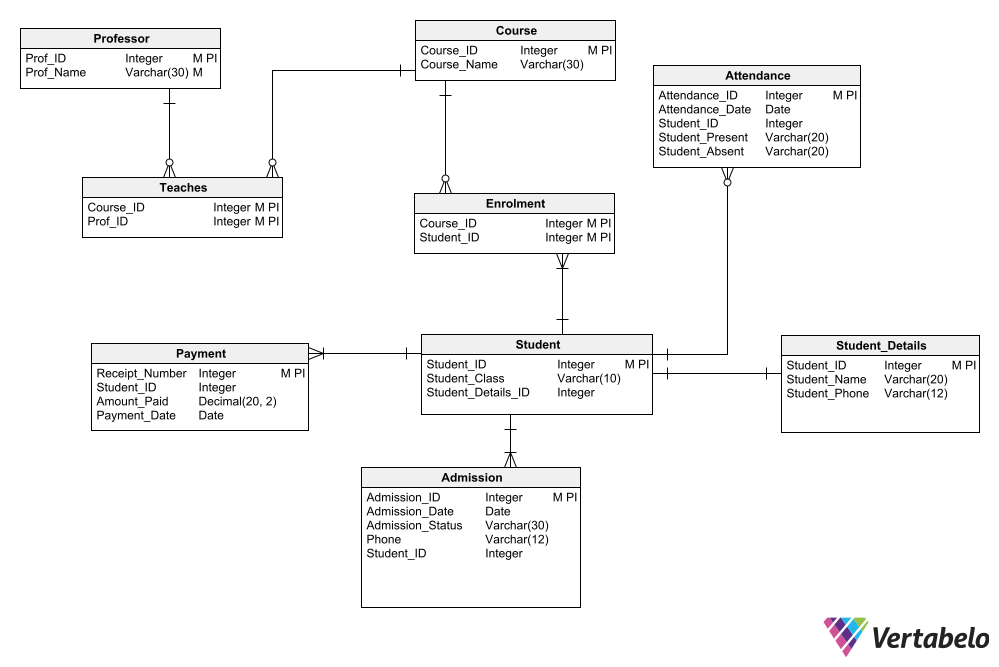
One bridge entity can be added as:

Teaches

Course\_ID(Foreign Key, INT)

Prof\_ID(Foreign Key, INT)

Here is the ER diagram:



g).

Sql commands to populate the tables:

-- Insert three records into the Student table

INSERT INTO Student (Student\_ID, Student\_Class, Student\_Details\_ID)

VALUES (101, '10A', 1),

(102, '10B', 2),

(103, '10C', 3);

-- Insert three records into the Student\_Details table

INSERT INTO Student\_Details (Student\_ID, Student\_Name, Student\_Phone)

VALUES (1, 'Alice Brown', '555-1234'),

(3, 'Bob Green', '555-5678'),

(2, 'Charlie Lee', '555-9012');

-- Insert three records into the Course table

INSERT INTO Course (Course\_ID, Course\_Name)

VALUES (1, 'Mathematics'),

(2, 'English'),

(3, 'Science');

-- Insert three records into the Admission table

INSERT INTO Admission (Admission\_ID, Admission\_Date, Admission\_Status, Phone, Student\_ID)

VALUES (1, '2022-09-01', 'Accepted', '555-1234', 101),

(2, '2022-09-01', 'Accepted', '555-5678', 102),

(3, '2022-09-02', 'Pending', '555-9012', 103);

-- Insert three records into the Attendance table

INSERT INTO Attendance (Attendance\_ID, Attendance\_Date, Student\_ID, Student\_Present, Student\_Absent)

VALUES (1, '2022-09-01', 101, 'Yes', NULL),

(2, '2022-09-01', 102, NULL, 'Yes'),

(3, '2022-09-02', 103, 'Yes', NULL);

-- Insert three records into the Enrolment table

INSERT INTO Enrolment (Course\_ID, Student\_ID)

VALUES (1, 101),

(2, 102),

(3, 103);

-- Insert three records into the Payment table

INSERT INTO Payment (Receipt\_Number, Student\_ID, Amount\_Paid, Payment\_Date)

VALUES (1, 101, 1000.00, '2022-09-05'),

(2, 102, 500.00, '2022-09-06'),

(3, 103, 800.00, '2022-09-07');

-- Insert three records into the Professor table

INSERT INTO Professor (Prof\_ID, Prof\_Name)

VALUES (1, 'John Smith'),

(2, 'Mary Johnson'),

(3, 'David Lee');

-- Insert three records into the Teaches table

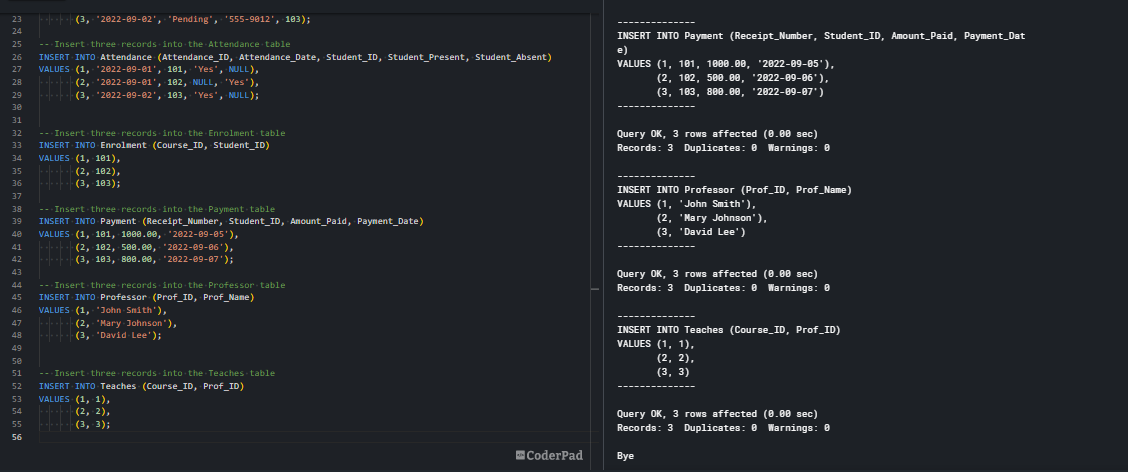
INSERT INTO Teaches (Course\_ID, Prof\_ID)

VALUES (1, 1),

(2, 2),

(3, 3);

Screenshot for execution statements for populating records;



h).

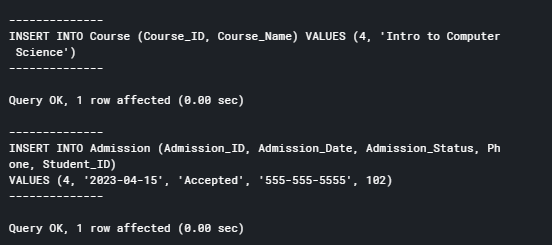
* Two insert statements are:

INSERT INTO Course (Course\_ID, Course\_Name) VALUES (4, 'Intro to Computer Science');

INSERT INTO Admission (Admission\_ID, Admission\_Date, Admission\_Status, Phone, Student\_ID)

VALUES (4, '2023-04-15', 'Accepted', '555-555-5555', 102);

Screenshot of the output:

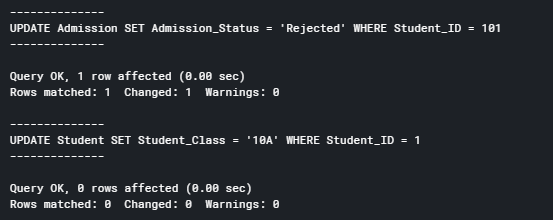


* Two update statements are:

UPDATE Admission SET Admission\_Status = 'Rejected' WHERE Student\_ID = 101;

UPDATE Student SET Student\_Class = '10A' WHERE Student\_ID = 1;

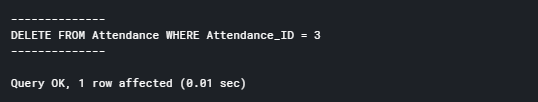
Output:



* One delete statement:

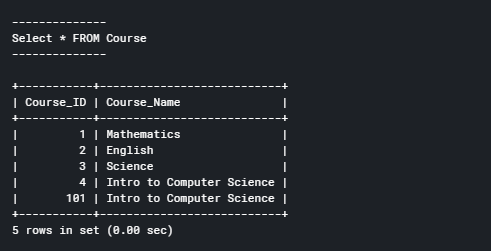
DELETE FROM Attendance WHERE Attendance\_ID = 3;

Output:



* Select \* FROM Course;

Output:



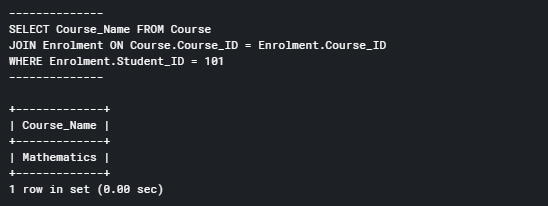
* 2 join statements are:

SELECT Course\_Name FROM Course

JOIN Enrolment ON Course.Course\_ID = Enrolment.Course\_ID

WHERE Enrolment.Student\_ID = 101;

Output:



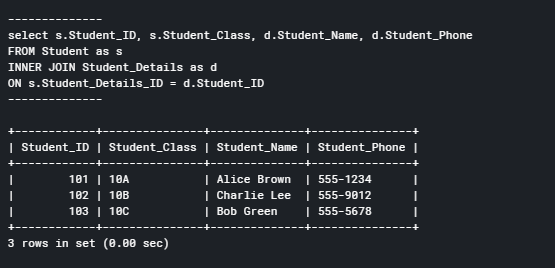
SELECT s.Student\_ID, s.Student\_Class, d.Student\_Name, d.Student\_Phone

FROM Student as s

INNER JOIN Student\_Details as d

ON s.Student\_Details\_ID = d.Student\_ID;

Output:

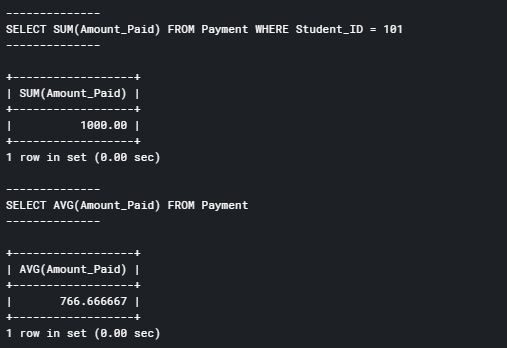


* 2 summary statements are:

SELECT SUM(Amount\_Paid) FROM Payment WHERE Student\_ID = 101;

SELECT AVG(Amount\_Paid) FROM Payment;

Output:



* Multi table query as name of course taught by “John Smith”.

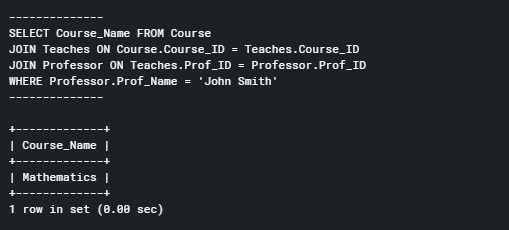
SELECT Course\_Name FROM Course

JOIN Teaches ON Course.Course\_ID = Teaches.Course\_ID

JOIN Professor ON Teaches.Prof\_ID = Professor.Prof\_ID

WHERE Professor.Prof\_Name = 'John Smith';

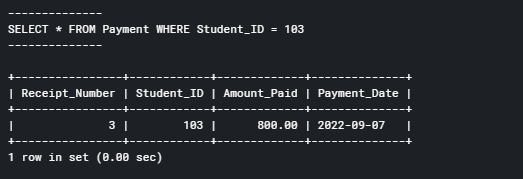
Output:



* A sql query:

SELECT \* FROM Payment WHERE Student\_ID = 103;

Output:



s). **Activity log**

29thof march 2023 – Worked on Relational Database Design Process

30thof march 2023 – Worked on Relational Database Design

31thof march 2023 – worked on Data Definition Language Scripts

1thof april 2023- Worked on Data Manipulation Language Scripts

2nd of april 2023 - Preparation of report