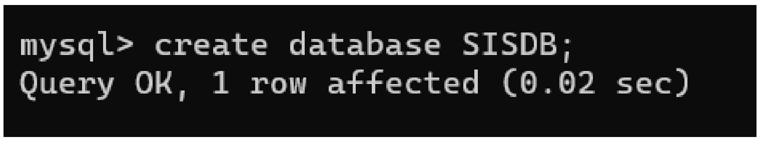
Assignment -2

Student Information System (SIS)

Submitted by : Mridul Bhardwaj

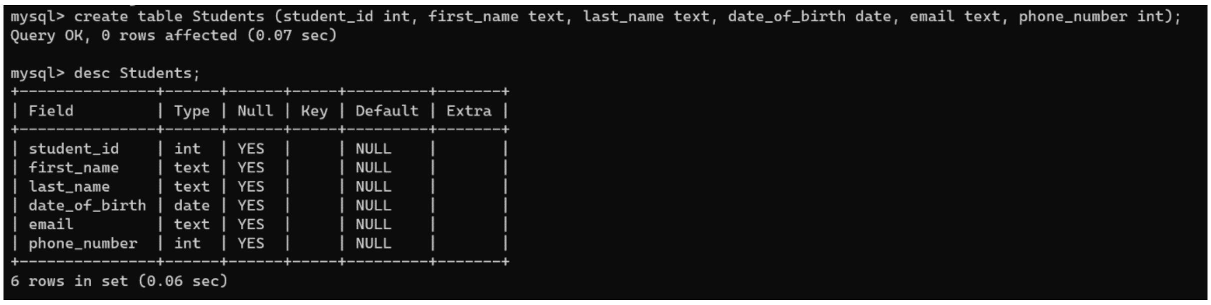
Task 1. Database Design:

1. Create the database named "SISDB"



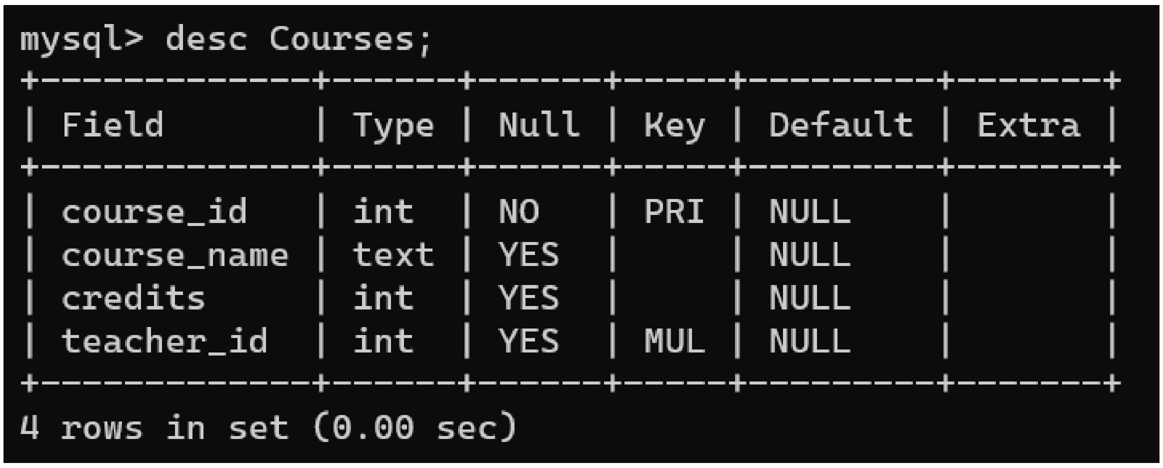
```
mysql> create database SISDB;
Query OK, 1 row affected (0.02 sec)
```

2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.
   a. Students



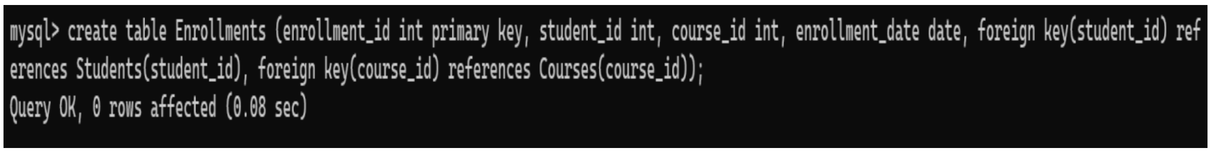
```
mysql> create table Students (student_id int, first_name text, last_name text, date_of_birth date, email text, phone_number int);
Query OK, 0 rows affected (0.07 sec)

mysql> desc Students;
+---------------+------+------+-----+---------+-------+
| Field         | Type | Null | Key | Default | Extra |
+---------------+------+------+-----+---------+-------+
| student_id    | int  | YES  |     | NULL    |       |
| first_name    | text | YES  |     | NULL    |       |
| last_name     | text | YES  |     | NULL    |       |
| date_of_birth | date | YES  |     | NULL    |       |
| email         | text | YES  |     | NULL    |       |
| phone_number  | int  | YES  |     | NULL    |       |
+---------------+------+------+-----+---------+-------+
6 rows in set (0.06 sec)
```

   b. Courses

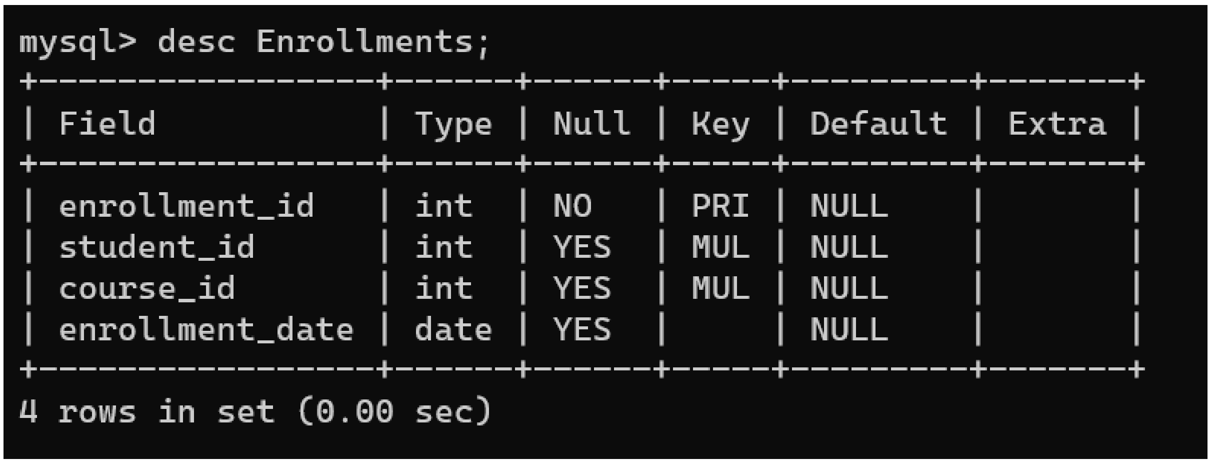


```
mysql> desc Courses;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| course_id   | int  | NO   | PRI | NULL    |       |
| course_name | text | YES  |     | NULL    |       |
| credits     | int  | YES  |     | NULL    |       |
| teacher_id  | int  | YES  | MUL | NULL    |       |
+-------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

   c. Enrollments

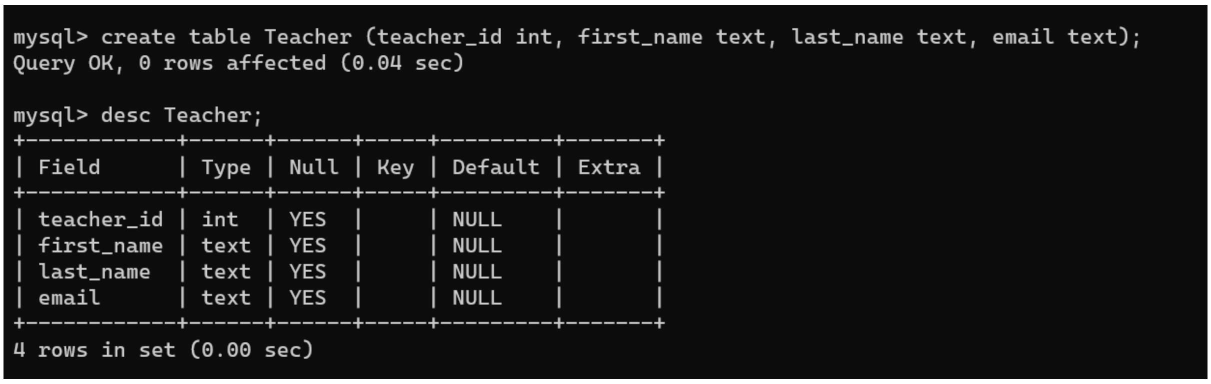


```
mysql> create table Enrollments (enrollment_id int primary key, student_id int, course_id int, enrollment_date date, foreign key(student_id) ref
erences Students(student_id), foreign key(course_id) references Courses(course_id));
Query OK, 0 rows affected (0.08 sec)
```



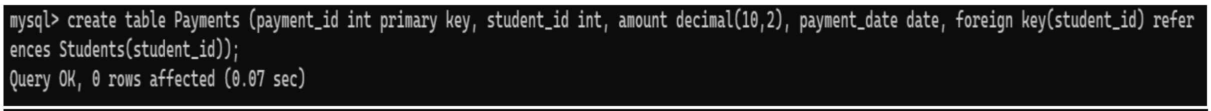
```
mysql> desc Enrollments;
+-----------------+------+------+-----+---------+-------+
| Field           | Type | Null | Key | Default | Extra |
+-----------------+------+------+-----+---------+-------+
| enrollment_id   | int  | NO   | PRI | NULL    |       |
| student_id      | int  | YES  | MUL | NULL    |       |
| course_id       | int  | YES  | MUL | NULL    |       |
| enrollment_date | date | YES  |     | NULL    |       |
+-----------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

d. Teacher

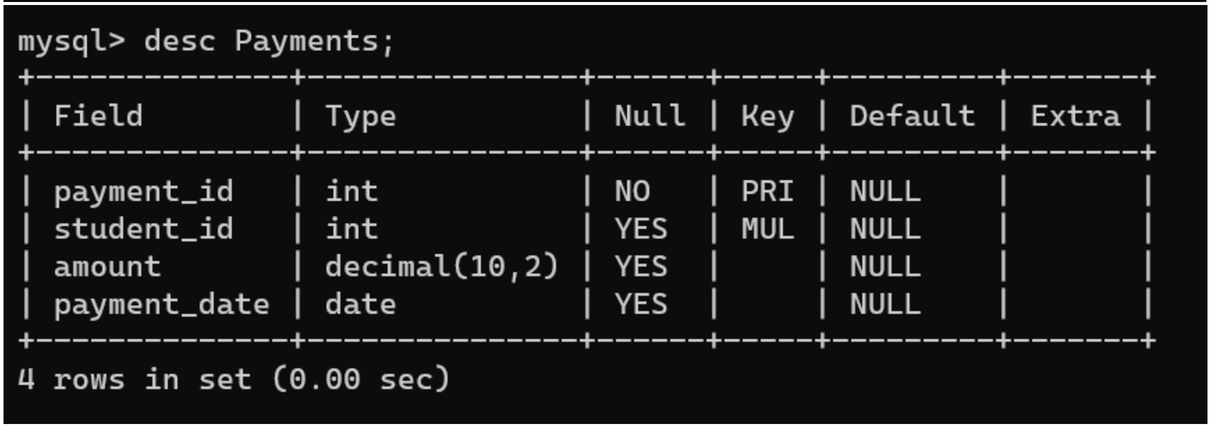


```
mysql> create table Teacher (teacher_id int, first_name text, last_name text, email text);
Query OK, 0 rows affected (0.04 sec)

mysql> desc Teacher;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| teacher_id | int  | YES  |     | NULL    |       |
| first_name | text | YES  |     | NULL    |       |
| last_name  | text | YES  |     | NULL    |       |
| email      | text | YES  |     | NULL    |       |
+------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

e. Payments

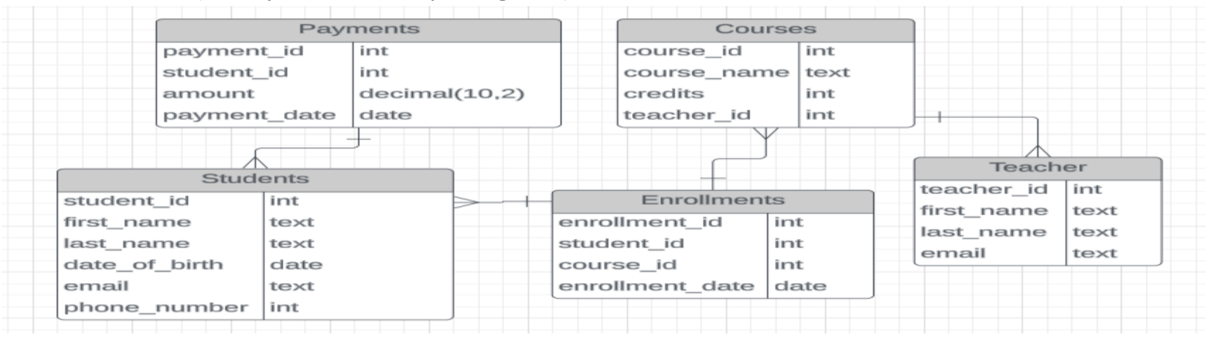


```
mysql> create table Payments (payment_id int primary key, student_id int, amount decimal(10,2), payment_date date, foreign key(student_id) refer
ences Students(student_id));
Query OK, 0 rows affected (0.07 sec)
```

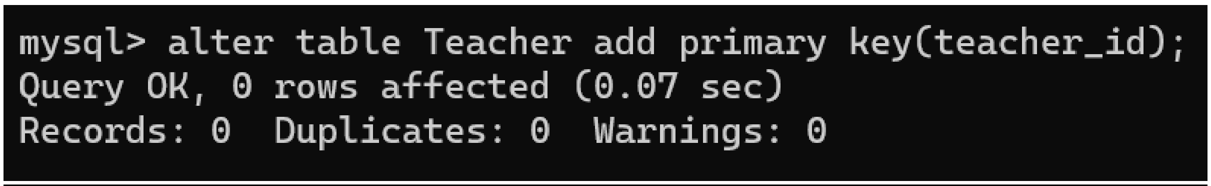


```
mysql> desc Payments;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| payment_id   | int           | NO   | PRI | NULL    |       |
| student_id   | int           | YES  | MUL | NULL    |       |
| amount       | decimal(10,2) | YES  |     | NULL    |       |
| payment_date | date          | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

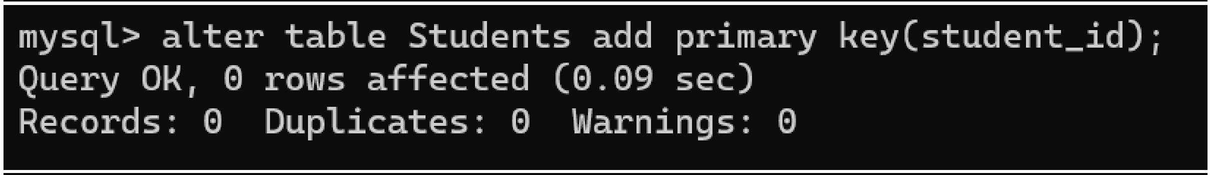
3. Create an ERD (Entity Relationship Diagram) for the database.



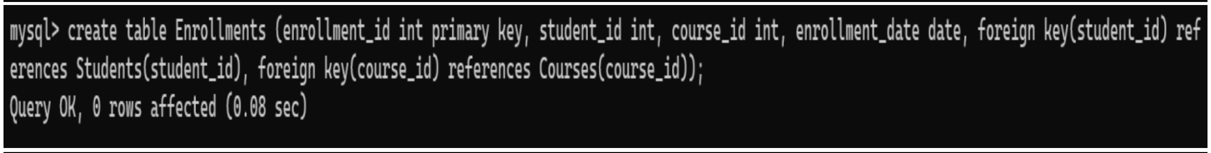
4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



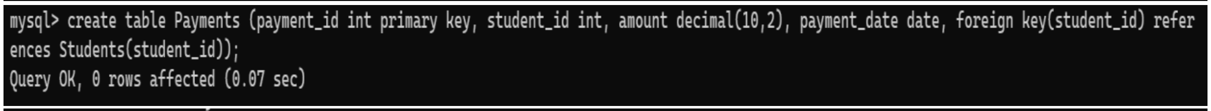
```
mysql> alter table Teacher add primary key(teacher_id);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



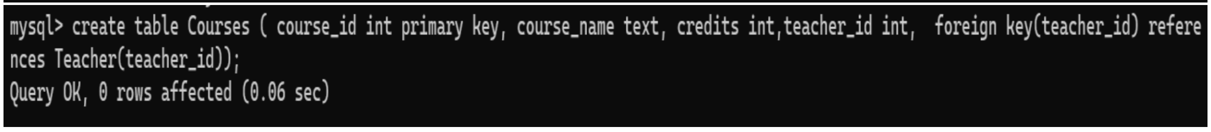
```
mysql> alter table Students add primary key(student_id);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



```
mysql> create table Enrollments (enrollment_id int primary key, student_id int, course_id int, enrollment_date date, foreign key(student_id) ref
erences Students(student_id), foreign key(course_id) references Courses(course_id));
Query OK, 0 rows affected (0.08 sec)
```

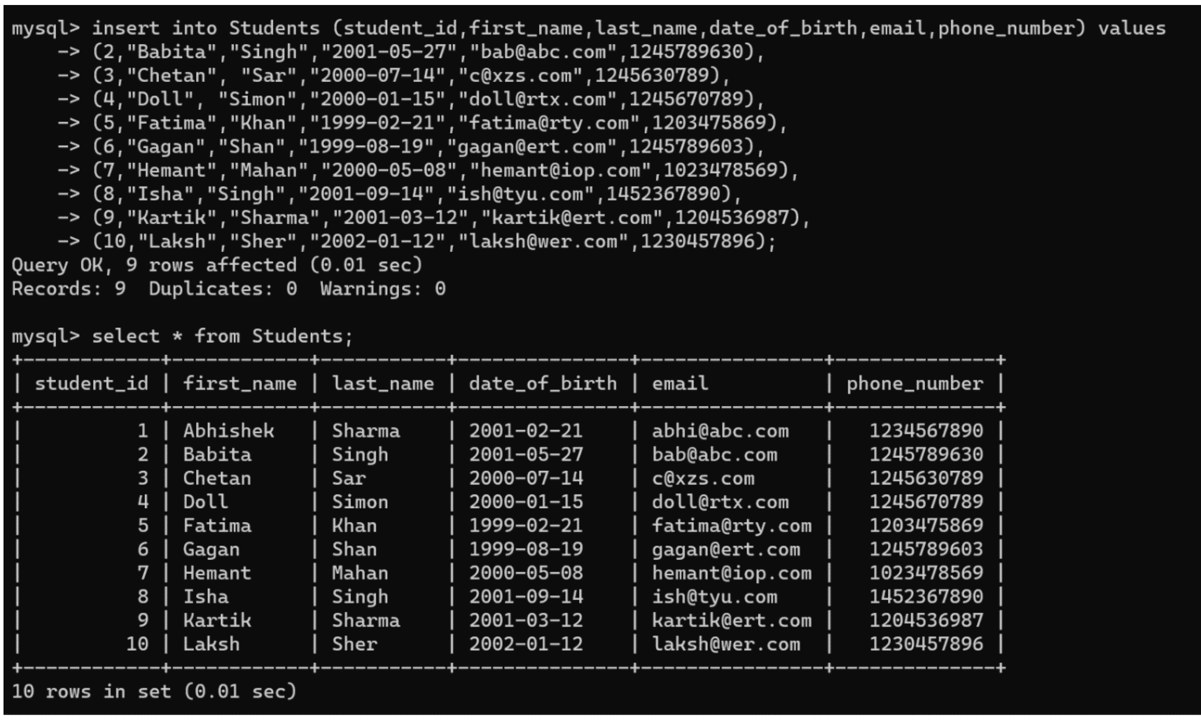


```
mysql> create table Payments (payment_id int primary key, student_id int, amount decimal(10,2), payment_date date, foreign key(student_id) refer
ences Students(student_id));
Query OK, 0 rows affected (0.07 sec)
```



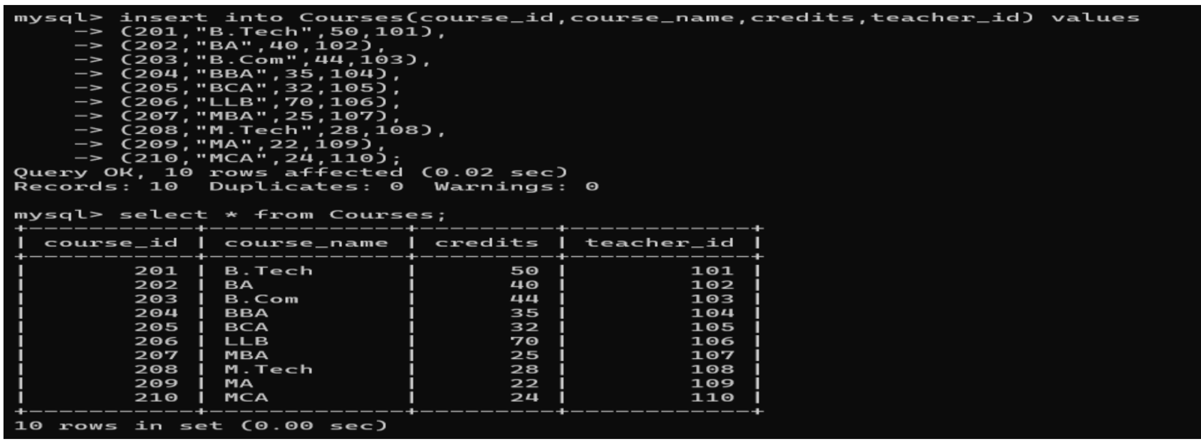
```
mysql> create table Courses ( course_id int primary key, course_name text, credits int,teacher_id int,  foreign key(teacher_id) refere
nces Teacher(teacher_id));
Query OK, 0 rows affected (0.06 sec)
```

5. Insert at least 10 sample records into each of the following tables.
   i. Students



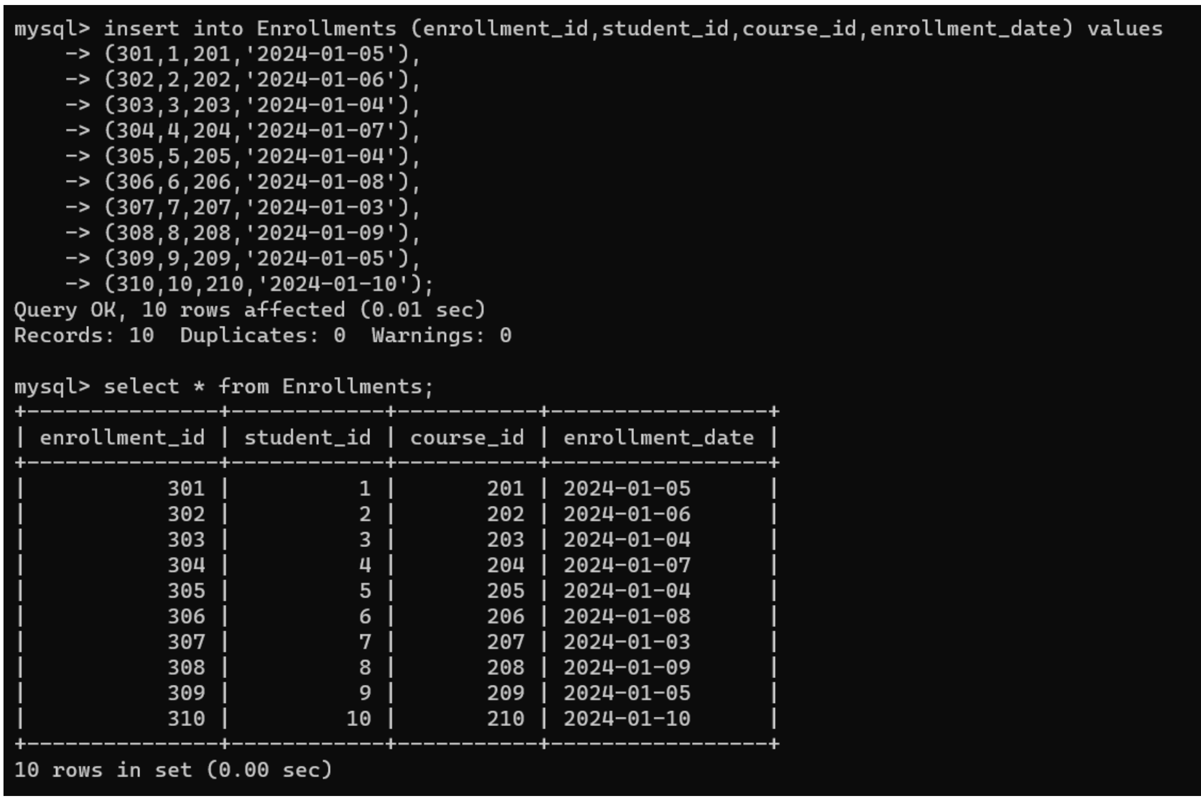
```
mysql> insert into Students (student_id,first_name,last_name,date_of_birth,email,phone_number) values
    -> (2,"Babita","Singh","2001-05-27","bab@abc.com",1245789630),
    -> (3,"Chetan", "Sar","2000-07-14","c@xzs.com",1245630789),
    -> (4,"Doll", "Simon","2000-01-15","doll@rtx.com",1245670789),
    -> (5,"Fatima","Khan","1999-02-21","fatima@rty.com",1203475869),
    -> (6,"Gagan","Shan","1999-08-19","gagan@ert.com",1245789603),
    -> (7,"Hemant","Mahan","2000-05-08","hemant@iop.com",1023478569),
    -> (8,"Isha","Singh","2001-09-14","ish@tyu.com",1452367890),
    -> (9,"Kartik","Sharma","2001-03-12","kartik@ert.com",1204536987),
    -> (10,"Laksh","Sher","2002-01-12","laksh@wer.com",1230457896);
Query OK, 9 rows affected (0.01 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> select * from Students;
+------------+------------+-----------+---------------+----------------+--------------+
| student_id | first_name | last_name | date_of_birth | email          | phone_number |
+------------+------------+-----------+---------------+----------------+--------------+
|          1 | Abhishek   | Sharma    | 2001-02-21    | abhi@abc.com   |   1234567890 |
|          2 | Babita     | Singh     | 2001-05-27    | bab@abc.com    |   1245789630 |
|          3 | Chetan     | Sar       | 2000-07-14    | c@xzs.com      |   1245630789 |
|          4 | Doll       | Simon     | 2000-01-15    | doll@rtx.com   |   1245670789 |
|          5 | Fatima     | Khan      | 1999-02-21    | fatima@rty.com |   1203475869 |
|          6 | Gagan      | Shan      | 1999-08-19    | gagan@ert.com  |   1245789603 |
|          7 | Hemant     | Mahan     | 2000-05-08    | hemant@iop.com |   1023478569 |
|          8 | Isha       | Singh     | 2001-09-14    | ish@tyu.com    |   1452367890 |
|          9 | Kartik     | Sharma    | 2001-03-12    | kartik@ert.com |   1204536987 |
|         10 | Laksh      | Sher      | 2002-01-12    | laksh@wer.com  |   1230457896 |
+------------+------------+-----------+---------------+----------------+--------------+
10 rows in set (0.01 sec)
```

   ii. Courses



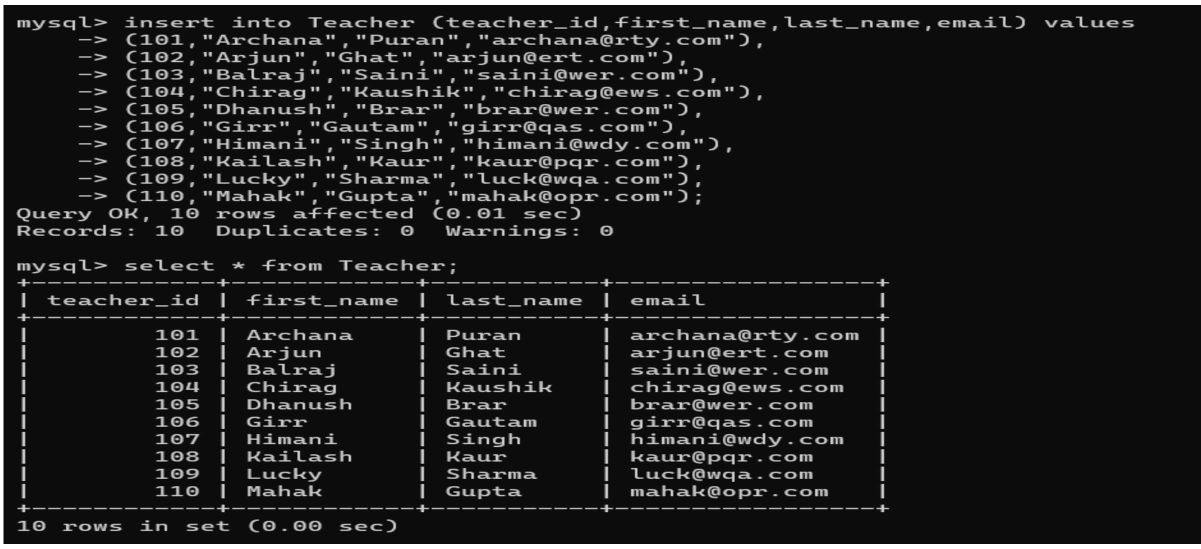
```
mysql> insert into Courses(course_id,course_name,credits,teacher_id) values
    -> (201,"B.Tech",50,101),
    -> (202,"BA",40,102),
    -> (203,"B.Com",44,103),
    -> (204,"BBA",35,104),
    -> (205,"BCA",32,105),
    -> (206,"LLB",70,106),
    -> (207,"MBA",25,107),
    -> (208,"M.Tech",28,108),
    -> (209,"MA",22,109),
    -> (210,"MCA",24,110);
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Courses;
+-----------+-------------+---------+------------+
| course_id | course_name | credits | teacher_id |
+-----------+-------------+---------+------------+
|       201 | B.Tech      |      50 |        101 |
|       202 | BA          |      40 |        102 |
|       203 | B.Com       |      44 |        103 |
|       204 | BBA         |      35 |        104 |
|       205 | BCA         |      32 |        105 |
|       206 | LLB         |      70 |        106 |
|       207 | MBA         |      25 |        107 |
|       208 | M.Tech      |      28 |        108 |
|       209 | MA          |      22 |        109 |
|       210 | MCA         |      24 |        110 |
+-----------+-------------+---------+------------+
10 rows in set (0.00 sec)
```

iii. Enrollments



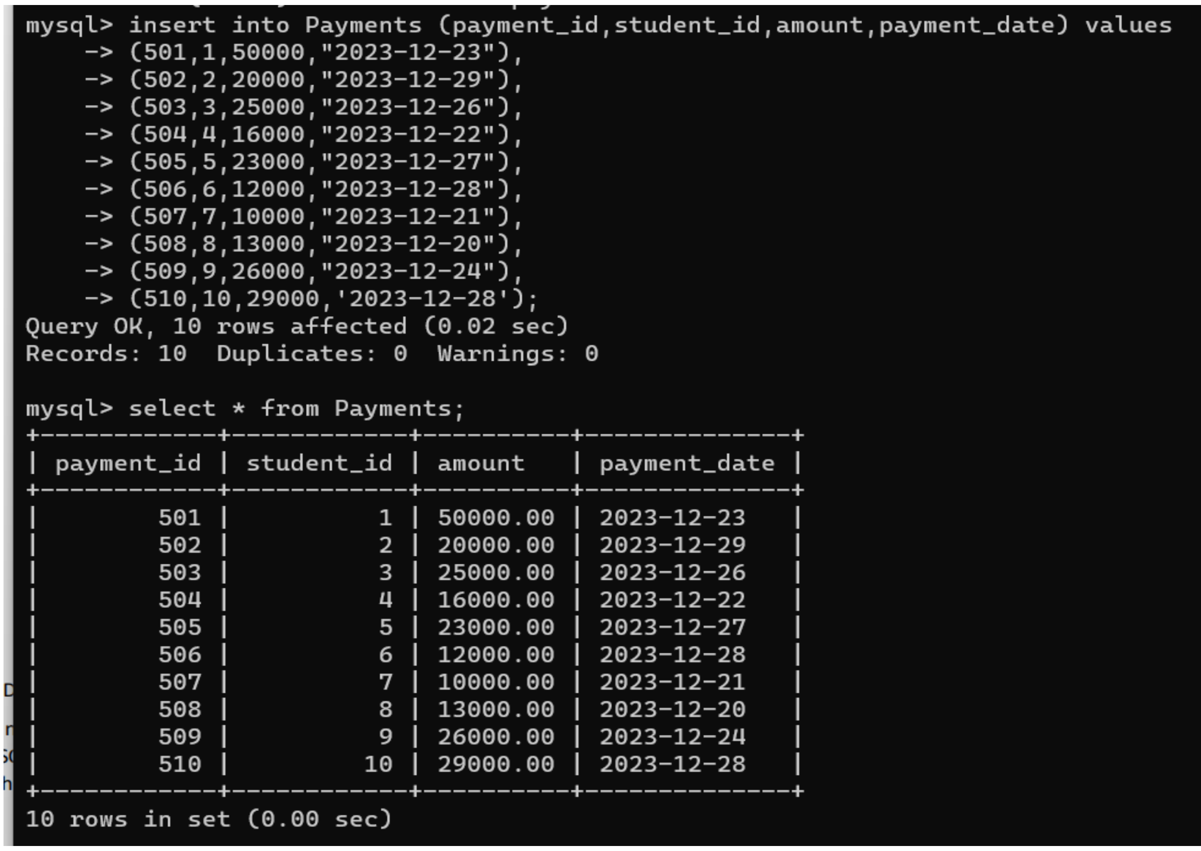
```
mysql> insert into Enrollments (enrollment_id,student_id,course_id,enrollment_date) values
    -> (301,1,201,'2024-01-05'),
    -> (302,2,202,'2024-01-06'),
    -> (303,3,203,'2024-01-04'),
    -> (304,4,204,'2024-01-07'),
    -> (305,5,205,'2024-01-04'),
    -> (306,6,206,'2024-01-08'),
    -> (307,7,207,'2024-01-03'),
    -> (308,8,208,'2024-01-09'),
    -> (309,9,209,'2024-01-05'),
    -> (310,10,210,'2024-01-10');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           301 |          1 |       201 | 2024-01-05      |
|           302 |          2 |       202 | 2024-01-06      |
|           303 |          3 |       203 | 2024-01-04      |
|           304 |          4 |       204 | 2024-01-07      |
|           305 |          5 |       205 | 2024-01-04      |
|           306 |          6 |       206 | 2024-01-08      |
|           307 |          7 |       207 | 2024-01-03      |
|           308 |          8 |       208 | 2024-01-09      |
|           309 |          9 |       209 | 2024-01-05      |
|           310 |         10 |       210 | 2024-01-10      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

iv. Teacher



```
mysql> insert into Teacher (teacher_id,first_name,last_name,email) values
    -> (101,"Archana","Puran","archana@rty.com"),
    -> (102,"Arjun","Ghat","arjun@ert.com"),
    -> (103,"Balraj","Saini","saini@wer.com"),
    -> (104,"Chirag","Kaushik","chirag@ews.com"),
    -> (105,"Dhanush","Brar","brar@wer.com"),
    -> (106,"Girr","Gautam","girr@qas.com"),
    -> (107,"Himani","Singh","himani@wdy.com"),
    -> (108,"Kailash","Kaur","kaur@pqr.com"),
    -> (109,"Lucky","Sharma","luck@wqa.com"),
    -> (110,"Mahak","Gupta","mahak@opr.com");
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Teacher;
+------------+------------+-----------+-----------------+
| teacher_id | first_name | last_name | email           |
+------------+------------+-----------+-----------------+
|        101 | Archana    | Puran     | archana@rty.com |
|        102 | Arjun      | Ghat      | arjun@ert.com   |
|        103 | Balraj     | Saini     | saini@wer.com   |
|        104 | Chirag     | Kaushik   | chirag@ews.com  |
|        105 | Dhanush    | Brar      | brar@wer.com    |
|        106 | Girr       | Gautam    | girr@qas.com    |
|        107 | Himani     | Singh     | himani@wdy.com  |
|        108 | Kailash    | Kaur      | kaur@pqr.com    |
|        109 | Lucky      | Sharma    | luck@wqa.com    |
|        110 | Mahak      | Gupta     | mahak@opr.com   |
+------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

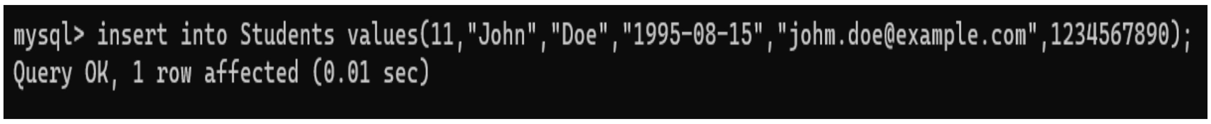
v. Payments



```
mysql> insert into Payments (payment_id,student_id,amount,payment_date) values
    -> (501,1,50000,"2023-12-23"),
    -> (502,2,20000,"2023-12-29"),
    -> (503,3,25000,"2023-12-26"),
    -> (504,4,16000,"2023-12-22"),
    -> (505,5,23000,"2023-12-27"),
    -> (506,6,12000,"2023-12-28"),
    -> (507,7,10000,"2023-12-21"),
    -> (508,8,13000,"2023-12-20"),
    -> (509,9,26000,"2023-12-24"),
    -> (510,10,29000,'2023-12-28');
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Payments;
+------------+------------+----------+--------------+
| payment_id | student_id | amount   | payment_date |
+------------+------------+----------+--------------+
|        501 |          1 | 50000.00 | 2023-12-23   |
|        502 |          2 | 20000.00 | 2023-12-29   |
|        503 |          3 | 25000.00 | 2023-12-26   |
|        504 |          4 | 16000.00 | 2023-12-22   |
|        505 |          5 | 23000.00 | 2023-12-27   |
|        506 |          6 | 12000.00 | 2023-12-28   |
|        507 |          7 | 10000.00 | 2023-12-21   |
|        508 |          8 | 13000.00 | 2023-12-20   |
|        509 |          9 | 26000.00 | 2023-12-24   |
|        510 |         10 | 29000.00 | 2023-12-28   |
+------------+------------+----------+--------------+
10 rows in set (0.00 sec)
```

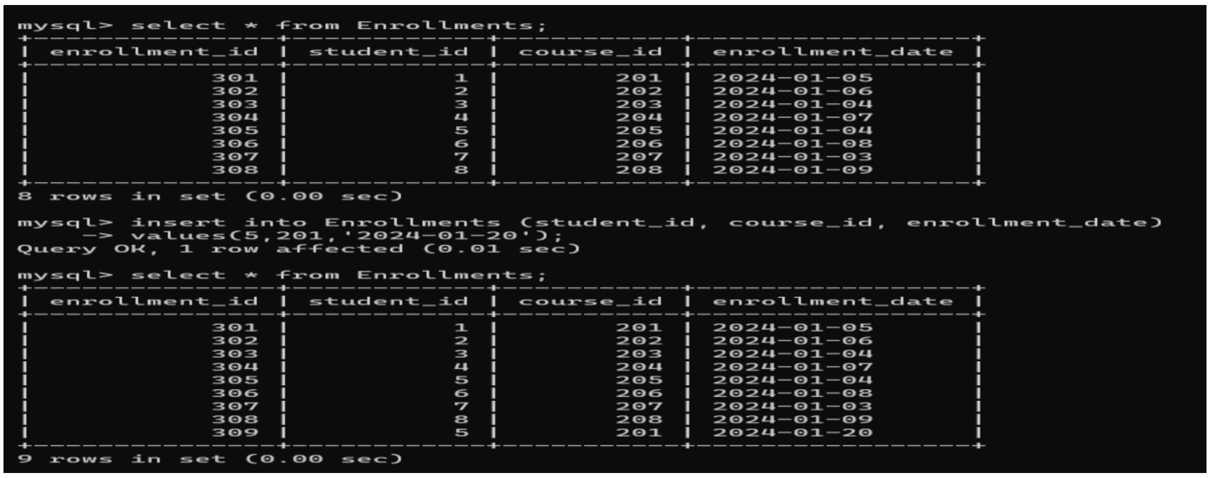
Tasks 2: Select, Where, Between, AND, LIKE:

1. Write an SQL query to insert a new student into the "Students" table with the following details:
   a. First Name: John
   b. Last Name: Doe
   c. Date of Birth: 1995-08-15
   d. Email: john.doe@example.com
   e. Phone Number: 1234567890



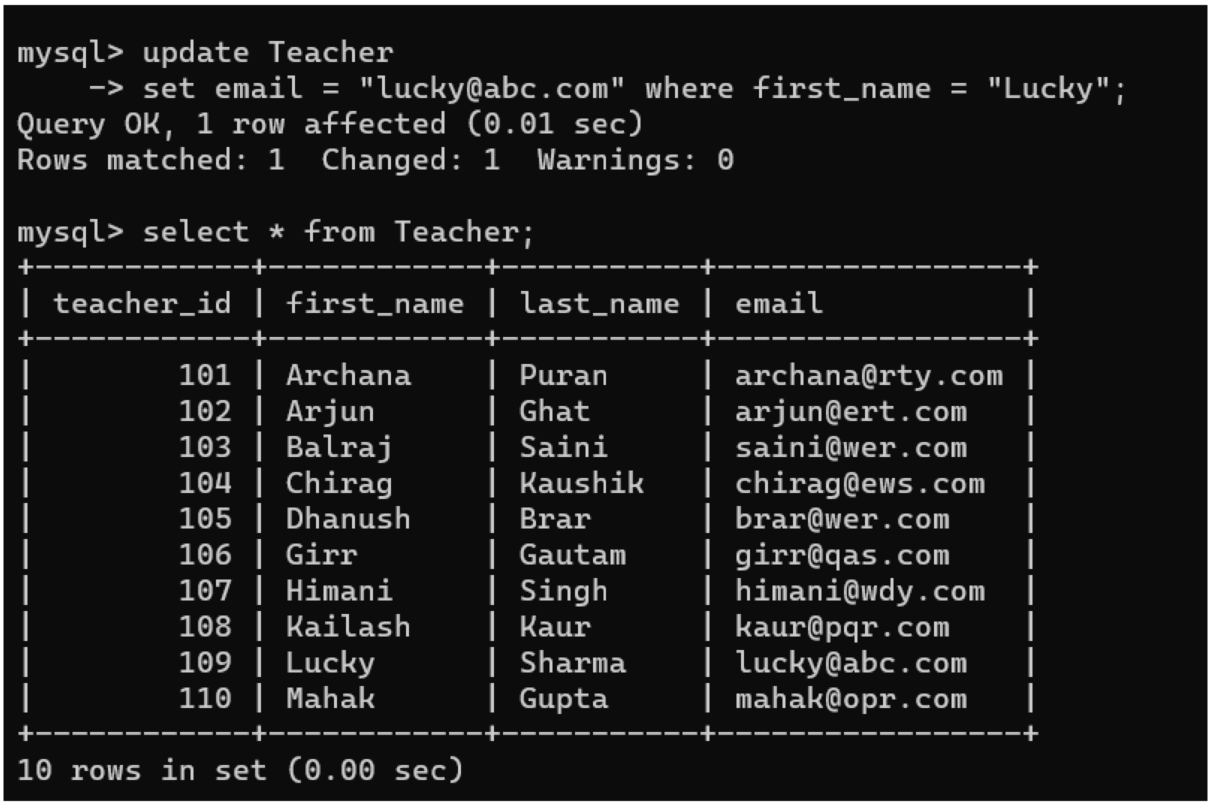
```
mysql> insert into Students values(11,"John","Doe","1995-08-15","johm.doe@example.com",1234567890);
Query OK, 1 row affected (0.01 sec)
```

2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.



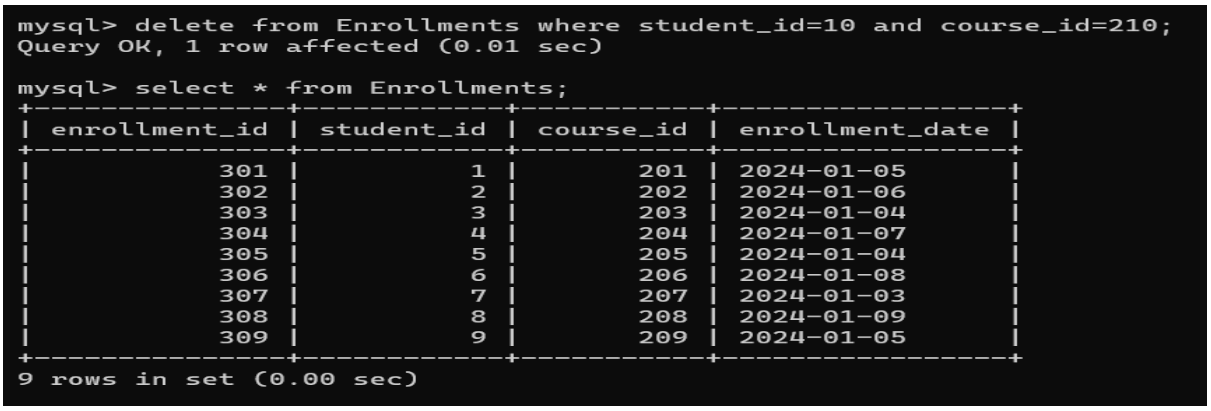
```
mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           301 |          1 |       201 | 2024-01-05      |
|           302 |          2 |       202 | 2024-01-06      |
|           303 |          3 |       203 | 2024-01-04      |
|           304 |          4 |       204 | 2024-01-07      |
|           305 |          5 |       205 | 2024-01-04      |
|           306 |          6 |       206 | 2024-01-08      |
|           307 |          7 |       207 | 2024-01-03      |
|           308 |          8 |       208 | 2024-01-09      |
+---------------+------------+-----------+-----------------+
8 rows in set (0.00 sec)

mysql> insert into Enrollments (student_id, course_id, enrollment_date)
    -> values(5,201,'2024-01-20');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           301 |          1 |       201 | 2024-01-05      |
|           302 |          2 |       202 | 2024-01-06      |
|           303 |          3 |       203 | 2024-01-04      |
|           304 |          4 |       204 | 2024-01-07      |
|           305 |          5 |       205 | 2024-01-04      |
|           306 |          6 |       206 | 2024-01-08      |
|           307 |          7 |       207 | 2024-01-03      |
|           308 |          8 |       208 | 2024-01-09      |
|           309 |          5 |       201 | 2024-01-20      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)
```

3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.



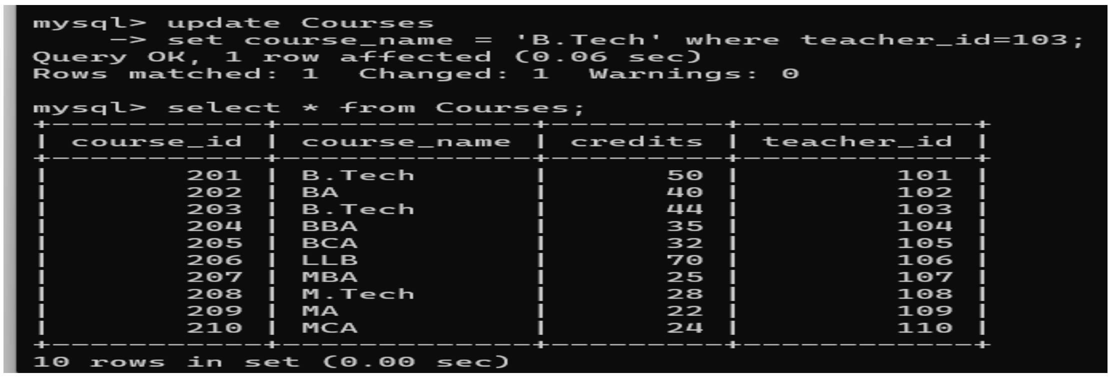
```
mysql> update Teacher
    -> set email = "lucky@abc.com" where first_name = "Lucky";
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Teacher;
+------------+------------+-----------+-----------------+
| teacher_id | first_name | last_name | email           |
+------------+------------+-----------+-----------------+
|        101 | Archana    | Puran     | archana@rty.com |
|        102 | Arjun      | Ghat      | arjun@ert.com   |
|        103 | Balraj     | Saini     | saini@wer.com   |
|        104 | Chirag     | Kaushik   | chirag@ews.com  |
|        105 | Dhanush    | Brar      | brar@wer.com    |
|        106 | Girr       | Gautam    | girr@qas.com    |
|        107 | Himani     | Singh     | himani@wdy.com  |
|        108 | Kailash    | Kaur      | kaur@pqr.com    |
|        109 | Lucky      | Sharma    | lucky@abc.com   |
|        110 | Mahak      | Gupta     | mahak@opr.com   |
+------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.



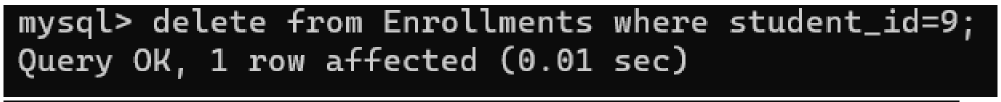
```
mysql> delete from Enrollments where student_id=10 and course_id=210;
Query OK, 1 row affected (0.01 sec)

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| enrollment_id | student_id | course_id | enrollment_date |
+---------------+------------+-----------+-----------------+
|           301 |          1 |       201 | 2024-01-05      |
|           302 |          2 |       202 | 2024-01-06      |
|           303 |          3 |       203 | 2024-01-04      |
|           304 |          4 |       204 | 2024-01-07      |
|           305 |          5 |       205 | 2024-01-04      |
|           306 |          6 |       206 | 2024-01-08      |
|           307 |          7 |       207 | 2024-01-03      |
|           308 |          8 |       208 | 2024-01-09      |
|           309 |          9 |       209 | 2024-01-05      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)
```

5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.

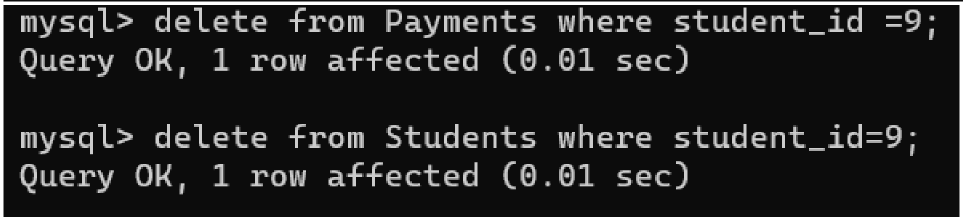


```
mysql> update Courses
    -> set course_name = 'B.Tech' where teacher_id=103;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Courses;
+-----------+-------------+---------+------------+
| course_id | course_name | credits | teacher_id |
+-----------+-------------+---------+------------+
|       201 | B.Tech      |      50 |        101 |
|       202 | BA          |      40 |        102 |
|       203 | B.Tech      |      44 |        103 |
|       204 | BBA         |      35 |        104 |
|       205 | BCA         |      32 |        105 |
|       206 | LLB         |      70 |        106 |
|       207 | MBA         |      25 |        107 |
|       208 | M.Tech      |      28 |        108 |
|       209 | MA          |      22 |        109 |
|       210 | MCA         |      24 |        110 |
+-----------+-------------+---------+------------+
10 rows in set (0.00 sec)
```

6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.

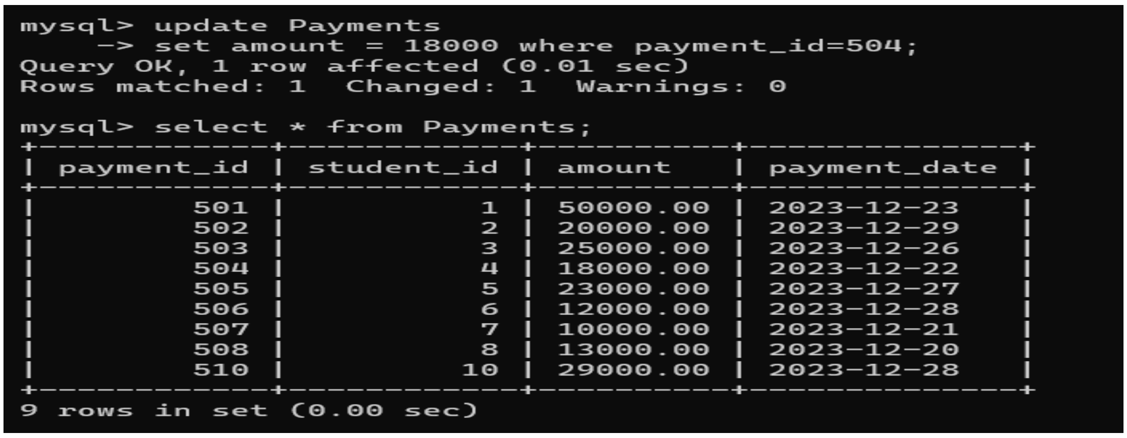


```
mysql> delete from Enrollments where student_id=9;
Query OK, 1 row affected (0.01 sec)
```



```
mysql> delete from Payments where student_id =9;
Query OK, 1 row affected (0.01 sec)

mysql> delete from Students where student_id=9;
Query OK, 1 row affected (0.01 sec)
```

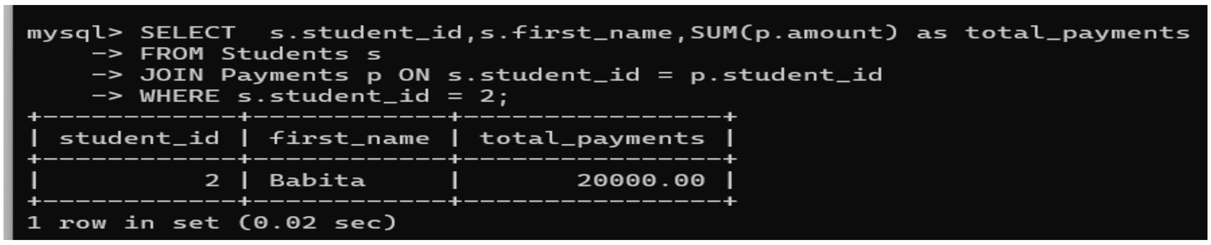
7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.



```
mysql> update Payments
    -> set amount = 18000 where payment_id=504;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Payments;
+------------+------------+----------+--------------+
| payment_id | student_id | amount   | payment_date |
+------------+------------+----------+--------------+
|        501 |          1 | 50000.00 | 2023-12-23   |
|        502 |          2 | 20000.00 | 2023-12-29   |
|        503 |          3 | 25000.00 | 2023-12-26   |
|        504 |          4 | 18000.00 | 2023-12-22   |
|        505 |          5 | 23000.00 | 2023-12-27   |
|        506 |          6 | 12000.00 | 2023-12-28   |
|        507 |          7 | 10000.00 | 2023-12-21   |
|        508 |          8 | 13000.00 | 2023-12-20   |
|        510 |         10 | 29000.00 | 2023-12-28   |
+------------+------------+----------+--------------+
9 rows in set (0.00 sec)
```

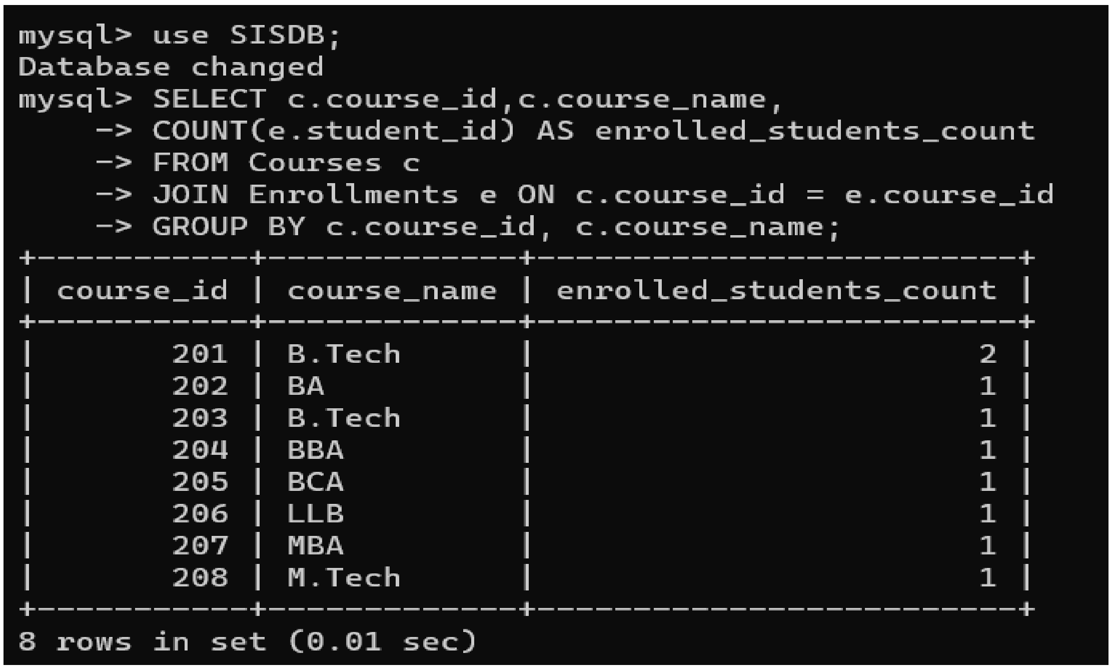
Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

1. Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.



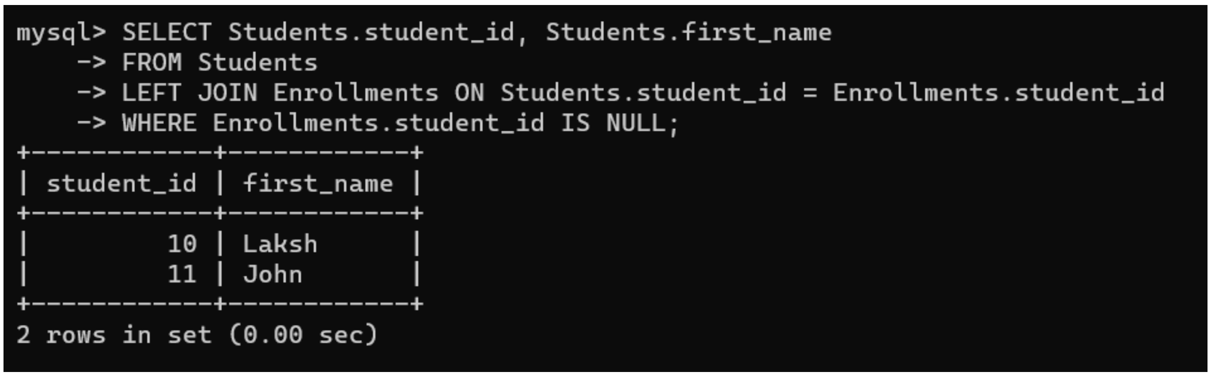
```
mysql> SELECT  s.student_id,s.first_name,SUM(p.amount) as total_payments
    -> FROM Students s
    -> JOIN Payments p ON s.student_id = p.student_id
    -> WHERE s.student_id = 2;
+------------+------------+----------------+
| student_id | first_name | total_payments |
+------------+------------+----------------+
|          2 | Babita     |       20000.00 |
+------------+------------+----------------+
1 row in set (0.02 sec)
```

2. Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.



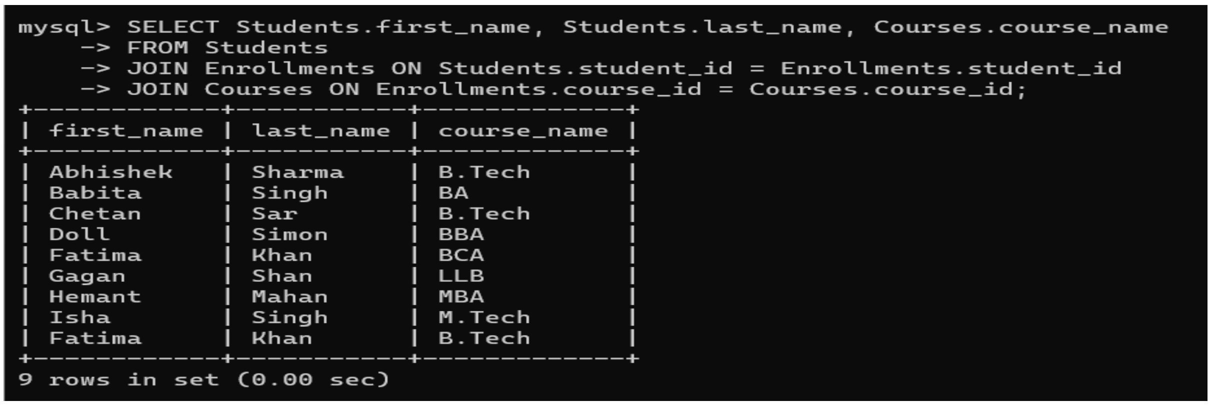
```
mysql> use SISDB;
Database changed
mysql> SELECT c.course_id,c.course_name,
    -> COUNT(e.student_id) AS enrolled_students_count
    -> FROM Courses c
    -> JOIN Enrollments e ON c.course_id = e.course_id
    -> GROUP BY c.course_id, c.course_name;
+-----------+-------------+-------------------------+
| course_id | course_name | enrolled_students_count |
+-----------+-------------+-------------------------+
|       201 | B.Tech      |                       2 |
|       202 | BA          |                       1 |
|       203 | B.Tech      |                       1 |
|       204 | BBA         |                       1 |
|       205 | BCA         |                       1 |
|       206 | LLB         |                       1 |
|       207 | MBA         |                       1 |
|       208 | M.Tech      |                       1 |
+-----------+-------------+-------------------------+
8 rows in set (0.01 sec)
```

3. Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.



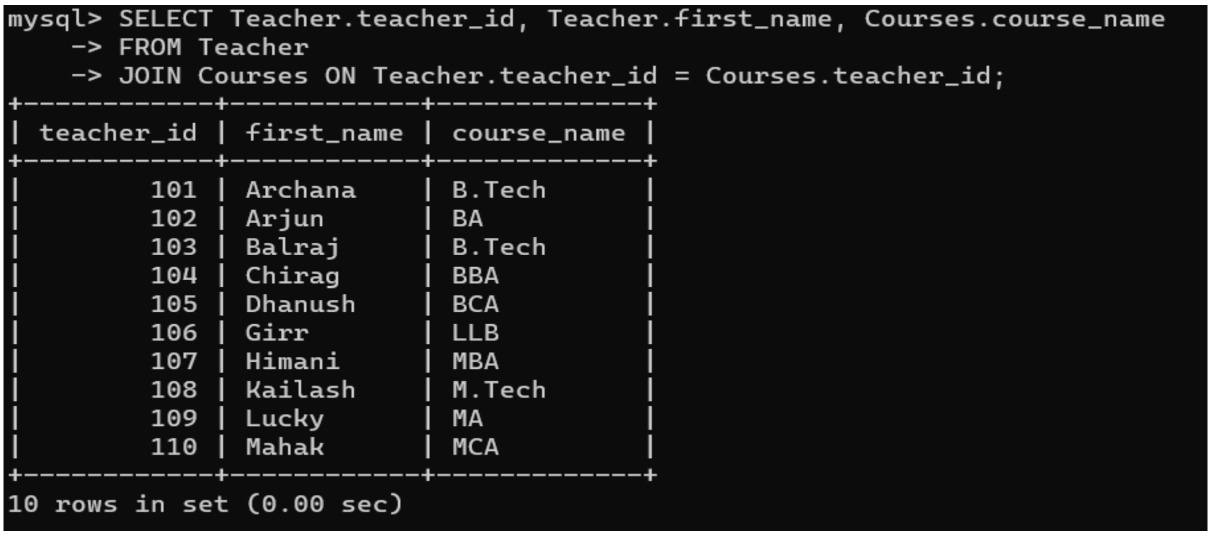
```
mysql> SELECT Students.student_id, Students.first_name
    -> FROM Students
    -> LEFT JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> WHERE Enrollments.student_id IS NULL;
+------------+------------+
| student_id | first_name |
+------------+------------+
|         10 | Laksh      |
|         11 | John       |
+------------+------------+
2 rows in set (0.00 sec)
```

4. Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.



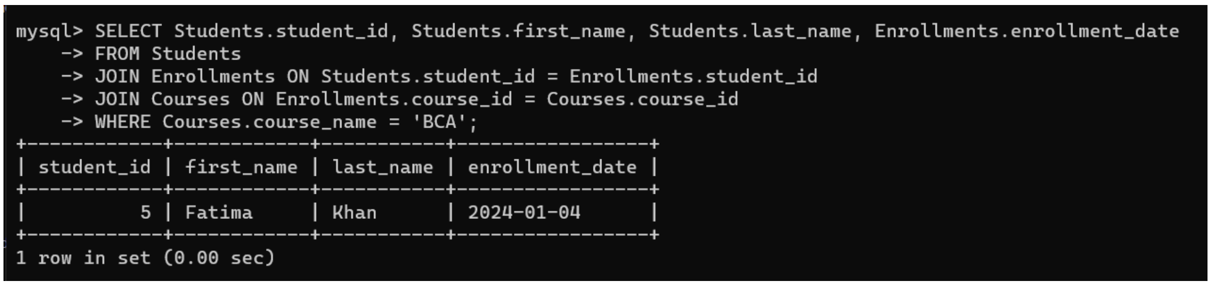
```
mysql> SELECT Students.first_name, Students.last_name, Courses.course_name
    -> FROM Students
    -> JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> JOIN Courses ON Enrollments.course_id = Courses.course_id;
+------------+-----------+-------------+
| first_name | last_name | course_name |
+------------+-----------+-------------+
| Abhishek   | Sharma    | B.Tech      |
| Babita     | Singh     | BA          |
| Chetan     | Sar       | B.Tech      |
| Doll       | Simon     | BBA         |
| Fatima     | Khan      | BCA         |
| Gagan      | Shan      | LLB         |
| Hemant     | Mahan     | MBA         |
| Isha       | Singh     | M.Tech      |
| Fatima     | Khan      | B.Tech      |
+------------+-----------+-------------+
9 rows in set (0.00 sec)
```

5. Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.



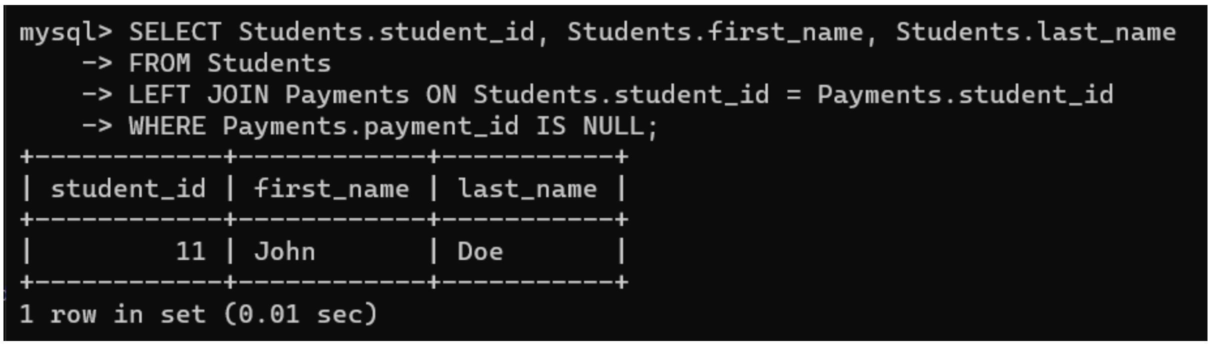
```
mysql> SELECT Teacher.teacher_id, Teacher.first_name, Courses.course_name
    -> FROM Teacher
    -> JOIN Courses ON Teacher.teacher_id = Courses.teacher_id;
+------------+------------+-------------+
| teacher_id | first_name | course_name |
+------------+------------+-------------+
|        101 | Archana    | B.Tech      |
|        102 | Arjun      | BA          |
|        103 | Balraj     | B.Tech      |
|        104 | Chirag     | BBA         |
|        105 | Dhanush    | BCA         |
|        106 | Girr       | LLB         |
|        107 | Himani     | MBA         |
|        108 | Kailash    | M.Tech      |
|        109 | Lucky      | MA          |
|        110 | Mahak      | MCA         |
+------------+------------+-------------+
10 rows in set (0.00 sec)
```

6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.



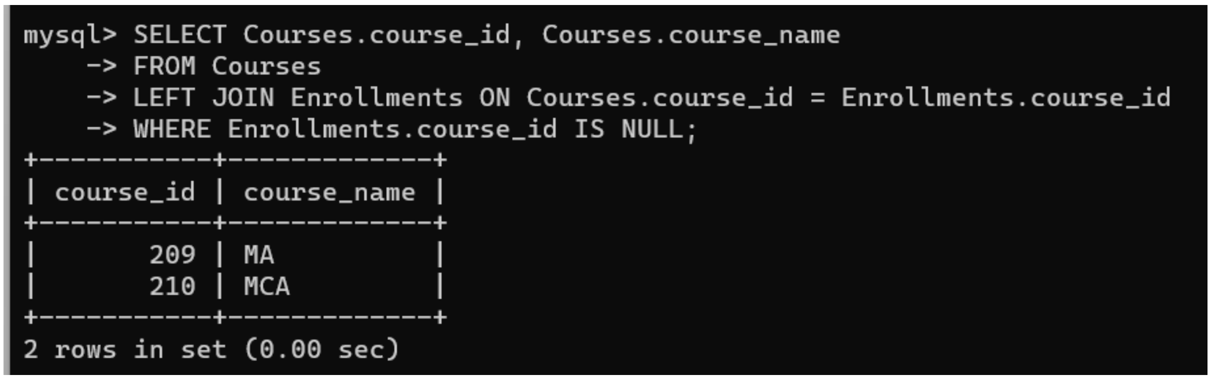
```
mysql> SELECT Students.student_id, Students.first_name, Students.last_name, Enrollments.enrollment_date
    -> FROM Students
    -> JOIN Enrollments ON Students.student_id = Enrollments.student_id
    -> JOIN Courses ON Enrollments.course_id = Courses.course_id
    -> WHERE Courses.course_name = 'BCA';
+------------+------------+-----------+-----------------+
| student_id | first_name | last_name | enrollment_date |
+------------+------------+-----------+-----------------+
|          5 | Fatima     | Khan      | 2024-01-04      |
+------------+------------+-----------+-----------------+
1 row in set (0.00 sec)
```

7. Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.



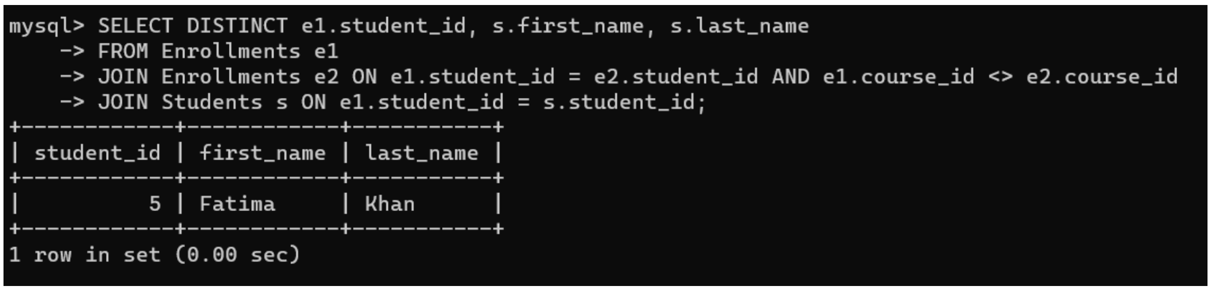
```
mysql> SELECT Students.student_id, Students.first_name, Students.last_name
    -> FROM Students
    -> LEFT JOIN Payments ON Students.student_id = Payments.student_id
    -> WHERE Payments.payment_id IS NULL;
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|         11 | John       | Doe       |
+------------+------------+-----------+
1 row in set (0.01 sec)
```

8. Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.



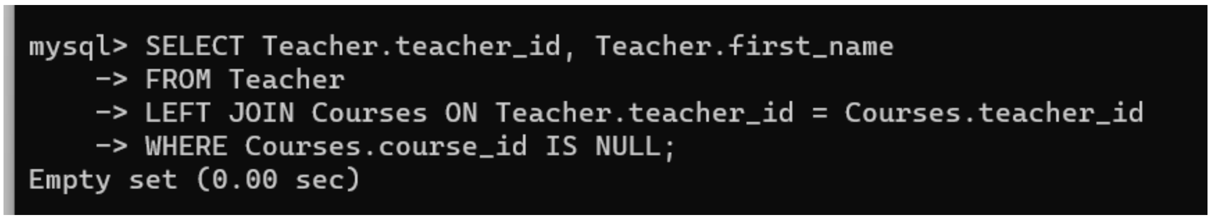
```
mysql> SELECT Courses.course_id, Courses.course_name
    -> FROM Courses
    -> LEFT JOIN Enrollments ON Courses.course_id = Enrollments.course_id
    -> WHERE Enrollments.course_id IS NULL;
+-----------+-------------+
| course_id | course_name |
+-----------+-------------+
|       209 | MA          |
|       210 | MCA         |
+-----------+-------------+
2 rows in set (0.00 sec)
```

9. Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.



```
mysql> SELECT DISTINCT e1.student_id, s.first_name, s.last_name
    -> FROM Enrollments e1
    -> JOIN Enrollments e2 ON e1.student_id = e2.student_id AND e1.course_id <> e2.course_id
    -> JOIN Students s ON e1.student_id = s.student_id;
+------------+------------+-----------+
| student_id | first_name | last_name |
+------------+------------+-----------+
|          5 | Fatima     | Khan      |
+------------+------------+-----------+
1 row in set (0.00 sec)
```

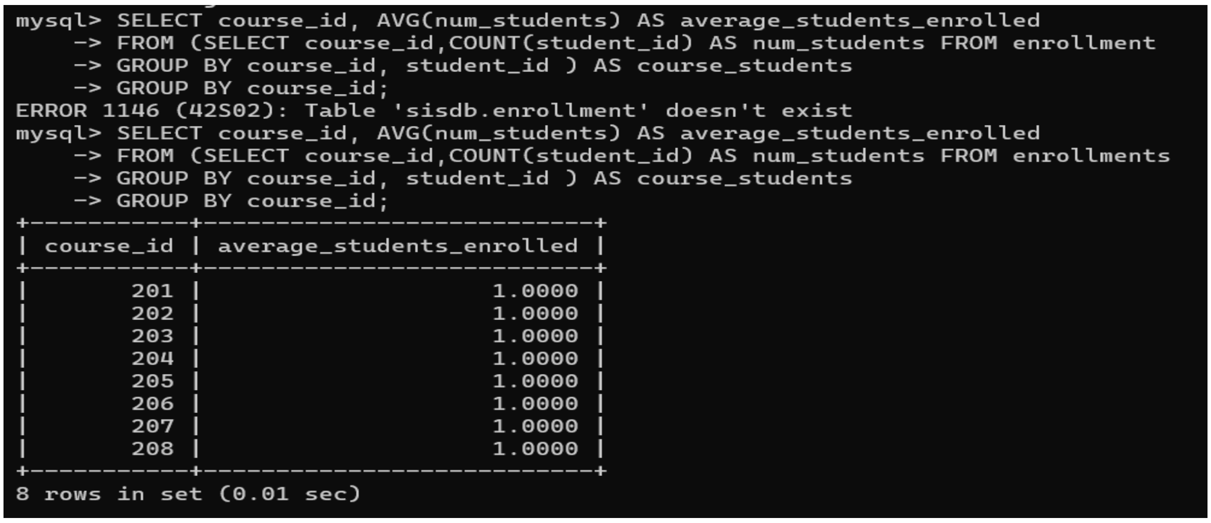
10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.



```
mysql> SELECT Teacher.teacher_id, Teacher.first_name
    -> FROM Teacher
    -> LEFT JOIN Courses ON Teacher.teacher_id = Courses.teacher_id
    -> WHERE Courses.course_id IS NULL;
Empty set (0.00 sec)
```

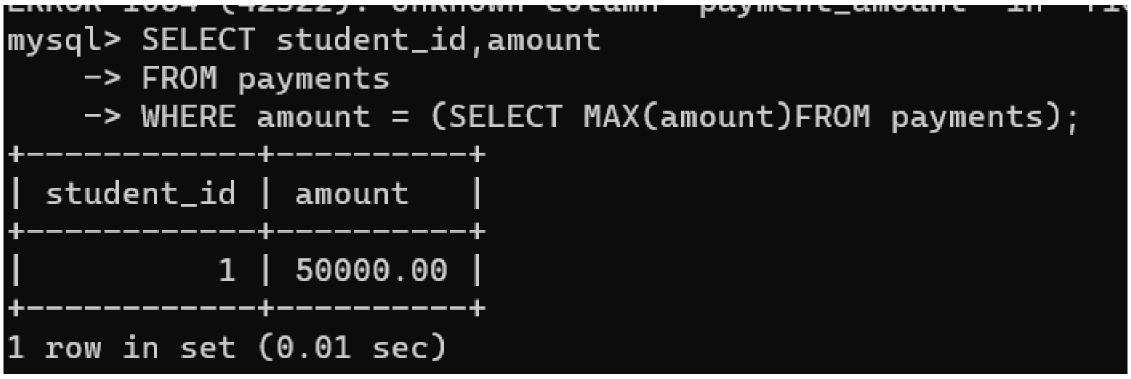
Task 4. Subquery and its type:

1. Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.



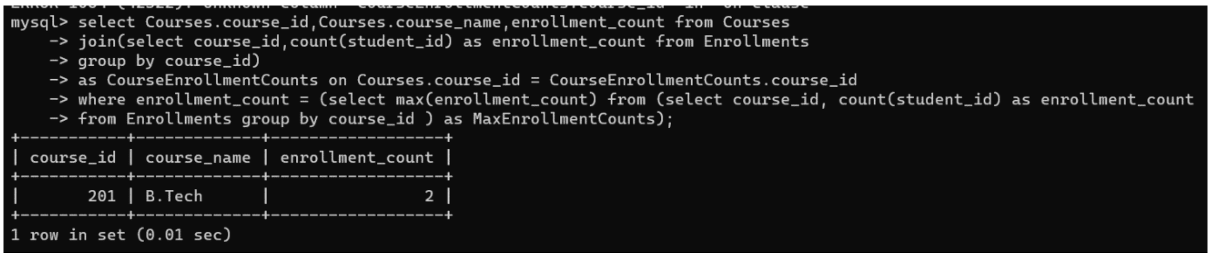
```
mysql> SELECT course_id, AVG(num_students) AS average_students_enrolled
    -> FROM (SELECT course_id,COUNT(student_id) AS num_students FROM enrollment
    -> GROUP BY course_id, student_id ) AS course_students
    -> GROUP BY course_id;
ERROR 1146 (42S02): Table 'sisdb.enrollment' doesn't exist
mysql> SELECT course_id, AVG(num_students) AS average_students_enrolled
    -> FROM (SELECT course_id,COUNT(student_id) AS num_students FROM enrollments
    -> GROUP BY course_id, student_id ) AS course_students
    -> GROUP BY course_id;
+-----------+---------------------------+
| course_id | average_students_enrolled |
+-----------+---------------------------+
|       201 |                    1.0000 |
|       202 |                    1.0000 |
|       203 |                    1.0000 |
|       204 |                    1.0000 |
|       205 |                    1.0000 |
|       206 |                    1.0000 |
|       207 |                    1.0000 |
|       208 |                    1.0000 |
+-----------+---------------------------+
8 rows in set (0.01 sec)
```

2. Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.



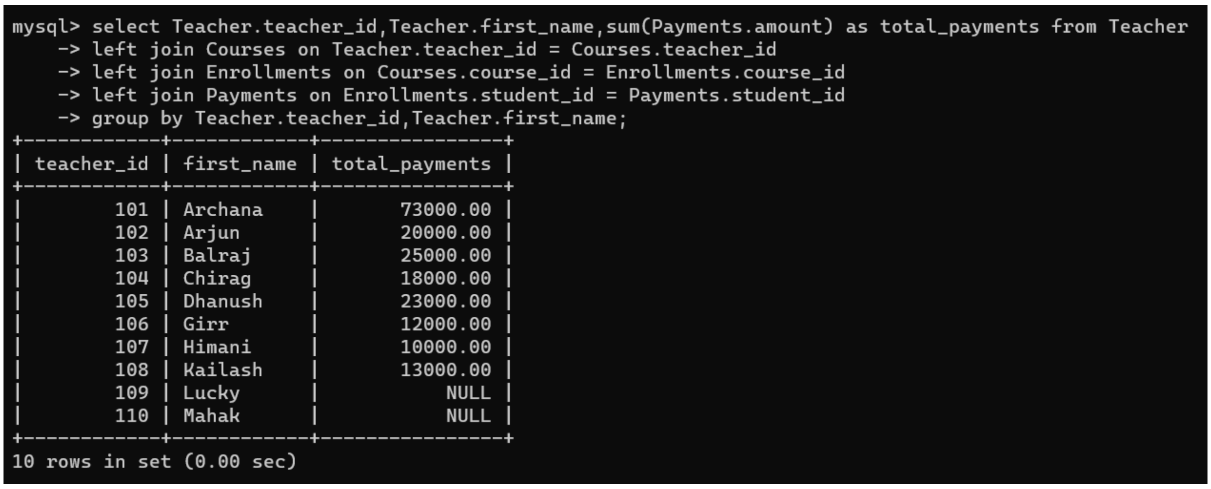
```
mysql> SELECT student_id,amount
    -> FROM payments
    -> WHERE amount = (SELECT MAX(amount)FROM payments);
+------------+----------+
| student_id | amount   |
+------------+----------+
|          1 | 50000.00 |
+------------+----------+
1 row in set (0.01 sec)
```

3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.



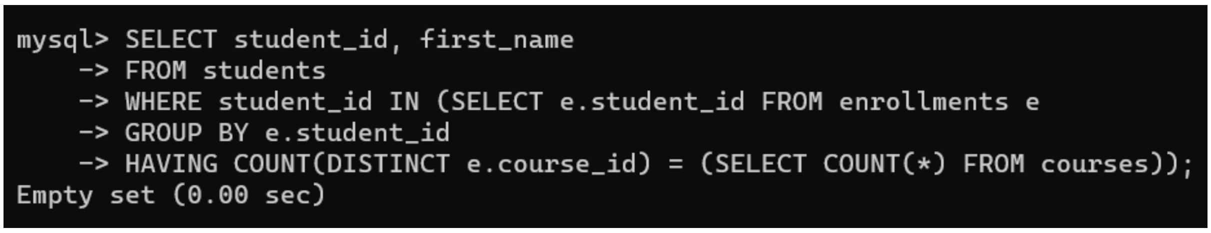
```
mysql> select Courses.course_id,Courses.course_name,enrollment_count from Courses
    -> join(select course_id,count(student_id) as enrollment_count from Enrollments
    -> group by course_id)
    -> as CourseEnrollmentCounts on Courses.course_id = CourseEnrollmentCounts.course_id
    -> where enrollment_count = (select max(enrollment_count) from (select course_id, count(student_id) as enrollment_count
    -> from Enrollments group by course_id ) as MaxEnrollmentCounts);
+-----------+-------------+------------------+
| course_id | course_name | enrollment_count |
+-----------+-------------+------------------+
|       201 | B.Tech      |                2 |
+-----------+-------------+------------------+
1 row in set (0.01 sec)
```

4. Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.



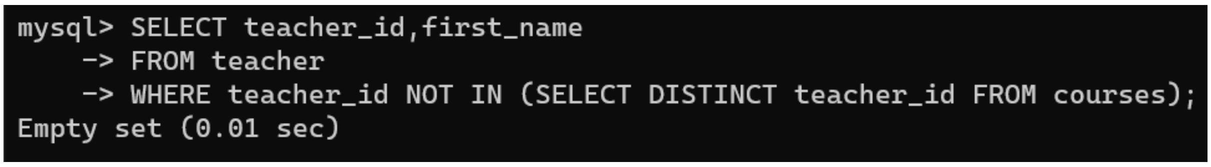
```
mysql> select Teacher.teacher_id,Teacher.first_name,sum(Payments.amount) as total_payments from Teacher
    -> left join Courses on Teacher.teacher_id = Courses.teacher_id
    -> left join Enrollments on Courses.course_id = Enrollments.course_id
    -> left join Payments on Enrollments.student_id = Payments.student_id
    -> group by Teacher.teacher_id,Teacher.first_name;
+------------+------------+----------------+
| teacher_id | first_name | total_payments |
+------------+------------+----------------+
|        101 | Archana    |       73000.00 |
|        102 | Arjun      |       20000.00 |
|        103 | Balraj     |       25000.00 |
|        104 | Chirag     |       18000.00 |
|        105 | Dhanush    |       23000.00 |
|        106 | Girr       |       12000.00 |
|        107 | Himani     |       10000.00 |
|        108 | Kailash    |       13000.00 |
|        109 | Lucky      |           NULL |
|        110 | Mahak      |           NULL |
+------------+------------+----------------+
10 rows in set (0.00 sec)
```

5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.



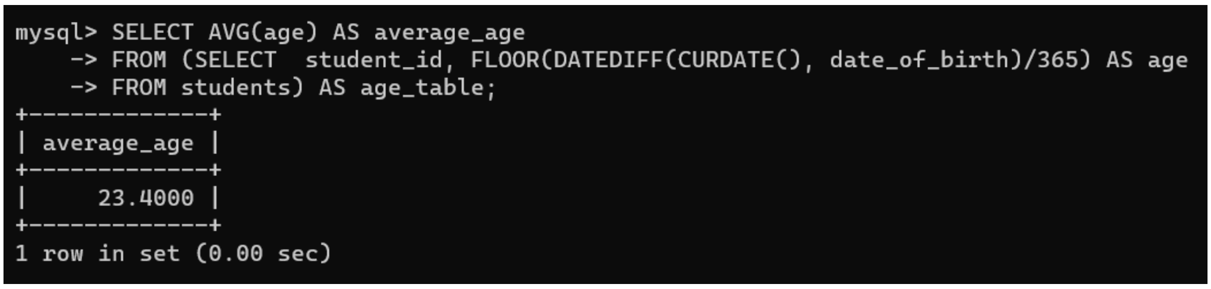
```
mysql> SELECT student_id, first_name
    -> FROM students
    -> WHERE student_id IN (SELECT e.student_id FROM enrollments e
    -> GROUP BY e.student_id
    -> HAVING COUNT(DISTINCT e.course_id) = (SELECT COUNT(*) FROM courses));
Empty set (0.00 sec)
```

6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.



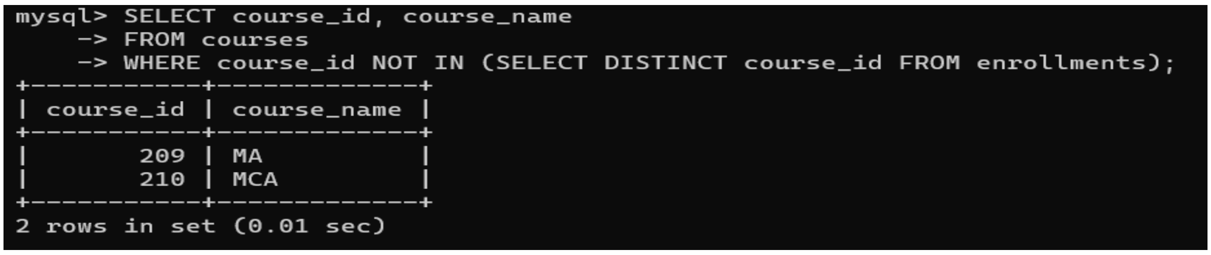
```
mysql> SELECT teacher_id,first_name
    -> FROM teacher
    -> WHERE teacher_id NOT IN (SELECT DISTINCT teacher_id FROM courses);
Empty set (0.01 sec)
```

7. Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.



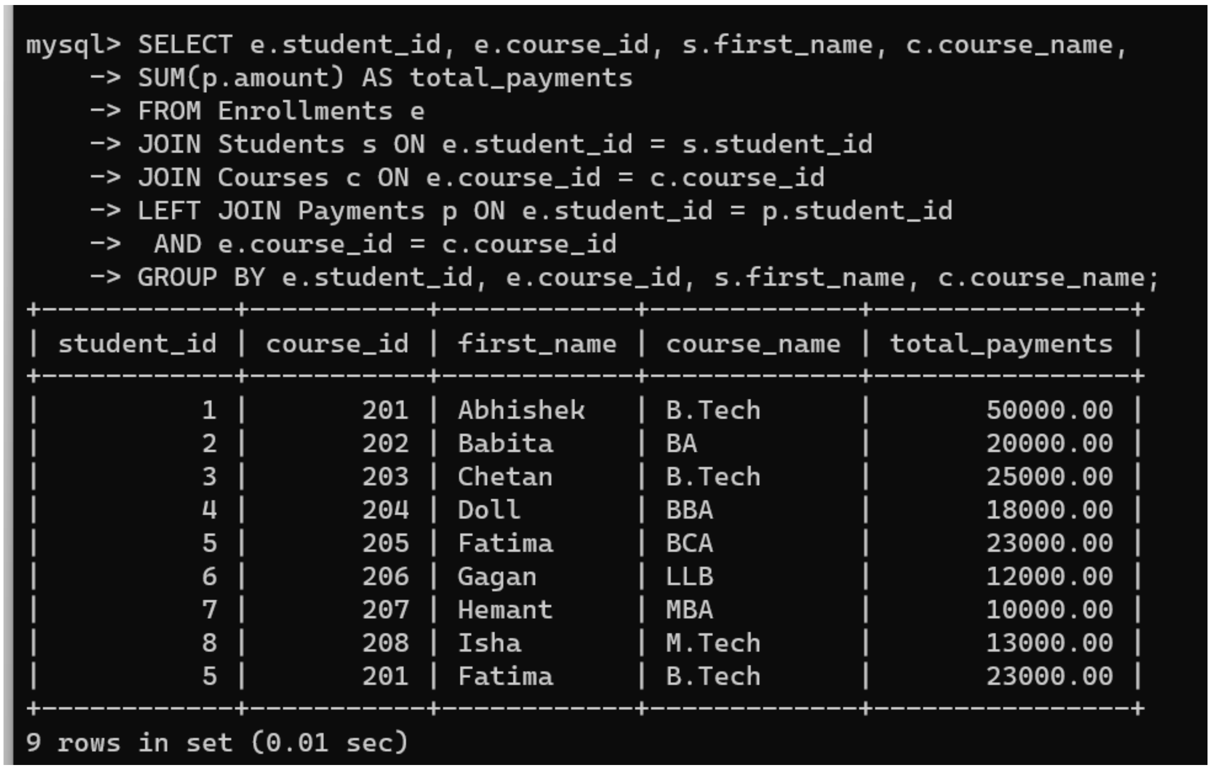
```
mysql> SELECT AVG(age) AS average_age
    -> FROM (SELECT  student_id, FLOOR(DATEDIFF(CURDATE(), date_of_birth)/365) AS age
    -> FROM students) AS age_table;
+-------------+
| average_age |
+-------------+
|     23.4000 |
+-------------+
1 row in set (0.00 sec)
```

8. Identify courses with no enrollments. Use subqueries to find courses without enrollment records.



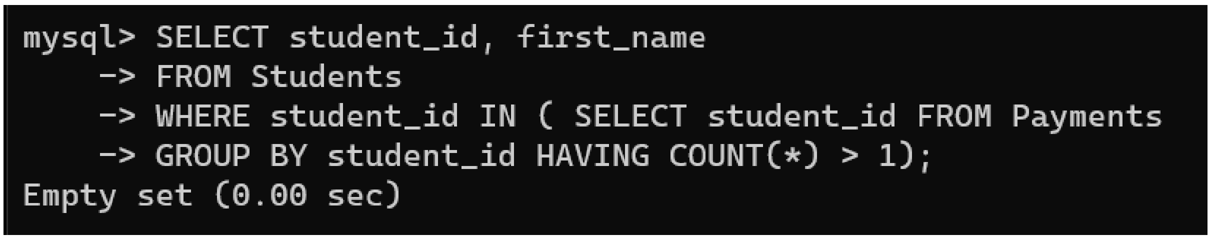
```
mysql> SELECT course_id, course_name
    -> FROM courses
    -> WHERE course_id NOT IN (SELECT DISTINCT course_id FROM enrollments);
+-----------+-------------+
| course_id | course_name |
+-----------+-------------+
|       209 | MA          |
|       210 | MCA         |
+-----------+-------------+
2 rows in set (0.01 sec)
```

9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.



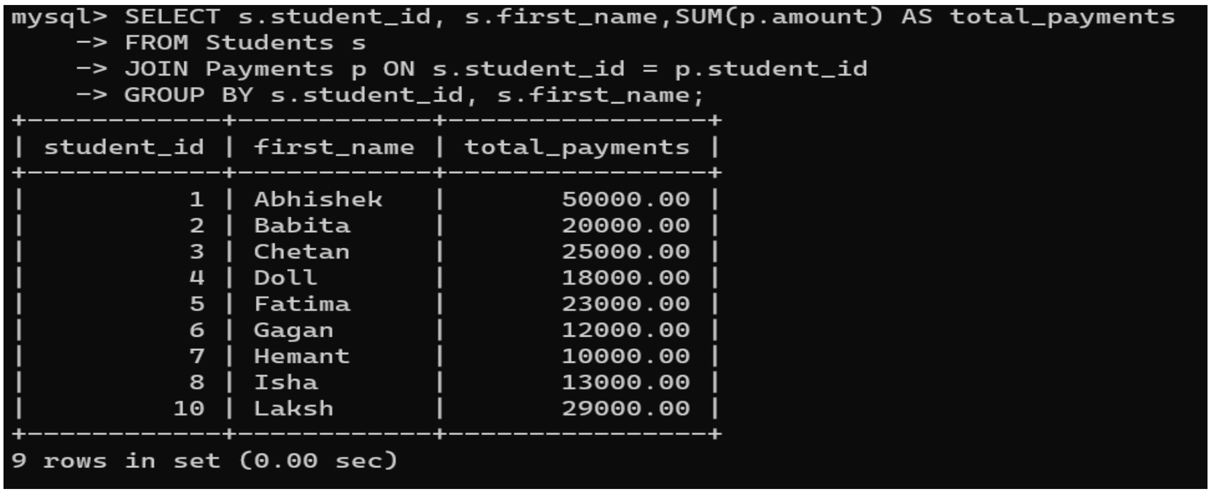
```
mysql> SELECT e.student_id, e.course_id, s.first_name, c.course_name,
    -> SUM(p.amount) AS total_payments
    -> FROM Enrollments e
    -> JOIN Students s ON e.student_id = s.student_id
    -> JOIN Courses c ON e.course_id = c.course_id
    -> LEFT JOIN Payments p ON e.student_id = p.student_id
    ->  AND e.course_id = c.course_id
    -> GROUP BY e.student_id, e.course_id, s.first_name, c.course_name;
+------------+-----------+------------+-------------+----------------+
| student_id | course_id | first_name | course_name | total_payments |
+------------+-----------+------------+-------------+----------------+
|          1 |       201 | Abhishek   | B.Tech      |       50000.00 |
|          2 |       202 | Babita     | BA          |       20000.00 |
|          3 |       203 | Chetan     | B.Tech      |       25000.00 |
|          4 |       204 | Doll       | BBA         |       18000.00 |
|          5 |       205 | Fatima     | BCA         |       23000.00 |
|          6 |       206 | Gagan      | LLB         |       12000.00 |
|          7 |       207 | Hemant     | MBA         |       10000.00 |
|          8 |       208 | Isha       | M.Tech      |       13000.00 |
|          5 |       201 | Fatima     | B.Tech      |       23000.00 |
+------------+-----------+------------+-------------+----------------+
9 rows in set (0.01 sec)
```

10. Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.



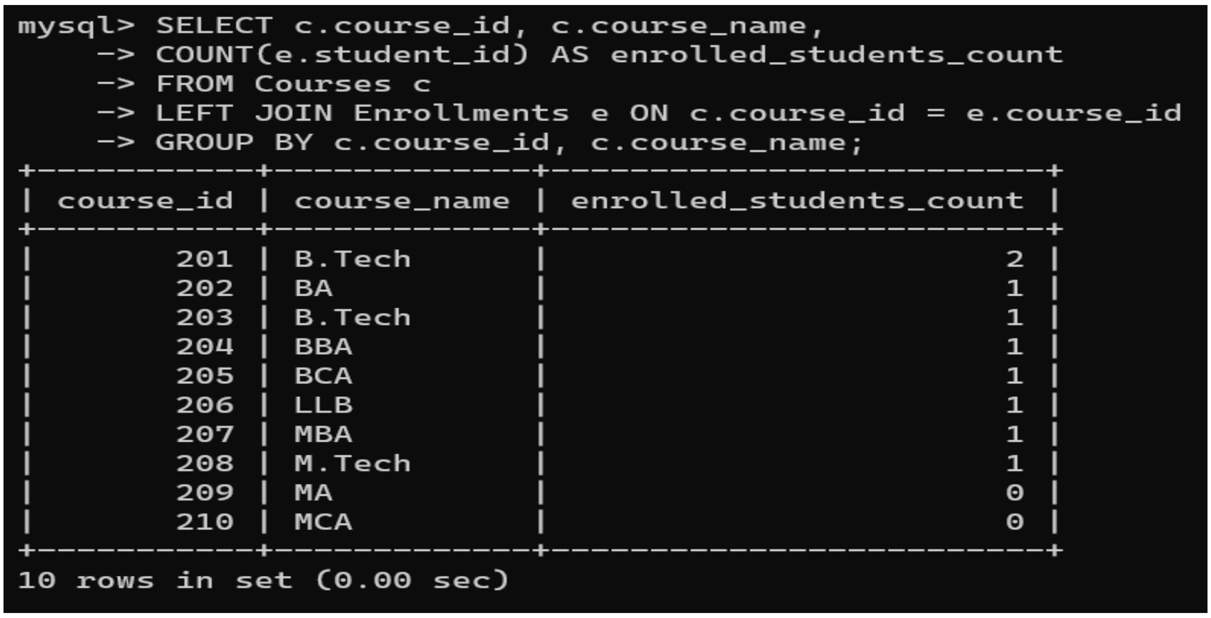
```
mysql> SELECT student_id, first_name
    -> FROM Students
    -> WHERE student_id IN ( SELECT student_id FROM Payments
    -> GROUP BY student_id HAVING COUNT(*) > 1);
Empty set (0.00 sec)
```

11. Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.



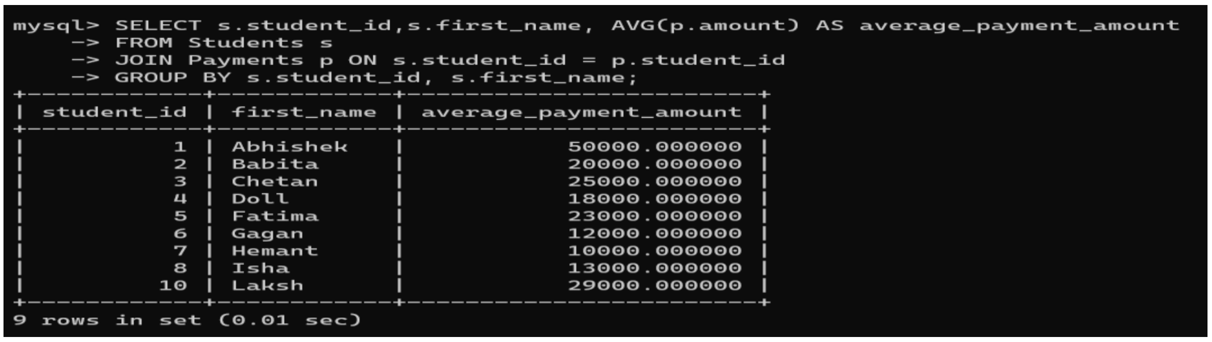
```
mysql> SELECT s.student_id, s.first_name,SUM(p.amount) AS total_payments
    -> FROM Students s
    -> JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id, s.first_name;
+------------+------------+----------------+
| student_id | first_name | total_payments |
+------------+------------+----------------+
|          1 | Abhishek   |       50000.00 |
|          2 | Babita     |       20000.00 |
|          3 | Chetan     |       25000.00 |
|          4 | Doll       |       18000.00 |
|          5 | Fatima     |       23000.00 |
|          6 | Gagan      |       12000.00 |
|          7 | Hemant     |       10000.00 |
|          8 | Isha       |       13000.00 |
|         10 | Laksh      |       29000.00 |
+------------+------------+----------------+
9 rows in set (0.00 sec)
```

12. Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.



```
mysql> SELECT c.course_id, c.course_name,
    -> COUNT(e.student_id) AS enrolled_students_count
    -> FROM Courses c
    -> LEFT JOIN Enrollments e ON c.course_id = e.course_id
    -> GROUP BY c.course_id, c.course_name;
+-----------+-------------+-------------------------+
| course_id | course_name | enrolled_students_count |
+-----------+-------------+-------------------------+
|       201 | B.Tech      |                       2 |
|       202 | BA          |                       1 |
|       203 | B.Tech      |                       1 |
|       204 | BBA         |                       1 |
|       205 | BCA         |                       1 |
|       206 | LLB         |                       1 |
|       207 | MBA         |                       1 |
|       208 | M.Tech      |                       1 |
|       209 | MA          |                       0 |
|       210 | MCA         |                       0 |
+-----------+-------------+-------------------------+
10 rows in set (0.00 sec)
```

13. Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.



```
mysql> SELECT s.student_id,s.first_name, AVG(p.amount) AS average_payment_amount
    -> FROM Students s
    -> JOIN Payments p ON s.student_id = p.student_id
    -> GROUP BY s.student_id, s.first_name;
+------------+------------+------------------------+
| student_id | first_name | average_payment_amount |
+------------+------------+------------------------+
|          1 | Abhishek   |           50000.000000 |
|          2 | Babita     |           20000.000000 |
|          3 | Chetan     |           25000.000000 |
|          4 | Doll       |           18000.000000 |
|          5 | Fatima     |           23000.000000 |
|          6 | Gagan      |           12000.000000 |
|          7 | Hemant     |           10000.000000 |
|          8 | Isha       |           13000.000000 |
|         10 | Laksh      |           29000.000000 |
+------------+------------+------------------------+
9 rows in set (0.01 sec)
```