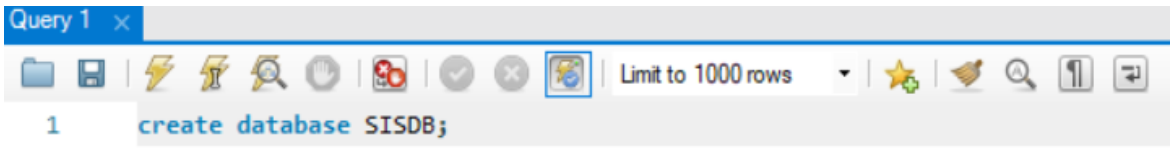
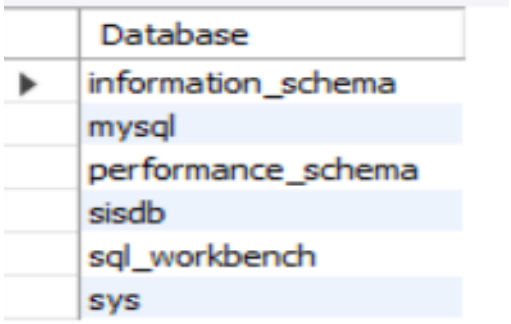
Task 1. Database Design

1. Create the database named "SISDB"

Query



Output



Database
information_schema
mysql
performance_schema
sisdb
sql_workbench
sys

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

b. Courses

c. Enrollments

d. Teacher

e. Payments

a. Students

Query

```sql
create table Students (
student_id int primary key auto_increment,
first_name varchar(100) not null,
last_name varchar(100) not null,
date_of_birth date not null,
email varchar(100) unique not null,
phone_number varchar(15)
) auto_increment = 101;
```

```sql
desc students;
```

Output

```
3  14:35:45  create table Students ( student_id int primary key auto_increment, first_name varchar(100) not null, last_name varc...  0 row(s) affected
3  06:00:55  desc students  6 row(s) returned
```

Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	NULL	auto_increment
first_name	varchar(100)	NO		NULL	
last_name	varchar(100)	NO		NULL	
date_of_birth	date	NO		NULL	
email	varchar(100)	NO	UNI	NULL	
phone_number	varchar(15)	YES		NULL	

b. Courses

Query

```sql
create table courses (
course_id int primary key auto_increment,
course_name varchar(150) not null,
credits int not null,
teacher_id int,
foreign key(teacher_id) references teacher(teacher_id)) auto_increment=201;
```

```sql
desc courses;
```

Output

```
12  14:50:57  create table courses ( course_id int primary key auto...  0 row(s) affected
4  06:02:15  desc courses  4 row(s) returned
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

Field	Type	Null	Key	Default	Extra
course_id	int	NO	PRI	NULL	auto_increment
course_name	varchar(150)	NO		NULL	
credits	int	NO		NULL	
teacher_id	int	YES	MUL	NULL	

c. Enrollments

Query

```
1  create table enrollments (
2  enrollment_id int primary key auto_increment,
3  student_id int not null,
4  course_id int not null,
5  enrollment_date date not null,
6  unique(student_id, course_id),
7  foreign key (student_id) references students(student_id),
8  foreign key (course_id) references courses(course_id)) auto_increment = 301;
```

```
1 • desc enrollments;
```

Output

```
14  14:57:13  create table enrollments ( enrollment_id int primary k...  0 row(s) affected
```

```
5  06:03:09  desc enrollments  4 row(s) returned
```

Field	Type	Null	Key	Default	Extra
enrollment_id	int	NO	PRI	NULL	auto_increment
student_id	int	NO	MUL	NULL	
course_id	int	NO	MUL	NULL	
enrollment_date	date	NO		NULL	

d. Teacher

Query

```
1  create table teacher (
2  teacher_id int primary key auto_increment,
3  first_name varchar(100) not null,
4  last_name varchar(100) not null,
5  email varchar(100) unique not null) auto_increment=401;
```

```
1 • desc teacher;
```

Output

```
5  14:41:57  create table Teacher ( teacher_id int primary key auto...  0 row(s) affected
```

```
6  06:03:58  desc teacher  4 row(s) returned
```

Field	Type	Null	Key	Default	Extra
teacher_id	int	NO	PRI	NULL	auto_increment
first_name	varchar(100)	NO		NULL	
last_name	varchar(100)	NO		NULL	
email	varchar(100)	NO	UNI	NULL	

e. Payments

Query

```
1  create table payments (
2  payment_id int primary key auto_increment,
3  student_id int,
4  amount int not null,
5  payment_date date not null,
6  foreign key (student_id) references students(student_id)) auto_increment=501;
```

```
1 • desc payments;
```

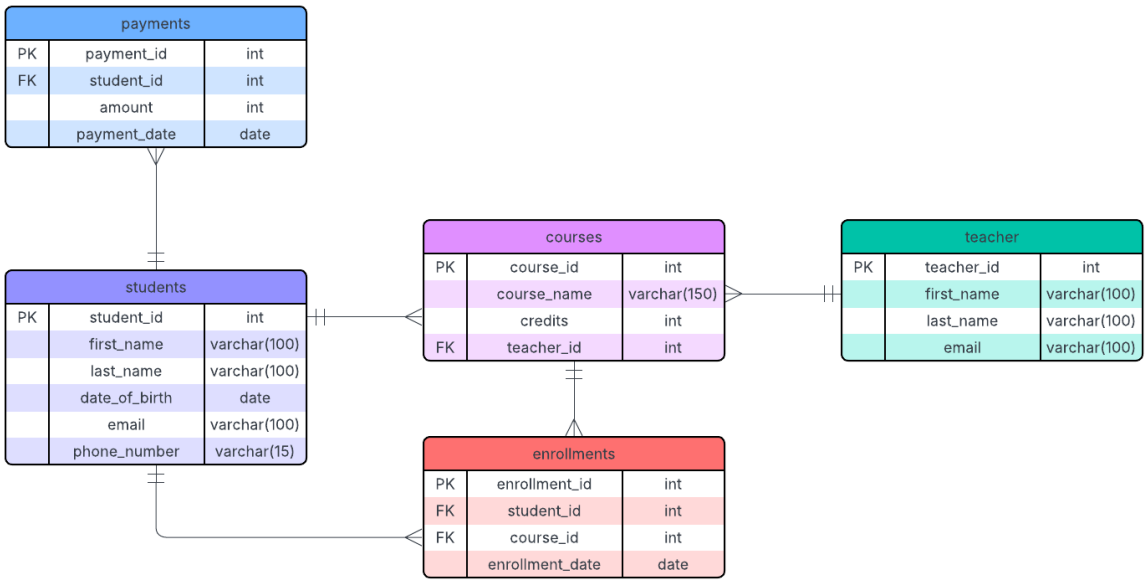
Output

```
19  15:10:47  create table payments ( payment_id int primary key ...  0 row(s) affected
```

```
7  06:04:58  desc payments  4 row(s) returned
```

Field	Type	Null	Key	Default	Extra
payment_id	int	NO	PRI	NULL	auto_increment
student_id	int	NO	MUL	NULL	
amount	int	NO		NULL	
payment_date	date	NO		NULL	

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



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

i. Students

ii. Courses

iii. Enrollments

iv. Teacher

v. Payments

i. Students

Query

```
insert into students(first_name, last_name, date_of_birth,email,phone_number) values
('Aarav', 'Sharma', '2002-05-14', 'aarav.sharma@gmail.com', '9876543210'),
('Isha', 'Verma', '2003-08-21', 'isha.verma@yahoo.com', '9876501234'),
('Rahul', 'Mehta', '2001-11-03', 'rahul.mehta@outlook.com', '9123456780'),
('Sneha', 'Patil', '2002-01-27', 'sneha.patil@gmail.com', '9988776655'),
('Karan', 'Singh', '2003-03-15', 'karan.singh@hotmail.com', '9001122334'),
('Divya', 'Reddy', '2002-09-12', 'divya.reddy@gmail.com', '9834567890'),
('Amit', 'Kumar', '2001-12-05', 'amit.kumar@gmail.com', '9823456789'),
('Neha', 'Joshi', '2003-07-19', 'neha.joshi@gmail.com', '9812345678'),
('Vikram', 'Das', '2002-04-25', 'vikram.das@yahoo.com', '9900112233'),
('Priya', 'Nair', '2001-10-10', 'priya.nair@outlook.com', '9798989898'),
('Rohan', 'Malhotra', '2003-02-28', 'rohan.malhotra@gmail.com', '9876543211'),
('Ananya', 'Gupta', '2002-07-17', 'ananya.gupta@gmail.com', '9876543212');
```

```
select * from students;
```

Output

```
24  15:23:04  insert into students(first_name, last_name, date_of_...  12 row(s) affected Records: 12  Duplicates: 0  Warn...
```

```
8  06:06:21  select * from students LIMIT 0, 1000  12 row(s) returned
```

student_id	first_name	last_name	date_of_birth	email	phone_number
103	Rahul	Mehta	2001-11-03	rahul.mehta@outlook.com	9123456780
104	Sneha	Patil	2002-01-27	sneha.patil@gmail.com	9988776655
105	Karan	Singh	2003-03-15	karan.singh@hotmail.com	9001122334
106	Divya	Reddy	2002-09-12	divya.reddy@gmail.com	9834567890
107	Amit	Kumar	2001-12-05	amit.kumar@gmail.com	9823456789
108	Neha	Joshi	2003-07-19	neha.joshi@gmail.com	9812345678
109	Vikram	Das	2002-04-25	vikram.das@yahoo.com	9900112233
110	Priya	Nair	2001-10-10	priya.nair@outlook.com	9798989898
111	Rohan	Malhotra	2003-02-28	rohan.malhotra@gmail.com	9876543211
112	Ananya	Gupta	2002-07-17	ananya.gupta@gmail.com	9876543212

ii. Courses

Query

```
insert into courses(course_name, credits, teacher_id) values
('Introduction to Computer Science', 3, 401),
('Advanced Programming', 4, 401),
('Database Systems', 4, 402),
('Data Structures and Algorithms', 4, 403),
('Operating Systems', 3, 404),
('Computer Networks', 3, 405),
('Artificial Intelligence', 4, 405),
('Web Development', 3, 406),
('Machine Learning', 4, 407),
('Cybersecurity Fundamentals', 3, 408),
('Cloud Computing', 3, 410),
('Software Engineering', 4, NULL),
('Mobile App Development', 3, NULL);
```

```
1 • select * from courses;
```

Output

```
28  15:32:51  insert into courses(course_name, credits, teacher_id...  13 row(s) affected Records: 13  Duplicates: 0  Warn...
9   06:08:01  select * from courses LIMIT 0, 1000                      13 row(s) returned
```

course_id	course_name	credits	teacher_id
201	Introduction to Computer Science	3	401
202	Advanced Programming	4	401
203	Database Systems	4	402
204	Data Structures and Algorithms	4	403
205	Operating Systems	3	404
206	Computer Networks	3	405
207	Artificial Intelligence	4	405
208	Web Development	3	406
209	Machine Learning	4	407
210	Cybersecurity Fundamentals	3	408
211	Cloud Computing	3	410
212	Software Engineering	4	NULL
213	Mobile App Development	3	NULL

iii. Enrollments

Query

```
insert into  enrollments (student_id, course_id, enrollment_date) values
(101, 201, '2024-01-10'),
(101, 203, '2024-01-10'),
(102, 201, '2024-02-15'),
(103, 205, '2024-02-15'),
(103, 209, '2024-03-20'),
(104, 202, '2024-04-05'),
(105, 206, '2024-04-05'),
(106, 205, '2024-06-12'),
(107, 204, '2024-07-01'),
(108, 201, '2024-07-01'),
(108, 208, '2024-08-18'),
(109, 204, '2024-09-10'),
(109, 202, '2024-09-10'),
(110, 210, '2024-10-25'),
(110, 207, '2024-11-30'),
(101, 204, '2024-12-01'),
(102, 203, '2024-12-05'),
(111, 201, '2024-12-10'),
(111, 202, '2024-12-10'),
(111, 203, '2024-12-10');
```

```
select * from enrollments;
```

Output

```
31  15:37:02  insert into  enrollments (student_id, course_id, enrollment_date) values (101, 201, '2024-01-10'), (101, 203, '2024-...  20 row(s) affected Records: 20  Duplicates: 0  Warnings: 0
```

```
10  06:09:06  select * from enrollments LIMIT 0, 1000
```

enrollment_id	student_id	course_id	enrollment_date
301	101	201	2024-01-10
302	101	203	2024-01-10
303	102	201	2024-02-15
304	103	205	2024-02-15
305	103	209	2024-03-20
306	104	202	2024-04-05
307	105	206	2024-04-05
308	106	205	2024-06-12
309	107	204	2024-07-01
310	108	201	2024-07-01
311	108	208	2024-08-18
312	109	204	2024-09-10
313	109	202	2024-09-10
314	110	210	2024-10-25

iv. Teacher

Query

```
insert into teacher (first_name, last_name, email) values
('Rajesh', 'Khanna', 'rajesh.khanna@university.edu'),
('Priyanka', 'Chopra', 'priyanka.chopra@university.edu'),
('Vikram', 'Seth', 'vikram.seth@university.edu'),
('Sunita', 'Rao', 'sunita.rao@university.edu'),
('Arjun', 'Kapoor', 'arjun.kapoor@university.edu'),
('Meera', 'Patel', 'meera.patel@university.edu'),
('Sanjay', 'Verma', 'sanjay.verma@university.edu'),
('Anjali', 'Desai', 'anjali.desai@university.edu'),
('Rahul', 'Bose', 'rahul.bose@university.edu'),
('Deepika', 'Sharma', 'deepika.sharma@university.edu'),
('Aditya', 'Roy', 'aditya.roy@university.edu');
```

```
select * from teacher;
```

Output

```
26  15:28:11  insert into teacher (first_name, last_name, email) val...  11 row(s) affected Records: 11  Duplicates: 0  Warn...
11  06:10:22  select * from teacher LIMIT 0, 1000                        11 row(s) returned
```

teacher_id	first_name	last_name	email
401	Rajesh	Khanna	rajesh.khanna@university.edu
402	Priyanka	Chopra	priyanka.chopra@university.edu
403	Vikram	Seth	vikram.seth@university.edu
404	Sunita	Rao	sunita.rao@university.edu
405	Arjun	Kapoor	arjun.kapoor@university.edu
406	Meera	Patel	meera.patel@university.edu
407	Sanjay	Verma	sanjay.verma@university.edu
408	Anjali	Desai	anjali.desai@university.edu
409	Rahul	Bose	rahul.bose@university.edu
410	Deepika	Sharma	deepika.sharma@university.edu
411	Aditya	Roy	aditya.roy@university.edu

v. Payments

Query

```
insert into payments (student_id, amount, payment_date) values
(101, 25000, '2024-01-05'),
(101, 28000, '2024-01-15'),
(102, 25000, '2024-02-10'),
(103, 35000, '2024-02-12'),
(103, 27000, '2024-03-18'),
(104, 30000, '2024-04-01'),
(105, 28000, '2024-04-03'),
(106, 35000, '2024-06-10'),
(107, 27000, '2024-06-28'),
(108, 25000, '2024-06-30'),
(108, 33000, '2024-08-15'),
(109, 27000, '2024-09-05'),
(109, 30000, '2024-09-08'),
(110, 22000, '2024-10-20'),
(110, 32000, '2024-11-25'),
(111, 35000, '2024-12-05'),
(112, 25000, '2024-12-08');
```

```
select * from payments;
```

Output

```
33  15:42:09  insert into payments (student_id, amount, payment_...  17 row(s) affected Records: 17  Duplicates: 0  Warn...
```

```
12  06:12:10  select * from payments LIMIT 0, 1000
```

payment_id	student_id	amount	payment_date
501	101	25000	2024-01-05
502	101	28000	2024-01-15
503	102	25000	2024-02-10
504	103	35000	2024-02-12
505	103	27000	2024-03-18
506	104	30000	2024-04-01
507	105	28000	2024-04-03
508	106	35000	2024-06-10
509	107	27000	2024-06-28
510	108	25000	2024-06-30
511	108	33000	2024-08-15
512	109	27000	2024-09-05
513	109	30000	2024-09-08
514	110	22000	2024-10-20
515	110	32000	2024-11-25
516	111	35000	2024-12-05
517	112	25000	2024-12-08
NULL	NULL	NULL	NULL

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

Query

```
insert into students(first_name, last_name, date_of_birth, email, phone_number) values
("John", "Doe", '1995-08-15', 'john.doe@example.com', '1234567890')
```

```
select * from students where student_id = 120;
```

Output

```
5  16:02:08  insert into students(first_name, last_name, date_of_birt...  1 row(s) affected
```

student_id	first_name	last_name	date_of_birth	email	phone_number
120	John	Doe	1995-08-15	john.doe@example.com	1234567890
NULL	NULL	NULL	NULL	NULL	NULL

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.

Query

```
insert into enrollments (student_id, course_id, enrollment_date) values
(120, 207, '2024-05-05');
```

```
select * from enrollments where student_id = 120 and course_id = 207;
```

Output

```
8  16:16:55  insert into enrollments (student_id, course_id, enrollme...  1 row(s) affected
```

enrollment_id	student_id	course_id	enrollment_date
321	120	207	2024-05-05
NULL	NULL	NULL	NULL

enrollment_id	student_id	course_id	enrollment_date
318	111	201	2024-12-10
319	111	202	2024-12-10
320	111	203	2024-12-10
321	120	207	2024-05-05
NULL	NULL	NULL	NULL

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

Query

```
update teacher set email = 'deepika.sharma@gmail.com' where teacher_id = 410;
```

```
select * from teacher where email like '%@gmail.com';
```

Output

```
18  16:27:53  update teacher set email = 'deepika.s@univerisity.edu' where teacher_id = 410    1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0
```

teacher_id	first_name	last_name	email
410	Deepika	Sharma	deepika.sharma@gmail.com
NULL	NULL	NULL	NULL

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.

Query

```
1   delete from enrollments where student_id = 111 and course_id = 201;
```

```
select * from enrollments where student_id=111 and course_id = 201;
```

Output

```
5  09:35:32  update teacher set email = 'deepika.sharma@gmail.com' where teacher_id = 410    1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0
3  09:21:12  select * from enrollments where student_id=111 and course_id = 201 LIMIT 0, 1000    0 row(s) returned
```

enrollment_id	student_id	course_id	enrollment_date
NULL	NULL	NULL	NULL

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

Query

```
1   update courses set teacher_id = 410 where course_id = 206;
```

```
2 •  select * from courses where course_id = 206 and teacher_id = 410;
```

Output

```
27  16:48:18  update courses set teacher_id = 410 where course_id = 206   1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0

6   09:23:32  select * from courses where course_id = 206 and teacher_id = 410 LIMIT 0, 1000   1 row(s) returned
```

course_id	course_name	credits	teacher_id
206	Computer Networks	3	410
NULL	NULL	NULL	NULL

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.

Query

```
1 •  delete from enrollments where student_id = 106;

1 •  select * from enrollments where student_id = 106;

1    delete from payments where student_id = 106;

1 •  select * from payments where student_id = 106;

1    delete from students where student_id = 106;

1 •  select * from students where student_id = 106;
```

Output

```
33  16:55:05  delete from enrollments where student_id = 106   1 row(s) affected
```

enrollment_id	student_id	course_id	enrollment_date
NULL	NULL	NULL	NULL

```
37  16:57:55  delete from payments where student_id = 106   1 row(s) affected
```

payment_id	student_id	amount	payment_date
NULL	NULL	NULL	NULL

```
39  17:02:47  delete from students where student_id = 106   1 row(s) affected
```

	student_id	first_name	last_name	date_of_birth	email	phone_number
*	NULL	NULL	NULL	NULL	NULL	NULL

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

Query

```
1  update payments set amount = 29500 where payment_id = 516 and amount between 28000 and 30000;
```

```
1  select * from payments where payment_id = 516;
```

Output

15 10:08:17 update payments set amount = 29500 where payment_id = 516 and amount between 28000 and 30000 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

	payment_id	student_id	amount	payment_date
▶	516	111	29500	2024-12-05
*	NULL	NULL	NULL	NULL

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.

Query

```
1 • select s.student_id, s.first_name, s.last_name, sum(p.amount)
2       from students as s join payments as p
3       on s.student_id = p.student_id
4       where s.student_id = 101
5       group by s.student_id, s.first_name, s.last_name;
```

Output

27 11:46:11 select s.student_id, s.first_name, s.last_name, sum(p.amount) from students as s join payments as p on s.student... 1 row(s) returned

	student_id	first_name	last_name	sum(p.amount)
▶	101	Aarav	Sharma	53000

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.

Query

```
select c.course_id, c.course_name, count(e.student_id) as student_count
from courses as c join enrollments as e
    on c.course_id = e.course_id
    group by c.course_id, c.course_name;
```

Output

35 13:01:01 select c.course_id, c.course_name, count(e.student_id) as student_count from courses as c join enrollments as e... 10 row(s) returned

	course_id	course_name	student_count
▶	201	Introduction to Computer Science	3
	203	Database Systems	3
	204	Data Structures and Algorithms	3
	205	Operating Systems	1
	209	Machine Learning	1
	202	Advanced Programming	3
	206	Computer Networks	1
	208	Web Development	1
	207	Artificial Intelligence	2
	210	Cybersecurity Fundamentals	1

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.

Query

```
select s.student_id, s.first_name, s.last_name
from students as s left join enrollments as e
    on s.student_id = e.student_id
    where e.enrollment_id is null;
```

Output

7 14:17:28 select s.student_id, s.first_name, s.last_name from students as s left join enrollments as e on s.student_id = e.stud... 1 row(s) returned

	student_id	first_name	last_name
▶	112	Ananya	Gupta

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.

Query

```
select s.first_name, s.last_name, c.course_name
from students as s join
    enrollments as e on s.student_id = e.student_id
    join courses as c on e.course_id = c.course_id;
```

Output

9 14:25:01 select s.first_name, s.last_name, c.course_name from students as s join enrollments as e on s.student_id = e.stud... 19 row(s) returned

	first_name	last_name	course_name
▶	Aarav	Sharma	Introduction to Computer Science
	Aarav	Sharma	Database Systems
	Aarav	Sharma	Data Structures and Algorithms
	Isha	Verma	Introduction to Computer Science
	Isha	Verma	Database Systems
	Rahul	Mehta	Operating Systems
	Rahul	Mehta	Machine Learning
	Sneha	Patil	Advanced Programming
	Karan	Singh	Computer Networks
	Amit	Kumar	Data Structures and Algorithms
	Neha	Joshi	Introduction to Computer Science
	Neha	Joshi	Web Development
	Vikram	Das	Advanced Programming
	Vikram	Das	Data Structures and Algorithms
	Priya	Nair	Artificial Intelligence
	Priya	Nair	Cybersecurity Fundamentals
	Rohan	Malhotra	Advanced Programming
	Rohan	Malhotra	Database Systems

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.

Query

```
select t.first_name, t.last_name, c.course_name
from teacher as t join courses as c
    on  t.teacher_id = c.teacher_id;
```

Output

11 14:32:09 select t.first_name, t.last_name, c.course_name from teacher as t join courses as c on t.teacher_id = c.teacher_j... 11 row(s) returned

first_name	last_name	course_name
Rajesh	Khanna	Introduction to Computer Science
Rajesh	Khanna	Advanced Programming
Priyanka	Chopra	Database Systems
Vikram	Seth	Data Structures and Algorithms
Sunita	Rao	Operating Systems
Deepika	Sharma	Computer Networks
Arjun	Kapoor	Artificial Intelligence
Meera	Patel	Web Development
Sanjay	Verma	Machine Learning
Anjali	Desai	Cybersecurity Fundamentals
Deepika	Sharma	Cloud Computing

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.

Query

```
select s.student_id, s.first_name, s.last_name, e.enrollment_date
from students as s join enrollments as e on s.student_id = e.student_id
    join courses as c on e.course_id = c.course_id where c.course_id = 202;
```

Output

25 14:47:02 select s.student_id, s.first_name, s.last_name, e.enrollment_date from students as s join enrollments as e on s.stu... 3 row(s) returned

student_id	first_name	last_name	enrollment_date
104	Sneha	Patil	2024-04-05
109	Vikram	Das	2024-09-10
111	Rohan	Malhotra	2024-12-10

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.

Query

```
select s.first_name, s.last_name
from students as s left join
    payments as p on s.student_id = p.student_id
    where p.payment_id is null;
```

Output

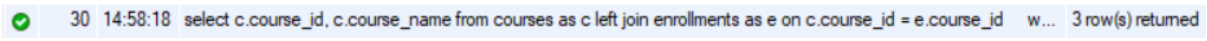
	first_name	last_name
▶	John	Doe

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.

Query

```
select c.course_id, c.course_name
from courses as c left join
    enrollments as e on c.course_id = e.course_id
    where e.enrollment_id is null;
```

Output



	course_id	course_name
▶	211	Cloud Computing
	212	Software Engineering
	213	Mobile App Development

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.

Query

```sql
select distinct e1.student_id
from enrollments as e1 join
    enrollments as e2
    on e1.student_id = e2.student_id
    and e1.course_id <> e2.course_id;
```

Output

student_id
101
102
103
108
109
110
111

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

Query

```sql
select t.first_name, t.last_name
from teacher as t left join
    courses as c on t.teacher_id = c.teacher_id
    where c.course_id is null;
```

Output

first_name	last_name
Rahul	Bose
Aditya	Roy

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.

Query

```
select avg(student_count) as avg_student_per_course
from (
select course_id, count(student_id) as student_count from enrollments
group by course_id) as course_count;
```

Output

```
17  07:56:27  select avg(student_count) as avg_student_per_course from  ( select course_id, count(student_id) as student_co...  1 row(s) returned
```

avg_student_per_course
1.9000

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.

Query

```
with student_totals as (
    select student_id, sum(amount) as total_payment
    from payments
    group by student_id)
select student_id, total_payment
from student_totals
where total_payment = ( select max(total_payment) from student_totals);
```

Output

```
33  08:28:11  with student_totals as ( select student_id, sum(amount) as total_payment  from payments  group by student_id...  1 row(s) returned
```

student_id	total_payment
103	62000

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

Query

```
with total_students as
(select course_id, count(student_id) as student_count
from enrollments
group by course_id)
select course_id, student_count from total_students
where student_count = (
    select max(student_count) from total_students);
```

Output

```
37  08:35:21  with total_students as (select course_id, count(student_id) as student_count from enrollments group by course_id...  4 row(s) returned
```

course_id	student_count
201	3
202	3
203	3
204	3

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

Query

```
select t.teacher_id, t.first_name,t.last_name,
ifnull((
select sum(p.amount)
from courses as c
join enrollments as e
on c.course_id = e.course_id
join payments as p
on e.student_id = p.student_id
where c.teacher_id = t.teacher_id), 0) as total_payments
from teacher t;
```

Output

```
17  11:19:41  select t.teacher_id, t.first_name,t.last_name, ifnull(( select sum(p.amount) from courses as c join enrollments as e...  11 row(s) returned
```

	teacher_id	first_name	last_name	total_payments
▶	401	Rajesh	Khanna	252500
	402	Priyanka	Chopra	107500
	403	Vikram	Seth	137000
	404	Sunita	Rao	62000
	405	Arjun	Kapoor	54000
	406	Meera	Patel	58000
	407	Sanjay	Verma	62000
	408	Anjali	Desai	54000
	409	Rahul	Bose	0
	410	Deepika	Sharma	28000
	411	Aditya	Roy	0

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

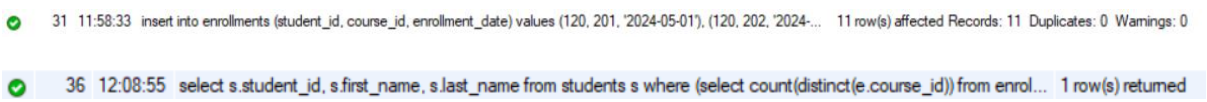
Query

Inserted the values first since no students is enrolled in all available courses

```
insert into enrollments (student_id, course_id, enrollment_date) values
(120, 201, '2024-05-01'),
(120, 202, '2024-05-02'),
(120, 203, '2024-05-03'),
(120, 204, '2024-05-04'),
(120, 205, '2024-05-05'),
(120, 208, '2024-05-21'),
(120, 209, '2024-05-22'),
(120, 210, '2024-05-23'),
(120, 211, '2024-05-22'),
(120, 212, '2024-05-09'),
(120, 213, '2024-05-25');
```

```
select s.student_id, s.first_name, s.last_name from students s
where (select count(distinct(e.course_id))
        from enrollments as e
        where e.student_id = s.student_id
    )=(select count(*) from courses);
```

Output



student_id	first_name	last_name
120	John	Doe
NULL	NULL	NULL

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

Query

```
select t.teacher_id, t.first_name,t.last_name from teacher as t
where not exists (
    select 1 from courses as c
    where c.teacher_id = t.teacher_id);
```

Output

50 12:57:02 select t.teacher_id, t.first_name,t.last_name from teacher as t where not exists (select 1 from courses as c wh... 2 row(s) returned

teacher_id	first_name	last_name
409	Rahul	Bose
411	Aditya	Roy
NULL	NULL	NULL

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

Query

```
select avg(student_age) as average_age
from (
    select
       timestampdiff(year, s.date_of_birth, CURDATE()) as student_age
    from students s
) as age_sub;
```

Output

51 13:03:03 select avg(student_age) as average_age from (select timestampdiff(year, s.date_of_birth, CURDATE()) as... 1 row(s) returned

average_age
22.9167

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

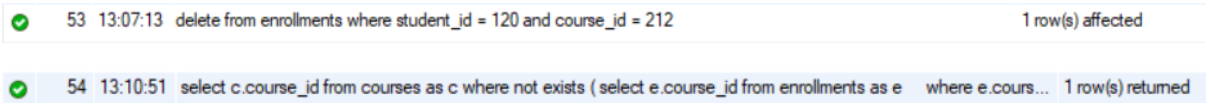
Query

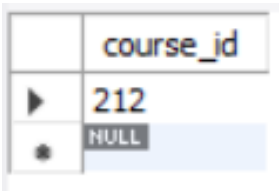
Deleting the course before identifying the course with no enrollments since every course is enrolled.

```
delete from enrollments where student_id = 120 and course_id = 212;

  select c.course_id from courses as c
where not exists (
    select e.course_id from enrollments as e
    where e.course_id = c.course_id);
```

Output





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

Query

```sql
select
    e.student_id,
    e.course_id,
    ifnull((
        select SUM(p.amount)
        from payments p
        where p.student_id = e.student_id
          and exists (
              select 1
              from enrollments e2
              where e2.student_id = p.student_id
              and e2.course_id = e.course_id
          )
    ),0) as total_payment
from enrollments e
group by e.student_id, e.course_id;
```

Output

```
61  13:28:13  select  e.student_id,  e.course_id,  ifnull((  select SUM(p.amount)  from payments p  where p.s...  31 row(s) returned
```

student_id	course_id	total_payment
101	201	53000
101	203	53000
101	204	53000
102	201	25000
102	203	25000
103	205	62000
103	209	62000
104	202	30000
105	206	28000
107	204	27000
108	201	58000
108	208	58000
109	202	57000
109	204	57000
110	207	54000
110	210	54000
111	202	29500
111	203	29500

student_id	course_id	total_payment
110	207	54000
110	210	54000
111	202	29500
111	203	29500
112	205	25000
120	201	0
120	202	0
120	203	0
120	204	0
120	205	0
120	206	0
120	207	0
120	208	0
120	209	0
120	210	0
120	211	0
120	213	0

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.

Query

```
select student_id, first_name, last_name
from students
where student_id in (
    select student_id
    from payments
    group by student_id
    having COUNT(*) > 1
);
```

Output

```
65  13:49:37  select student_id, first_name, last_name from students where student_id in (  select student_id  from payment...  5 row(s) returned
```

student_id	first_name	last_name
101	Aarav	Sharma
103	Rahul	Mehta
108	Neha	Joshi
109	Vikram	Das
110	Priya	Nair

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.

Query

```
select s.student_id, s.first_name, s.last_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, s.last_name;
```

Output

```
66  13:54:49  select s.student_id, s.first_name, s.last_name, sum(p.amount) as total_payments from students s join payments p ...  11 row(s) returned
```

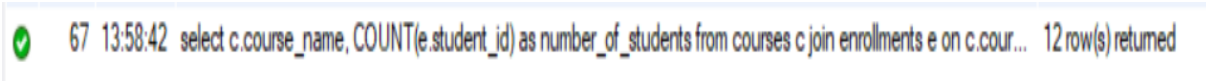
	student_id	first_name	last_name	total_payments
▶	101	Aarav	Sharma	53000
	102	Isha	Verma	25000
	103	Rahul	Mehta	62000
	104	Sneha	Patil	30000
	105	Karan	Singh	28000
	107	Amit	Kumar	27000
	108	Neha	Joshi	58000
	109	Vikram	Das	57000
	110	Priya	Nair	54000
	111	Rohan	Malhotra	29500
	112	Ananya	Gupta	25000

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.

Query

```
select c.course_name, COUNT(e.student_id) as number_of_students
from courses c
join enrollments e on c.course_id = e.course_id
group by c.course_name;
```

Output



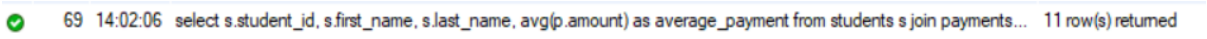
	course_name	number_of_students
▶	Introduction to Computer Science	4
	Advanced Programming	4
	Database Systems	4
	Data Structures and Algorithms	4
	Operating Systems	3
	Computer Networks	2
	Artificial Intelligence	2
	Web Development	2
	Machine Learning	2
	Cybersecurity Fundamentals	2
	Cloud Computing	1
	Mobile App Development	1

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.

Query

```sql
select s.student_id, s.first_name, s.last_name,
avg(p.amount) as average_payment from students s
join payments p on s.student_id = p.student_id
group by s.student_id, s.first_name, s.last_name;
```

Output



student_id	first_name	last_name	average_payment
101	Aarav	Sharma	26500.0000
102	Isha	Verma	25000.0000
103	Rahul	Mehta	31000.0000
104	Sneha	Patil	30000.0000
105	Karan	Singh	28000.0000
107	Amit	Kumar	27000.0000
108	Neha	Joshi	29000.0000
109	Vikram	Das	28500.0000
110	Priya	Nair	27000.0000
111	Rohan	Malhotra	29500.0000
112	Ananya	Gupta	25000.0000