


# ASSIGNMENT

## (STUDENT INFORMATION SYSTEM)

### Task 1. Database Design:

1. Create the database named "SISDB"

*create database SISD;*

O/P:  5 14:19:44 create database SISD

1 row(s) affected

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.

NOTE: I have used **Use SISD** command to make sure that I don't need to specify the database name every time

#### a. Students

```
CREATE TABLE Students(Student_id int(20) primary key,  
first_name varchar(30),  
last_name varchar(30),  
date_of_birth date,  
email varchar(30),  
phone_number varchar(10));
```

#### b. Courses

```
create table Courses(course_id int(20) primary key,  
course_name varchar(30),  
credits int(10),  
teacher_id int(30));
```

#### c. Enrollments

```
create table Enrollments(enrollment_id int(20) primary key,  
student_id int(20),  
course_id int(20),  
enrollment_date date);
```

#### d. Teacher

```
create table Teacher(teacher_id int(20),  
first_name varchar(30),  
last_name varchar(30),  
email varchar(30));
```

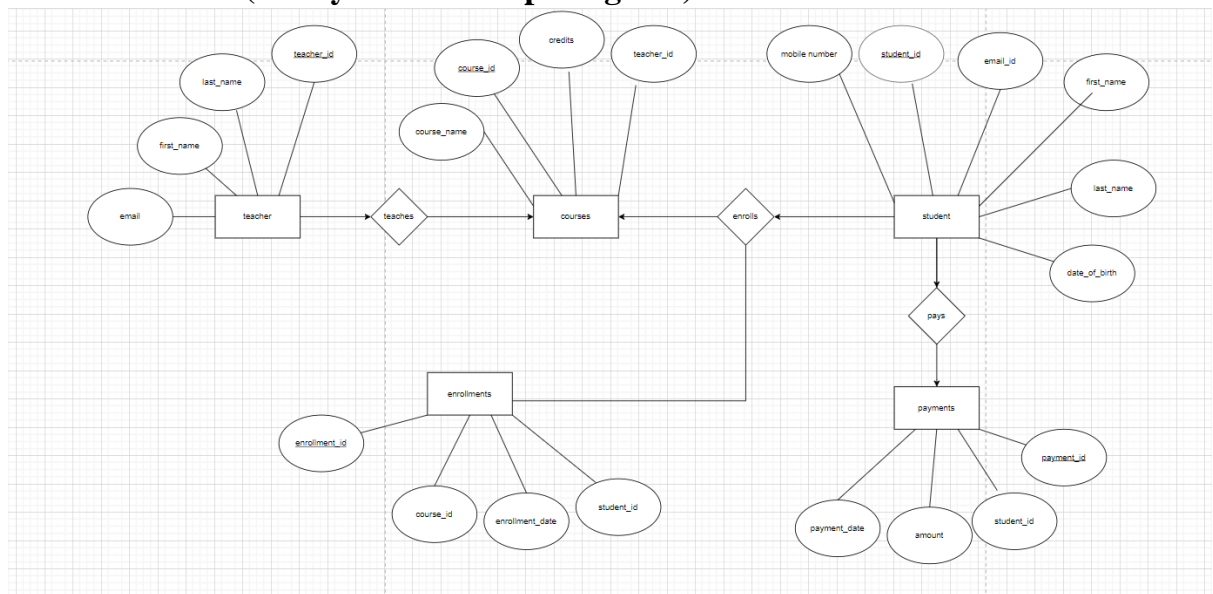
#### e. Payments

```
create table Payments(payment_id int(20) primary key,  
student_id int(20),  
amount int(20),  
payment_date date);
```

o/p:

#	Time	Action	Message	Duration / Fetch
8	14:22:18	CREATE TABLE Students(student_id int(20) primary key, first_name varchar(30), last_name varchar(30), date...	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release...	0.032 sec
9	14:24:29	create table Courses(course_id int(20) primary key, course_name varchar(30), credits int(10), teacher_id int(30))	0 row(s) affected, 3 warning(s): 1681 Integer display width is deprecated and will be removed in a future release...	0.015 sec
10	14:26:23	create table Enrollments(enrollment_id int(20) primary key, student_id int(20), course_id int(20), enrollment_date ...	0 row(s) affected, 3 warning(s): 1681 Integer display width is deprecated and will be removed in a future release...	0.032 sec
11	14:27:40	create table Teacher(teacher_id int(20), first_name varchar(30), last_name varchar(30), email varchar(30))	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release...	0.031 sec
13	14:32:07	create table Payments(payment_id int(20) primary key, student_id int(20), amount int(20), payment_date date)	0 row(s) affected, 3 warning(s): 1681 Integer display width is deprecated and will be removed in a future release...	0.015 sec

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



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

Now we should specify foreign key constraints for the tables

For courses :Adding teacher\_id as foreign key

*alter table courses add constraint fk1courses*

*foreign key(teacher\_id) references Teacher(teacher\_id);*

For Enrollments: adding student\_id as foreign key

*alter table enrollments*

*add constraint fk1enroll*

*foreign key(student\_id) references students(student\_id);*

For Enrollments: adding course\_id from courses as foreign key

*alter table enrollments*

*add constraint fk2enroll*

*foreign key(course\_id) references courses(course\_id);*

For payments:adding student\_id from students table as foreign key

*alter table payments*

*add constraint fk1payments*

*foreign key(student\_id) references students(student\_id);*

*o/p:*

✓	19	14:38:14	alter table courses add constraint fk1courses foreign key(teacher_id) references Teacher(teacher_id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.094 sec
✓	25	14:42:42	alter table enrollments add constraint fk2enroll foreign key(course_id) references courses(course_id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.078 sec
✓	26	14:44:09	alter table payments add constraint fk1payments foreign key(student_id) references students(student_id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.078 sec
✓	22	14:42:03	alter table enrollments add constraint fk1enroll foreign key(student_id) references students(student_id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.078 sec

Schema of tables now will be:

Students:

	Field	Type	Null	Key	Default	Extra
►	Student_id	int	NO	PRI	NULL	
	first_name	varchar(30)	YES		NULL	
	last_name	varchar(30)	YES		NULL	
	date_of_birth	date	YES		NULL	
	email	varchar(30)	YES		NULL	
	phone_number	varchar(10)	YES		NULL	

Courses:

	Field	Type	Null	Key	Default	Extra
►	course_id	int	NO	PRI	NULL	
	course_name	varchar(30)	YES		NULL	
	credits	int	YES		NULL	
	teacher_id	int	YES	MUL	NULL	

*Teacher:*

	Field	Type	Null	Key	Default	Extra
►	teacher_id	int	NO	PRI	NULL	
	first_name	varchar(30)	YES		NULL	
	last_name	varchar(30)	YES		NULL	
	email	varchar(30)	YES		NULL	

*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	

*Payments:*

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

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

#### **i. Students**

*INSERT INTO Students VALUES*

*(501,'pavan','kumar',str\_to\_date('10-5-2001','%d-%m-%Y'),'pavan@123',9848222101),*

*(* *502,'Soma','Shekar',STR\_to\_date('2002-08-15','%Y-%m-%d'),'soma@123.com',1234567802),*

*(503,'Akhilesh','Kumar',str\_to\_date('2001-11-20','%Y-%m-%d'),'akhilesh@123.com',1234567803')*

(504, 'Sai', 'Krishna', STR\_TO\_DATE('2002-03-25', '%Y-%m-%d'), 'sai@123.com', '1234567804'),

(505, 'Rani', 'Devi', STR\_TO\_DATE('2001-09-30', '%Y-%m-%d'), 'rani@123.com', '1234567805'),

(506, 'Anjali', 'Sharma', STR\_TO\_DATE('2003-07-05', '%Y-%m-%d'), 'anjali@123.com', '1234567806'),

(507, 'Jasmine', 'Kaur', STR\_TO\_DATE('2002-01-15', '%Y-%m-%d'), 'jasmine@123.com', '1234567807'),

(508, 'Priya', 'Patel', STR\_TO\_DATE('2001-04-20', '%Y-%m-%d'), 'priya@123.com', '1234567808'),

(509, 'Rahul', 'Sharma', STR\_TO\_DATE('2002-06-12', '%Y-%m-%d'), 'rahul@123.com', '1234567809'),

(510, 'Vijay', 'Kumar', STR\_TO\_DATE('2001-10-18', '%Y-%m-%d'), 'vijay@123.com', '1234567810'),

(511, 'Ganesh', 'Patil', STR\_TO\_DATE('2003-02-28', '%Y-%m-%d'), 'ganesh@123.com', '1234567811'),

(512, 'Amit', 'Singh', STR\_TO\_DATE('2001-07-03', '%Y-%m-%d'), 'amit@123.com', '1234567812'),

(513, 'Ravi', 'Reddy', STR\_TO\_DATE('2002-11-07', '%Y-%m-%d'), 'ravi@123.com', '1234567813'),

(514, 'Ajay', 'Khan', STR\_TO\_DATE('2003-05-15', '%Y-%m-%d'), 'ajay@123.com', '1234567814'),

(515, 'Deepak', 'Verma', STR\_TO\_DATE('2001-08-22', '%Y-%m-%d'), 'deepak@123.com', '1234567815'),

(516, 'Sneha', 'Gupta', STR\_TO\_DATE('2001-12-08', '%Y-%m-%d'), 'sneha@123.com', '1234567816'),

(517, 'Kavita', 'Choudhary', STR\_TO\_DATE('2002-04-27', '%Y-%m-%d'), 'kavita@123.com', '1234567817'),

(518, 'Neha', 'Yadav', STR\_TO\_DATE('2003-09-14', '%Y-%m-%d'), 'neha@123.com', '1234567818');

## ii. Courses

*INSERT INTO Courses VALUES*

(1, 'Java Full Stack', 3, 15201),

(2, 'Node.js', 4, 15202),

(3, 'React.js', 3, 15203),  
(4, 'Python', 2, 15204),  
(5, 'Artificial Intelligence (AI)', 4, 15205),  
(6, 'Machine Learning (ML)', 3, 15206),  
(7, 'Cyber Security', 4, 15207),  
(8, 'Ethical Hacking', 3, 15208),  
(9, 'C#', 4, 15209),  
(10, 'Salesforce', 2, 15210);

### iii. Enrollments

*INSERT INTO enrollments VALUES*

(1, 501, 5, STR\_TO\_DATE('2020-07-12', '%Y-%m-%d')),  
(2, 501, 7, STR\_TO\_DATE('2021-05-29', '%Y-%m-%d')),  
(3, 501, 3, STR\_TO\_DATE('2021-11-18', '%Y-%m-%d')),  
(4, 501, 2, STR\_TO\_DATE('2022-08-03', '%Y-%m-%d')),  
(5, 501, 8, STR\_TO\_DATE('2022-12-20', '%Y-%m-%d')),  
(6, 502, 9, STR\_TO\_DATE('2020-09-05', '%Y-%m-%d')),  
(7, 502, 4, STR\_TO\_DATE('2021-03-17', '%Y-%m-%d')),  
(8, 502, 1, STR\_TO\_DATE('2021-10-22', '%Y-%m-%d')),  
(9, 503, 3, STR\_TO\_DATE('2020-11-30', '%Y-%m-%d')),  
(10, 503, 10, STR\_TO\_DATE('2021-07-08', '%Y-%m-%d')),  
(11, 504, 6, STR\_TO\_DATE('2022-01-15', '%Y-%m-%d')),  
(12, 504, 1, STR\_TO\_DATE('2022-05-27', '%Y-%m-%d')),  
(13, 505, 7, STR\_TO\_DATE('2020-10-10', '%Y-%m-%d')),  
(14, 505, 10, STR\_TO\_DATE('2021-04-03', '%Y-%m-%d')),  
(15, 505, 5, STR\_TO\_DATE('2021-12-12', '%Y-%m-%d')),  
(16, 506, 2, STR\_TO\_DATE('2020-08-25', '%Y-%m-%d')),  
(17, 506, 6, STR\_TO\_DATE('2021-02-14', '%Y-%m-%d')),  
(18, 507, 9, STR\_TO\_DATE('2020-06-08', '%Y-%m-%d')),  
(19, 507, 3, STR\_TO\_DATE('2021-01-28', '%Y-%m-%d')),

```

(20, 508, 1, STR_TO_DATE('2020-09-20', '%Y-%m-%d')),
(21, 509, 8, STR_TO_DATE('2022-02-03', '%Y-%m-%d')),
(22, 510, 7, STR_TO_DATE('2022-06-19', '%Y-%m-%d')),
(23, 511, 5, STR_TO_DATE('2020-12-05', '%Y-%m-%d')),
(24, 512, 2, STR_TO_DATE('2021-05-15', '%Y-%m-%d')),
(25, 513, 4, STR_TO_DATE('2021-11-25', '%Y-%m-%d')),
(26, 514, 6, STR_TO_DATE('2020-07-22', '%Y-%m-%d')),
(27, 514, 8, STR_TO_DATE('2021-03-10', '%Y-%m-%d')),
(28, 515, 3, STR_TO_DATE('2020-11-14', '%Y-%m-%d')),
(29, 515, 1, STR_TO_DATE('2021-07-05', '%Y-%m-%d')),
(30, 516, 10, STR_TO_DATE('2020-08-18', '%Y-%m-%d')),
(31, 517, 4, STR_TO_DATE('2021-02-25', '%Y-%m-%d')),
(32, 518, NULL, NULL);

```

#### iv. Teacher

```

INSERT INTO Teacher(teacher_id, first_name, last_name, email)
VALUES
(15201, 'Satya', 'Sharma', 'satya123@gmail.com'),
(15202, 'Nageswar Rao', 'D', 'nageswar235@gmail.com'),
(15203, 'Lakshmi', 'Kumar', 'lakshmi723@hotmail.com'),
(15204, 'Uma', 'Sankar', 'uma123@gmail.com'),
(15205, 'Manikanteswari', 'Reddy', 'manikanteswari235@gmail.com'),
(15206, 'Nagavallika', 'N', 'nagavallika723@gmail.com'),
(15207, 'Hiranmayee', 'M', 'hiranmayee123@gmail.com'),
(15208, 'Ujwala', 'Sai', 'ujwala235@hotmail.com'),
(15209, 'Sahil', 'Yadav', 'sahil123@gmail.com'),
(15210, 'Nagaraju', 'S', 'rohan235@gmail.com'),
(15211, 'Anjali', 'Chowdary', 'anjali723@gmail.com'),

```

(15212, 'Karthik', 'M', 'karthik123@gmail.com');

v. Payments

```
INSERT INTO Payments (payement_id, student_id, amount,
payment_date) VALUES
```

```
(13459, 501, 100, STR_TO_DATE('2022-02-15', '%Y-%m-%d')),
(13460, 502, 150, STR_TO_DATE('2022-04-20', '%Y-%m-%d')),
(13461, 503, 200, STR_TO_DATE('2022-06-10', '%Y-%m-%d')),
(13462, 504, 120, STR_TO_DATE('2022-08-05', '%Y-%m-%d')),
(13463, 505, 180, STR_TO_DATE('2022-10-22', '%Y-%m-%d')),
(13464, 506, 220, STR_TO_DATE('2022-12-15', '%Y-%m-%d')),
(13465, 507, 130, STR_TO_DATE('2023-01-18', '%Y-%m-%d')),
(13466, 508, 170, STR_TO_DATE('2023-03-07', '%Y-%m-%d')),
(13467, 509, 210, STR_TO_DATE('2023-05-10', '%Y-%m-%d')),
(13468, 510, 140, STR_TO_DATE('2023-07-03', '%Y-%m-%d')),
(13469, 511, 190, STR_TO_DATE('2023-09-25', '%Y-%m-%d')),
(13470, 512, 230, STR_TO_DATE('2023-11-12', '%Y-%m-%d')),
(13471, 513, 150, STR_TO_DATE('2023-01-05', '%Y-%m-%d')),
(13472, 514, 200, STR_TO_DATE('2023-03-28', '%Y-%m-%d')),
(13473, 515, 250, STR_TO_DATE('2023-05-15', '%Y-%m-%d'));
```

**TASK-2:**

**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](mailto:john.doe@example.com)**

**e. Phone Number: 1234567890**

```
insert into students values(519,'John','Doe',str_to_date('1995-08-15','%Y-%m-%d'),'john.doe@example.com','1234567890');
```

o/p:1 row(s) affected



**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**

```
insert into enrollments values(33,519,2,str_to_date('21-03-2024','%d-%m-%Y'));
```

*o/p:Query OK, 1 row affected (0.00 sec)*

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

```
update teacher set email='satyasharma@gmail.com' where teacher_id=15201;
```

*o/p:Query OK, 1 row affected (0.01 sec)*

*Rows matched: 1 Changed: 1 Warnings: 0*

**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.

```
delete from enrollments where student_id=507 and course_id=3;
```

*o/p:Query OK, 1 row affected (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**

```
update courses set teacher_id=15211 where course_id=10;
```

*Query OK, 1 row affected (0.00 sec)*

*Rows matched: 1 Changed: 1 Warnings: 0*

**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**

```
delete from enrollments where student_id=517;
```

*/\*deleting enrollments data so we don't get referential integrity problem*

*Query OK, 1 row affected (0.00 sec)*

```
mysql> delete from students where student_id=517;
```

*/\* deleting from students table;*

*Query OK, 1 row affected (0.00 sec)*

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

*update payments set amount=10000 where student\_id=510;*

*Query OK, 1 row affected (0.00 sec)*

*Rows matched: 1 Changed: 1 Warnings: 0*

### 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

```
select s.student_id,sum(amount)
from students s join payments p
on s.student_id=p.student_id
group by student_id;
```

student_id	sum(amount)
501	100
502	150
503	200
504	120
505	180
506	220
507	130
508	170
509	210
510	10000
511	190
512	230
513	150
514	200
515	250

15 rows in set (0.00 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.

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

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

course_name	course_id	count(student_id)
Java Full Stack	1	4
Node.js	2	4
React.js	3	3
Python	4	2
Artificial Intelligence (AI)	5	3
Machine Learning (ML)	6	3
Cyber Security	7	3
Ethical Hacking	8	3
C#	9	2
Salesforce	10	3

```
10 rows in set (0.00 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.

```
select s.student_id from students s left join enrollments e
```

```
-> on s.student_id=e.student_id
```

```
-> where e.student_id is NULL;
```

```
+-----+
| student_id |
+-----+
|          518 |
+-----+
1 row 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.

```
select s.first_name,s.last_name,c.course_id,c.course_name
```

```
from students s join enrollments e on s.student_id=e.student_id
```

```
join courses c on c.course_id=e.course_id;
```

first_name	last_name	course_id	course_name
Soma	Shekar	1	Java Full Stack
Sai	Krishna	1	Java Full Stack
Priya	Patel	1	Java Full Stack
Deepak	Verma	1	Java Full Stack
pavan	kumar	2	Node.js
Anjali	Sharma	2	Node.js
Amit	Singh	2	Node.js
John	Doe	2	Node.js
pavan	kumar	3	React.js
Akhilesh	Kumar	3	React.js
Deepak	Verma	3	React.js
Soma	Shekar	4	Python
Ravi	Reddy	4	Python
pavan	kumar	5	Artificial Intelligence (AI)
Rani	Devi	5	Artificial Intelligence (AI)
Ganesh	Patil	5	Artificial Intelligence (AI)
Sai	Krishna	6	Machine Learning (ML)
Anjali	Sharma	6	Machine Learning (ML)
Ajay	Khan	6	Machine Learning (ML)
pavan	kumar	7	Cyber Security
Rani	Devi	7	Cyber Security
Vijay	Kumar	7	Cyber Security
pavan	kumar	8	Ethical Hacking
Rahul	Sharma	8	Ethical Hacking
Ajay	Khan	8	Ethical Hacking
Soma	Shekar	9	C#
Jasmine	Kaur	9	C#
Akhilesh	Kumar	10	Salesforce
Rani	Devi	10	Salesforce
Sneha	Gupta	10	Salesforce

30 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

```
select t.teacher_id,t.first_name,last_name ,course_name
```

```
-> from teacher t join courses c on c.teacher_id=t.teacher_id;
```

teacher_id	first_name	last_name	course_name
15201	Satya	Sharma	Java Full Stack
15202	Nageswar Rao	D	Node.js
15203	Lakshmi	Kumar	React.js
15204	Uma	Sankar	Python
15205	Manikanteswari	Reddy	Artificial Intelligence (AI)
15206	Nagavallika	N	Machine Learning (ML)
15207	Hiranmayee	M	Cyber Security
15208	Ujwala	Sai	Ethical Hacking
15209	Sahil	Yadav	C#
15211	Anjali	Chowdary	Salesforce

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**

*/\*FOR ALL COURSE*

*select s.student\_id,s.first\_name,s.last\_name*

*-> ,c.course\_name,e.enrollment\_date*

*-> from students s join enrollments e on e.student\_id=s.student\_id*

*-> join courses c on e.course\_id=c.course\_id;*

student_id	first_name	last_name	course_name	enrollment_date
502	Soma	Shekar	Java Full Stack	2021-10-22
504	Sai	Krishna	Java Full Stack	2022-05-27
508	Priya	Patel	Java Full Stack	2020-09-20
515	Deepak	Verma	Java Full Stack	2021-07-05
501	pavan	kumar	Node.js	2022-08-03
506	Anjali	Sharma	Node.js	2020-08-25
512	Amit	Singh	Node.js	2021-05-15
519	John	Doe	Node.js	2024-03-21
501	pavan	kumar	React.js	2021-11-18
503	Akhilesh	Kumar	React.js	2020-11-30
515	Deepak	Verma	React.js	2020-11-14
502	Soma	Shekar	Python	2021-03-17
513	Ravi	Reddy	Python	2021-11-25
501	pavan	kumar	Artificial Intelligence (AI)	2020-07-12
505	Rani	Devi	Artificial Intelligence (AI)	2021-12-12
511	Ganesh	Patil	Artificial Intelligence (AI)	2020-12-05
504	Sai	Krishna	Machine Learning (ML)	2022-01-15
506	Anjali	Sharma	Machine Learning (ML)	2021-02-14
514	Ajay	Khan	Machine Learning (ML)	2020-07-22
501	pavan	kumar	Cyber Security	2021-05-29
505	Rani	Devi	Cyber Security	2020-10-10
510	Vijay	Kumar	Cyber Security	2022-06-19
501	pavan	kumar	Ethical Hacking	2022-12-20
509	Rahul	Sharma	Ethical Hacking	2022-02-03
514	Ajay	Khan	Ethical Hacking	2021-03-10
502	Soma	Shekar	C#	2020-09-05
507	Jasmine	Kaur	C#	2020-06-08
503	Akhilesh	Kumar	Salesforce	2021-07-08
505	Rani	Devi	Salesforce	2021-04-03
516	Sneha	Gupta	Salesforce	2020-08-18

30 rows in set (0.00 sec)

*/\* for specific course*

*select s.student\_id,s.first\_name,s.last\_name*

```
,c.course_name,e.enrollment_date
from students s join enrollments e on e.student_id=s.student_id
join courses c on e.course_id=c.course_id
where c.course_id=2;
```

student_id	first_name	last_name	course_name	enrollment_date
501	pavan	kumar	Node.js	2022-08-03
506	Anjali	Sharma	Node.js	2020-08-25
512	Amit	Singh	Node.js	2021-05-15
519	John	Doe	Node.js	2024-03-21

4 rows 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

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

```
mysql> select s.student_id,s.first_name,s.last_name
-> from students s left join payments p
-> on s.student_id=p.student_id
-> where p.payment_date is null;
```

student_id	first_name	last_name
516	Sneha	Gupta
518	Neha	Yadav
519	John	Doe

3 rows in set (0.00 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

```
select c.course_id,c.course_name
from courses c left join enrollments e
on c.course_id=e.course_id
where e.enrollment_date is NULL;
```

o/p: Empty 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.

```
select e1.student_id from
-> enrollments e1 join enrollments e2
-> on e1.student_id=e2.student_id
-> group by e1.student_id
-> having count(e2.course_id)>1;
```

student_id
501
502
503
504
505
506
514
515

8 rows 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.

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

teacher_id	first_name	last_name
15210	Nagaraju	S
15212	Karthik	M

2 rows in 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.

```
select (sum(k1)/count(c.course_id)) as ans
  from (select e.course_id,count(e.student_id) as k1
        from enrollments e
        group by course_id) as k,courses c;
```

o/p:

ans
3.0000

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.

```
select *from students s
  where s.student_id in(
    select student_id
    from payments
    group by student_id
    having max(amount)=(select max(amount) from payments));
```

o/p:

Student_id	first_name	last_name	date_of_birth	email	phone_number
510	Vijay	Kumar	2001-10-18	vijay@123.com	1234567810

1 row in set (0.00 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

```
select course_id,course_name
  from courses
  where course_id in(
    select e.course_id from enrollments e
    group by e.course_id
    having count(e.course_id)=(select max(c) from (select course_id,count(course_id) as c from
    enrollments group by course_id) as k));
```



course_id	course_name
1	Java Full Stack
2	Node.js

2 rows in set (0.00 sec)

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

```
select t.teacher_id, sum(amount)
from teacher t join courses c on c.teacher_id=t.teacher_id
join enrollments e on e.course_id= c.course_id
join students s on s.student_id = e.student_id
join payments p on p.student_id=s.student_id
group by t.teacher_id;
```

teacher_id	sum(amount)
15205	470
15207	10280
15203	550
15202	550
15208	510
15209	280
15204	300
15201	690
15211	380
15206	540

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.**

```
select student_id, concat(first_name, ' ', last_name) as name
from students where student_id in(
select e.student_id from enrollments e
group by e.student_id
having count(enrollment_id)=(select count(course_id) from courses) );
```

O/P: Empty set (0.01 sec)

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

```
select teacher_id,concat(first_name,' ',last_name)
from teacher t
where t.teacher_id not in(
select teacher_id from courses c);
```

*o/p:*

```
+-----+-----+
| teacher_id | concat(first_name,' ',last_name) |
+-----+-----+
|      15210 | Nagaraju S                       |
|      15212 | Karthik M                       |
+-----+-----+
2 rows in set (0.00 sec)
```

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

```
select avg(age) from (
select timestampdiff(year,date_of_birth,curdate()) as age from students) as s;
```

```
+-----+
| avg(age) |
+-----+
|    21.7778 |
+-----+
1 row in set (0.00 sec)
```

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

```
select * from courses
where course_id not in(
select distinct course_id from enrollments);
```

course_id	course_name	credits	teacher_id
11	sql	4	15204

1 row in set (0.00 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

```
select s.student_id,concat(s.first_name,' ',s.last_name) as name,sum1 from
(select student_id,sum(amount) as sum1
from payments p
group by p.student_id) as k,students s
where k.student_id=s.student_id;
```

student_id	name	sum1
501	pavan kumar	100
502	Soma Shekar	1650
503	Akhilesh Kumar	200
504	Sai Krishna	120
505	Rani Devi	180
506	Anjali Sharma	220
507	Jasmine Kaur	130
508	Priya Patel	170
509	Rahul Sharma	210
510	Vijay Kumar	10000
511	Ganesh Patil	190
512	Amit Singh	230
513	Ravi Reddy	150
514	Ajay Khan	200
515	Deepak Verma	250

15 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

```
select s.student_id,concat(s.first_name,' ',s.last_name) as name
from students s
where student_id in(
select student_id from payments p
group by p.student_id
having count(p.student_id)>1);
```

student_id	name
502	Soma Shekar

1 row in 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

```
select s.student_id,concat(s.first_name,' ',s.last_name) as name,sum(amount) from
students s join payments p
on s.student_id=p.student_id
group by p.student_id;
```

student_id	name	sum(amount)
501	pavan kumar	100
502	Soma Shekar	1650
503	Akhilesh Kumar	200
504	Sai Krishna	120
505	Rani Devi	180
506	Anjali Sharma	220
507	Jasmine Kaur	130
508	Priya Patel	170
509	Rahul Sharma	210
510	Vijay Kumar	10000
511	Ganesh Patil	190
512	Amit Singh	230
513	Ravi Reddy	150
514	Ajay Khan	200
515	Deepak Verma	250

15 rows in set (0.00 sec)

11. 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.

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

course_name	number_of_students
Java Full Stack	4
Node.js	4
React.js	3
Python	2
Artificial Intelligence (AI)	3
Machine Learning (ML)	3
Cyber Security	3
Ethical Hacking	3
C#	2
Salesforce	3

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

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

student_id	avg(amount)
501	100.0000
502	825.0000
503	200.0000
504	120.0000
505	180.0000
506	220.0000
507	130.0000
508	170.0000
509	210.0000
510	10000.0000
511	190.0000
512	230.0000
513	150.0000
514	200.0000
515	250.0000

15 rows in set (0.00 sec)