

Student Information System (SIS)

Database Tables:

The SIS database consists of the following tables:

1. Students

- student_id (Primary Key)
- first_name
- last_name
- date_of_birth
- email
- phone_number

2. Courses

- course_id (Primary Key)
- course_name
- credits
- teacher_id (Foreign Key)

3. Enrollments

- enrollment_id (Primary Key)
- student_id (Foreign Key)
- course_id (Foreign Key)
- enrollment_date

4. Teacher

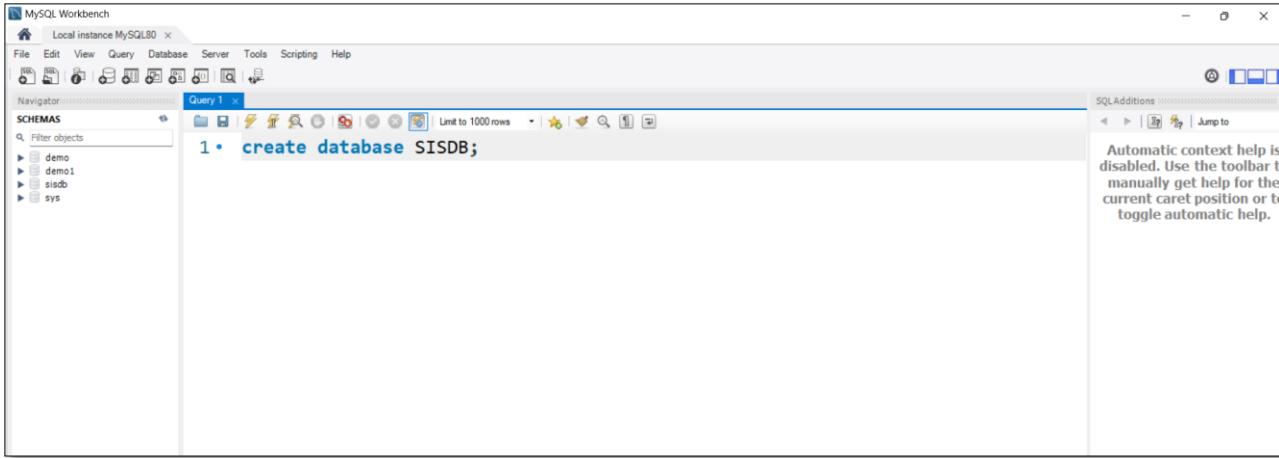
- teacher_id (Primary Key)
- first_name
- last_name
- email

5. Payments

- payment_id (Primary Key)
- student_id (Foreign Key)
- amount
- payment_date

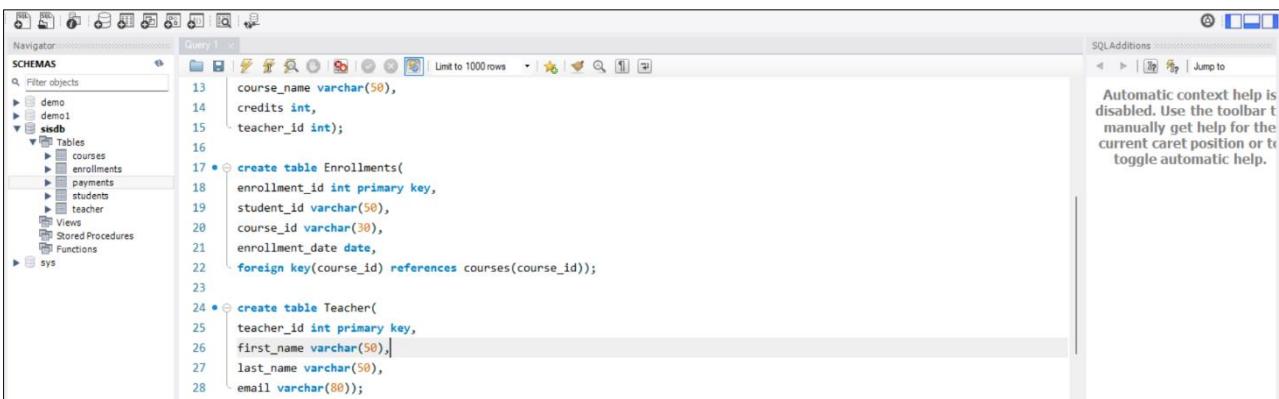
Task 1. Database Design:

1. Create the database named "SISDB"



The screenshot shows the MySQL Workbench interface. In the top-left corner, it says "Local instance MySQL80". The "Navigator" pane on the left lists several schemas: demo, demo1, sisdb, and sys. The "Query 1" pane contains the SQL command: `1 • create database SISDB;`. The status bar at the bottom of the query pane says "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

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



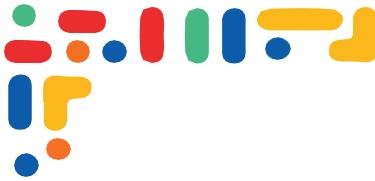
The screenshot shows the MySQL Workbench interface. The "Navigator" pane on the left shows the "Tables" section under the sisdb schema, which includes courses, enrollments, payments, students, and teacher tables. The "Query 1" pane contains the following SQL code:

```

13 course_name varchar(50),
14 credits int,
15 teacher_id int);
16
17 • create table Enrollments(
18 enrollment_id int primary key,
19 student_id varchar(50),
20 course_id varchar(30),
21 enrollment_date date,
22 foreign key(course_id) references courses(course_id));
23
24 • create table Teacher(
25 teacher_id int primary key,
26 first_name varchar(50),
27 last_name varchar(50),
28 email varchar(80));

```

The status bar at the bottom of the query pane says "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."



MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

SCHEMAS demo demo1 sisdb

Tables: courses, enrollments, payments, students, teacher, Views, Stored Procedures, Functions

sys

Query 1

```
1 • create database SISDB;
2 • create table Students(
3     student_id int,
4     first_name varchar(50),
5     last_name varchar(60),
6     date_of_birth varchar(15),
7     email varchar(80),
8     phone_number int,
9     primary key(student_id));
10 • alter table Students modify date_of_birth date;
11
12 • create table Courses(
13     course_id varchar(30) primary key,
14     course_name varchar(50),
15     credits int,
16     teacher_id int);
17
18 • create table Enrollments(
```

Output:

#	Time	Action	Message	Duration / Fetch
7	09:51:25	create table Enrollments(enrollment_id int primary key,student_id varchar(50),course_id varchar(30),enrollment_date...	0 row(s) affected	0.031 sec
8	09:55:28	create table Teacher(teacher_id int primary key, first_name varchar(50), last_name varchar(50), email varchar(80))	0 row(s) affected	0.031 sec
9	09:57:52	create table Payments(payment_id int primary key, student_id int, amount int, payment_date date, foreign key(stude...	Error Code: 3780. Referencing column 'student_id' and referenced column 'student_id' in foreign key constraint 'pay...	0.000 sec
10	09:58:13	create table Payments(payment_id int primary key, student_id int, amount int, payment_date date, foreign key(stude...	0 row(s) affected	0.031 sec
11	10:09:19	alter table Students modify date_of_birth date	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.094 sec

Object Info Session

1 29°C Haze

ENG IN 10:09 21-04-2024

Functions sys

Administration Schemas

Information

Table: students

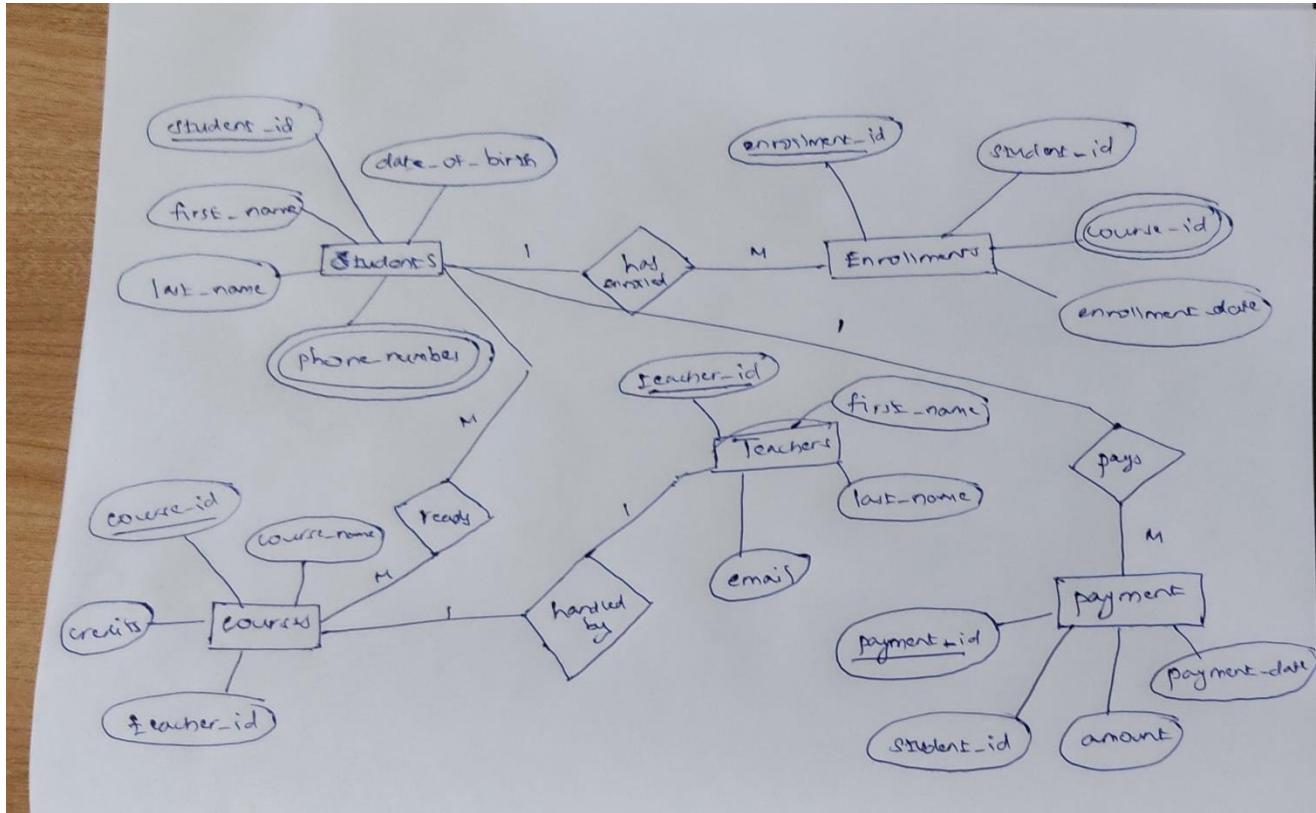
Columns: student_id int PK, first_name varchar(50), last_name varchar(60), date_of_birth date, email varchar(80), phone_number int

Output:

#	Time	Action	Message	Duration / Fetch
7	09:51:25	create table Enrollments(enrollment_id int primary key,student_id varchar(50),course_id varchar(30),enrollment_date...	0 row(s) affected	0.031 sec
8	09:55:28	create table Teacher(teacher_id int primary key, first_name varchar(50), last_name varchar(50), email varchar(80))	0 row(s) affected	0.031 sec
9	09:57:52	create table Payments(payment_id int primary key, student_id int, amount int, payment_date date, foreign key(stude...	Error Code: 3780. Referencing column 'student_id' and referenced column 'student_id' in foreign key constraint 'pay...	0.000 sec
10	09:58:13	create table Payments(payment_id int primary key, student_id int, amount int, payment_date date, foreign key(stude...	0 row(s) affected	0.031 sec
11	10:09:19	alter table Students modify date_of_birth date	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.094 sec

Object Info Session

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



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

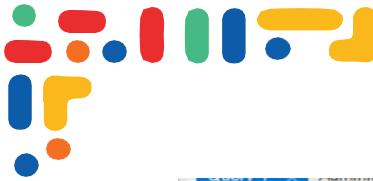
```

38
39 • alter table courses add constraint fk foreign key(teacher_id) references Teacher(teacher_id);
40
41

```

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

- Students
- Courses
- Enrollments
- Teacher
- Payments



HEXAWARE

```
52     values(101,'Rani','K','2009-11-2','swaze@gmail.com',7658901265);
53
54 •   insert into Students
55     values(146,'Swetha','Ram','2012-6-2','ss1234@gmail.com',7658952165);
56
57 •   insert into Students
58     values
59     (111,'Vani','Sambath','2005-04-3','vanis@gmail.com',1248901261),
60     (117,'Surya','Murthy','1978-3-1','surya@yahoo.com',9996754312),
61     (124,'Vikram','Anderson','1890-3-8','vikramrst45@gmail.com',8236782165),
62     (130,'Madhu','Ram','2012-8-3','mdram345@gmail.com',7658954321),
63     (115,'Kavin','Singh','2016-8-6','ss1234@gmail.com',7658952165),
64     (143,'Alad','Khan','2017-5-1','khanaala@yahoo.com',9465123216);
65
66
67 •   insert into Teacher
68     values
69     (123,'Vara','Lakshmi','vj20@gmail.com'),
```



```
80
81 •   insert into courses
82     values
83     ('19CB101','Digital marketing', 15 ,112 ),
84     ('19CB105','Operating System', 17 ,133 ),
85     ('201CS224','Networking', 30 , 150),
86     ('202CS134','Automata Theory', 20 , 123),
87     ('201IS134','Financial Management', 10 , 174),
88     ('201IS568','IT Project Management', 35 , 160),
89     ('201ME134','CAD Modelling', 25 , 128),
90     ('201ME136','Statistics', 35 , 124),
91     ('18CB145','Boolean Algebra', 25 , 132),
92     ('19CB111','Data Structures and Algorithms', 50 , 137),
93     ('19EE678','Principles of Electronic Engineering', 15 , 118);
94
95 •   insert into Enrollments
96     values
```

Output :


```
104
105 •   insert into Enrollments values
106     (301889,130,'18CB145','2001-10-01'),
107     (435612,124,'201IS568','2002-07-31');
108
109 •   insert into Enrollments values
110     (456765,115,'201CS224','2001-09-01');
111
112 •   insert into Enrollments values
113     (45765,146,'19CB105','2001-09-01');
114
115 •   insert into Enrollments values
116     (455765,80,'201ME134','2000-12-11');
117
118 •   insert into Enrollments values
119     (345123,100,'19CB105','2004-05-01');
```

```
116      (455765,80,'201ME134','2000-12-11');  
117  
118 •  insert into Enrollments values  
119      (345123,100,'19CB105','2004-05-01');  
120  
121  
122 •  insert into payments  
123      values (123456,100,1000,'2004-05-02'),  
124      (123457,80,2000,'2000-12-12'),  
125      (123500,130,1500,'2001-11-01'),  
126      (123459,100,3000,'2004-05-02'),  
127      (123501,124,1500,'2003-07-1'),  
128      (123600,117,2000,'2007-07-05'),  
129      (123455,111,3500,'2001-08-02');  
130
```

```
66  
67 •  insert into Teacher  
68      values  
69      (123,'Vara','Lakshmi','vj20@gmail.com'),  
70      (124,'Swarna','Latha','Swarnal342@gmail.com'),  
71      (128,'Rama','Swamy','Rs1234@gmail.com'),  
72      (133,'Jay','Shetty','jswere34@gmail.com'),  
73      (132,'Keerthi','Pandian','kpkp2020@gmail.com'),  
74      (137,'Srinikethan','M','srinik@yahoo.com'),  
75      (150,'Manoj','Kumar','heromj@gmail.com'),  
76      (118,'Saravana','Moorthy','sv20sv@gmail.com'),  
77      (160,'Rani Lakshmi','Bhai','bhai1987@gmail.com'),  
78      (174,'Cheran','K','chek99@yahoo.com'),  
79      (112,'Sri','Ram','srm198@gmail.com');  
80
```

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

```
135  
136 •  insert into students values(190,'John','Doe','1995-08-15','john.doe@example.com',1234567890);  
137
```

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.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with tables: triggers, payments, students, and teacher. The main area shows the following SQL code:

```

137
138 • insert into enrollments
139     values(123457,80,'2011S134','2000-05-01');

```

The 'Output' tab shows the execution results in a table:

#	Time	Action	Message	Duration / Fetch
45	10:41:31	insert into payments values (1236032,146,1000,'2004-05-02),(123667,115,2000,2...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.016 sec
46	10:41:51	select * from payments LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
47	10:44:38	insert into student values('190','John','Doe','1995-08-15');john.doe@example.com','12...	Error Code: 1146. Table 'ssid.student' doesn't exist	0.000 sec
48	10:44:50	insert into students values('190','John','Doe','1995-08-15');john.doe@example.com','1...	1 row(s) affected	0.015 sec
49	10:50:02	insert into enrollments values(123457,80,2011S134,2000-05-01)	Error Code: 1292. Incorrect date value: '1994' for column 'enrollment_date' at row 1	0.000 sec
50	10:50:26	insert into enrollments values(123457,80,2011S134,'2000-05-01')	1 row(s) affected	0.016 sec

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

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with tables: Foreign Keys, Triggers, payments, students, teacher, and teachers. The main area shows the following SQL code:

```

141 • update Teacher set email='vara145@gmail.com' where teacher_id=123;

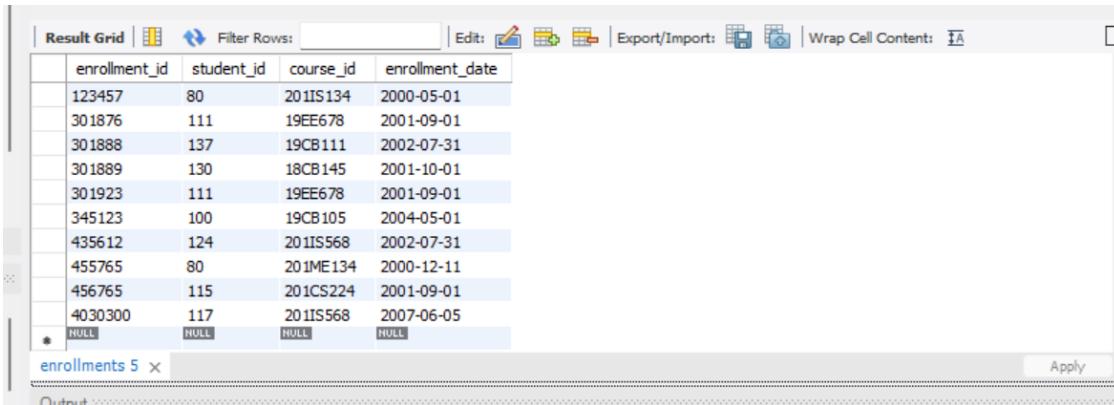
```

The 'Output' tab shows the execution results in a table:

#	Time	Action	Message	Duration / Fetch
48	10:44:50	insert into students values('190','John','Doe','1995-08-15');john.doe@example.com','1...	1 row(s) affected	0.015 sec
49	10:50:02	insert into enrollments values(123457,80,2011S134,2000-05-01)	Error Code: 1292. Incorrect date value: '1994' for column 'enrollment_date' at row 1	0.000 sec
50	10:50:26	insert into enrollments values(123457,80,2011S134,'2000-05-01')	1 row(s) affected	0.016 sec
51	10:53:08	update Teachers set email='vara145@gmail.com' where email='vj20@gmail.com'	Error Code: 1146. Table 'ssid.teachers' doesn't exist	0.000 sec
52	10:53:25	update Teacher set email='vara145@gmail.com' where email='vj20@gmail.com'	Error Code: 1175. You are using safe update mode and you tried to update a table ...	0.000 sec
53	10:54:42	update Teacher set email='vara145@gmail.com' where teacher_id=123	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 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.

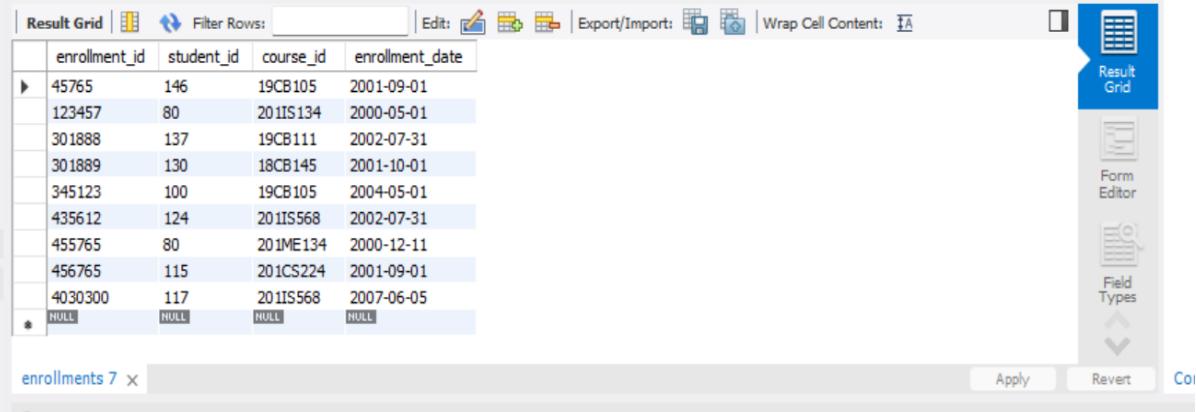
Before deleting:



	enrollment_id	student_id	course_id	enrollment_date
1	123457	80	201IS134	2000-05-01
2	301876	111	19EEE678	2001-09-01
3	301888	137	19CB111	2002-07-31
4	301889	130	18CB145	2001-10-01
5	301923	111	19EEE678	2001-09-01
6	345123	100	19CB105	2004-05-01
7	435612	124	201IS568	2002-07-31
8	455765	80	201ME134	2000-12-11
9	456765	115	201CS224	2001-09-01
10	4030300	117	201IS568	2007-06-05
*	NULL	NULL	NULL	NULL

After deleting

```
143
144 •  delete from enrollments where student_id = 111 and course_id='19EE678';
145 •  SELECT*FROM enrollments;
146
```



	enrollment_id	student_id	course_id	enrollment_date
1	45765	146	19CB105	2001-09-01
2	123457	80	201IS134	2000-05-01
3	301888	137	19CB111	2002-07-31
4	301889	130	18CB145	2001-10-01
5	345123	100	19CB105	2004-05-01
6	435612	124	201IS568	2002-07-31
7	455765	80	201ME134	2000-12-11
8	456765	115	201CS224	2001-09-01
9	4030300	117	201IS568	2007-06-05
*	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.

Before updating

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

course_id	course_name	credits	teacher_id
19CB101	Digital marketing	15	112
19CB105	Operating System	17	133
19CB111	Data Structures and Algorithms	50	137
19EE678	Principles of Electronic Engineering	15	118
201CS224	Networking	30	150
201IS134	Financial Management	10	174
201IS568	IT Project Management	35	160
201ME134	CAD Modelling	25	128
201ME136	Statistics	35	124
202CS134	Automata Theory	20	123
NULL	NULL	NULL	NULL

Courses 8 X Apply

Output:

After updating

File | Select | Insert | Delete | Find | Replace | Refresh | Limit to 1000 rows | Favorites | Help |

```
147 • select*from Courses;
148
149 • update Courses
150     set teacher_id= '112'
151     where course_id= '202CS134';
152
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

course_id	course_name	credits	teacher_id
18CB145	Boolean Algebra	25	132
19CB101	Digital marketing	15	112
19CB105	Operating System	17	133
19CB111	Data Structures and Algorithms	50	137
19EE678	Principles of Electronic Engineering	15	118
201CS224	Networking	30	150
201IS134	Financial Management	10	174
201IS568	IT Project Management	35	160
201ME134	CAD Modelling	25	128
201ME136	Statistics	35	124
202CS134	Automata Theory	20	112

Courses 10 X Apply Revert

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

To maintain referential integrity, I have modified the foreign key constraint.

```

159
160 • alter table payments
161 drop foreign key payments_ibfk_1;
162
163 • alter table payments
164 add constraint payments_ibfk_2
165 foreign key (student_id)
166 references students(student_id)
167 on delete cascade;
168
169 • delete from students
170 where student_id = 80;
171
172
173
174
175

```

After deleting

Payments table

	payment_id	student_id	amount	payment_date
▶	122500	143	1500	2001-05-01
	123455	111	3500	2001-08-02
	123456	100	1000	2004-05-02
	123459	100	3000	2004-05-02
	123500	130	1500	2001-11-01
	123501	124	1500	2003-07-01
	123600	117	2000	2007-07-05
	123667	115	2000	2001-06-01
	1236032	146	1000	2004-05-02
*	HULL	HULL	HULL	HULL

Students table

	student_id	first_name	last_name	date_of_birth	email	phone_number
▶	100	Rama	Manoj	2005-12-01	rama123@gmail.com	9976234165
	101	Rani	K	2009-11-02	swaze@gmail.com	7658901265
	111	Vani	Sambath	2005-04-03	vanis@gmail.com	1248901261
	115	Kavin	Singh	2016-08-06	ss1234@gmail.com	7658952165
	117	Surya	Murthy	1978-03-01	surya@yahoo.com	9996754312
	124	Vikram	Anderson	1890-03-08	vikramrst45@gmail.com	8236782165
	130	Madhu	Ram	2012-08-03	mdram345@gmail.com	7658954321
	143	Alad	Khan	2017-05-01	khanaala@yahoo.com	9465123216
	146	Swetha	Ram	2012-06-02	ss1234@gmail.com	7658952165
	190	John	Doe	1995-08-15	john.doe@example.com	1234567890
*	HULL	HULL	HULL	HULL	HULL	HULL

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

Before updating

	payment_id	student_id	amount	payment_date
▶	122500	143	1500	2001-05-01
	123455	111	3500	2001-08-02
	123456	100	1000	2004-05-02
	123459	100	3000	2004-05-02
	123500	130	1500	2001-11-01
	123501	124	1500	2003-07-01
	123600	117	2000	2007-07-05
	123667	115	2000	2001-06-01
	1236032	146	1000	2004-05-02
*	NULL	NULL	NULL	NULL

payments 14 ×

After updating

	payment_id	student_id	amount	payment_date
▶	122500	143	1500	2001-05-01
	123455	111	3500	2001-08-02
	123456	100	1000	2004-05-02
	123459	100	3000	2004-05-02
	123500	130	1500	2001-11-01
	123501	124	1500	2003-07-01
▶	123600	117	1200	2007-07-05
	123667	115	2000	2001-06-01
	1236032	146	1000	2004-05-02
*	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.

Students table

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	student_id	first_name	last_name	date_of_birth	email	phone_number
▶	100	Rama	Manoj	2005-12-01	rama123@gmail.com	9976234165
	101	Rani	K	2009-11-02	swaze@gmail.com	7658901265
	111	Vani	Sambath	2005-04-03	vanis@gmail.com	1248901261
	115	Kavin	Singh	2016-08-06	ss1234@gmail.com	7658952165
	117	Surya	Murthy	1978-03-01	surya@yahoo.com	9996754312
	124	Vikram	Anderson	1890-03-08	vikramrst45@gmail.com	8236782165
	130	Madhu	Ram	2012-08-03	mdram345@gmail.com	7658954321
	143	Alad	Khan	2017-05-01	khanaala@yahoo.com	9465123216
	146	Swetha	Ram	2012-06-02	ss1234@gmail.com	7658952165
	190	John	Doe	1995-08-15	john.doe@example.com	1234567890
*	NULL	NULL	NULL	NULL	NULL	NULL

students 16 x payments 17 | Output:

Payments table

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	payment_id	student_id	amount	payment_date
▶	122500	143	1500	2001-05-01
	123455	111	3500	2001-08-02
	123456	100	1000	2004-05-02
	123459	100	3000	2004-05-02
	123500	130	1500	2001-11-01
	123501	124	1500	2003-07-01
	123600	117	1200	2007-07-05
	123667	115	2000	2001-06-01
	1236032	146	1000	2004-05-02
*	NULL	NULL	NULL	NULL

After join operation

```

178
179 •   select s.student_id,s.first_name,sum(p.amount) as amount from students as s inner join
180     payments as p on s.student_id=p.student_id group by s.student_id;
181
182

```

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

	student_id	first_name	amount
▶	100	Rama	4000
	111	Vani	3500
	115	Kavin	2000
	117	Surya	1200
	124	Vikram	1500
	130	Madhu	1500
	143	Alad	1500
	146	Swetha	1000

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.

```
235  
236 •  select c.course_name, count(e.student_id) as number_of_students  
237   from courses as c  
238   inner join enrollments as e  
239   on c.course_id = e.course_id  
240   group by c.course_id;  
241
```

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

	course_name	number_of_students
▶	Operating System	2
	Financial Management	1
	Data Structures and Algorithms	1
	Boolean Algebra	1
	IT Project Management	2
	CAD Modelling	1
	Networking	1

Result 7 x

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.

```
186 •  select students.first_name, students.last_name  
187   from Students  
188   left join Enrollments ON Students.student_id = Enrollments.student_id  
189   where Enrollments.student_id IS NULL;  
190  
191
```

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

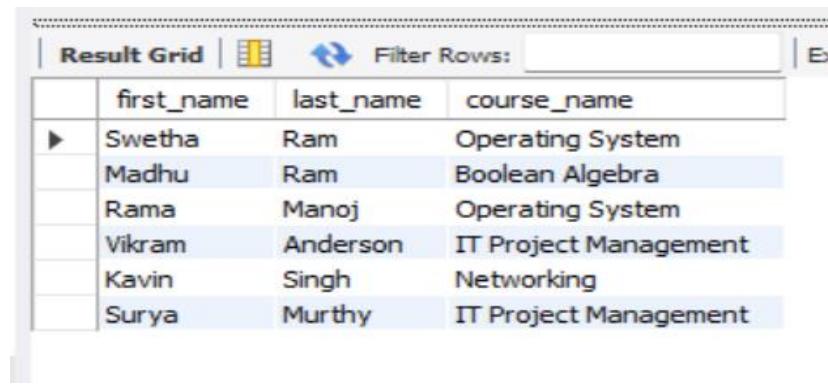
	first_name	last_name
▶	Rani	K
	Vani	Sambath
	Alad	Khan
	John	Doe
	Kili	M
	Sanjeev	Kumar

Result 1 x

Read Only

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.

```
170  
191 • select s.first_name,s.last_name,c.course_name  
192   from students as s  
193   inner join enrollments as e  
194   on e.student_id=s.student_id  
195   inner join Courses as c  
196   on c.course_id = e.course_id;  
197  
198  
199
```



The screenshot shows a database result grid with the following data:

	first_name	last_name	course_name
▶	Swetha	Ram	Operating System
	Madhu	Ram	Boolean Algebra
	Rama	Manoj	Operating System
	Vikram	Anderson	IT Project Management
	Kavin	Singh	Networking
	Surya	Murthy	IT Project Management

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.

```
197  
198 • select t.first_name,t.last_name,c.course_name  
199   from teacher as t  
200   inner join courses as c  
201   on c.teacher_id = t.teacher_id;  
202  
203  
204
```

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

	first_name	last_name	course_name
▶	Sri	Ram	Digital marketing
	Sri	Ram	Automata Theory
	Saravana	Moorthy	Principles of Electronic Engineering
	Swarna	Latha	Statistics
	Rama	Swamy	CAD Modelling
	Keerthi	Pandian	Boolean Algebra
	Jay	Shetty	Operating System
	Srinikethan	M	Data Structures and Algorithms
	Manoj	Kumar	Networking
	Rani Lakshmi	Bhai	IT Project Management
	Cheran	K	Financial Management

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 1 | Result Grid | Filter Rows: Export: Wrap Cell Content:

```
203 • select s.first_name,e.enrollment_date,c.course_name
204   from students as s inner join
205     enrollments as e
206   on s.student_id=e.student_id
207   inner join courses as c |
208     on e.course_id = c.course_id;
?QQ
```

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

	first_name	enrollment_date	course_name
▶	Swetha	2001-09-01	Operating System
	Madhu	2001-10-01	Boolean Algebra
	Rama	2004-05-01	Operating System
	Vikram	2002-07-31	IT Project Management
	Kavin	2001-09-01	Networking
	Surya	2007-06-05	IT Project Management

Result 2 | Read Only

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 1

```

209
210 •   select s.first_name
211     from students as s
212     left join payments as p
213       on s.student_id = p.student_id
214     where p.amount is null;
215

```

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

first_name
Rani
John
Kili
Sanjeev

Result 3

Action Output

#	Time	Action	Message
2	09:36:45	select s.first_name, e.enrollment_date, c.course_name from students as s inner join ...	Error Code: 1054. Unknown column 'e.enrollment_date' in Field ...
3	09:37:13	select s.first_name, e.enrollment_date, c.course_name from students as s inner join ...	6 row(s) returned
4	09:38:17	select s.first_name, e.enrollment_date, c.course_name from students as s inner join ...	6 row(s) returned
5	09:42:47	select s.first_name from students as s left join payments as p on s.student_id = p.s...	4 row(s) returned

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 2

```

215
216
217 •   select c.course_name from courses as c |
218     left join enrollments as e
219       on c.course_id = e.course_id
220     where e.course_id is null;
221

```

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

course_name
Digital marketing
Principles of Electronic Engineering
Statistics
Automata Theory

Result 4

Read Only

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.

```
223 •  select e1.student_id, count(e1.course_id) as num_courses_enrolled
224   from enrollments as e1
225   inner join enrollments as e2 on e1.student_id = e2.student_id
226   where e1.course_id >> e2.course_id
227   group by e1.student_id
228   having count(e1.course_id) > 1;
229
```

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content:

student_id	num_courses_enrolled
80	2

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.

```
230 •  select t.first_name
231   from teacher as t
232   left join courses as c
233   on t.teacher_id = c.teacher_id
234   where c.course_id is null;
235
236
```

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content:

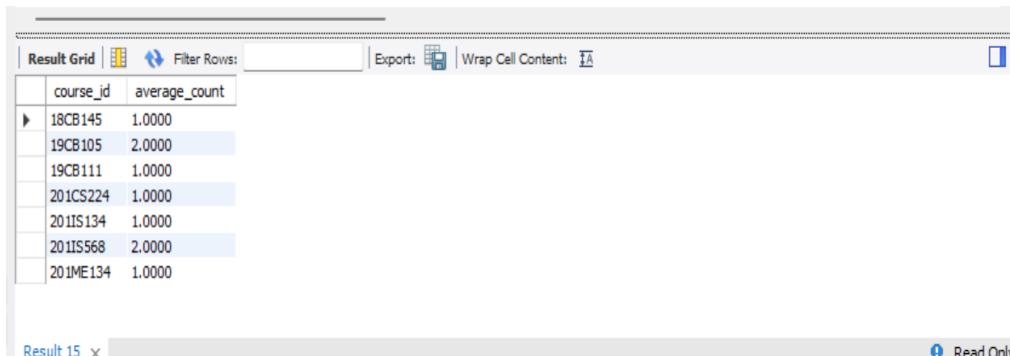
first_name
Vara

Result 6 x Read Only

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.

```
241  
242 •  select course_id, avg(student_count) as average_count  
243   ⏺ from (  
244     select course_id, count(student_id) as student_count  
245       from enrollments  
246      group by course_id  
247   ) as course_counts  
248   group by course_id;  
249
```



course_id	average_count
18CB145	1.0000
19CB105	2.0000
19CB111	1.0000
201CS224	1.0000
201IS134	1.0000
201IS568	2.0000
201ME134	1.0000

Avg returns values in decimal point. Here 1 is denoted as 1.0000

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.

```
1  
2 •  select s.student_id, s.first_name, s.last_name, p.amount  
3   from students as s  
4   inner join payments as p on s.student_id = p.student_id  
5   where p.amount =  
5     select MAX(amount)  
6       from payments  
7   ;  
8  
9  
L
```

Result Grid				
	student_id	first_name	last_name	amount
▶	111	Vani	Sambath	3500

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

```

258 •   select c.course_name, max(e.enroll) as max_enroll
259   ⋮   from courses as c inner join (
260     ⋮       select course_id, count(*) as enroll
261     ⋮         from enrollments
262     ⋮           group by course_id
263   ⋮ ) as e on c.course_id = e.course_id
264   ⋮   group by c.course_name
265   ⋮   having max(e.enroll) = (
266     ⋮       select max(enroll_count)
267   ⋮       from (
268         ⋮           select count(*) as enroll_count
269         ⋮             from enrollments
270         ⋮               group by course_id
271     ⋮   ) as subquery
272   ⋮ );
273

```

Result Grid		
	course_name	max_enroll
▶	Operating System	2
	IT Project Management	2

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

```

259 •   select t.teacher_id, t.first_name as teacher_first_name,
260   t.last_name as teacher_last_name,sum(p.amount) as total_payments
261   from teacher t
262   inner join courses c
263   on t.teacher_id = c.teacher_id
264   inner join enrollments e
265   on c.course_id = e.course_id
266   inner join payments p
267   on e.student_id = p.student_id
268   group by t.teacher_id, t.first_name, t.last_name;
269
270

```

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

teacher_id	teacher_first_name	teacher_last_name	total_payments
133	Jay	Shetty	5000
132	Keerthi	Pandian	1500
160	Rani Lakshmi	Bhai	2700
150	Manoj	Kumar	2000

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

```

258
259 •   select
260     s.student_id,
261     s.first_name,
262     s.last_name
263   from
264     students s
265   where |
266     (select count(distinct course_id) from courses) =
267     (select count(distinct course_id)
268      from enrollments e
269      where e.student_id = s.student_id);
270
271
272

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

student_id	first_name	last_name
HULL	HULL	HULL

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

```

•   select t.first_name
    from teacher as t
    where t.teacher_id not in
    (
      select c.teacher_id
      from courses as c);

```

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

first_name
Vara

teacher 24 x Read Only

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

The screenshot shows a database query editor window titled "Query 1". The query code is as follows:

```
270
271 • select avg(age) as average_age
272   from
273   ( select
274     round(datediff(current_date(), date_of_birth) / 365, 0) as age
275   from students
276   ) as student_age;
277
```

The result grid shows one row with the value 29.3333 under the column "average_age".

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

The screenshot shows a database query editor window. The query code is as follows:

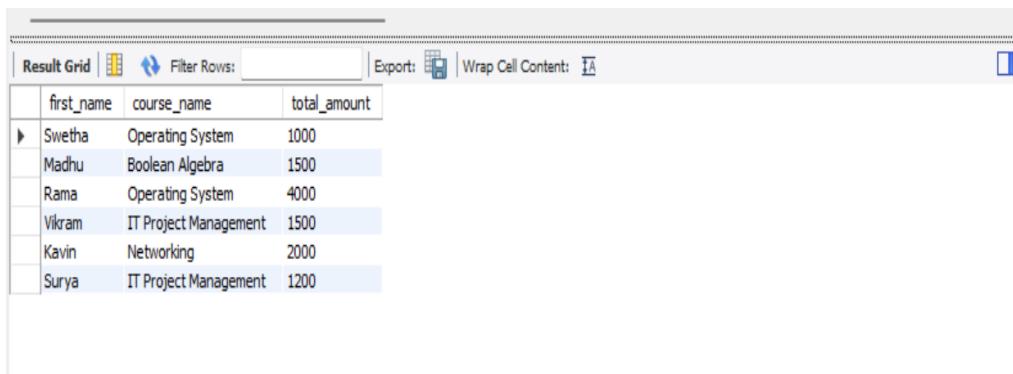
```
277
278 • select c.course_id, c.course_name
279   from courses c
280   where
281   c.course_id not in (
282   select distinct course_id
283   from enrollments );
```

The result grid shows the following data:

course_id	course_name
19CB101	Digital marketing
19EE678	Principles of Electronic Engineering
201ME136	Statistics
202CS134	Automata Theory
*	NULL

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.first_name, c.course_name,
  ⊂ (   select sum(p.amount)
      from payments p |
      where p.student_id = s.student_id
      and e.course_id = c.course_id
    ) as total_amount
  from students s
  inner join
    enrollments e on s.student_id = e.student_id
  inner join
    courses c on e.course_id = c.course_id;
```



The screenshot shows a database result grid with three columns: first_name, course_name, and total_amount. The data is as follows:

	first_name	course_name	total_amount
▶	Swetha	Operating System	1000
	Madhu	Boolean Algebra	1500
	Rama	Operating System	4000
	Vikram	IT Project Management	1500
	Kavin	Networking	2000
	Surya	IT Project Management	1200

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

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
		first_name	last_name	
▶		Rama	Manoj	

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, 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;
```

| Result Grid |            |            | Filter Rows:   | Export: | Wrap Cell Content: |
|-------------|------------|------------|----------------|---------|--------------------|
|             | student_id | first_name | total_payments |         |                    |
| ▶           | 143        | Alad       | 1500           |         |                    |
|             | 111        | Vani       | 3500           |         |                    |
|             | 100        | Rama       | 4000           |         |                    |
|             | 130        | Madhu      | 1500           |         |                    |
|             | 124        | Vikram     | 1500           |         |                    |
|             | 117        | Surya      | 1200           |         |                    |
|             | 115        | Kavin      | 2000           |         |                    |
|             | 146        | Swetha     | 1000           |         |                    |

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.

```
308
309 • select c.course_name, count(e.student_id) as count_students
310 from courses as c
311 left join enrollments as e
312 on c.course_id = e.course_id
313 group by course_name;
314
315
316
```

315

| Result Grid |                                      | Filter Rows:   | Export: | Wrap Cell Content: |
|-------------|--------------------------------------|----------------|---------|--------------------|
|             | course_name                          | count_students |         |                    |
| ▶           | Boolean Algebra                      | 1              |         |                    |
|             | Digital marketing                    | 0              |         |                    |
|             | Operating System                     | 2              |         |                    |
|             | Data Structures and Algorithms       | 1              |         |                    |
|             | Principles of Electronic Engineering | 0              |         |                    |
|             | Networking                           | 1              |         |                    |
|             | Financial Management                 | 1              |         |                    |
|             | IT Project Management                | 2              |         |                    |
|             | CAD Modelling                        | 1              |         |                    |
|             | Statistics                           | 0              |         |                    |
|             | Automata Theory                      | 0              |         |                    |

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.

```
315 • select avg(p.amount)
316 from payments p
317 left join students s on p.student_id = s.student_id;
318
319
```

315

| Result Grid |               | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|---------------|--------------|---------|--------------------|
|             | avg(p.amount) |              |         |                    |
| ▶           | 1800.0000     |              |         |                    |