University Database

A database management system (DBMS) is a collection of interconnected data and a set of applications for accessing that data. A database is a collection of data that contains information relevant to a company. A database management system's primary purpose is to provide an easy and efficient method for storing and retrieving database information.

Full Schema

classroom (building, room_number, capacity)

department(dept_name, building, budget)

course(course_id, title, dept_name, credits)

instructor(ID, name, dept_name, salary)

section(course_id, sec_id, semester, year, building, room_number, time_slot_id)

teaches(ID, course_id, sec id, semester, year)

student(ID, name, dept name, tot_cred)

takes(ID, course_id, sec_id, semester, year, grade)

advisor(s_ID, i_ID)

time slot(time_slot_id, day, start_time, end_time)

prereq(course_id, prereq_id)

Figure: Schema of the university database.

The university database is a critical system designed to handle interconnected information necessary for both academic and administrative functions. It encompasses key components such as **Classroom**, which holds data about physical spaces, including building names and room capacities; **Department**, which manages academic units based on names, locations, and budgets; and **Course**, which provides details like course IDs, titles, and credit hours.

The **Instructor** entity captures faculty details and associates them with their respective departments, while the **Section** entity specifies course offerings for each semester, along with classroom allocations. The **Teaches** table defines the connection between instructors and the sections they lead, ensuring clarity in teaching assignments.

Students are represented in the **Student** entity, which stores information such as student IDs, names, and total earned credits. The **Takes** table monitors student course enrollments and records their grades. Moreover, the **Advisor** entity oversees student-faculty advising relationships, improving academic guidance and support.

The **Time Slot** entity schedules classes, while the **Prereq** table outlines the prerequisites for courses. This comprehensive database streamlines the management of academic processes, maintains data consistency, and supports the institution's educational objectives, thereby enriching the experiences of both students and faculty.

Important Considerations

1. **Data Integrity**: Ensure foreign key relationships are correctly implemented to maintain consistency across tables, such as linking dept_name in course to the department table.

2. **Normalization**: Regularly review the schema to minimize redundancy and avoid duplicating information across multiple tables

3. **Access Control**: Implement strict access controls to protect sensitive data, especially student records and financial information.

4. **Backup and Recovery**: Establish a regular backup schedule and a clear recovery process to safeguard data against loss.

5. **Performance Optimization**: Monitor and optimize database performance to efficiently handle high query volumes, particularly during registration periods.

6. **Scalability**: Design the schema to accommodate growth, allowing for easy addition of courses, students, and departments.

7. **Documentation**: Maintain clear documentation of the schema and entity relationships to facilitate maintenance and future development.

Schema Diagram for the University Database

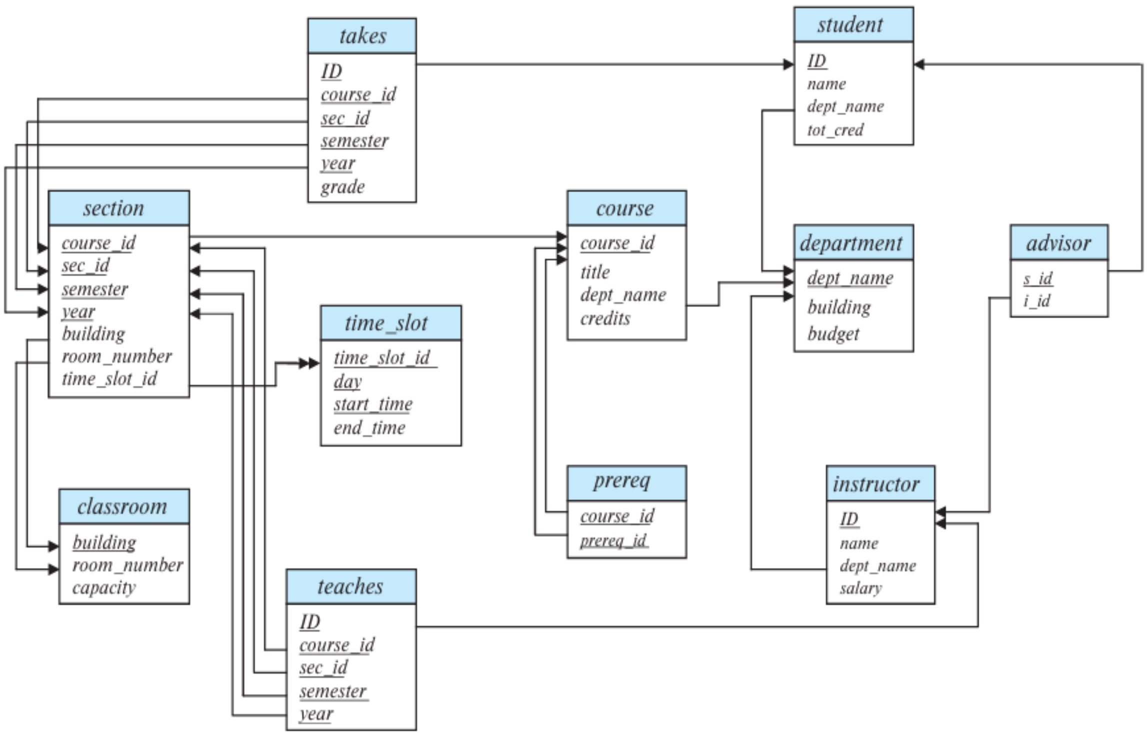


Figure: Schema diagram for the university database.

E-R Diagram for University Database

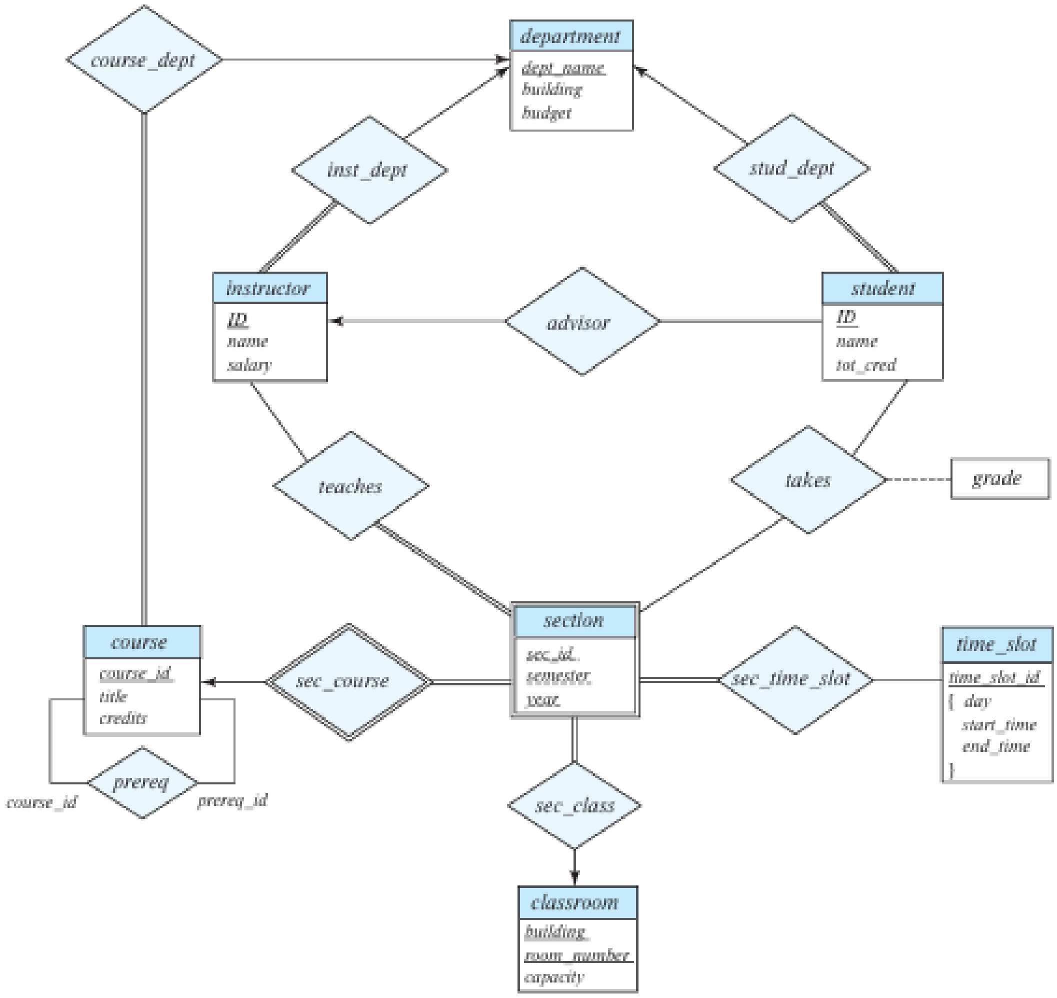


Figure: E-R diagram for a university database.

Problem no – 01

Problem Name – Write SQL queries using integrity constraints to create tables for a database.

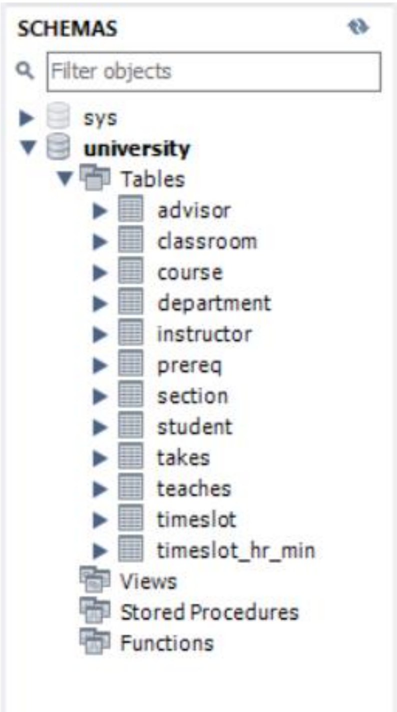
Query –

```
CREATE TABLE classroom (
    building VARCHAR(15),
    room_number VARCHAR(7),
    capacity NUMERIC(4,0),
    PRIMARY KEY (building, room_number));
CREATE TABLE department (
    dept_name VARCHAR(20),
    building VARCHAR(15),
    budget NUMERIC(12,2) CHECK (budget > 0),
    PRIMARY KEY (dept_name),
    FOREIGN KEY (building) REFERENCES classroom(building));
CREATE TABLE course (
    course_id VARCHAR(7),
    title VARCHAR(50),
    dept_name VARCHAR(20),
    credits NUMERIC(2,0) CHECK (credits > 0),
    PRIMARY KEY (course_id),
    FOREIGN KEY (dept_name) REFERENCES department(dept_name) ON
DELETE SET NULL );
CREATE TABLE instructor (
    ID VARCHAR(5),
    name VARCHAR(20) NOT NULL,
    dept_name VARCHAR(20),
    salary NUMERIC(8,2) CHECK (salary > 29000),
    PRIMARY KEY (ID),
    FOREIGN KEY (dept_name) REFERENCES department(dept_name) ON
DELETE SET NULL );
CREATE TABLE section (
    course_id VARCHAR(8),
    sec_id VARCHAR(8),
    semester VARCHAR(6) CHECK (semester IN ('Fall', 'Winter',
'Spring', 'Summer')),
    year NUMERIC(4,0) CHECK (year > 1701 AND year < 2100),
    building VARCHAR(15),
    room_number VARCHAR(7),
    time_slot_id VARCHAR(4),
    PRIMARY KEY (course_id, sec_id, semester, year),
    FOREIGN KEY (course_id) REFERENCES course(course_id) ON DELETE
CASCADE,
    FOREIGN KEY (building, room_number) REFERENCES
```

```
classroom(building, room_number) ON DELETE SET NULL );
CREATE TABLE teaches (
    ID VARCHAR(5),
    course_id VARCHAR(8),
    sec_id VARCHAR(8),
    semester VARCHAR(6),
    year NUMERIC(4,0),
    PRIMARY KEY (ID, course_id, sec_id, semester, year),
    FOREIGN KEY (course_id, sec_id, semester, year) REFERENCES
section(course_id, sec_id, semester, year) ON DELETE CASCADE,
    FOREIGN KEY (ID) REFERENCES instructor(ID) ON DELETE CASCADE
);
CREATE TABLE student (
    ID VARCHAR(5),
    name VARCHAR(20) NOT NULL,
    dept_name VARCHAR(20),
    tot_cred NUMERIC(3,0) CHECK (tot_cred >= 0),
    PRIMARY KEY (ID),
    FOREIGN KEY (dept_name) REFERENCES department(dept_name) ON
DELETE SET NULL );
CREATE TABLE takes (
    ID VARCHAR(5),
    course_id VARCHAR(8),
    sec_id VARCHAR(8),
    semester VARCHAR(6),
    year NUMERIC(4,0),
    grade VARCHAR(2),
    PRIMARY KEY (ID, course_id, sec_id, semester, year),
    FOREIGN KEY (course_id, sec_id, semester, year) REFERENCES
section(course_id, sec_id, semester, year) ON DELETE CASCADE,
    FOREIGN KEY (ID) REFERENCES student(ID) ON DELETE CASCADE );
CREATE TABLE advisor (
    s_ID VARCHAR(5),
    i_ID VARCHAR(5),
    PRIMARY KEY (s_ID, i_ID),
    FOREIGN KEY (i_ID) REFERENCES instructor(ID),
    FOREIGN KEY (s_ID) REFERENCES student(ID) ON DELETE CASCADE );
CREATE TABLE prereq (
    course_id VARCHAR(8),
    prereq_id VARCHAR(8),
    PRIMARY KEY (course_id, prereq_id),
    FOREIGN KEY (course_id) REFERENCES course(course_id) ON DELETE
CASCADE,
    FOREIGN KEY (prereq_id) REFERENCES course(course_id) );
CREATE TABLE timeslot (
    time_slot_id VARCHAR(4),
```

```
    day VARCHAR(1) CHECK (day IN ('M', 'T', 'W', 'R', 'F', 'S',
'U')),
    start_time TIME,
    end_time TIME,
    PRIMARY KEY (time_slot_id, day, start_time) );
CREATE TABLE timeslot_hr_min (
    time_slot_id VARCHAR(4),
    day VARCHAR(1),
    start_hr NUMERIC(2) CHECK (start_hr >= 0 AND start_hr < 24),
    start_min NUMERIC(2) CHECK (start_min >= 0 AND start_min <
60),
    end_hr NUMERIC(2) CHECK (end_hr >= 0 AND end_hr < 24),
    end_min NUMERIC(2) CHECK (end_min >= 0 AND end_min < 60),
    PRIMARY KEY (time_slot_id, day, start_hr, start_min)
);
```

Output –



Conclusion – In conclusion we can say that we have successfully created the tables of the pre-designed university database.

Problem no – 02

Problem Name – Write SQL queries to insert values into tables in the university database.

Query –

```
INSERT INTO classroom (building, room_number, capacity) VALUES
('Packard', '101', 500),
('Painter', '514', 10),
('Taylor', '3128', 70),
('Watson', '100', 30),
('Watson', '120', 50);
INSERT INTO department (dept_name, building, budget) VALUES
('Biology', 'Watson', 90000),
('Comp. Sci.', 'Taylor', 100000),
('Elec. Eng.', 'Taylor', 85000),
('Finance', 'Painter', 120000),
('History', 'Painter', 50000),
('Music', 'Packard', 80000),
('Physics', 'Watson', 70000);
INSERT INTO course (course_id, title, dept_name, credits) VALUES
('BIO-101', 'Intro. to Biology', 'Biology', 4),
('BIO-301', 'Genetics', 'Biology', 4),
('BIO-399', 'Computational Biology', 'Biology', 3),
('CS-101', 'Intro. to Computer Science', 'Comp. Sci.', 4),
('CS-190', 'Game Design', 'Comp. Sci.', 4),
('CS-315', 'Robotics', 'Comp. Sci.', 3),
('CS-319', 'Image Processing', 'Comp. Sci.', 3),
('CS-347', 'Database System Concepts', 'Comp. Sci.', 3),
('EE-181', 'Intro. to Digital Systems', 'Elec. Eng.', 3),
('FIN-201', 'Investment Banking', 'Finance', 3),
('HIS-351', 'World History', 'History', 3),
('MU-199', 'Music Video Production', 'Music', 3),
('PHY-101', 'Physical Principles', 'Physics', 4);
INSERT INTO instructor (ID, name, dept_name, salary) VALUES
('10101', 'Srinivasan', 'Comp. Sci.', 65000),
('12121', 'Wu', 'Finance', 90000),
('15151', 'Mozart', 'Music', 40000),
('22222', 'Einstein', 'Physics', 95000),
('32343', 'El Said', 'History', 60000),
('33456', 'Gold', 'Physics', 87000),
('45565', 'Katz', 'Comp. Sci.', 75000),
('58583', 'Califieri', 'History', 62000),
('76543', 'Singh', 'Finance', 80000),
('76766', 'Crick', 'Biology', 72000),
('83821', 'Brandt', 'Comp. Sci.', 92000),
```

```
('98345', 'Kim', 'Elec. Eng.', 80000);
INSERT INTO section (course_id, sec_id, semester, year, building,
room_number, time_slot_id) VALUES
('BIO-101', '1', 'Summer', 2017, 'Painter', '514', 'B'),
('BIO-301', '1', 'Summer', 2018, 'Painter', '514', 'A'),
('CS-101', '1', 'Fall', 2017, 'Packard', '101', 'H'),
('CS-101', '1', 'Spring', 2018, 'Packard', '101', 'F'),
('CS-190', '1', 'Spring', 2017, 'Taylor', '3128', 'E'),
('CS-190', '2', 'Spring', 2017, 'Taylor', '3128', 'A'),
('CS-315', '1', 'Spring', 2018, 'Watson', '120', 'D'),
('CS-319', '1', 'Spring', 2018, 'Watson', '100', 'B'),
('CS-319', '2', 'Spring', 2018, 'Taylor', '3128', 'C'),
('CS-347', '1', 'Fall', 2017, 'Taylor', '3128', 'A'),
('EE-181', '1', 'Spring', 2017, 'Taylor', '3128', 'C'),
('FIN-201', '1', 'Spring', 2018, 'Packard', '101', 'B'),
('HIS-351', '1', 'Spring', 2018, 'Painter', '514', 'C'),
('MU-199', '1', 'Spring', 2018, 'Packard', '101', 'D'),
('PHY-101', '1', 'Fall', 2017, 'Watson', '100', 'A');
INSERT INTO teaches (ID, course_id, sec_id, semester, year) VALUES
('10101', 'CS-101', '1', 'Fall', 2017),
('10101', 'CS-315', '1', 'Spring', 2018),
('10101', 'CS-347', '1', 'Fall', 2017),
('12121', 'FIN-201', '1', 'Spring', 2018),
('15151', 'MU-199', '1', 'Spring', 2018),
('22222', 'PHY-101', '1', 'Fall', 2017),
('32343', 'HIS-351', '1', 'Spring', 2018),
('45565', 'CS-101', '1', 'Spring', 2018),
('45565', 'CS-319', '1', 'Spring', 2018),
('76766', 'BIO-101', '1', 'Summer', 2017),
('76766', 'BIO-301', '1', 'Summer', 2018),
('83821', 'CS-190', '1', 'Spring', 2017),
('83821', 'CS-190', '2', 'Spring', 2017),
('83821', 'CS-319', '2', 'Spring', 2018),
('98345', 'EE-181', '1', 'Spring', 2017);
INSERT INTO student (ID, name, dept_name, tot_cred) VALUES
('00128', 'Zhang', 'Comp. Sci.', 102),
('12345', 'Shankar', 'Comp. Sci.', 32),
('19991', 'Brandt', 'History', 80),
('23121', 'Chavez', 'Finance', 110),
('44553', 'Peltier', 'Physics', 56),
('45678', 'Levy', 'Physics', 46),
('54321', 'Williams', 'Comp. Sci.', 54),
('55739', 'Sanchez', 'Music', 38),
('70557', 'Snow', 'Physics', 0),
('76543', 'Brown', 'Comp. Sci.', 58),
('76653', 'Aoi', 'Elec. Eng.', 60),
```

```
('98765', 'Bourikas', 'Elec. Eng.', 98),
('98988', 'Tanaka', 'Biology', 120);
INSERT INTO takes (ID, course_id, sec_id, semester, year, grade)
VALUES
('00128', 'CS-101', 1, 'Fall', 2017, 'A'),
('00128', 'CS-347', 1, 'Fall', 2017, 'A-'),
('12345', 'CS-101', 1, 'Fall', 2017, 'C'),
('12345', 'CS-190', 2, 'Spring', 2017, 'A'),
('12345', 'CS-315', 1, 'Spring', 2018, 'A'),
('12345', 'CS-347', 1, 'Fall', 2017, 'A'),
('19991', 'HIS-351', 1, 'Spring', 2018, 'B'),
('23121', 'FIN-201', 1, 'Spring', 2018, 'C+'),
('44553', 'PHY-101', 1, 'Fall', 2017, 'B-'),
('45678', 'CS-101', 1, 'Fall', 2017, 'F'),
('45678', 'CS-101', 1, 'Spring', 2018, 'B+'),
('45678', 'CS-319', 1, 'Spring', 2018, 'B'),
('54321', 'CS-101', 1, 'Fall', 2017, 'A-'),
('54321', 'CS-190', 2, 'Spring', 2017, 'B+'),
('55739', 'MU-199', 1, 'Spring', 2018, 'A-'),
('76543', 'CS-101', 1, 'Fall', 2017, 'A'),
('76543', 'CS-319', 2, 'Spring', 2018, 'A'),
('76653', 'EE-181', 1, 'Spring', 2017, 'C'),
('98765', 'CS-101', 1, 'Fall', 2017, 'C-'),
('98765', 'CS-315', 1, 'Spring', 2018, 'B'),
('98988', 'BIO-101', 1, 'Summer', 2017, 'A'),
('98988', 'BIO-301', 1, 'Summer', 2018, NULL);
INSERT INTO advisor (s_id, i_id) VALUES
('00128', '45565'),
('12345', '10101'),
('23121', '76543'),
('44553', '22222'),
('45678', '22222'),
('76543', '45565'),
('76653', '98345'),
('98765', '98345'),
('98988', '76766');
INSERT INTO timeslot (time_slot_id, day, start_time, end_time)
VALUES
('A', 'M', '08-00', '08-50'),
('A', 'W', '08-00', '08-50'),
('A', 'F', '08-00', '08-50'),
('B', 'M', '09-00', '09-50'),
('B', 'W', '09-00', '09-50'),
('B', 'F', '09-00', '09-50'),
('C', 'M', '11-00', '11-50'),
('C', 'W', '11-00', '11-50'),
```

```
('C', 'F', '11-00', '11-50'),
('D', 'M', '13-00', '13-50'),
('D', 'W', '13-00', '13-50'),
('D', 'F', '13-00', '13-50'),
('E', 'T', '10-30', '11-45'),
('E', 'R', '10-30', '11-45'),
('F', 'T', '14-30', '15-45'),
('F', 'R', '14-30', '15-45'),
('G', 'M', '16-00', '16-50'),
('G', 'W', '16-00', '16-50'),
('G', 'F', '16-00', '16-50'),
('H', 'W', '10-00', '12-30');
INSERT INTO prereq (course_id, prereq_id) VALUES
('BIO-301', 'BIO-101'),
('BIO-399', 'BIO-101'),
('CS-190', 'CS-101'),
('CS-315', 'CS-101'),
('CS-319', 'CS-101'),
('CS-347', 'CS-101'),
('EE-181', 'PHY-101');
INSERT INTO timeslot_hr_min (time_slot_id, day, start_hr,
start_min, end_hr, end_min) VALUES
('A', 'M', 8, 0, 8, 50),
('A', 'W', 8, 0, 8, 50),
('A', 'F', 8, 0, 8, 50),
('B', 'M', 9, 0, 9, 50),
('B', 'W', 9, 0, 9, 50),
('B', 'F', 9, 0, 9, 50),
('C', 'M', 11, 0, 11, 50),
('C', 'W', 11, 0, 11, 50),
('C', 'F', 11, 0, 11, 50),
('D', 'M', 13, 0, 13, 50),
('D', 'W', 13, 0, 13, 50),
('D', 'F', 13, 0, 13, 50),
('E', 'T', 10, 30, 11, 45),
('E', 'R', 10, 30, 11, 45),
('F', 'T', 14, 30, 15, 45),
('F', 'R', 14, 30, 15, 45),
('G', 'M', 16, 0, 16, 50),
('G', 'W', 16, 0, 16, 50),
('G', 'F', 16, 0, 16, 50),
('H', 'W', 10, 0, 12, 30);
```

Output –

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4
NULL	NULL	NULL	NULL

building	room_number	capacity
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50
NULL	NULL	NULL

s_ID	i_ID
12345	10101
44553	22222
45678	22222
00128	45565
76543	45565
23121	76543
98988	76766
76653	98345
98765	98345
NULL	NULL

dept_name	building	budget
Biology	Watson	90000.00
Comp. Sci.	Taylor	100000.00
Elec. Eng.	Taylor	85000.00
Finance	Painter	120000.00
History	Painter	50000.00
Music	Packard	80000.00
Physics	Watson	70000.00
NULL	NULL	NULL

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A
NULL	NULL	NULL	NULL	NULL	NULL	NULL

course_id	prereq_id
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101
NULL	NULL

ID	course_id	sec_id	semester	year
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
10101	CS-101	1	Fall	2017
45565	CS-101	1	Spring	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
10101	CS-315	1	Spring	2018
45565	CS-319	1	Spring	2018
83821	CS-319	2	Spring	2018
10101	CS-347	1	Fall	2017
98345	EE-181	1	Spring	2017
12121	FIN-201	1	Spring	2018
32343	HIS-351	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
NULL	NULL	NULL	NULL	NULL

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120
NULL	NULL	NULL	NULL

ID	course_id	sec_id	semester	year	grade
00128	CS-101	1	Fall	2017	A
00128	CS-347	1	Fall	2017	A-
12345	CS-101	1	Fall	2017	C
12345	CS-190	2	Spring	2017	A
12345	CS-315	1	Spring	2018	A
12345	CS-347	1	Fall	2017	A
19991	HIS-351	1	Spring	2018	B
23121	FIN-201	1	Spring	2018	C+
44553	PHY-101	1	Fall	2017	B-
45678	CS-101	1	Fall	2017	F
45678	CS-101	1	Spring	2018	B+
45678	CS-319	1	Spring	2018	B
54321	CS-101	1	Fall	2017	A-
54321	CS-190	2	Spring	2017	B+
55739	MU-199	1	Spring	2018	A-
76543	CS-101	1	Fall	2017	A
76543	CS-319	2	Spring	2018	A
76653	EE-181	1	Spring	2017	C
98765	CS-101	1	Fall	2017	C-
98765	CS-315	1	Spring	2018	B
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	NULL
NULL	NULL	NULL	NULL	NULL	NULL

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00
NULL	NULL	NULL	NULL

time_slot_id	day	start_hr	start_min	end_hr	end_min
A	F	8	0	8	50
A	M	8	0	8	50
A	W	8	0	8	50
B	F	9	0	9	50
B	M	9	0	9	50
B	W	9	0	9	50
C	F	11	0	11	50
C	M	11	0	11	50
C	W	11	0	11	50
D	F	13	0	13	50
D	M	13	0	13	50
D	W	13	0	13	50
E	R	10	30	11	45
E	T	10	30	11	45
F	R	14	30	15	45
F	T	14	30	15	45
G	F	16	0	16	50
G	M	16	0	16	50
G	W	16	0	16	50
H	W	10	0	12	30
NULL	NULL	NULL	NULL	NULL	NULL

time_slot_id	day	start_time	end_time
A	F	08:00:00	08:50:00
A	M	08:00:00	08:50:00
A	W	08:00:00	08:50:00
B	F	09:00:00	09:50:00
B	M	09:00:00	09:50:00
B	W	09:00:00	09:50:00
C	F	11:00:00	11:50:00
C	M	11:00:00	11:50:00
C	W	11:00:00	11:50:00
D	F	13:00:00	13:50:00
D	M	13:00:00	13:50:00
D	W	13:00:00	13:50:00
E	R	10:30:00	11:45:00
E	T	10:30:00	11:45:00
F	R	14:30:00	15:45:00
F	T	14:30:00	15:45:00
G	F	16:00:00	16:50:00
G	M	16:00:00	16:50:00
G	W	16:00:00	16:50:00
H	W	10:00:00	12:30:00
NULL	NULL	NULL	NULL

Conclusion – In conclusion we can say that we have successfully inserted the data in the database.

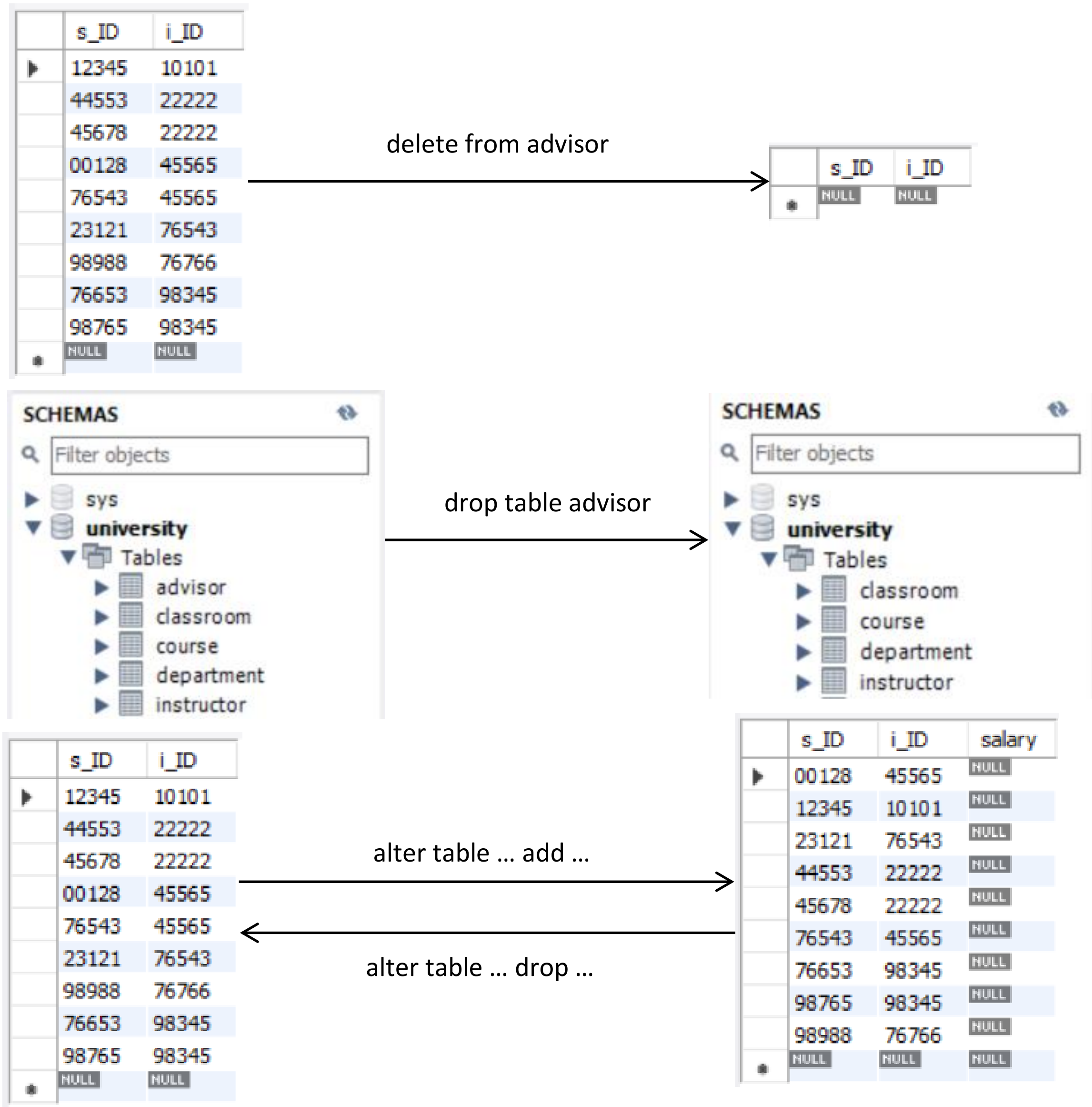
Problem no – 03

Problem Name – Write SQL queries using delete, drop table, alter table command.

Query –

```
delete from advisor;
drop table advisor;
alter table advisor add salary numeric(8,0);
alter table advisor drop salary;
```

Output –



Conclusion – These commands are essential for managing data (DELETE), removing tables (DROP TABLE), and modifying table structures (ALTER TABLE) effectively in a database. In conclusion, we can say that we have successfully written SQL queries using delete, drop table, alter table command.

Problem no – 04

Problem Name – Write a query searching for an attribute.

Query –

```
select  dept_name  from  student;
select  ID,name  from  student;
```

Output –

dept_name
Biology
Comp. Sci.
Comp. Sci.
Comp. Sci.
Comp. Sci.
Elec. Eng.
Elec. Eng.
Finance
History
Music
Physics
Physics
Physics

ID	name
00128	Zhang
12345	Shankar
19991	Brandt
23121	Chavez
44553	Peltier
45678	Levy
54321	Williams
55739	Sanchez
70557	Snow
76543	Brown
76653	Aoi
98765	Bourikas
98988	Tanaka
NULL	NULL

Conclusion – These queries demonstrate how to retrieve specific attributes (columns) from a table using the SELECT statement. This is a fundamental SQL operation for extracting relevant data from a database.

Problem no – 05

Problem Name – Write queries by implementing the DISTINCT and ALL keywords.

Query –

```
select all dept_name from instructor;
select distinct dept_name from instructor;
```

Output –

dept_name
Biology
Comp. Sci.
Comp. Sci.
Comp. Sci.
Elec. Eng.
Finance
Finance
History
History
Music
Physics
Physics

all dept_name

dept_name
Biology
Comp. Sci.
Elec. Eng.
Finance
History
Music
Physics

distinct dept_name

Conclusion – From the output of the queries we have observed the behaviors of the "**DISTINCT**" and "**ALL**" keywords.

Problem no – 06

Problem Name – Write queries using arithmetic, logical and relational operators.

Query & Output –

Query	Output
`select ID, name, dept_name, salary *1.1 from instructor;`	see table 1 below
`select name from instructor where dept_name = 'Comp. Sci.' and salary > 70000;`	see table 2 below
`select name, instructor.dept_name, building from instructor, department where instructor.dept_name= department.dept_name;`	see table 3 below
`select name, course_id from instructor, teaches where instructor.ID= teaches.ID and instructor.dept_name = 'Comp. Sci.';`	see table 4 below

```
select ID, name, dept_name,
salary *1.1 from instructor;
```

ID	name	dept_name	salary *1.1
10101	Srinivasan	Comp. Sci.	71500.000
12121	Wu	Finance	99000.000
15151	Mozart	Music	44000.000
22222	Einstein	Physics	104500.000
32343	El Said	History	66000.000
33456	Gold	Physics	95700.000
45565	Katz	Comp. Sci.	82500.000
58583	Califieri	History	68200.000
76543	Singh	Finance	88000.000
76766	Crick	Biology	79200.000
83821	Brandt	Comp. Sci.	101200.000
98345	Kim	Elec. Eng.	88000.000

```
select name from instructor
where dept_name = 'Comp. Sci.'
and salary > 70000;
```

name
Katz
Brandt

```
select name,
instructor.dept_name, building
from instructor, department
where instructor.dept_name=
department.dept_name;
```

name	dept_name	building
Mozart	Music	Packard
Wu	Finance	Painter
Singh	Finance	Painter
El Said	History	Painter
Califieri	History	Painter
Srinivasan	Comp. Sci.	Taylor
Katz	Comp. Sci.	Taylor
Brandt	Comp. Sci.	Taylor
Kim	Elec. Eng.	Taylor
Crick	Biology	Watson
Einstein	Physics	Watson
Gold	Physics	Watson

```
select name, course_id from
instructor, teaches
where instructor.ID=
teaches.ID and
instructor.dept_name = 'Comp.
Sci.';
```

name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Katz	CS-101
Katz	CS-319
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319

Conclusion – In conclusion, we can say that we have successfully written SQL queries using arithmetic, logical, and relational operators.

Problem no – 07

Problem name – Write queries using renaming (AS clause) operation.

Query & **Output –**

Query	Output
SELECT DISTINCT T.name FROM instructor as T, instructor as S WHERE T.salary > S.salary and S.dept_name = "Biology";	name Wu Einstein Gold Katz Singh Brandt Kim

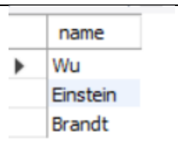
Conclusion – In this SQL query we can see that the same instructor table is used twice to get every distinct names of the instructors that has more salary than the one who have the least salary. In other word this query is used to get the names of the instructors that do not have the least salary. But in order to get the names we needed to use the same instructor table twice, the touple of name from the T is inserted into the output query every time T’s salary is more that the S’s salary. In conclusion we can say that we have successfully wrote a SQL query using “**AS**” clause.

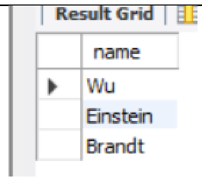
Problem no – 08

Problem name – Write queries using **BETWEEN** keyword and comparison operations.

Query & **Output –**

Query	Output

`select name`
`from instructor`
`where salary between 90000 and 100000;`
	
	name
	Wu
	Einstein
	Brandt

`select name`
`from instructor`
`where (salary>= 90000) and (salary<=100000);`
	
	Result Grid
	name
	Wu
	Einstein
	Brandt

Conclusion– These queries highlight two methods to filter data based on a range condition for the salary attribute in the instructor table. The first query uses the BETWEEN keyword, a concise and intuitive way to specify a range. The second query employs explicit comparison operators, providing an alternative way to express the same logic. So therefore, we can say that we have successfully written queries using BETWEEN keyword and comparison operations.

Problem no – 09

Problem name – Write queries using aggregate functions. (AVG, MAX, MIN, SUM, COUNT)

Query & Output –

Query	Output
select avg (salary) from instructor where dept_name= 'Comp. Sci.';	avg (salary) 77333.333333
select dept_name, count(ID) from instructor group by dept_name;	dept_name count(ID) Biology 1 Comp. Sci. 3 Elec. Eng. 1 Finance 2 History 2 Music 1 Physics 2
select sum(salary) from instructor;	sum(salary) 898000.01
select max(salary) from instructor;	max(salary) 95000.00
select min(salary) from instructor;	min(salary) 40000.00

Conclusion – These queries showcase using SQL aggregate functions, such as AVG, COUNT, SUM, MAX, and MIN, to perform calculations on groups of rows. Each function serves a distinct purpose, from calculating averages to determining the total, maximum, or minimum values, or counting rows. Thus, we can say that we have successfully written queries using aggregate functions. (AVG, MAX, MIN, SUM, COUNT).

Problem no – 10

Problem name – Write subqueries for fetching specific data and show the usages of SOME and ALL clauses before the subqueries.

Query & Output –

Query	Output

`SELECT instructor.name`
`FROM instructor`
`WHERE instructor.salary > some(SELECT instructor.salary FROM instructor WHERE instructor.dept_name = 'Biology');`
	name
	Wu
	Einstein
	Gold
	Katz
	Singh
	Brandt
	Kim

`SELECT instructor.name`
`FROM instructor`
`WHERE instructor.salary < all(SELECT instructor.salary FROM instructor WHERE instructor.dept_name = 'Biology');`
	name
	Srinivasan
	Mozart
	El Said
	Califieri

Conclusion – In the first query one of them gives all the name that has more salary than any of the instructors in the biology department using the "**SOME**" and in the second query using the "**ALL**" we find all the names of the instructors that less than all of the instructors in the biology department.

Problem no – 11

Problem name – Write queries using string operations, attribute specification, and ORDER BY clause.

Query & Output–

<table>
<tr><th>Query</th><th>Output</th></tr>
<tr><td>
<pre>SELECT dept_name FROM
department
WHERE building LIKE
'%Taylor%';</pre>
</td><td>
<table>
<tr><th>dept_name</th></tr>
<tr><td>Comp. Sci.</td></tr>
<tr><td>Elec. Eng.</td></tr>
<tr><td>NULL</td></tr>
</table>
</td></tr>
<tr><td>
<pre>SELECT instructor.*
FROM instructor, teaches
WHERE instructor.ID =
teaches.ID;</pre>
</td><td>
<table>
<tr><th>ID</th><th>name</th><th>dept_name</th><th>salary</th></tr>
<tr><td>10101</td><td>Srinivasan</td><td>Comp. Sci.</td><td>65000.00</td></tr>
<tr><td>10101</td><td>Srinivasan</td><td>Comp. Sci.</td><td>65000.00</td></tr>
<tr><td>10101</td><td>Srinivasan</td><td>Comp. Sci.</td><td>65000.00</td></tr>
<tr><td>12121</td><td>Wu</td><td>Finance</td><td>90000.00</td></tr>
<tr><td>15151</td><td>Mozart</td><td>Music</td><td>40000.00</td></tr>
<tr><td>22222</td><td>Einstein</td><td>Physics</td><td>95000.00</td></tr>
<tr><td>32343</td><td>El Said</td><td>History</td><td>60000.00</td></tr>
<tr><td>45565</td><td>Katz</td><td>Comp. Sci.</td><td>75000.00</td></tr>
<tr><td>45565</td><td>Katz</td><td>Comp. Sci.</td><td>75000.00</td></tr>
<tr><td>76766</td><td>Crick</td><td>Biology</td><td>72000.01</td></tr>
<tr><td>76766</td><td>Crick</td><td>Biology</td><td>72000.01</td></tr>
</table>
</td></tr>
<tr><td>
<pre>SELECT name
FROM instructor
WHERE dept_name = 'Comp. Sci.'
ORDER BY name;</pre>
</td><td>
Result Grid
<table>
<tr><th>name</th></tr>
<tr><td>Brandt</td></tr>
<tr><td>Katz</td></tr>
<tr><td>Srinivasan</td></tr>
</table>
</td></tr>
<tr><td>
<pre>SELECT *
FROM instructor
ORDER BY salary DESC;</pre>
</td><td>
Result Grid | Filter Rows:
<table>
<tr><th>ID</th><th>name</th><th>dept_name</th><th>salary</th></tr>
<tr><td>22222</td><td>Einstein</td><td>Physics</td><td>95000.00</td></tr>
<tr><td>83821</td><td>Brandt</td><td>Comp. Sci.</td><td>92000.00</td></tr>
<tr><td>12121</td><td>Wu</td><td>Finance</td><td>90000.00</td></tr>
<tr><td>33456</td><td>Gold</td><td>Physics</td><td>87000.00</td></tr>
<tr><td>76543</td><td>Singh</td><td>Finance</td><td>80000.00</td></tr>
<tr><td>98345</td><td>Kim</td><td>Elec. Eng.</td><td>80000.00</td></tr>
<tr><td>45565</td><td>Katz</td><td>Comp. Sci.</td><td>75000.00</td></tr>
<tr><td>76766</td><td>Crick</td><td>Biology</td><td>72000.01</td></tr>
<tr><td>10101</td><td>Srinivasan</td><td>Comp. Sci.</td><td>65000.00</td></tr>
<tr><td>58583</td><td>Califieri</td><td>History</td><td>62000.00</td></tr>
<tr><td>32343</td><td>El Said</td><td>History</td><td>60000.00</td></tr>
</table>
</td></tr>
</table>

Conclusion – In conclusion, we can say that we have successfully have written the SQL query to do string matching operation and the sorting operation using "**ORDER BY**".

Problem no – 12

Problem name – Write queries using set operations (UNION, INTERSECT, EXCEPT).

Query & Output –

Query	Output
(select course_id from section where semester = 'Fall' and year= 2017) union (select course_id from section where semester = 'Spring' and year= 2018);	course_id CS-101 CS-347 PHY-101 FIN-201 MU-199 HIS-351 CS-319 CS-315
(select course_id from section where semester = 'Fall' and year= 2017) intersect (select course_id from section where semester = 'Spring' and year= 2018);	course_id CS-101
(select course_id from section where semester = 'Fall' and year= 2017) except (select course_id from section where semester = 'Spring' and year= 2018);	course_id CS-347 PHY-101

Conclusion – With the outputs and comparisons from the datasets we can say that we have successfully written queries using set operations (UNION, INTERSECT, EXCEPT).

Problem no – 13

Problem name – Write queries using set membership, set comparison, and testing for empty relationships.

Query & Output –

Query	Output

select distinct course_id
from section
where semester = 'Fall' and year= 2017
and course_id in (
    select course_id
    from section
    where semester = 'Spring' and year=
2018
);
	course_id
	CS-101

select distinct course_id
from section
where semester = 'Fall' and year= 2017
and course_id not in (
     select course_id
     from section
     where semester = 'Spring' and year=
2018
);
	course_id
	CS-347
	PHY-101

select distinct T.name from instructor as T, instructor as S where T.salary > S.salary and S.dept_name = 'Biology';	name Wu Einstein Gold Katz Singh Brandt Kim
select name from instructor where salary > some (select salary from instructor where dept_name = 'Biology');	name Wu Einstein Gold Katz Singh Brandt Kim

```
select name
from instructor
where salary > all (
     select salary
     from instructor
    where dept_name = 'Biology'
);
```

name
Wu
Einstein
Gold
Katz
Singh
Brandt
Kim

```
select course_id
from section as S
where semester = 'Fall' and year = 2017
and exists(
     select *
    from section as T
    where semester = 'Spring' and year=
2018 and S.course_id = T.course_id
);
```

course_id
CS-101

Conclusion – In conclusion we can say that we have successfully wrote queries to use set membership, set comparison, and testing for empty relationships using “**IN**”, “**SOME**”, “**ALL**” and “**EXISTS**” clauses.

Problem no – 14

Problem name – Write queries on multiple relations and the use of NATURAL JOIN keyword.

Query & Output –

<table>
<tr><th>Query</th><th>Output</th></tr>
<tr><td>select name, course_id
from student natural join
takes;</td><td>name | course_id
Brown | CS-319
Chavez | FIN-201
Levy | CS-101
Levy | CS-101
Levy | CS-319
Peltier | PHY-101
Sanchez | MU-199
Shankar | CS-101
Shankar | CS-190
Shankar | CS-315
Shankar | CS-347
Tanaka | BIO-101
Tanaka | BIO-301
Williams | CS-101
Williams | CS-190
Zhang | CS-101
Zhang | CS-347</td></tr>
<tr><td>select name, title
from student natural join
takes, course
where takes.course_id =
course.course_id;</td><td>name | title
Tanaka | Intro. to Biology
Tanaka | Genetics
Zhang | Intro. to Computer Science
Shankar | Zhang Computer Science
Levy | Intro. to Computer Science
Williams | Intro. to Computer Science
Brown | Intro. to Computer Science
Bourikas | Intro. to Computer Science
Levy | Intro. to Computer Science
Shankar | Game Design
Williams | Game Design
Shankar | Robotics
Bourikas | Robotics
Levy | Image Processing
Brown | Image Processing
Zhang | Database System Concepts
Shankar | Database System Concepts</td></tr>
</table>

Conclusion – In conclusion we can say that we have successfully written queries on multiple relations with the use of NATURAL JOIN keyword.

Problem no – 15

Problem name – Write queries using different types of joins (INNER JOIN, LEFT JOIN, RIGHT JOIN).

Query & Output –

Query

```
select *
from  student join takes
on student.ID  = takes.ID;
```

Output

ID	name	dept_name	tot_cred	ID	course_id	sec_id	semester	year	grade
00128	Zhang	Comp. Sci.	102	00128	CS-101	1	Fall	2017	A
00128	Zhang	Comp. Sci.	102	00128	CS-347	1	Fall	2017	A-
12345	Shankar	Comp. Sci.	32	12345	CS-101	1	Fall	2017	C
12345	Shankar	Comp. Sci.	32	12345	CS-190	2	Spring	2017	A
12345	Shankar	Comp. Sci.	32	12345	CS-315	1	Spring	2018	A
12345	Shankar	Comp. Sci.	32	12345	CS-347	1	Fall	2017	A
19991	Brandt	History	80	19991	HIS-351	1	Spring	2018	B
23121	Chavez	Finance	110	23121	FIN-201	1	Spring	2018	C+
44553	Peltier	Physics	56	44553	PHY-101	1	Fall	2017	B-
45678	Levy	Physics	46	45678	CS-101	1	Fall	2017	F
45678	Levy	Physics	46	45678	CS-101	1	Spring	2018	B+
45678	Levy	Physics	46	45678	CS-319	1	Spring	2018	B
54321	Williams	Comp. Sci.	54	54321	CS-101	1	Fall	2017	A-
54321	Williams	Comp. Sci.	54	54321	CS-190	2	Spring	2017	B+
55739	Sanchez	Music	38	55739	MU-199	1	Spring	2018	A-
76543	Brown	Comp. Sci.	58	76543	CS-101	1	Fall	2017	A
76543	Brown	Comp. Sci.	58	76543	CS-319	2	Spring	2018	A

```
select *
from course
natural left outer join
prereq;
```

course_id	title	dept_name	credits	prereq_id
BIO-101	Intro. to Biology	Biology	4	NULL
BIO-301	Genetics	Biology	4	BIO-101
BIO-399	Computational Biology	Biology	3	BIO-101
CS-101	Intro. to Computer Science	Comp. Sci.	4	NULL
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	CS-101
CS-319	Image Processing	Comp. Sci.	3	CS-101
CS-347	Database System Concepts	Comp. Sci.	3	CS-101
EE-181	Intro. to Digital Systems	Elec. Eng.	3	PHY-101
FIN-201	Investment Banking	Finance	3	NULL

```
select *
from course natural right
outer join prereq;
```

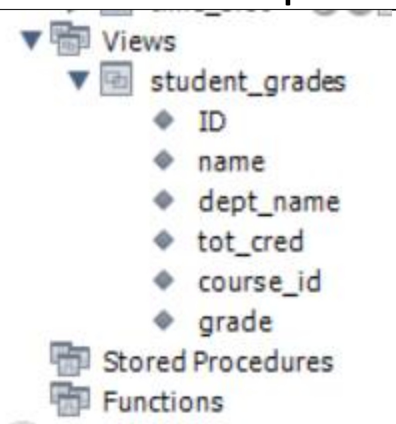
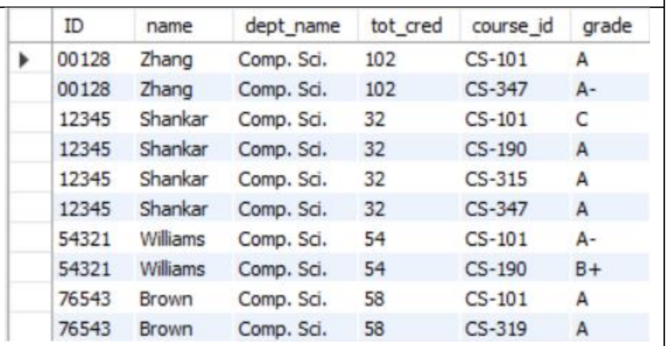
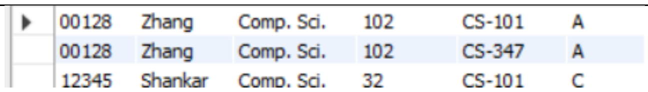
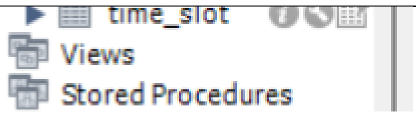
course_id	prereq_id	title	dept_name	credits
BIO-301	BIO-101	Genetics	Biology	4
BIO-399	BIO-101	Computational Biology	Biology	3
CS-190	CS-101	Game Design	Comp. Sci.	4
CS-315	CS-101	Robotics	Comp. Sci.	3
CS-319	CS-101	Image Processing	Comp. Sci.	3
CS-347	CS-101	Database System Concepts	Comp. Sci.	3
EE-181	PHY-101	Intro. to Digital Systems	Elec. Eng.	3

Conclusion – In the command we can observe that simple using "**JOIN**" command using "**ON**" keyword returns an inner join and LEFT or RIGHT outer join requires the usage of explicit command. In conclusion, we can say that we have successfully wrote SQL commands using inner join, left join and right join.

Problem no – 16

Problem name – Write SQL queries to create and manipulate views for displaying student details with their respective course names and grades.

Query and Output –

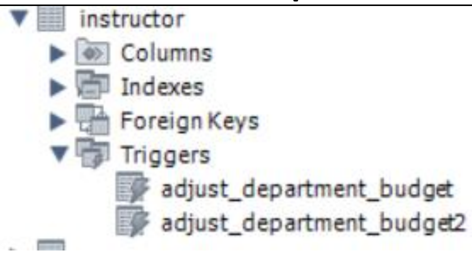
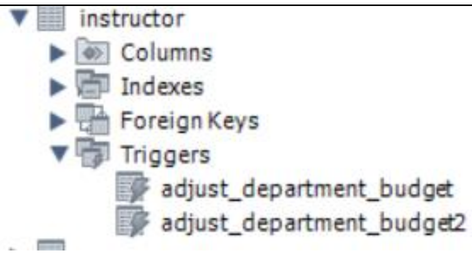
<table>
<tr><th>Query</th><th>Output</th></tr>
<tr><td>create view student_grades as
select
student.ID as ID,
student.name as name,
student.dept_name as
dept_name, student.tot_cred
as tot_cred, takes.course_id
as course_id, takes.grade as
grade from student, takes
where student.ID = takes.ID;</td><td>
</td></tr>
<tr><td>select * from student_grades
where dept_name='Comp. Sci.';</td><td>
<table>
<tr><th>ID</th><th>name</th><th>dept_name</th><th>tot_cred</th><th>course_id</th><th>grade</th></tr>
<tr><td>00128</td><td>Zhang</td><td>Comp. Sci.</td><td>102</td><td>CS-101</td><td>A</td></tr>
<tr><td>00128</td><td>Zhang</td><td>Comp. Sci.</td><td>102</td><td>CS-347</td><td>A-</td></tr>
<tr><td>12345</td><td>Shankar</td><td>Comp. Sci.</td><td>32</td><td>CS-101</td><td>C</td></tr>
<tr><td>12345</td><td>Shankar</td><td>Comp. Sci.</td><td>32</td><td>CS-190</td><td>A</td></tr>
<tr><td>12345</td><td>Shankar</td><td>Comp. Sci.</td><td>32</td><td>CS-315</td><td>A</td></tr>
<tr><td>12345</td><td>Shankar</td><td>Comp. Sci.</td><td>32</td><td>CS-347</td><td>A</td></tr>
<tr><td>54321</td><td>Williams</td><td>Comp. Sci.</td><td>54</td><td>CS-101</td><td>A-</td></tr>
<tr><td>54321</td><td>Williams</td><td>Comp. Sci.</td><td>54</td><td>CS-190</td><td>B+</td></tr>
<tr><td>76543</td><td>Brown</td><td>Comp. Sci.</td><td>58</td><td>CS-101</td><td>A</td></tr>
<tr><td>76543</td><td>Brown</td><td>Comp. Sci.</td><td>58</td><td>CS-319</td><td>A</td></tr>
</table></td></tr>
<tr><td>update takes
set grade='A'
where ID = 128 and course_id
= 'CS-347';</td><td>
<table>
<tr><td>00128</td><td>Zhang</td><td>Comp. Sci.</td><td>102</td><td>CS-101</td><td>A</td></tr>
<tr><td>00128</td><td>Zhang</td><td>Comp. Sci.</td><td>102</td><td>CS-347</td><td>A</td></tr>
<tr><td>12345</td><td>Shankar</td><td>Comp. Sci.</td><td>32</td><td>CS-101</td><td>C</td></tr>
</table></td></tr>
<tr><td>drop view student_grades;</td><td>
</td></tr>
</table>

Conclusion – In the query we cannot implement the view to modify the grade of Zang to ‘A’. Usually these kinds of operation are why view is used. But a database admin can easily have access to the takes table. So, this gives an added layer of security to the data. In conclusion we can say that we have successfully have written SQL queries to create and manipulate views for displaying student details with their respective course names and grades.

Problem no – 17

Problem name – Write SQL queries to create a trigger that automatically updates the budget of a department in the department table whenever an instructor's salary is updated in the instructor table.

Query & Output –

<table>
<tr><th>Query</th><th>Output</th></tr>
<tr><td><pre>
DELIMITER $$
CREATE TRIGGER
adjust_department_budget
AFTER INSERT ON instructor
FOR EACH ROW
BEGIN
IF (NEW.salary IS NOT NULL)
THEN
UPDATE department
SET budget = budget +
(NEW.salary/1000)
WHERE dept_name =
NEW.dept_name;
     END IF;
END;
DELIMITER;
</pre></td><td>
</td></tr>
<tr><td><pre>
DELIMITER $$
CREATE TRIGGER
adjust_department_budget2
AFTER UPDATE
ON instructor
FOR EACH ROW
BEGIN
IF (NEW.salary IS NOT NULL
AND OLD.salary IS NOT NULL)
THEN
UPDATE department SET budget
= budget + (NEW.salary -
OLD.salary)/1000
WHERE dept_name =
NEW.dept_name;
end if;
IF NEW.salary IS NOT NULL and
OLD.salary IS NULL THEN
update department
set budget = budget +
new.salary/1000
where dept_name =
new.dept_name;
END IF;
</pre></td><td>
</td></tr>
</table>

```
END;
DELIMITER;
```

```
update instructor
set salary = salary*1.10;
select * from department;
```

dept_name	building	budget
Biology	Watson	90007.20
Comp. Sci.	Taylor	100023.20
Elec. Eng.	Taylor	85008.00
Finance	Painter	120017.00
History	Painter	50012.20

Conclusion – In conclusion we can say that we have successfully wrote a trigger in SQL that automatically updates the budget of a department in the department table whenever an instructor's salary is updated in the instructor table or a new instructor in hired in the department.

Problem no – 18

Problem name – Write SQL queries to create an index on the room_number column of the classroom table to speed up searches for classrooms based on their room numbers. Additionally, demonstrate how to drop this index if it is no longer needed.

Query –

```
CREATE INDEX idx_room_number
ON classroom(room_number);

SELECT building, room_number, capacity
FROM classroom
WHERE room_number = '101';

-- to see wheather database is using index yes or not
EXPLAIN SELECT building, room_number, capacity
FROM classroom
WHERE room_number = '101';

DROP INDEX idx_room_number
ON classroom(room_number);
```

Output–

	building	room_number	capacity
▶	Packard	101	500
*	NULL	NULL	NULL

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	classroom	NULL	ref	idx_room_num	idx_room_number	30	const	1	100.00	NULL

Conclusion – Indexes significantly enhance performance for queries involving search and filter operations by reducing the time required to locate data. They are essential for improving read performance in large datasets but should be used judiciously due to the additional storage and maintenance overhead they introduce. Regular monitoring and optimization of indexes are crucial for maintaining a balanced database system.