Lab Number 1

Create the following relational database and write query for the questions that follow:

Employee (person-name, street, city)
Works (person-name, company-name, salary)
Company (company-name, city)
Manages (person-name, manager-name)

Table Schema

```
CREATE TABLE Employee (
    person_name VARCHAR(50) PRIMARY KEY,
    street VARCHAR(100),
    city VARCHAR(50)
);
```

Field Types

#	Field	Schema	Table	Type
1	person_name	qn1	employee	VARCHAR
2	street	qn1	employee	VARCHAR
3	city	qn1	employee	VARCHAR

```
CREATE TABLE Works (
    person_name VARCHAR(50),
    company_name VARCHAR(50),
    salary DECIMAL(10, 2),
    PRIMARY KEY (person_name, company_name),
    FOREIGN KEY (person_name) REFERENCES Employee(person_name)
);
```

Field Types

#	Field	Schema	Table	Type
1	person_name	qn1	works	VARCHAR
2	company_name	qn1	works	VARCHAR
3	salary	qn1	works	DECIMAL

```
CREATE TABLE Company (
    company_name VARCHAR(50) PRIMARY KEY,
    city VARCHAR(50)
);
```

Field Types

#	Field	Schema	Table	Type
1	company_name	qn1	company	VARCHAR
2	city	qn1	company	VARCHAR

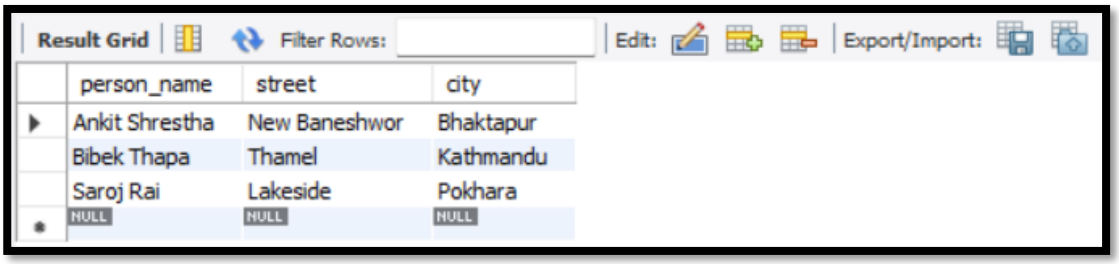
```
CREATE TABLE Manages (
        person_name VARCHAR(50),
        manager_name VARCHAR(50),
        PRIMARY KEY (person_name),
        FOREIGN KEY (person_name) REFERENCES Employee(person_name),
        FOREIGN KEY (manager_name) REFERENCES Employee(person_name)
);
```

Field Types

#	Field	Schema	Table	Type
1	person_name	qn1	manages	VARCHAR
2	manager_name	qn1	manages	VARCHAR

Data Insertion

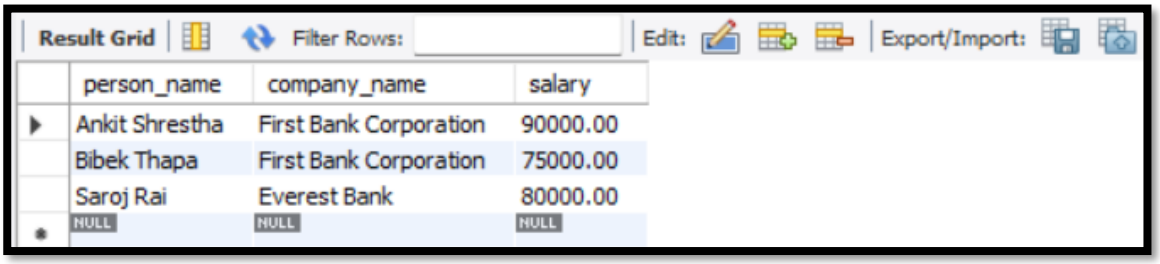
```
-- Insert into Employee table
        INSERT INTO Employee (person_name, street, city) VALUES
        ('Bibek Thapa', 'Thamel', 'Kathmandu'),
        ('Saroj Rai', 'Lakeside', 'Pokhara'),
        ('Ankit Shrestha', 'New Baneshwor', 'Bhaktapur');
        SELECT * FROM Employee;
```



Result Grid | Filter Rows: | Edit: | Export/Import:

person_name	street	city
Ankit Shrestha	New Baneshwor	Bhaktapur
Bibek Thapa	Thamel	Kathmandu
Saroj Rai	Lakeside	Pokhara
NULL	NULL	NULL

```
-- Insert into Works table
        INSERT INTO Works (person_name, company_name, salary) VALUES
('Bibek Thapa', 'First Bank Corporation', 75000),
('Saroj Rai', 'Everest Bank', 80000),
('Ankit Shrestha', 'First Bank Corporation', 90000);
        SELECT * FROM Works;
```

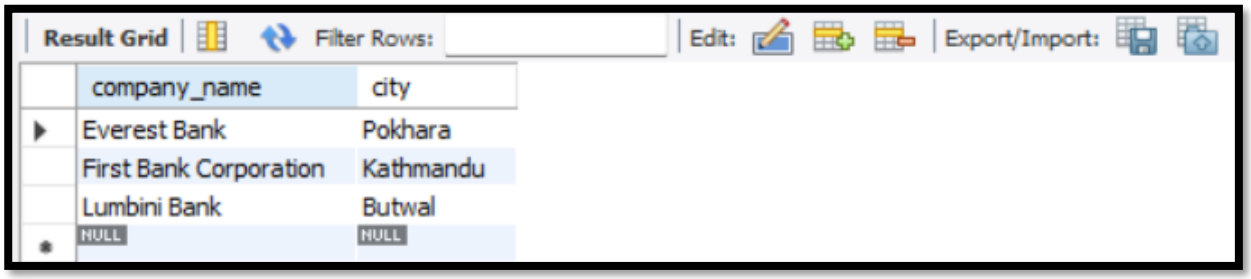


Result Grid | Filter Rows: | Edit: | Export/Import:

person_name	company_name	salary
Ankit Shrestha	First Bank Corporation	90000.00
Bibek Thapa	First Bank Corporation	75000.00
Saroj Rai	Everest Bank	80000.00
NULL	NULL	NULL

-- Insert into Company table
INSERT INTO Company (company_name, city) VALUES
('First Bank Corporation', 'Kathmandu'),
('Everest Bank', 'Pokhara'),
('Lumbini Bank', 'Butwal');

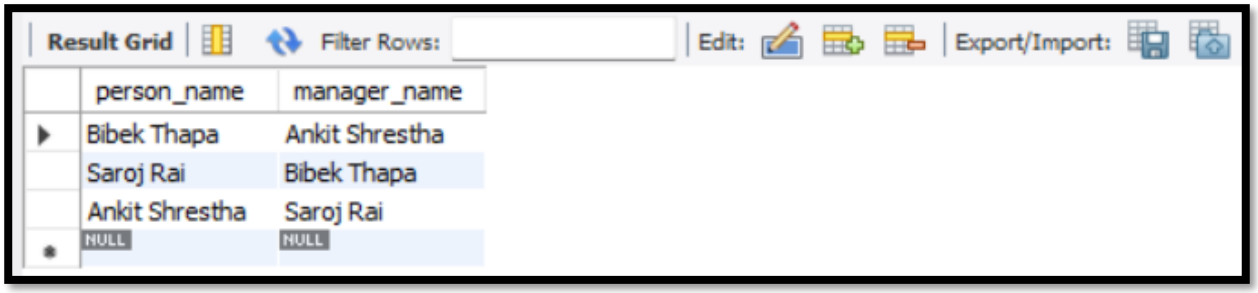
SELECT * FROM Company;



company_name	city
Everest Bank	Pokhara
First Bank Corporation	Kathmandu
Lumbini Bank	Butwal
NULL	NULL

-- Insert records into Manages table
INSERT INTO Manages (person_name, manager_name) VALUES
('Bibek Thapa', 'Ankit Shrestha'),
('Saroj Rai', 'Bibek Thapa'),
('Ankit Shrestha', 'Saroj Rai');

SELECT * FROM Manages;

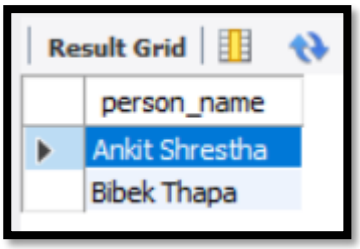


person_name	manager_name
Bibek Thapa	Ankit Shrestha
Saroj Rai	Bibek Thapa
Ankit Shrestha	Saroj Rai
NULL	NULL

Queries

a). Find the names of all employees who work for First Bank Corporation.

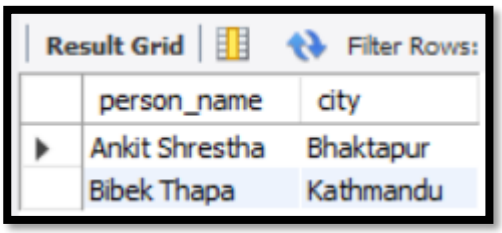
```
SELECT person_name
FROM Works
WHERE company_name = 'First Bank Corporation';
```



person_name
Ankit Shrestha
Bibek Thapa

b). Find the names and cities of residence of all employees who work for First Bank Corporation.

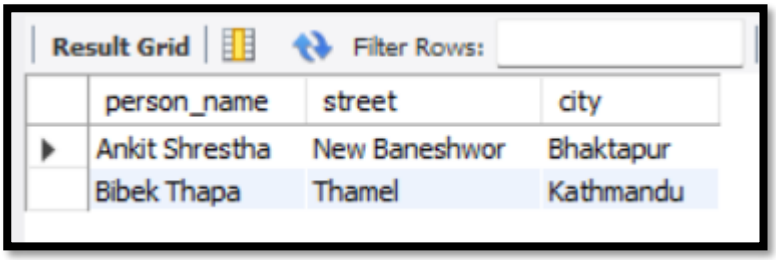
```
SELECT e.person_name, e.city
FROM Employee e
JOIN Works w ON e.person_name = w.person_name
WHERE w.company_name = 'First Bank Corporation';
```



person_name	city
Ankit Shrestha	Bhaktapur
Bibek Thapa	Kathmandu

c). Find the names, street addresses, and cities of residence of all employees who work for First Bank Corporation and earn more than Rs 10,000 per annum.

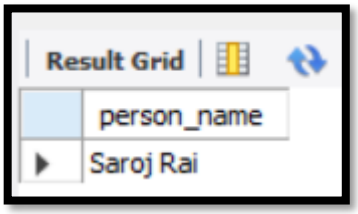
```
SELECT e.person_name, e.street, e.city
FROM Employee e
JOIN Works w ON e.person_name = w.person_name
WHERE w.company_name = 'First Bank Corporation' AND w.salary > 10000;
```



person_name	street	city
Ankit Shrestha	New Baneshwor	Bhaktapur
Bibek Thapa	Thamel	Kathmandu

d). Find the name of all employees in this database who do not work for First Bank Corporation.

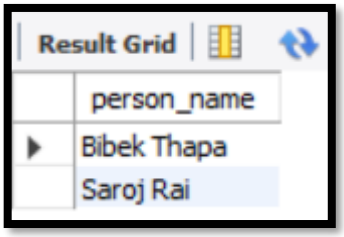
```
SELECT person_name
FROM Works
WHERE company_name != 'First Bank Corporation';
```



person_name
Saroj Rai

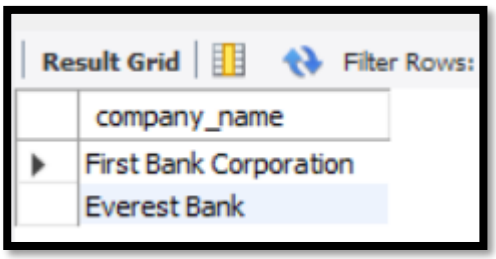
e). Find all employees in the database who live in the same cities as the companies for which they work.

```
SELECT e.person_name
FROM Employee e
JOIN Works w ON e.person_name = w.person_name
JOIN Company c ON w.company_name = c.company_name
WHERE e.city = c.city;
```



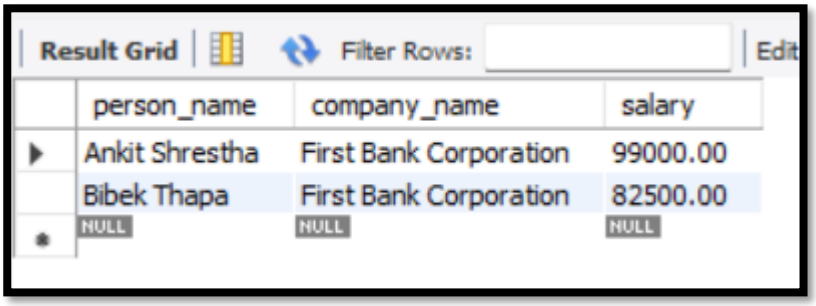
f). Find all companies in which average salary of employee is more than 5000.

```
SELECT company_name
FROM Works
GROUP BY company_name
HAVING AVG(salary) > 5000;
```



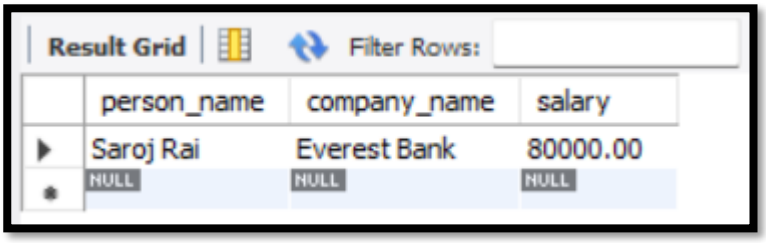
g). Update the salary of all the employees who work for First Bank Corporation by 10%.

```
SET SQL_SAFE_UPDATES = 0;
UPDATE Works
SET salary = salary * 1.10
WHERE company_name = 'First Bank Corporation';
SET SQL_SAFE_UPDATES = 1;
SELECT * FROM Works
WHERE company_name = 'First Bank Corporation';
```



h) Delete the records of all employees who work for First Bank Corporation.

```
DELETE FROM Works
WHERE company_name = 'First Bank Corporation';
SELECT * FROM Works;
```



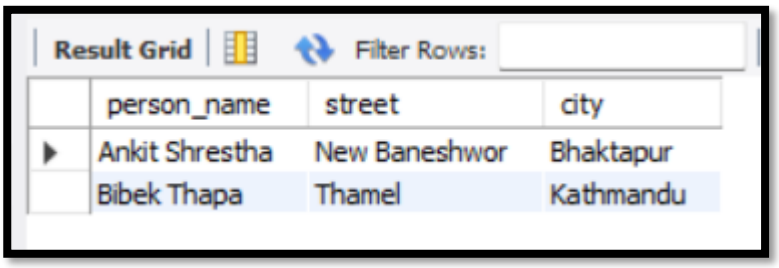
Result Grid | Filter Rows:

person_name	company_name	salary
Saroj Rai	Everest Bank	80000.00
NULL	NULL	NULL

i) Create a view to find the names, street addresses, and cities of residence of all employees who work for First Bank Corporation and earn more than Rs 10,000 per annum.

```
CREATE VIEW high_earning_fbc_employees AS
SELECT e.person_name, e.street, e.city
FROM Employee e
JOIN Works w ON e.person_name = w.person_name
WHERE w.company_name = 'First Bank Corporation' AND w.salary > 10000;

SELECT * FROM high_earning_fbc_employees;
```



Result Grid | Filter Rows:

person_name	street	city
Ankit Shrestha	New Baneshwor	Bhaktapur
Bibek Thapa	Thamel	Kathmandu

Lab number 2

For the following relations -
Members (mid, name, design, age)
Books (Bid, Btitle, BAuthor, Bpublisher, Bprice)
Reserves (mid, Bid, date)
Where Bid is book identification, Btitle is Book title, Bpublisher is book publisher, Bprice is Book price, mid is Members identification, and Design is designation.

(a) Create the above tables in MySQL:

Table Schema:

```
CREATE TABLE Members (
   mid INT PRIMARY KEY,
   name VARCHAR(100),
   design VARCHAR(50),
   age INT
);
```

Field Types

#	Field	Schema	Table	Type
1	mid	qn2	members	INT
2	name	qn2	members	VARCHAR
3	design	qn2	members	VARCHAR
4	age	qn2	members	INT

```
CREATE TABLE Books (
   Bid INT PRIMARY KEY,
   Btitle VARCHAR(100),
   BAuthor VARCHAR(100),
   Bpublisher VARCHAR(100),
   Bprice DECIMAL(10, 2)
);
```

Field Types

#	Field	Schema	Table	Type
1	Bid	qn2	books	INT
2	Btitle	qn2	books	VARCHAR
3	BAuthor	qn2	books	VARCHAR
4	Bpublisher	qn2	books	VARCHAR
5	Bprice	qn2	books	DECIMAL

```
CREATE TABLE Reserves (mid INT, Bid INT, date DATE, PRIMARY KEY (mid, Bid),
FOREIGN KEY (mid) REFERENCES Members(mid), FOREIGN KEY (Bid) REFERENCES
Books(Bid));
```

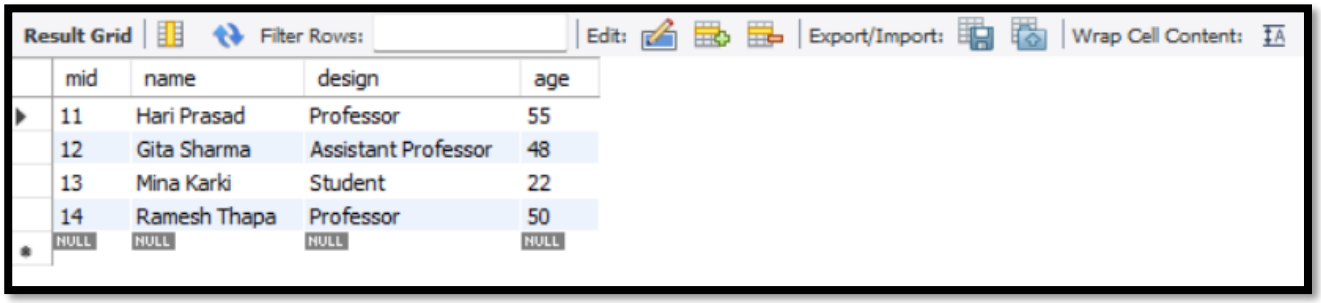
Field Types

#	Field	Schema	Table	Type
1	mid	qn2	reserves	INT
2	Bid	qn2	reserves	INT
3	date	qn2	reserves	DATE

(b) Insert any three records or more in each of the above tables and display them.

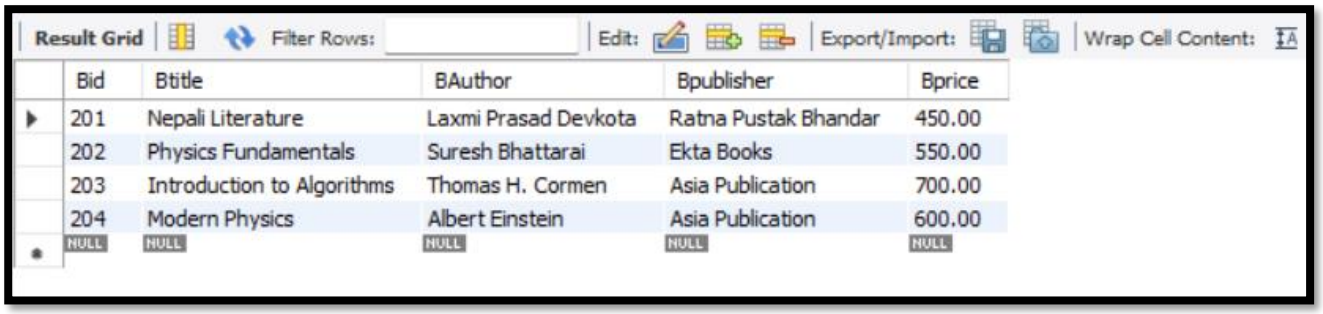
Data Insertion:

INSERT INTO Members (mid, name, design, age) VALUES
(11, 'Hari Prasad', 'Professor', 55),
(12, 'Gita Sharma', 'Assistant Professor', 48),
(13, 'Mina Karki', 'Student', 22),
(14, 'Ramesh Thapa', 'Professor', 50);



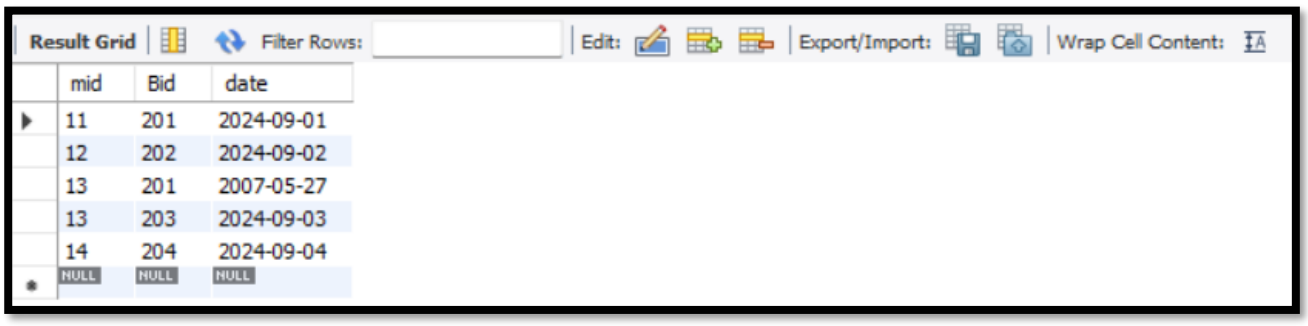
mid	name	design	age
11	Hari Prasad	Professor	55
12	Gita Sharma	Assistant Professor	48
13	Mina Karki	Student	22
14	Ramesh Thapa	Professor	50
NULL	NULL	NULL	NULL

INSERT INTO Books (Bid, Btitle, BAuthor, Bpublisher, Bprice) VALUES
(201, 'Nepali Literature', 'Laxmi Prasad Devkota', 'Ratna Pustak Bhandar', 450),
(202, 'Physics Fundamentals', 'Suresh Bhattarai', 'Ekta Books', 550),
(203, 'Introduction to Algorithms', 'Thomas H. Cormen', 'Asia Publication', 700),
(204, 'Modern Physics', 'Albert Einstein', 'Asia Publication', 600);



Bid	Btitle	BAuthor	Bpublisher	Bprice
201	Nepali Literature	Laxmi Prasad Devkota	Ratna Pustak Bhandar	450.00
202	Physics Fundamentals	Suresh Bhattarai	Ekta Books	550.00
203	Introduction to Algorithms	Thomas H. Cormen	Asia Publication	700.00
204	Modern Physics	Albert Einstein	Asia Publication	600.00
NULL	NULL	NULL	NULL	NULL

INSERT INTO Reserves (mid, Bid, date) VALUES
(11, 201, '2024-09-01'),
(12, 202, '2024-09-02'),
(13, 203, '2024-09-03'),
(13, 201, '2007-05-27'),
(14, 204, '2024-09-04');



mid	Bid	date
11	201	2024-09-01
12	202	2024-09-02
13	201	2007-05-27
13	203	2024-09-03
14	204	2024-09-04
NULL	NULL	NULL

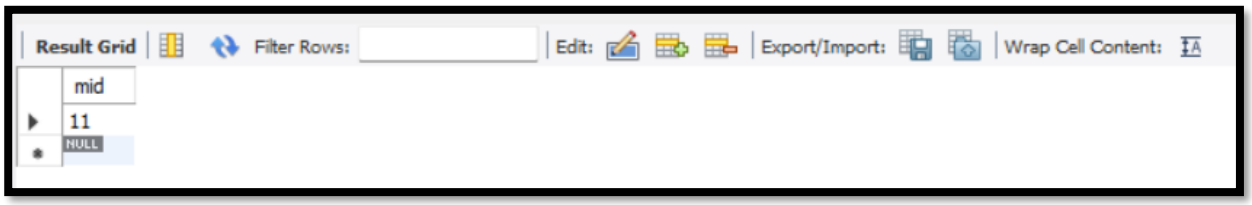
(c) Write the SQL for each of the following queries.

a) List the title of books reserved by professors older than 45 years.

```
SELECT B.Btitle
FROM Books B
JOIN Reserves R ON B.Bid = R.Bid
JOIN Members M ON R.mid = M.mid
WHERE M.design = 'Professor' AND M.age > 45;
```

b) Find ids of members who have not reserved books costing more than Rs. 500.

```
SELECT DISTINCT M.mid
FROM Members M
WHERE M.mid NOT IN (
SELECT R.mid
FROM Reserves R
JOIN Books B ON R.Bid = B.Bid
WHERE B.Bprice > 500
);
```



c) Find the author and title of books reserved on 27-May-2007.

```
SELECT B.BAuthor, B.Btitle
FROM Books B
JOIN Reserves R ON B.Bid = R.Bid
WHERE R.date = '2024-08-05';
```

d) Find the names of members who have reserved all books.

```
SELECT M.name
FROM Members M
WHERE NOT EXISTS (
SELECT B.Bid
FROM Books B
WHERE B.Bid NOT IN (
SELECT R.Bid
FROM Reserves R
WHERE R.mid = M.mid
));
```

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

	name

Result: There is no any member who reserved all books.

e) Update the price of all the books by Rs 100 whose publisher name is 'Asia Publication'

```
UPDATE Books
SET Bprice = Bprice + 100
WHERE Bpublisher = 'Asia Publication';
SELECT * FROM Books
```

	Bid	Btitle	BAuthor	Bpublisher	Bprice
▶	201	Nepali Literature	Laxmi Prasad Devkota	Ratna Pustak Bhandar	450.00
	202	Physics Fundamentals	Suresh Bhattarai	Ekta Books	550.00
	203	Introduction to Algorithms	Thomas H. Cormen	Asia Publication	800.00
	204	Modern Physics	Albert Einstein	Asia Publication	700.00
*	NULL	NULL	NULL	NULL	NULL

Result: Bprice for Bid 203 and 204 whose publisher name is 'Asia Publication' is increased by Rs.100.

f) Delete the records of all members whose age is less than 18.

```
DELETE FROM Members
WHERE age < 18;
```

	mid	name	design	age
▶	11	Hari Prasad	Professor	55
	12	Gita Sharma	Assistant Professor	48
	13	Mina Karki	Student	22
	14	Ramesh Thapa	Professor	50
*	NULL	NULL	NULL	NULL

Result: Since there is no member whose age is less than 18 so it remains unchanged.

Lab number 3

Consider the following relational schema and write the relational algebra expression and SQL for the following.

Supplier (supplier-id, supplier-name, city)
Supplies (supplier-id, part-id, quantity)
Parts (part-id, part-name, color, weight)

Table Schema:

```
CREATE TABLE Supplier (
    supplier_id INT PRIMARY KEY,
    supplier_name VARCHAR(255) NOT NULL,
    city VARCHAR(100)
);
```

Field Types

#	Field	Schema	Table	Type
1	supplier_id	qn3	supplier	INT
2	supplier_name	qn3	supplier	VARCHAR
3	city	qn3	supplier	VARCHAR

```
CREATE TABLE Parts (
    part_id INT PRIMARY KEY,
    part_name VARCHAR(255) NOT NULL,
    color VARCHAR(50),
    weight DECIMAL(10, 2)
);
```

Field Types

#	Field	Schema	Table	Type
1	part_id	qn3	parts	INT
2	part_name	qn3	parts	VARCHAR
3	color	qn3	parts	VARCHAR
4	weight	qn3	parts	DECIMAL

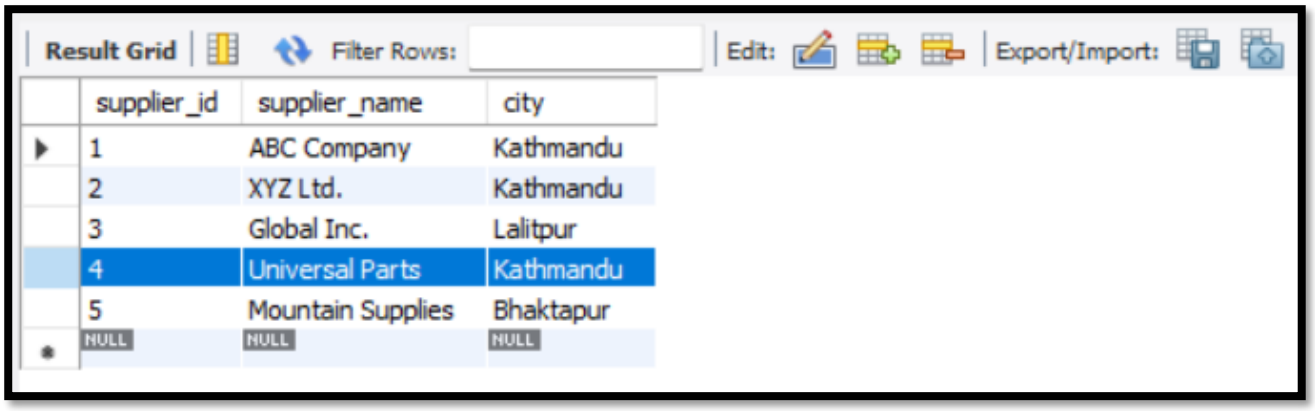
```
CREATE TABLE Supplies (
    supplier_id INT,
    part_id INT,
    quantity INT,
    PRIMARY KEY (supplier_id, part_id),
    FOREIGN KEY (supplier_id) REFERENCES Supplier(supplier_id),
    FOREIGN KEY (part_id) REFERENCES Parts(part_id)
);
```

Field Types

#	Field	Schema	Table	Type
1	supplier_id	qn3	supplies	INT
2	part_id	qn3	supplies	INT
3	quantity	qn3	supplies	INT

Data Insertion

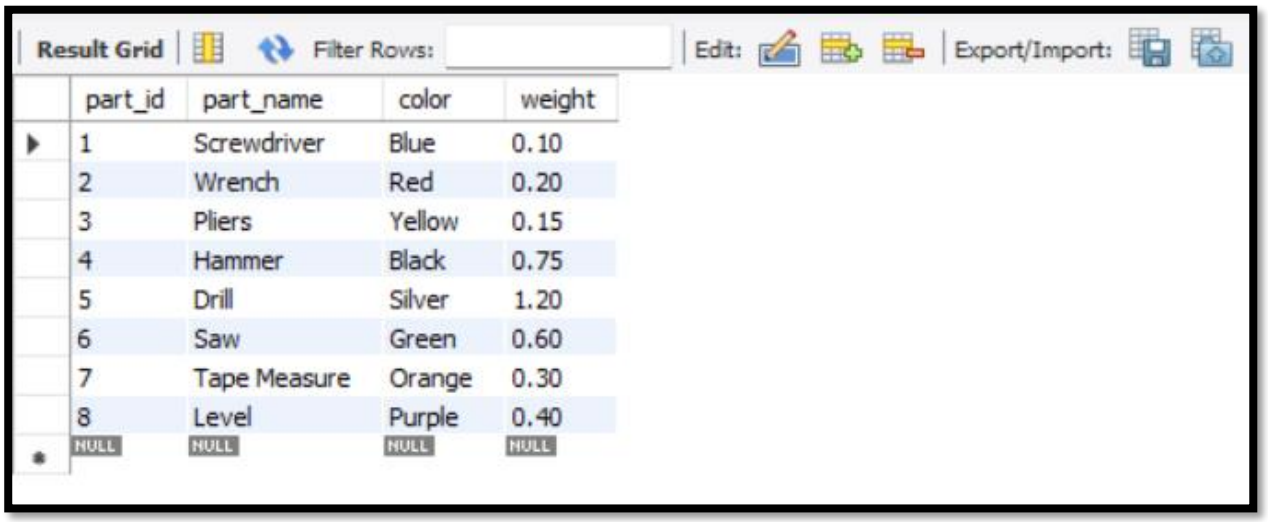
```
INSERT INTO Supplier (supplier_id, supplier_name, city)
VALUES
(1, 'ABC Company', 'Kathmandu'),
(2, 'XYZ Ltd.', 'Kathmandu'),
(3, 'Global Inc.', 'Lalitpur'),
(4, 'Universal Parts', 'Kathmandu'),
(5, 'Mountain Supplies', 'Bhaktapur');
SELECT * FROM Supplier
```



Result Grid | Filter Rows: | Edit: | Export/Import:

supplier_id	supplier_name	city
1	ABC Company	Kathmandu
2	XYZ Ltd.	Kathmandu
3	Global Inc.	Lalitpur
4	Universal Parts	Kathmandu
5	Mountain Supplies	Bhaktapur
NULL	NULL	NULL

```
INSERT INTO Parts (part_id, part_name, color, weight)
VALUES
(1, 'Screwdriver', 'Blue', 0.10),
(2, 'Wrench', 'Red', 0.20),
(3, 'Pliers', 'Yellow', 0.15),
(4, 'Hammer', 'Black', 0.75),
(5, 'Drill', 'Silver', 1.20),
(6, 'Saw', 'Green', 0.60),
(7, 'Tape Measure', 'Orange', 0.30),
(8, 'Level', 'Purple', 0.40);
SELECT * FROM Parts
```

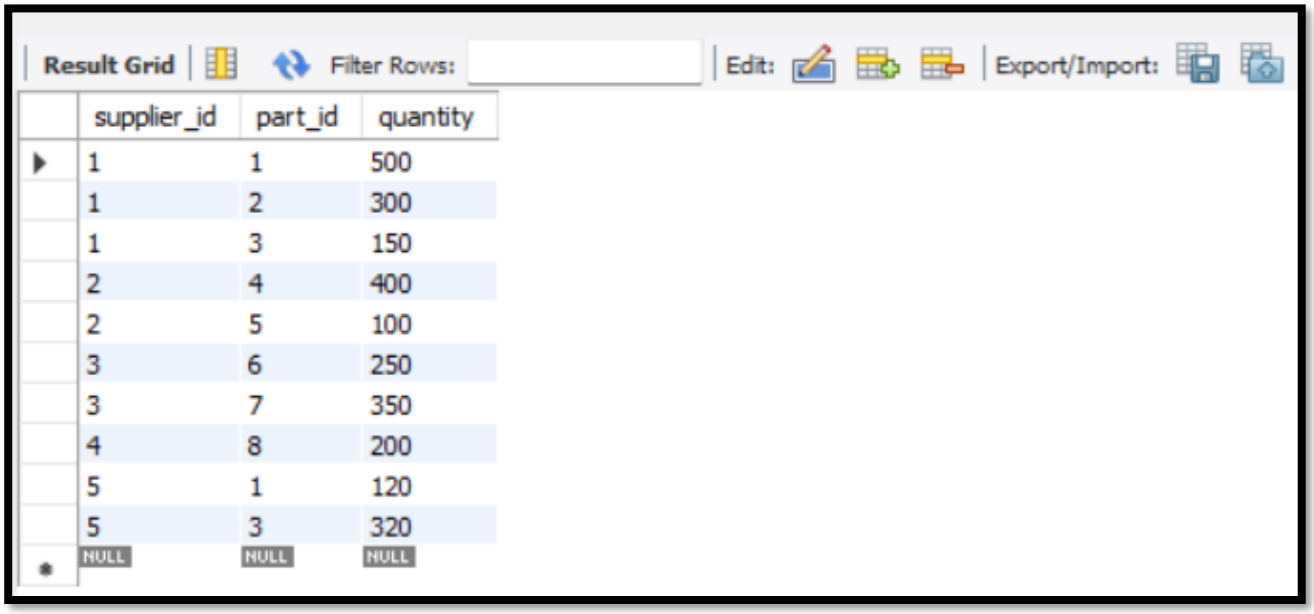


Result Grid | Filter Rows: | Edit: | Export/Import:

part_id	part_name	color	weight
1	Screwdriver	Blue	0.10
2	Wrench	Red	0.20
3	Pliers	Yellow	0.15
4	Hammer	Black	0.75
5	Drill	Silver	1.20
6	Saw	Green	0.60
7	Tape Measure	Orange	0.30
8	Level	Purple	0.40
NULL	NULL	NULL	NULL

```
INSERT INTO Supplies (supplier_id, part_id, quantity)
VALUES
(1, 1, 500),
(1, 2, 300),
(1, 3, 150),
(2, 4, 400),
(2, 5, 100),
(3, 6, 250),
(3, 7, 350),
(4, 8, 200),
(5, 1, 120),
(5, 3, 320);

SELECT * FROM Supplies
```

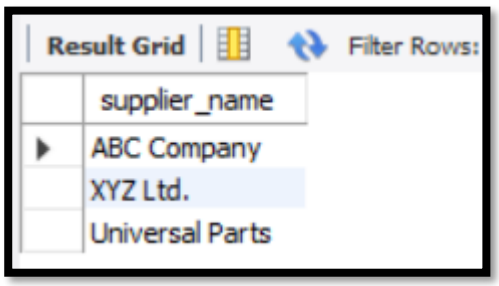


supplier_id	part_id	quantity
1	1	500
1	2	300
1	3	150
2	4	400
2	5	100
3	6	250
3	7	350
4	8	200
5	1	120
5	3	320
NULL	NULL	NULL

Queries:

a) Find the name of all supplier located in city "Kathmandu".

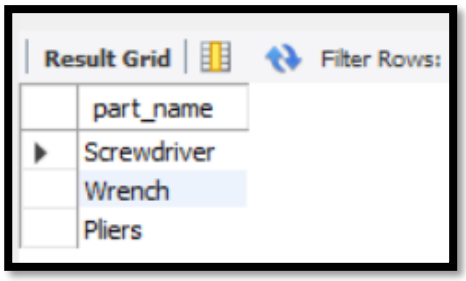
```
SELECT supplier_name
FROM Supplier
WHERE city = 'Kathmandu';
```



supplier_name
ABC Company
XYZ Ltd.
Universal Parts

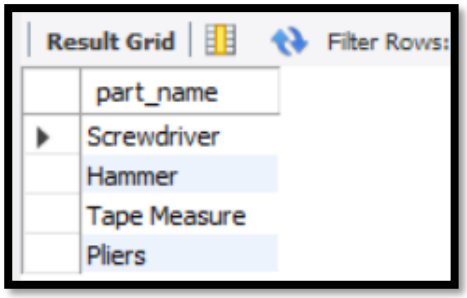
b)Find the name of all parts supplied by "ABC Company".

```
SELECT DISTINCT p.part_name
FROM Parts p
JOIN Supplies s ON p.part_id = s.part_id
JOIN Supplier sup ON s.supplier_id = sup.supplier_id
WHERE sup.supplier_name = 'ABC Company';
```



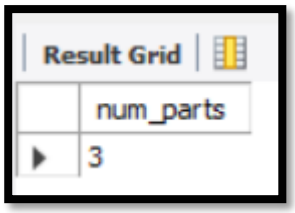
c) Find the name of all parts that are supplied in quantity greater than 300.

```
SELECT DISTINCT p.part_name
FROM Parts p
JOIN Supplies s ON p.part_id = s.part_id
WHERE s.quantity > 300;
```



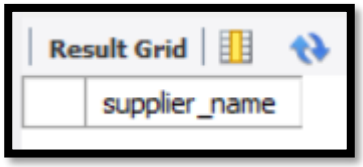
d)Find the number of parts supplied by "ABC Company'.

```
SELECT COUNT(DISTINCT s.part_id) AS num_parts
FROM Supplies s
JOIN Supplier sup ON s.supplier_id = sup.supplier_id
WHERE sup.supplier_name = 'ABC Company';
```



e) Find the name of all suppliers who supply more than 30 different parts.

```
SELECT sup.supplier_name
FROM Supplies s
JOIN Supplier sup ON s.supplier_id = sup.supplier_id
GROUP BY sup.supplier_id, sup.supplier_name
HAVING COUNT(DISTINCT s.part_id) > 30;
```



Lab Number 4

Consider the following relational schema and write the SQL for the following.

student(id, name)
enrolledIn(id, code)
subject(code, lecturer)

Table Schema

```
CREATE TABLE student (
   id INT PRIMARY KEY,
   name VARCHAR(100)
);
```

Field Types

#	Field	Schema	Table	Type
1	id	qn4_kiran	student	INT
2	name	qn4_kiran	student	VARCHAR

```
CREATE TABLE subject (
   code VARCHAR(10) PRIMARY KEY,
   lecturer VARCHAR(100)
);
```

Field Types

#	Field	Schema	Table	Type
1	code	qn4_kiran	subject	VARCHAR
2	lecturer	qn4_kiran	subject	VARCHAR

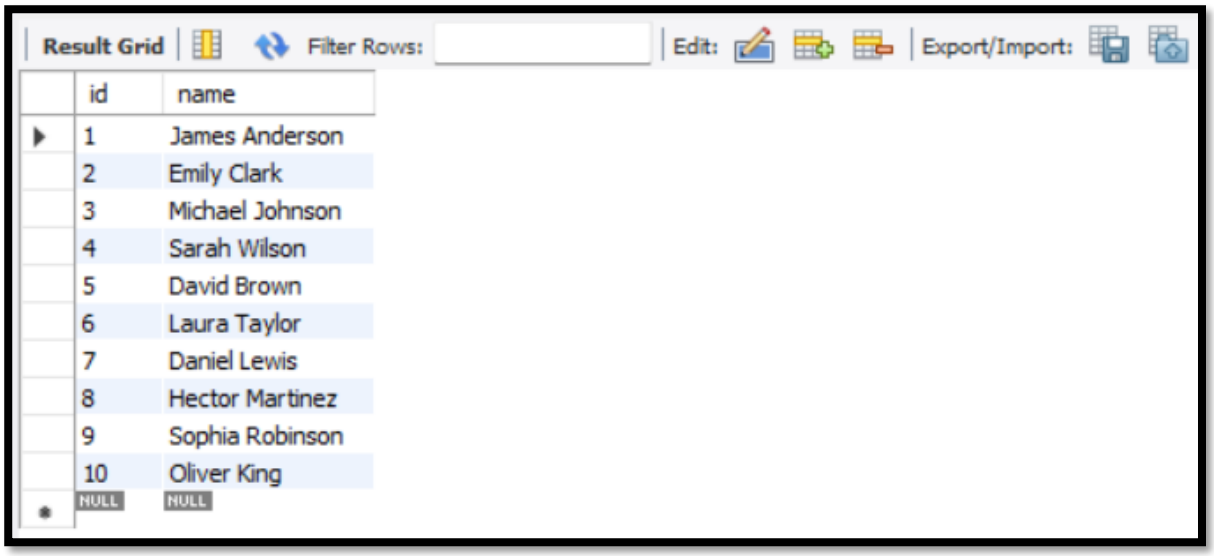
```
CREATE TABLE enrolledIn (
   id INT,
   code VARCHAR(10),
   PRIMARY KEY (id, code),
   FOREIGN KEY (id) REFERENCES student(id),
   FOREIGN KEY (code) REFERENCES subject(code)
);
```

Field Types

#	Field	Schema	Table	Type
1	id	qn4_kiran	enrolledin	INT
2	code	qn4_kiran	enrolledin	VARCHAR

Data Insertion

```
INSERT INTO student (id, name)
VALUES
(1, 'James Anderson'),
(2, 'Emily Clark'),
(3, 'Michael Johnson'),
(4, 'Sarah Wilson'),
(5, 'David Brown'),
(6, 'Laura Taylor'),
(7, 'Daniel Lewis'),
(8, 'Hector Martinez'),
(9, 'Sophia Robinson'),
(10, 'Oliver King');
SELECT * FROM student
```



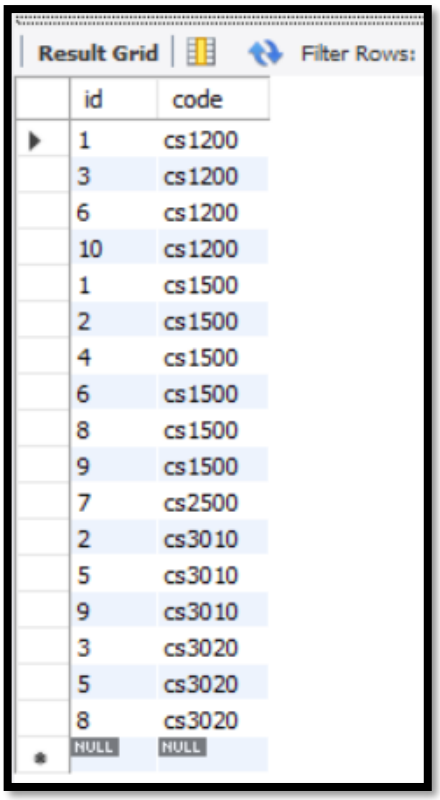
id	name
1	James Anderson
2	Emily Clark
3	Michael Johnson
4	Sarah Wilson
5	David Brown
6	Laura Taylor
7	Daniel Lewis
8	Hector Martinez
9	Sophia Robinson
10	Oliver King
NULL	NULL

```
INSERT INTO subject (code, lecturer)
VALUES
('cs1500', 'Dr. Alice Cooper'),
('cs1200', 'Dr. Bob Green'),
('cs3020', 'Dr. Carol White'),
('cs3010', 'Dr. Derek Black'),
('cs2500', 'Dr. Fiona Adams');
SELECT * FROM subject;
```

code	lecturer
cs1200	Dr. Bob Green
cs1500	Dr. Alice Cooper
cs2500	Dr. Fiona Adams
cs3010	Dr. Derek Black
cs3020	Dr. Carol White
NULL	NULL

```
INSERT INTO enrolledIn (id, code)
VALUES
(1, 'cs1500'),
(1, 'cs1200'),
(2, 'cs1500'),
(2, 'cs3010'),
(3, 'cs3020'),
(3, 'cs1200'),
(4, 'cs1500'),
(5, 'cs3010'),
(5, 'cs3020'),
(6, 'cs1500'),
(6, 'cs1200'),
(7, 'cs2500'),
(8, 'cs3020'),
(8, 'cs1500'),
(9, 'cs3010'),
(9, 'cs1500'),
(10, 'cs1200');

SELECT * FROM enrolledIn
```

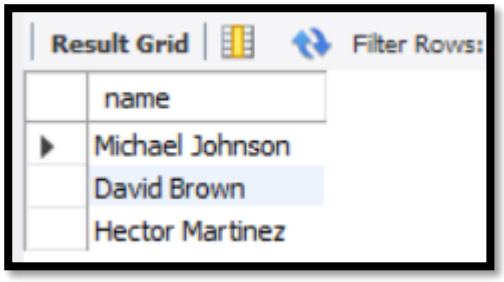


id	code
1	cs1200
3	cs1200
6	cs1200
10	cs1200
1	cs1500
2	cs1500
4	cs1500
6	cs1500
8	cs1500
9	cs1500
7	cs2500
2	cs3010
5	cs3010
9	cs3010
3	cs3020
5	cs3020
8	cs3020
NULL	NULL

Queries:

a) What are the names of students enrolled in cs3020?

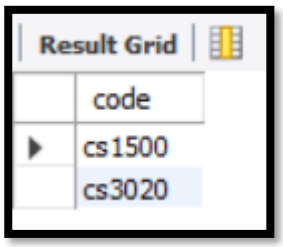
```
SELECT student.name
FROM student
JOIN enrolledIn ON student.id = enrolledIn.id
WHERE enrolledIn.code = 'cs3020';
```



name
Michael Johnson
David Brown
Hector Martinez

b) Which subjects is Hector taking?

```
SELECT subject.code
FROM subject
JOIN enrolledIn ON subject.code = enrolledIn.code
JOIN student ON student.id = enrolledIn.id
WHERE student.name = 'Hector Martinez';
```



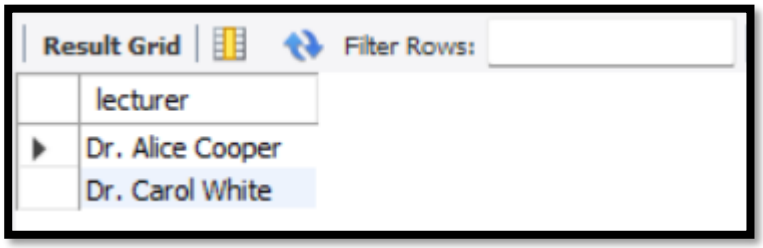
code
cs1500
cs3020

c) Who teaches cs1500?

```
SELECT lecturer
FROM subject
WHERE code = 'cs1500';
```

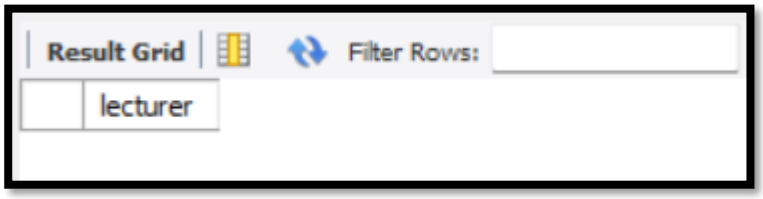
d) Who teaches cs1500 or cs3020?

```
SELECT lecturer
FROM subject
WHERE code = 'cs1500' OR code = 'cs3020';
```



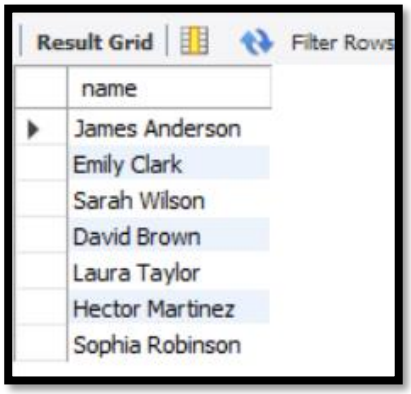
e) Who teaches at least two different subjects?

```
SELECT lecturer
FROM subject
GROUP BY lecturer
HAVING COUNT(DISTINCT code) >= 2;
```



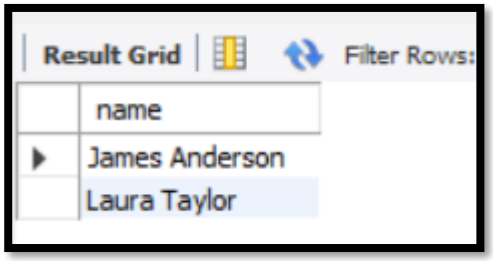
f) What are the names of students in cs1500 or cs3010?

```
SELECT DISTINCT student.name
FROM student
JOIN enrolledIn ON student.id = enrolledIn.id
WHERE enrolledIn.code = 'cs1500' OR enrolledIn.code = 'cs3010';
```



g) What are the names of students in both cs1500 and cs1200?

```
SELECT student.name
FROM student
JOIN enrolledIn e1 ON student.id = e1.id
JOIN enrolledIn e2 ON student.id = e2.id
WHERE e1.code = 'cs1500' AND e2.code = 'cs1200';
```

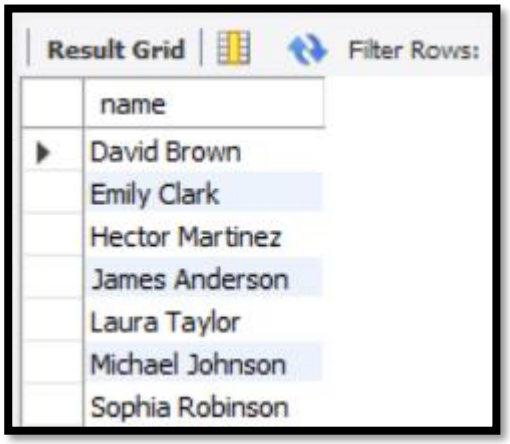


Result Grid | Filter Rows:

name
James Anderson
Laura Taylor

h) What are the names of students in at least two different subjects?

```
SELECT student.name
FROM student
JOIN enrolledIn ON student.id = enrolledIn.id
GROUP BY student.name
HAVING COUNT(DISTINCT enrolledIn.code) >= 2;
```

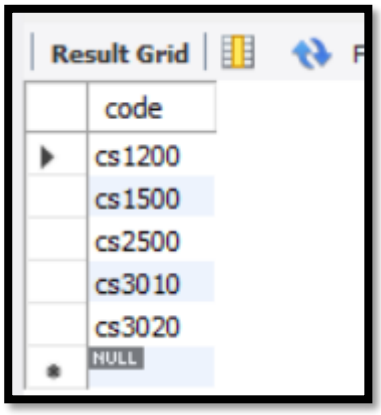


Result Grid | Filter Rows:

name
David Brown
Emily Clark
Hector Martinez
James Anderson
Laura Taylor
Michael Johnson
Sophia Robinson

i)What are the codes of all the subjects taught?

```
SELECT code
FROM subject;
```



Result Grid | F

code
cs1200
cs1500
cs2500
cs3010
cs3020
NULL

j) What are the names of all the students?

```
SELECT name
FROM student;
```

Result Grid | Filter Rows

name
James Anderson
Emily Clark
Michael Johnson
Sarah Wilson
David Brown
Laura Taylor
Daniel Lewis
Hector Martinez
Sophia Robinson
Oliver King

k) What are the names of all the students in cs1500?

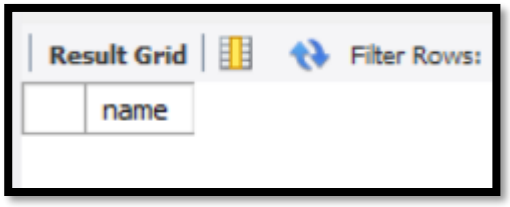
```
SELECT student.name
FROM student
JOIN enrolledIn ON student.id = enrolledIn.id
WHERE enrolledIn.code = 'cs1500';
```

Result Grid | Filter Rows:

name
James Anderson
Emily Clark
Sarah Wilson
Laura Taylor
Hector Martinez
Sophia Robinson

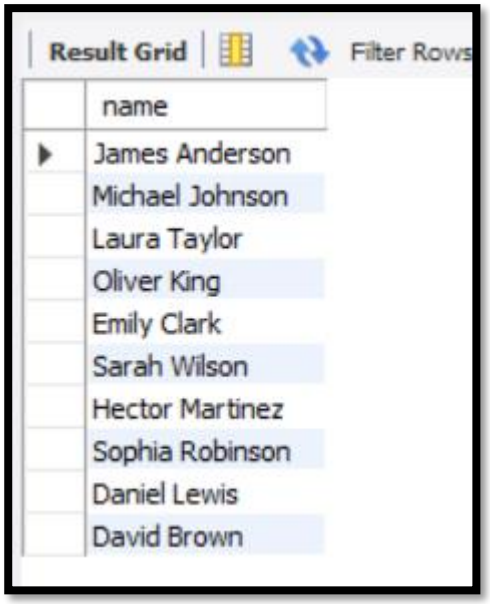
l) What are the names of students taking a subject taught by Dr. James Morgan.

```
SELECT DISTINCT student.name
FROM student
JOIN enrolledIn ON student.id = enrolledIn.id
JOIN subject ON enrolledIn.code = subject.code
WHERE subject.lecturer = 'Dr. James Morgan';
```



m) What are the names of students who are taking a subject not taught by Dr. James Morgan?

```
SELECT DISTINCT student.name
FROM student
JOIN enrolledIn ON student.id = enrolledIn.id
JOIN subject ON enrolledIn.code = subject.code
WHERE subject.lecturer <> 'Dr. James Morgan';
```



Lab Number 5

Consider the following relational database.

Student(snum: integer, sname: string, major: string, level: string, age: integer)
Class(name: string, meets at: time, room: string, fid: integer)
Enrolled(snum: integer, cname: string)
Faculty(fid: integer, fname: string, deptid: integer)

Table Schema

```
CREATE TABLE Student (
    snum INTEGER PRIMARY KEY,
    sname VARCHAR(255) NOT NULL,
    major VARCHAR(255),
    level VARCHAR(255),
    age INTEGER
);
```

Field Types

#	Field	Schema	Table	Type
1	snum	kiran_qn5	student	INT
2	sname	kiran_qn5	student	VARCHAR
3	major	kiran_qn5	student	VARCHAR
4	level	kiran_qn5	student	VARCHAR
5	age	kiran_qn5	student	INT

```
CREATE TABLE Faculty (
    fid INTEGER PRIMARY KEY,
    fname VARCHAR(255) NOT NULL,
    deptid INTEGER
);
```

Field Types

#	Field	Schema	Table	Type
1	fid	kiran_qn5	faculty	INT
2	fname	kiran_qn5	faculty	VARCHAR
3	deptid	kiran_qn5	faculty	INT

```
CREATE TABLE Class (
    name VARCHAR(255) PRIMARY KEY,
    meets_at TIME,
    room VARCHAR(255),
    fid INTEGER,
    FOREIGN KEY (fid) REFERENCES Faculty(fid)
);
```

Field Types

#	Field	Schema	Table	Type
1	name	kiran_qn5	class	VARCHAR
2	meets_at	kiran_qn5	class	TIME
3	room	kiran_qn5	class	VARCHAR
4	fid	kiran_qn5	class	INT

```
CREATE TABLE Enrolled (
    snum INTEGER,
    cname VARCHAR(255),
    PRIMARY KEY (snum, cname),
    FOREIGN KEY (snum) REFERENCES Student(snum),
    FOREIGN KEY (cname) REFERENCES Class(name)
);
```

Field Types

#	Field	Schema	Table	Type
1	snum	kiran_qn5	enrolled	INT
2	cname	kiran_qn5	enrolled	VARCHAR

Data Insertion

```
INSERT INTO Student (snum, sname, major, level, age)
VALUES
(1001, 'Suman Acharya', 'Computer Science', 'JR', 22),
(1002, 'Amelia Brown', 'Mathematics', 'JR', 21),
(1003, 'Hans Müller', 'History', 'SR', 25),
(1004, 'Pooja Sharma', 'History', 'SO', 19);
SELECT * FROM Student
```

Result Grid | Filter Rows: | Edit: | Export/Import:

snum	sname	major	level	age
1001	Suman Acharya	Computer Science	JR	22
1002	Amelia Brown	Mathematics	JR	21
1003	Hans Müller	History	SR	25
1004	Pooja Sharma	History	SO	19
NULL	NULL	NULL	NULL	NULL

```
INSERT INTO Faculty (fid, fname, deptid)
VALUES
(2001, 'Dr. Olivia Thompson', 1),
(2002, 'Prof. Peter Schmidt', 2),
(2003, 'I. Teach', 3),
(2004, 'Dr. Ravi Koirala', 4);
SELECT * FROM Faculty
```

Result Grid | Filter Rows:

fid	fname	deptid
2001	Dr. Olivia Thompson	1
2002	Prof. Peter Schmidt	2
2003	I. Teach	3
2004	Dr. Ravi Koirala	4
NULL	NULL	NULL

```
INSERT INTO Class (name, meets_at, room, fid)
VALUES
('CS101', '09:00:00', 'Room 101', 2001),
('MATH202', '10:30:00', 'Room 202', 2002),
('PHYS303', '13:00:00', 'Room 303', 2003),
('HIST404', '15:00:00', 'R128', 2004);
SELECT * FROM Class
```

Result Grid | Filter Rows:

name	meets_at	room	fid
CS101	09:00:00	Room 101	2001
HIST404	15:00:00	R128	2004
MATH202	10:30:00	Room 202	2002
PHYS303	13:00:00	Room 303	2003
NULL	NULL	NULL	NULL

```
INSERT INTO Enrolled (snum, cname)
VALUES
(1001, 'CS101'),
(1002, 'PHYS303'),
(1003, 'HIST404'),
(1004, 'PHYS303');
SELECT * FROM Enrolled
```

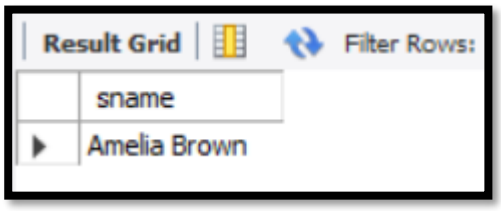
Result Grid | Filter Rows:

snum	cname
1001	CS101
1003	HIST404
1002	PHYS303
1004	PHYS303
NULL	NULL

Queries:

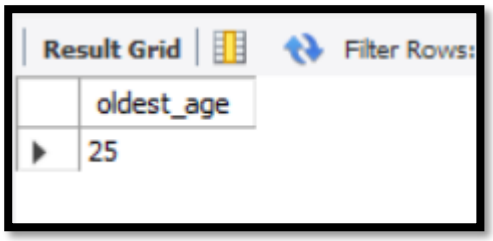
a) Find the names of all Juniors (Level = JR) who are enrolled in a class taught by I. Teach.

```
SELECT s.sname
FROM Student s
JOIN Enrolled e ON s.snum = e.snum
JOIN Class c ON e.cname = c.name
JOIN Faculty f ON c.fid = f.fid
WHERE s.level = 'JR' AND f.fname = 'I. Teach';
```



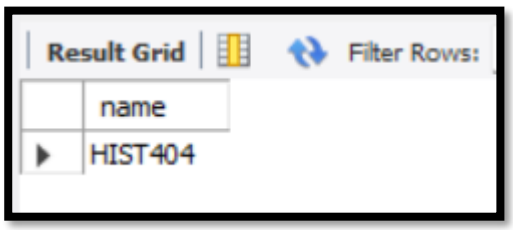
b) Find the age of the oldest student who is either a History major or is enrolled in a course taught by I. Teach.

```
SELECT MAX(s.age) AS oldest_age
FROM Student s
WHERE s.major = 'History'
  OR EXISTS (
     SELECT 1
     FROM Enrolled e
     JOIN Class c ON e.cname = c.name
     JOIN Faculty f ON c.fid = f.fid
     WHERE e.snum = s.snum AND f.fname = 'I. Teach'
);
```



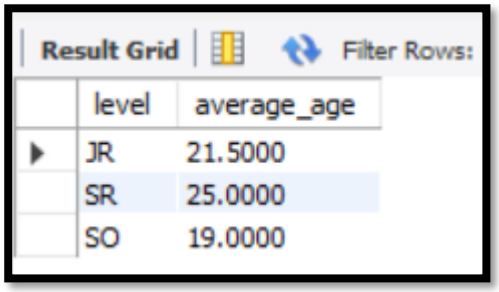
c) Find the names of all classes that either meet in room R128 or have five or more students enrolled.

```
SELECT c.name
FROM Class c
LEFT JOIN Enrolled e ON c.name = e.cname
GROUP BY c.name, c.room
HAVING c.room = 'R128'
  OR COUNT(e.snum) >= 5;
```



d) Print the Level and the average age of students for that Level, for each Level.

```
SELECT s.level, AVG(s.age) AS average_age
FROM Student s
GROUP BY s.level;
```

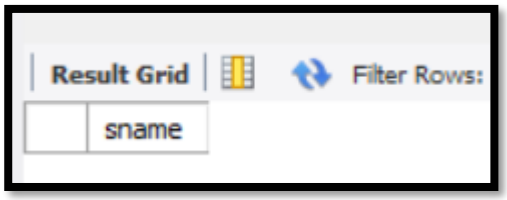


Result Grid | Filter Rows:

level	average_age
JR	21.5000
SR	25.0000
SO	19.0000

e) Find the names of students who are not enrolled in any class.

```
SELECT s.sname
FROM Student s
LEFT JOIN Enrolled e ON s.snum = e.snum
WHERE e.snum IS NULL;
```

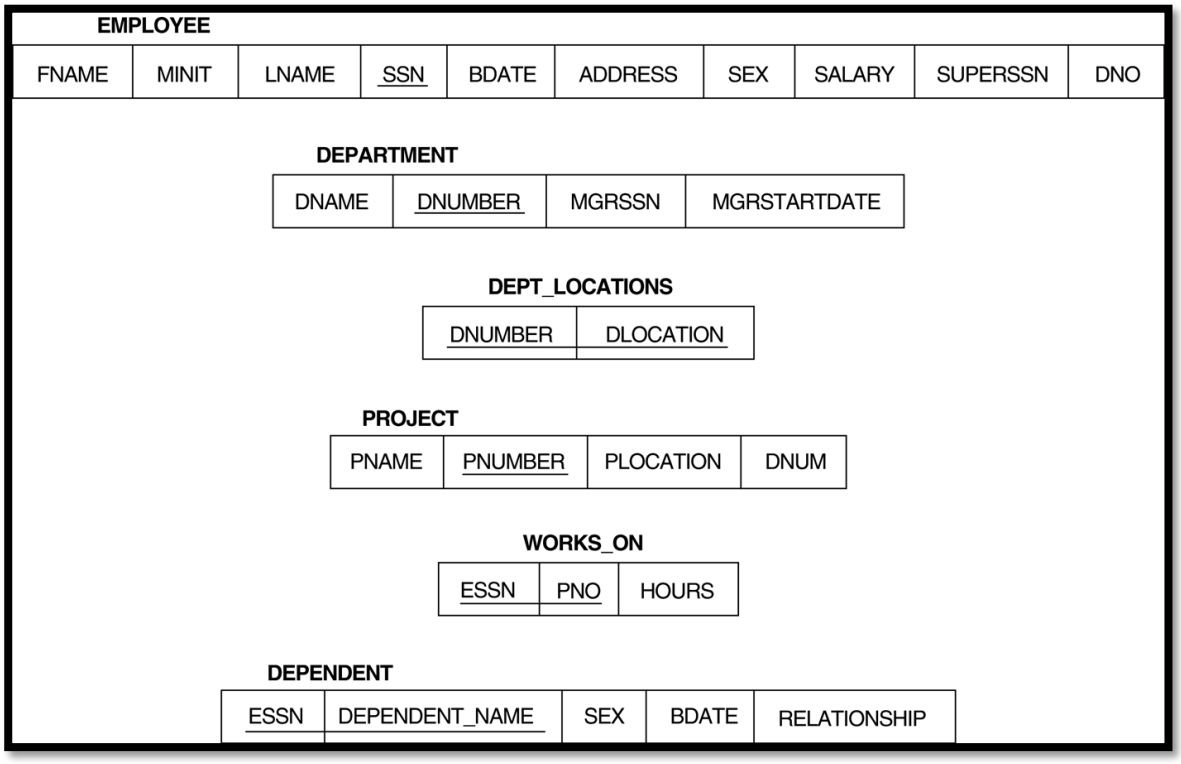


Result Grid | Filter Rows:

sname

Lab Number 6

Consider the following database schema. What are the referential integrity constraints that should be held on the schema? Write appropriate SQL DDL statements to define the database.



Referential Integrity Constraints

1. **Employee to Department (**Dno**):**
   - **Constraint:** Dno in employee must match an existing Dnumber in department.
   - **Action:** If a department is deleted, the corresponding employees should either have their Dno set to NULL or be deleted. (ON DELETE SET NULL or ON DELETE CASCADE).

2. **Employee to Employee (**Super_ssn**):**
   - **Constraint:** Super_ssn in employee must match an existing Ssn in employee.
   - **Action:** If a supervisor is deleted, the corresponding subordinates should either have their Super_ssn set to NULL or be deleted. (ON DELETE SET NULL or ON DELETE CASCADE).

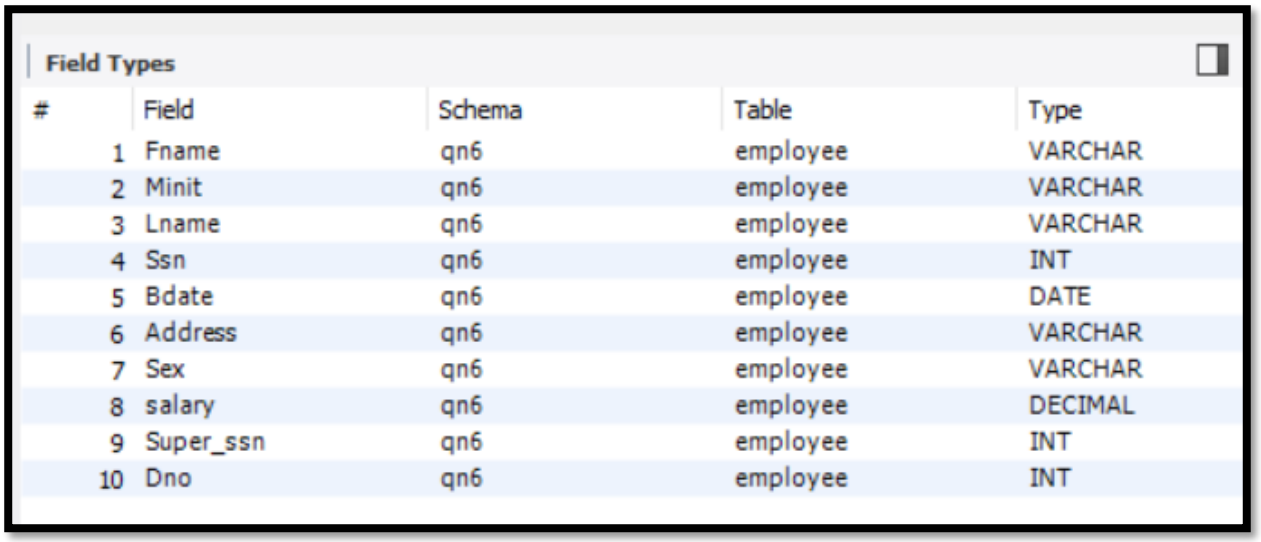
3. **Department to Employee (**Mgr_ssn**):**
   - **Constraint:** Mgr_ssn in department must match an existing Ssn in employee.
   - **Action:** If an employee who is a manager is deleted, the corresponding department's Mgr_ssn should be set to NULL or the department should be deleted. (ON DELETE SET NULL or ON DELETE CASCADE).

4. **Project to Department (**Dnum**):**
   - **Constraint:** Dnum in project must match an existing Dnumber in department.
   - **Action:** If a department is deleted, the corresponding projects should either have their Dnum set to NULL or be deleted. (ON DELETE SET NULL or ON DELETE CASCADE).

5. **Works_On to Employee (**Essn**):**
    - **Constraint:** Essn in works_on must match an existing Ssn in employee.
    - **Action:** If an employee is deleted, the corresponding records in works_on should be deleted. (ON DELETE CASCADE).
6. **Works_On to Project (**Pno**):**
    - **Constraint:** Pno in works_on must match an existing Pnumber in project.
    - **Action:** If a project is deleted, the corresponding records in works_on should be deleted. (ON DELETE CASCADE).
7. **Dependent to Employee (**Essn**):**
    - **Constraint:** Essn in dependent must match an existing Ssn in employee.
    - **Action:** If an employee is deleted, the corresponding dependents should be deleted. (ON DELETE CASCADE).
8. **Dept_Locations to Department (**Dnumber**):**
    - **Constraint:** Dnumber in dept_locations must match an existing Dnumber in department.
    - **Action:** If a department is deleted, the corresponding records in dept_locations should be deleted. (ON DELETE CASCADE).

Table Schema

```
CREATE TABLE employee (
   Fname VARCHAR(30),
   Minit VARCHAR(10),
   Lname VARCHAR(40),
   Ssn INT PRIMARY KEY,
   Bdate DATE,
   Address VARCHAR(60),
   Sex VARCHAR(1),
   Salary DECIMAL(8,2),
   Super_ssn INT,
   Dno INT;
);
```



Field Types

#	Field	Schema	Table	Type
1	Fname	qn6	employee	VARCHAR
2	Minit	qn6	employee	VARCHAR
3	Lname	qn6	employee	VARCHAR
4	Ssn	qn6	employee	INT
5	Bdate	qn6	employee	DATE
6	Address	qn6	employee	VARCHAR
7	Sex	qn6	employee	VARCHAR
8	salary	qn6	employee	DECIMAL
9	Super_ssn	qn6	employee	INT
10	Dno	qn6	employee	INT

```
CREATE TABLE department (
    Dname VARCHAR(30),
    Dnumber INT PRIMARY KEY,
    Mgr_ssn INT,
    Mgr_start_date DATE
);
```

Field Types

#	Field	Schema	Table	Type
1	Dname	qn6	department	VARCHAR
2	Dnumber	qn6	department	INT
3	Mgr_ssn	qn6	department	INT
4	Mgr_start_date	qn6	department	DATE

```
CREATE TABLE project (
    Pname VARCHAR(50),
    Pnumber INT PRIMARY KEY,
    Plocation VARCHAR(50),
    Dnum INT,
    FOREIGN KEY (Dnum) REFERENCES department(Dnumber) ON DELETE SET NULL
);
```

Field Types

#	Field	Schema	Table	Type
1	Pname	qn6	project	VARCHAR
2	Pnumber	qn6	project	INT
3	Plocation	qn6	project	VARCHAR
4	Dnum	qn6	project	INT

```
CREATE TABLE works_on (
    Essn INT,
    Pno INT,
    Hours DECIMAL(5,2),
    PRIMARY KEY (Essn, Pno),
    FOREIGN KEY (Essn) REFERENCES employee(Ssn) ON DELETE CASCADE,
    FOREIGN KEY (Pno) REFERENCES project(Pnumber) ON DELETE CASCADE
);
```

Field Types

#	Field	Schema	Table	Type
1	Essn	qn6	works_on	INT
2	Pno	qn6	works_on	INT
3	Hours	qn6	works_on	DECIMAL

```
CREATE TABLE dependent (
    Essn INT,
    dependent_name VARCHAR(50),
    Sex VARCHAR(1),
    Bdate DATE,
    Relationship VARCHAR(40),
    PRIMARY KEY (Essn, dependent_name),
    FOREIGN KEY (Essn) REFERENCES employee(Ssn) ON DELETE CASCADE
);
```

Field Types

#	Field	Schema	Table	Type
1	Essn	qn6	dependent	INT
2	dependent_name	qn6	dependent	VARCHAR
3	Sex	qn6	dependent	VARCHAR
4	Bdate	qn6	dependent	DATE
5	Relationship	qn6	dependent	VARCHAR

```
CREATE TABLE dept_locations (
    Dnumber INT,
    Dlocation VARCHAR(40),
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES department(Dnumber)
);
```

Field Types

#	Field	Schema	Table	Type
1	Dnumber	qn6	dept_locations	INT
2	Dlocation	qn6	dept_locations	VARCHAR

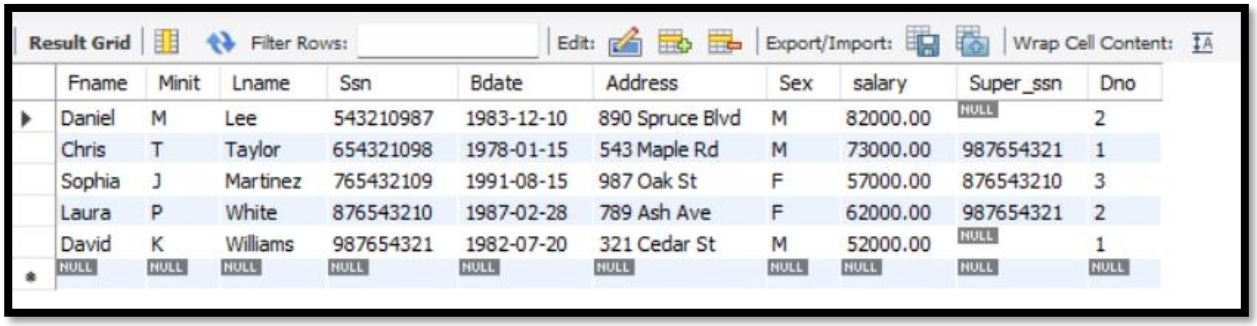
Data Insertion

```
INSERT INTO department (Dname, Dnumber, Mgr_ssn, Mgr_start_date)
VALUES
('Innovation', 1, NULL, '2020-01-01'),
('Marketing', 2, NULL, '2019-03-15'),
('Operations', 3, NULL, '2021-06-10');
```

Result Grid | Filter Rows: | Edit: | Export/Import:

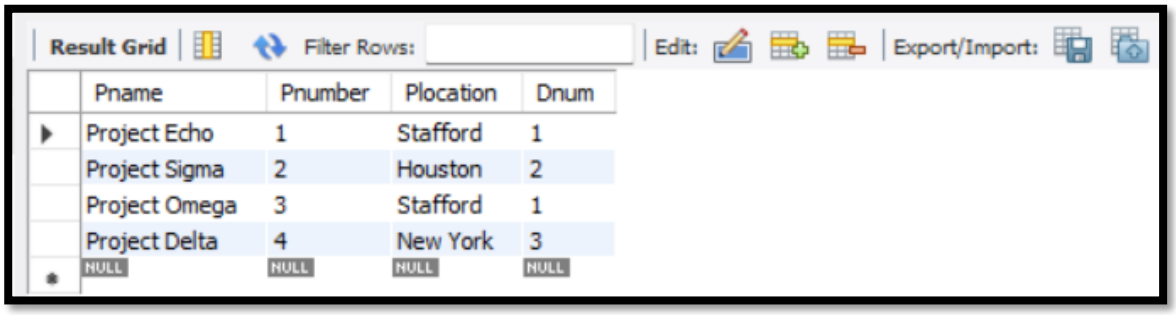
Dname	Dnumber	Mgr_ssn	Mgr_start_date
Innovation	1	NULL	2020-01-01
Marketing	2	NULL	2019-03-15
Operations	3	NULL	2021-06-10
NULL	NULL	NULL	NULL

INSERT INTO employee (Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno)
VALUES
('David', 'K', 'Williams', 987654321, '1982-07-20', '321 Cedar St', 'M', 52000.00, NULL, 1),
('Laura', 'P', 'White', 876543210, '1987-02-28', '789 Ash Ave', 'F', 62000.00, 987654321, 2),
('Sophia', 'J', 'Martinez', 765432109, '1991-08-15', '987 Oak St', 'F', 57000.00, 876543210, 3),
('Chris', 'T', 'Taylor', 654321098, '1978-01-15', '543 Maple Rd', 'M', 73000.00, 987654321, 1),
('Daniel', 'M', 'Lee', 543210987, '1983-12-10', '890 Spruce Blvd', 'M', 82000.00, NULL, 2);



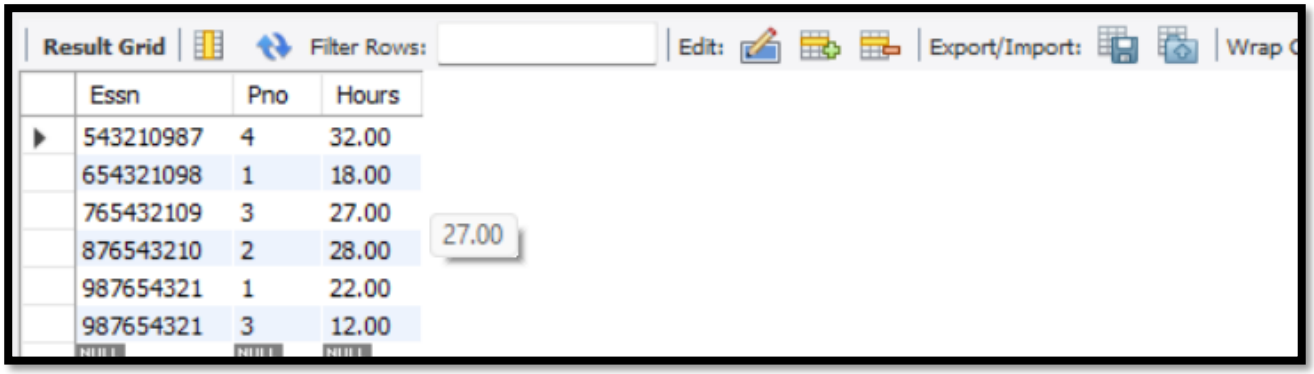
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	salary	Super_ssn	Dno
Daniel	M	Lee	543210987	1983-12-10	890 Spruce Blvd	M	82000.00	NULL	2
Chris	T	Taylor	654321098	1978-01-15	543 Maple Rd	M	73000.00	987654321	1
Sophia	J	Martinez	765432109	1991-08-15	987 Oak St	F	57000.00	876543210	3
Laura	P	White	876543210	1987-02-28	789 Ash Ave	F	62000.00	987654321	2
David	K	Williams	987654321	1982-07-20	321 Cedar St	M	52000.00	NULL	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

INSERT INTO project (Pname, Pnumber, Plocation, Dnum)
VALUES
('Project Echo', 1, 'Stafford', 1),
('Project Sigma', 2, 'Houston', 2),
('Project Omega', 3, 'Stafford', 1),
('Project Delta', 4, 'New York', 3);



Pname	Pnumber	Plocation	Dnum
Project Echo	1	Stafford	1
Project Sigma	2	Houston	2
Project Omega	3	Stafford	1
Project Delta	4	New York	3
NULL	NULL	NULL	NULL

INSERT INTO works_on (Essn, Pno, Hours)
VALUES
(987654321, 1, 22.00),
(876543210, 2, 28.00),
(765432109, 3, 27.00),
(654321098, 1, 18.00),
(543210987, 4, 32.00),
(987654321, 3, 12.00);



Essn	Pno	Hours
543210987	4	32.00
654321098	1	18.00
765432109	3	27.00
876543210	2	28.00
987654321	1	22.00
987654321	3	12.00

```
INSERT INTO dependent (Essn, dependent_name, Sex, Bdate, Relationship)
VALUES
(987654321, 'Olivia', 'F', '2012-09-18', 'Daughter'),
(876543210, 'Ryan', 'M', '2014-10-30', 'Son'),
(876543210, 'Sophia', 'F', '2016-03-25', 'Daughter'),
(765432109, 'Ethan', 'M', '2018-07-10', 'Son'),
(543210987, 'Mia', 'F', '2020-01-05', 'Daughter');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

Essn	dependent_name	Sex	Bdate	Relationship
543210987	Mia	F	2020-01-05	Daughter
765432109	Ethan	M	2018-07-10	Son
876543210	Ryan	M	2014-10-30	Son
876543210	Sophia	F	2016-03-25	Daughter
987654321	Olivia	F	2012-09-18	Daughter
NULL	NULL	NULL	NULL	NULL

```
INSERT INTO dept_locations (Dnumber, Dlocation)
VALUES
(1, 'Stafford'),
(2, 'Houston'),
(3, 'New York');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

Dnumber	Dlocation
1	Stafford
2	Houston
3	New York
NULL	NULL

Queries:

a) Retrieve the name and address of all employees who work for the 'Innovation' department.

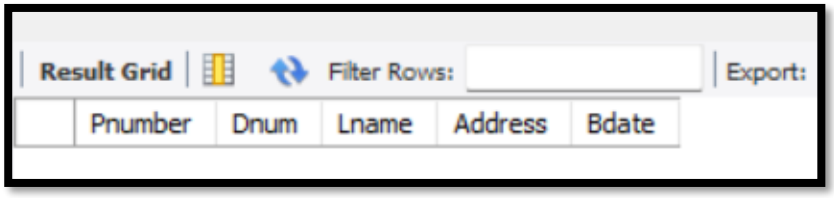
```
SELECT e.Fname, e.Minit, e.Lname, e.Address
FROM employee e
JOIN department d ON e.Dno = d.Dnumber
WHERE d.Dname = Innovation';
```

Result Grid | Filter Rows: | Export:

Fname	Minit	Lname	Address
Chris	T	Taylor	543 Maple Rd
David	K	Williams	321 Cedar St

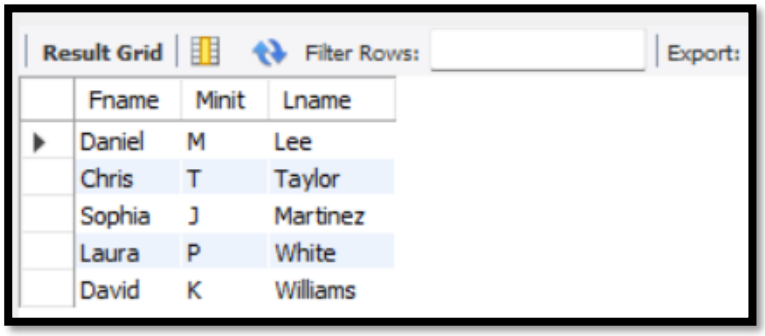
b) For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date.

```
SELECT p.Pnumber, p.Dnum, e.Lname, e.Address, e.Bdate
FROM project p
JOIN department d ON p.Dnum = d.Dnumber
JOIN employee e ON d.Mgr_ssn = e.Ssn
WHERE p.Plocation = 'Stafford';
```



c) Find the names of employees who work on all the projects controlled by department number 5.

```
SELECT e.Fname, e.Minit, e.Lname
FROM employee e
WHERE NOT EXISTS (
SELECT p.Pnumber
FROM project p
WHERE p.Dnum = 5
AND NOT EXISTS (
SELECT w.Essn
FROM works_on w
WHERE w.Essn = e.Ssn
AND w.Pno = p.Pnumber
));
```

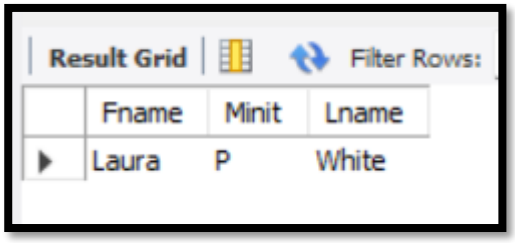


d) Make a list of project numbers for projects that involve an employee whose last name is ‘Williams’, either as a worker or as a manager of the department that controls the project.

```
SELECT DISTINCT p.Pnumber
FROM project p
LEFT JOIN works_on w ON p.Pnumber = w.Pno
LEFT JOIN employee e1 ON w.Essn = e1.Ssn
LEFT JOIN department d ON p.Dnum = d.Dnumber
LEFT JOIN employee e2 ON d.Mgr_ssn = e2.Ssn
WHERE e1.Lname = 'Williams' OR e2.Lname = ‘Williams’;
```

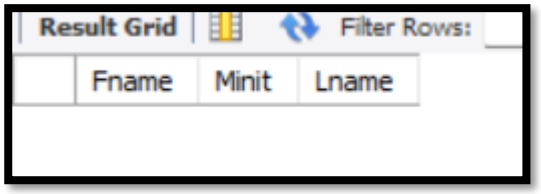
e) List the names of all employees with two or more dependents

```
SELECT e.Fname, e.Minit, e.Lname
FROM employee e
JOIN dependent d ON e.Ssn = d.Essn
GROUP BY e.Ssn, e.Fname, e.Minit, e.Lname
HAVING COUNT(d.dependent_name) >= 2;
```



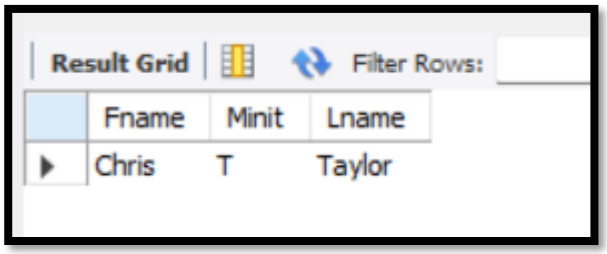
f) List the names of managers who have at least one dependent

```
SELECT e.Fname, e.Minit, e.Lname
FROM employee e
JOIN department d ON e.Ssn = d.Mgr_ssn
WHERE EXISTS (
SELECT 1
FROM dependent dep
WHERE dep.Essn = e.Ssn
);
```



g) Retrieve the names of employees who have no dependents.

```
SELECT e.Fname, e.Minit, e.Lname
FROM employee e
LEFT JOIN dependent d ON e.Ssn = d.Essn
WHERE d.Essn IS NULL;
```



Lab number 7

What is the view? And explain the advantages of view. Write command to create view of employees working in IT department.

Theory:

A **view** in SQL is a virtual table that is derived from a query applied to one or more base tables. Unlike a physical table, a view does not store data itself but provides a way to present data stored in the database in a specific format. Views are primarily used for simplifying complex queries, ensuring data security, and abstracting the underlying database structure from the end users.

Key Characteristics of a View:

- **Virtual Table:** A view behaves like a table when queried, but it does not store data itself. It pulls data from other tables based on the SQL query that defines the view.
- **Dynamic Data:** Since a view reflects the result of a query, any changes in the underlying tables are immediately visible when the view is queried.
- **Filtered Data Presentation:** Views can filter and present only specific rows and columns from the underlying tables, based on conditions specified in the view's defining query.

Advantages of Using Views in SQL

Views offer several advantages in database management, enhancing both the functionality and security of a database. Below are some of the key benefits:

1. **Simplification of Complex Queries:**

- **Ease of Use:** Views allow users to encapsulate complex SQL queries into a simple view. This means that instead of writing and understanding a complex query each time, users can query the view as if it were a simple table.
- **Reusability:** Once a view is created, it can be reused across different queries and by different users, saving time and reducing the likelihood of errors.

2. **Enhanced Security:**

- **Data Access Control:** Views can be used to restrict access to specific rows or columns of a table. For example, sensitive information such as salaries can be excluded from the view, allowing users to access only the non-sensitive data.
- **Access Permissions:** Permissions can be granted on views rather than on the underlying base tables. This allows administrators to control who can access specific data without exposing the entire table.

3. **Data Abstraction:**

- **Schema Simplification:** Views provide an abstraction layer over the underlying table schema. Users do not need to know the complexity of the database schema; they can interact with a simplified version of the data.
- **Data Representation:** Views can present data in a different format or structure than the underlying tables. For example, a view can aggregate data, combine columns, or even join multiple tables to present a unified dataset.

4. **Consistency and Integrity:**

- **Consistency in Data Representation:** Since views are derived from base tables, any changes in the underlying data are automatically reflected in the view. This ensures that users always see the most current and consistent data.
- **Centralized Logic:** Business logic, such as calculations or data transformations, can be centralized in a view. This ensures that the logic is applied consistently across all applications that use the view.

5. **Data Aggregation and Reporting:**

- **Predefined Queries:** Views can be used to create predefined queries for reporting purposes. For example, a view can be created to summarize sales data by month, providing a ready-to-use dataset for reporting.
- **Performance Optimization:** In some cases, views can improve query performance by simplifying the query execution plan. This is particularly true for materialized views, which store the query results and can be refreshed periodically.

1. **Independence from Physical Data Structure:**

- **Schema Evolution:** Views provide a level of independence from the physical structure of the database. If the schema of the underlying tables changes, the view can often be updated to accommodate the changes without requiring changes to the queries that use the view.
- **Backward Compatibility:** Views can be used to maintain backward compatibility with older applications when the underlying database schema changes. By creating views that mimic the old table structure, applications can continue to function without modification.

Procedure to create view of employees working in IT department.

1) Create the Employee Table: Ensure that the database contains an Employee table with relevant fields.

```
CREATE TABLE Employee (
   emp_id INTEGER PRIMARY KEY,
   emp_name VARCHAR(255),
   department VARCHAR(255)
);
```

Field Types

#	Field	Schema	Table	Type
1	emp_id	kiran	employee	INT
2	emp_name	kiran	employee	VARCHAR
3	department	kiran	employee	VARCHAR

2) Insert Sample Data: Populate the Employee table with sample data, including employees from various departments

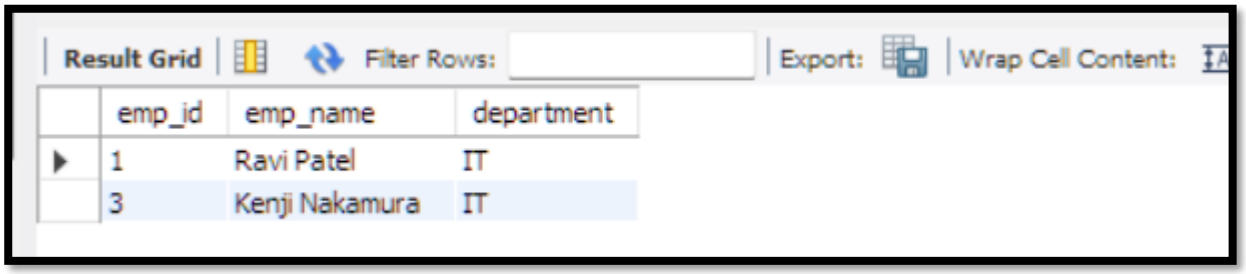
```
INSERT INTO Employee (emp_id, emp_name, department)
VALUES
(1, 'Ravi Patel', 'IT'),
(2, 'Neha Singh', 'HR'),
(3, 'Kenji Nakamura', 'IT'),
(4, 'Maya Rathi', 'Finance');.
```

Result Grid | Filter Rows: | Edit:

emp_id	emp_name	department
1	Ravi Patel	IT
2	Neha Singh	HR
3	Kenji Nakamura	IT
4	Maya Rathi	Finance
NULL	NULL	NULL

3)Create and Query the View: Create a view to filter and display only the employees working in the IT department and Retrieve data from the IT_Employees view to verify that it correctly displays only IT department employees.

```
CREATE VIEW IT_Employees AS
SELECT emp_id, emp_name, department
FROM Employee
WHERE department = 'IT';
SELECT * FROM IT_Employees;
```



emp_id	emp_name	department
1	Ravi Patel	IT
3	Kenji Nakamura	IT

Conclusion

This lab demonstrated how to create and use SQL views in a relational database. The IT_Employees view provided a simplified and secure way to access data related to employees in the IT department. Views are a powerful tool in database management, offering benefits such as simplification of complex queries, data security, and reusability. By using views, database administrators and users can efficiently manage and retrieve data, ensuring consistency and security in the process.