Project Work Of Database Management System

Tribhuvan University

Amrit Science Campus

Thamel, Kathmandu

Submitted By: Bishnu Chalise

Faculty: CSIT

Roll No.: 34

Section: A

Submitted To: Er. Yograj Joshi

Internal Examiner:

Signature: ________________

External Examiner

Signature: ______________

Name: Bishnu Chalise

Roll No: 34

Index

S.N	Questions	Date	Signature
1.	Lab No.1	2081/05/20	
2.	Lab No.2	2081/05/20	
3.	Lab No.3	2081/05/21	
4.	Lab No.4	2081/05/22	
5.	Lab No.5	2081/05/22	
6.	Lab No.6	2081/05/23	
7.	Lab No.7	2081/05/24	

[redacted]

Database

A **database** is an organized collecon of structured informaon, or data, typically stored electronically in a computer system. Databases are used to store, retrieve, and manage data efficiently.

DBMS (Database Management System)

A **Database Management System (DBMS)** is soware that interacts with the database, applicaons, and users to capture and analyze data. The DBMS provides a systemac way to create, retrieve, update, and manage data.

Constraints

Constraints are rules enforced on data columns in a database table to ensure the integrity, accuracy, and reliability of the data. They restrict the type of data that can be stored in a table.

Types of Constraints

- **NOT NULL**: Ensures that a column cannot have a NULL value.
- **UNIQUE**: Ensures all values in a column are unique.
- **PRIMARY KEY**: Uniquely identifies each row in a table.
- **FOREIGN KEY**: Ensures referential integrity between two tables.
- **CHECK**: Ensures that the values in a column satisfy a specific condition.
- **DEFAULT**: Sets a default value for a column if no value is specified.

Primary Key

A **Primary Key** is a column or a set of columns in a table that uniquely idenfies each row in that table. It ensures that each record in the table is unique and that no primary key column can contain NULL values.

Key Characteristics

- Uniqueness: Each value must be unique across the table.
- Non-NULL: The primary key column(s) cannot have NULL values.
- Stability: The value of the primary key should not change over time.

Example

```
CREATE TABLE employees(
    employee_id INT NOT NULL,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    PRIMARY KEY (employee_id)
);
```

Foreign Key

A Foreign Key is a column or a set of columns in one table that refers to the primary key in another table. It is used to establish a link between the data in the two tables and ensures referential integrity.

Characteristic

- Referential Integrity: The foreign key ensures that the value in the child table exists in the parent table.

- Relaonship: Establishes a relaonship between two tables (one-to-many, many-to-one, or many-to-many).
- Cascading Acons: Can be configured to perform acons like CASCADE, SET NULL, or RESTRICT when the referenced data is updated or deleted.

Example

```
CREATE TABLE departments (
    department_id INT NOT NULL,
    department_name VARCHAR(50) NOT NULL,
    PRIMARY KEY (department_id)
);

CREATE TABLE employees (
    employee_id INT NOT NULL,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    department_id INT,
    PRIMARY KEY (employee_id),
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

SQL Commands and Snippets

Create Database

```
CREATE DATABASE database_name;
```

Delete Database

```
DROP DATABASE database_name;
```

Create Table

Create command creates a new database with the specified name.

```
CREATE TABLE table_name (
    column1 datatype constraints,
    column2 datatype constraints,
    ...
);
```

Delete Table

Delete command deletes the specified database and all its associated data and objects.

```
DROP TABLE table_name;
```

Alter Table

Alter command is used to modify the structure of an existing table in a database. It allows you to:

- Add a new column to the table with a specified name, data type, and constraints.
- Change the data type, size, or constraints of an existing column.
- Remove an existing column from the table.

```
ALTER TABLE table_name ADD column_name datatype constraints;
ALTER TABLE table_name MODIFY COLUMN column_name datatype constraints;
ALTER TABLE table_name DROP COLUMN column_name;
```

View Schema

Provides detailed informaon about the columns in a specified table, including data types and constraints.

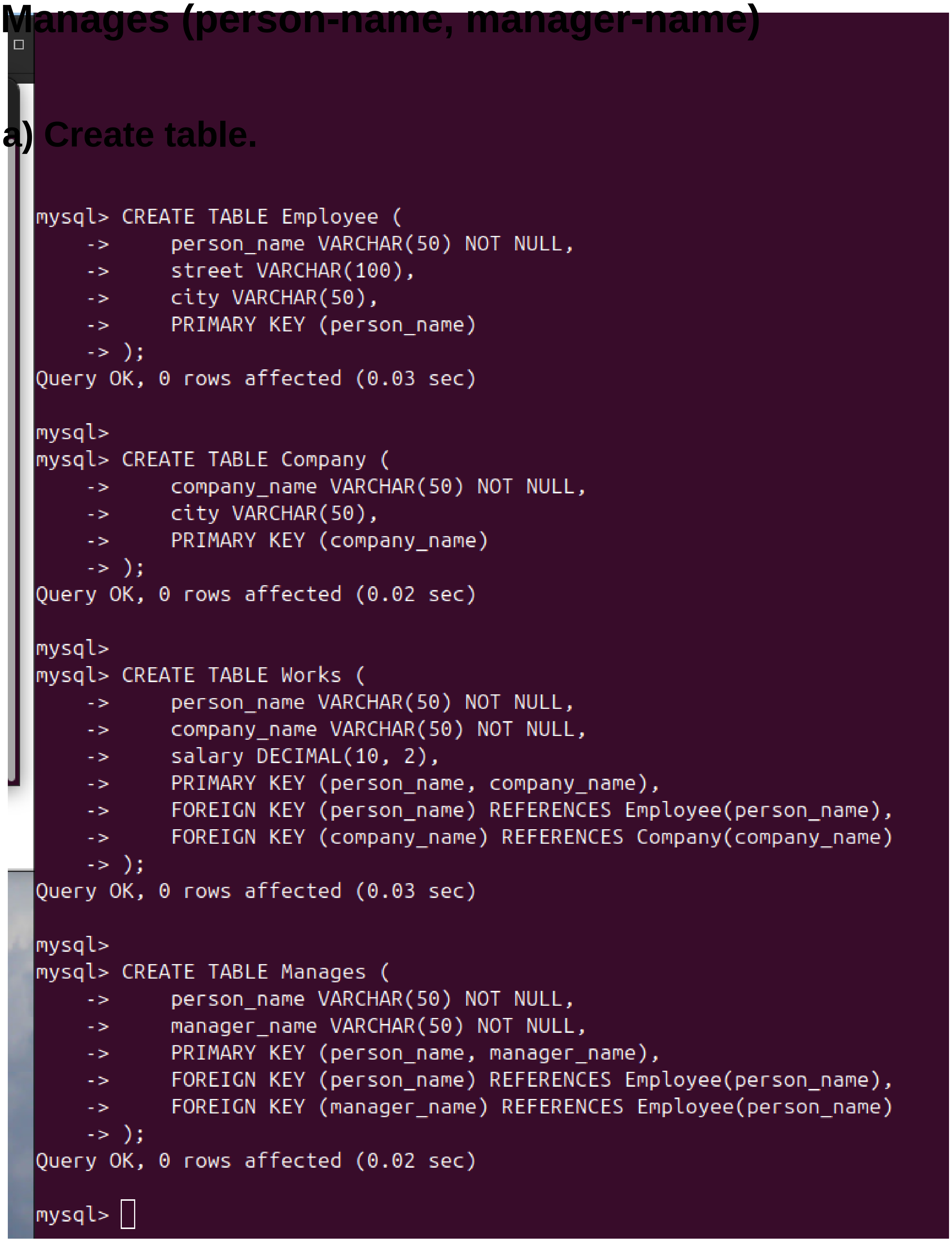
```
DESCRIBE table_name;
```

Insert Data

Inserts new rows into a table with specified column values.

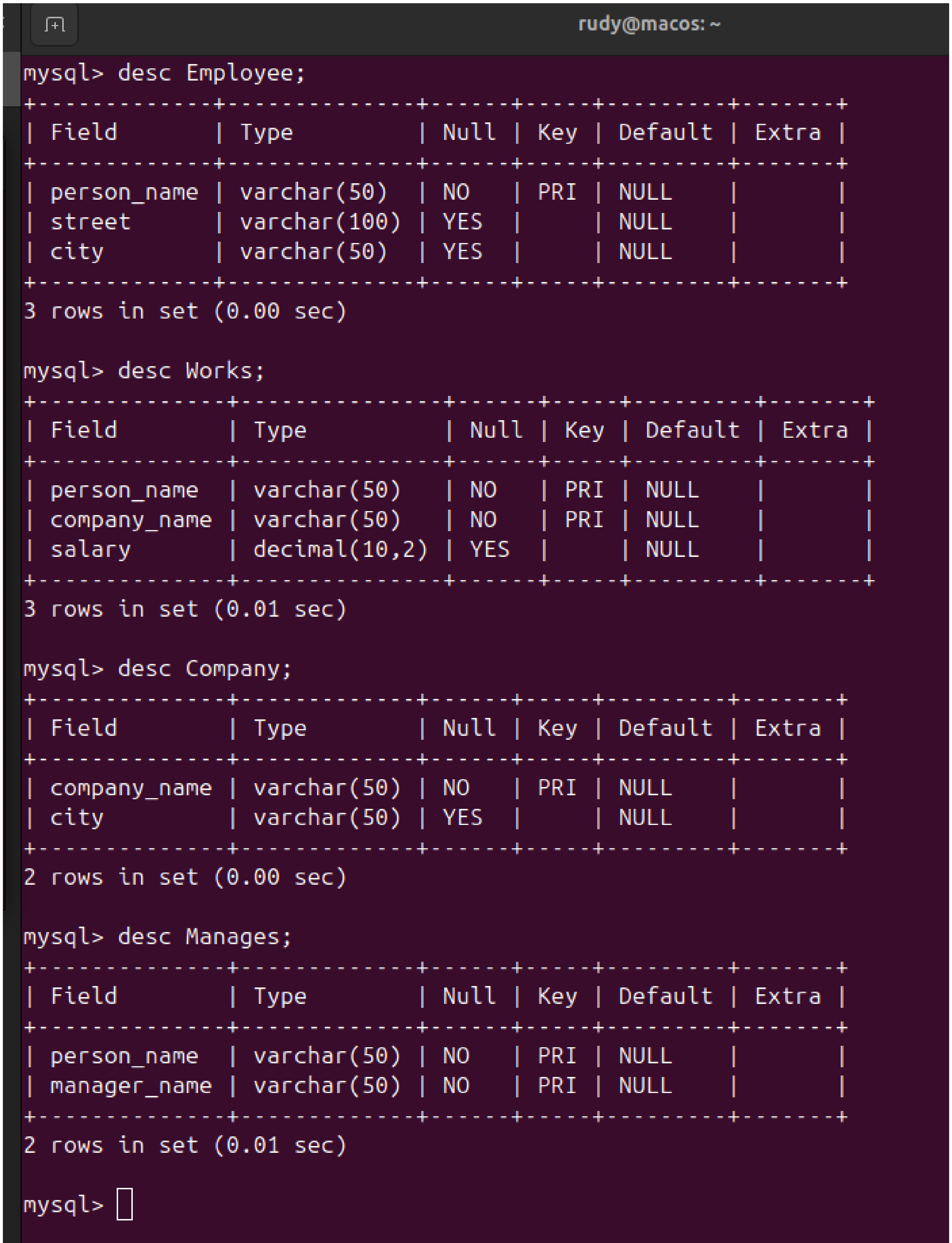
```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

1. Consider the following relational database, where the primary keys are underlined.
Employee (person-name, street, city)
Works (person-name, company-name, salary)
Company (company-name, city)
Manages (person-name, manager-name)



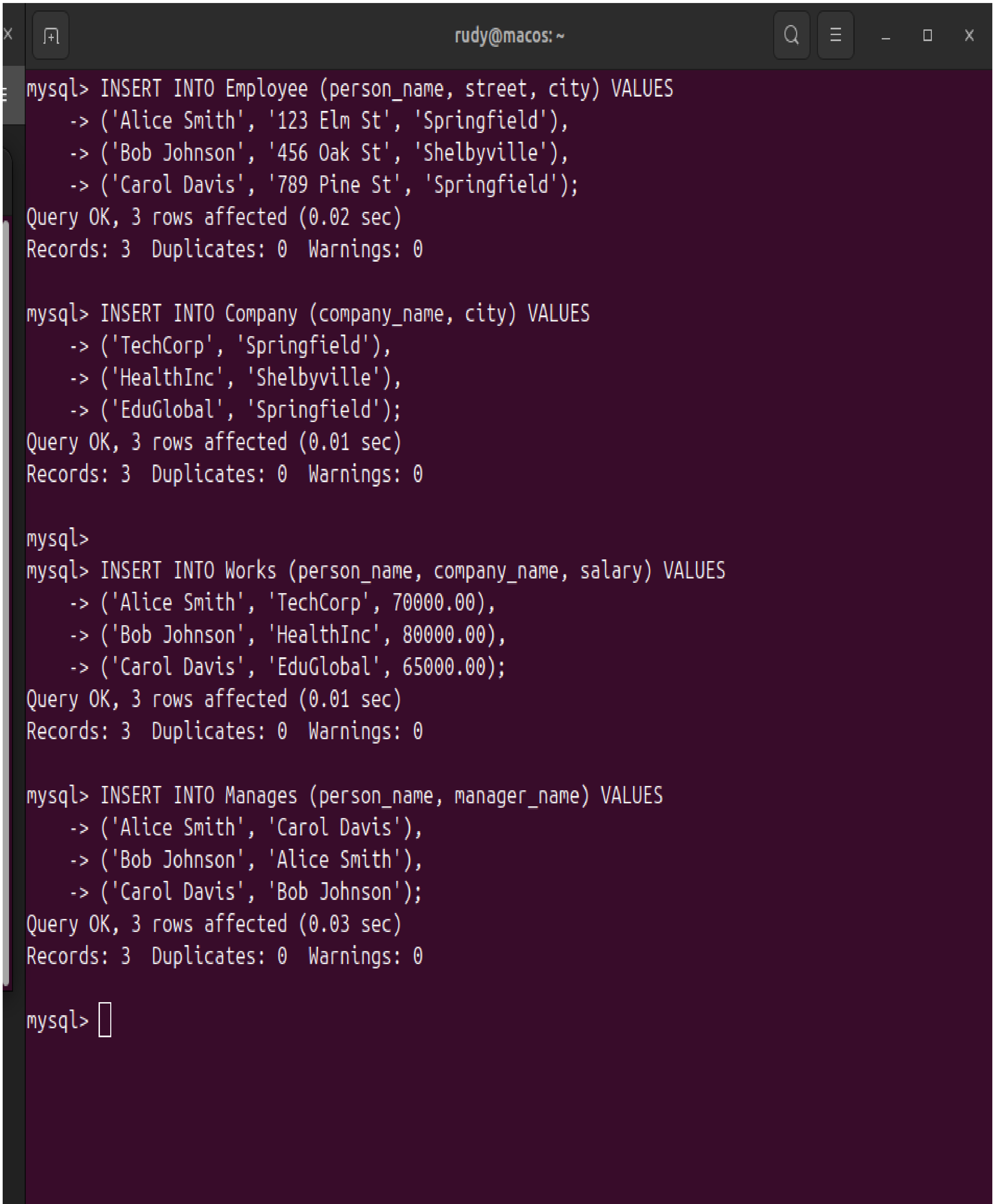
a) Create table.

```
mysql> CREATE TABLE Employee (
    ->     person_name VARCHAR(50) NOT NULL,
    ->     street VARCHAR(100),
    ->     city VARCHAR(50),
    ->     PRIMARY KEY (person_name)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> CREATE TABLE Company (
    ->     company_name VARCHAR(50) NOT NULL,
    ->     city VARCHAR(50),
    ->     PRIMARY KEY (company_name)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE Works (
    ->     person_name VARCHAR(50) NOT NULL,
    ->     company_name VARCHAR(50) NOT NULL,
    ->     salary DECIMAL(10, 2),
    ->     PRIMARY KEY (person_name, company_name),
    ->     FOREIGN KEY (person_name) REFERENCES Employee(person_name),
    ->     FOREIGN KEY (company_name) REFERENCES Company(company_name)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> CREATE TABLE Manages (
    ->     person_name VARCHAR(50) NOT NULL,
    ->     manager_name VARCHAR(50) NOT NULL,
    ->     PRIMARY KEY (person_name, manager_name),
    ->     FOREIGN KEY (person_name) REFERENCES Employee(person_name),
    ->     FOREIGN KEY (manager_name) REFERENCES Employee(person_name)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> 
```

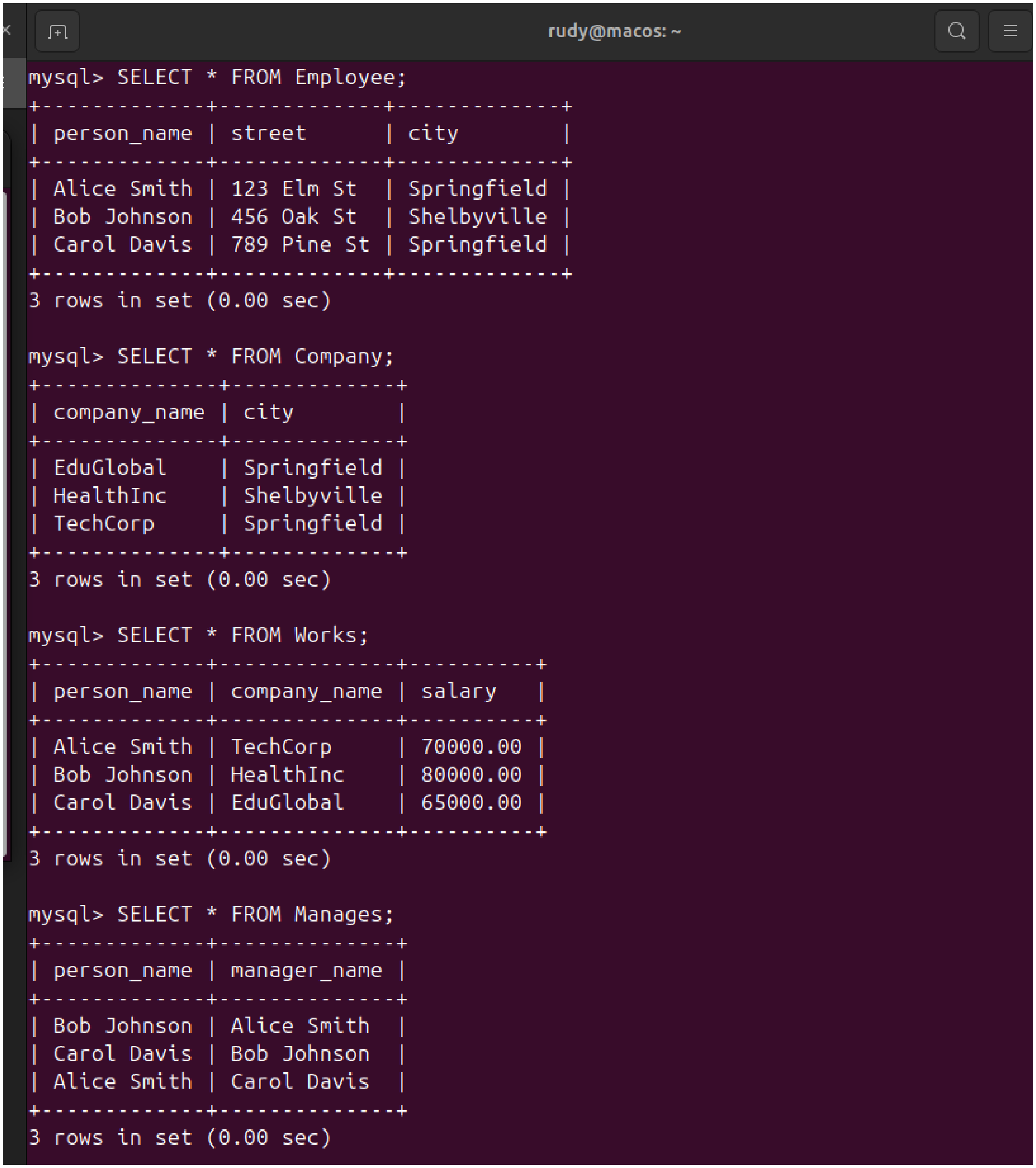

rudy@macos: ~

```
mysql> desc Employee;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| person_name | varchar(50)  | NO   | PRI | NULL    |       |
| street      | varchar(100) | YES  |     | NULL    |       |
| city        | varchar(50)  | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> desc Works;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| person_name  | varchar(50)   | NO   | PRI | NULL    |       |
| company_name | varchar(50)   | NO   | PRI | NULL    |       |
| salary       | decimal(10,2) | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> desc Company;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| company_name | varchar(50) | NO   | PRI | NULL    |       |
| city         | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> desc Manages;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| person_name  | varchar(50) | NO   | PRI | NULL    |       |
| manager_name | varchar(50) | NO   | PRI | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> 
```

b) Insert any three records in each above table and display them.



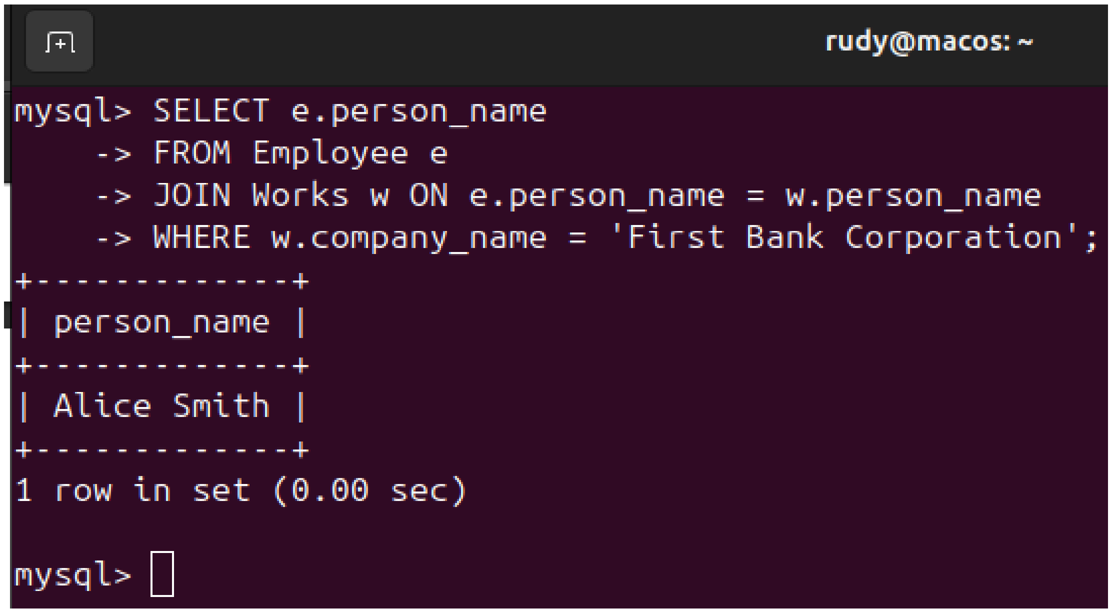
```
mysql> INSERT INTO Employee (person_name, street, city) VALUES
    -> ('Alice Smith', '123 Elm St', 'Springfield'),
    -> ('Bob Johnson', '456 Oak St', 'Shelbyville'),
    -> ('Carol Davis', '789 Pine St', 'Springfield');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Company (company_name, city) VALUES
    -> ('TechCorp', 'Springfield'),
    -> ('HealthInc', 'Shelbyville'),
    -> ('EduGlobal', 'Springfield');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO Works (person_name, company_name, salary) VALUES
    -> ('Alice Smith', 'TechCorp', 70000.00),
    -> ('Bob Johnson', 'HealthInc', 80000.00),
    -> ('Carol Davis', 'EduGlobal', 65000.00);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Manages (person_name, manager_name) VALUES
    -> ('Alice Smith', 'Carol Davis'),
    -> ('Bob Johnson', 'Alice Smith'),
    -> ('Carol Davis', 'Bob Johnson');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> 
```



```
mysql> SELECT * FROM Employee;
+-------------+-------------+-------------+
| person_name | street      | city        |
+-------------+-------------+-------------+
| Alice Smith | 123 Elm St  | Springfield |
| Bob Johnson | 456 Oak St  | Shelbyville |
| Carol Davis | 789 Pine St | Springfield |
+-------------+-------------+-------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM Company;
+--------------+-------------+
| company_name | city        |
+--------------+-------------+
| EduGlobal    | Springfield |
| HealthInc    | Shelbyville |
| TechCorp     | Springfield |
+--------------+-------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM Works;
+-------------+--------------+----------+
| person_name | company_name | salary   |
+-------------+--------------+----------+
| Alice Smith | TechCorp     | 70000.00 |
| Bob Johnson | HealthInc    | 80000.00 |
| Carol Davis | EduGlobal    | 65000.00 |
+-------------+--------------+----------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM Manages;
+-------------+--------------+
| person_name | manager_name |
+-------------+--------------+
| Bob Johnson | Alice Smith  |
| Carol Davis | Bob Johnson  |
| Alice Smith | Carol Davis  |
+-------------+--------------+
3 rows in set (0.00 sec)
```

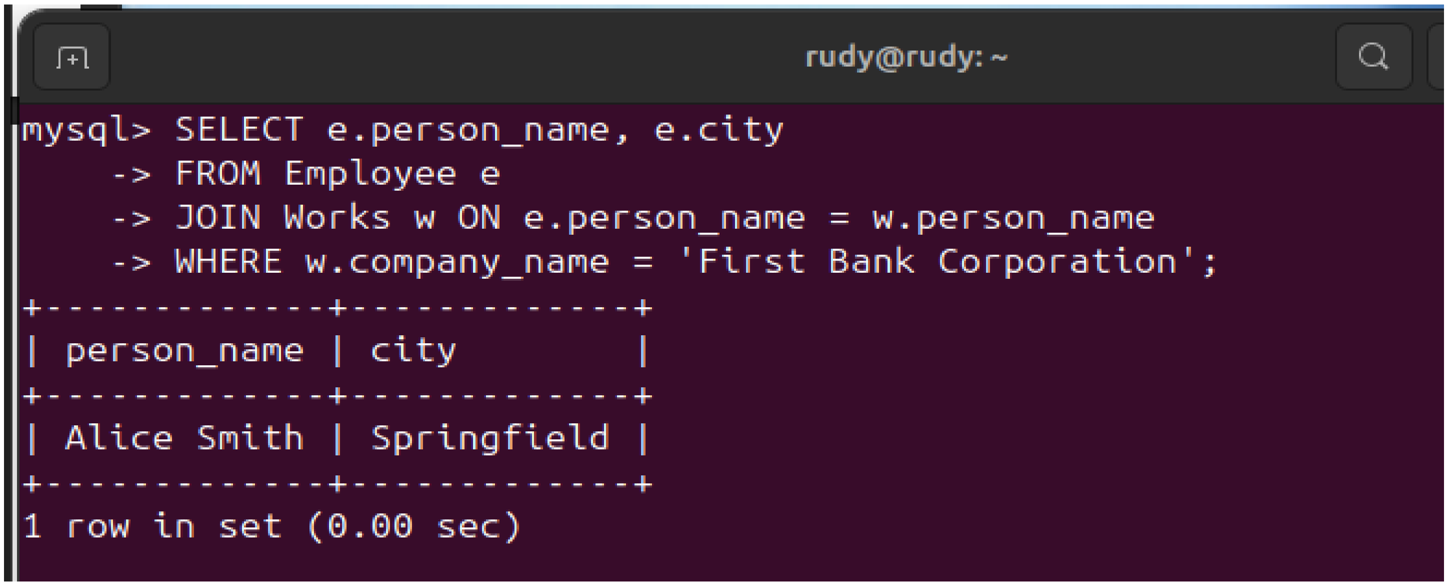
C.

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



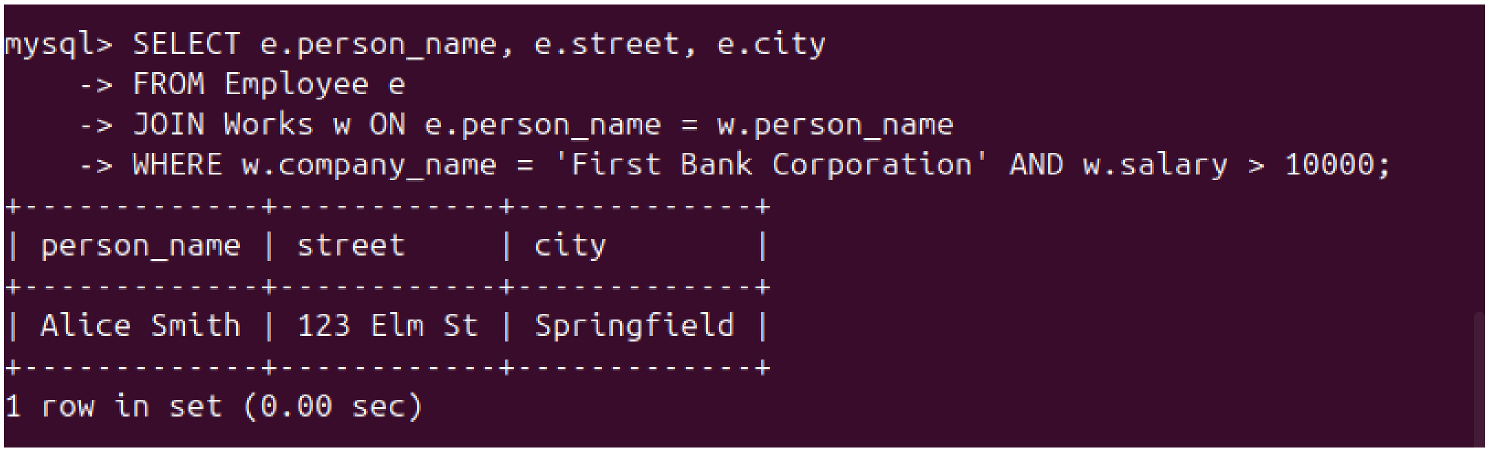
```
mysql> SELECT e.person_name
    -> FROM Employee e
    -> JOIN Works w ON e.person_name = w.person_name
    -> WHERE w.company_name = 'First Bank Corporation';
+-------------+
| person_name |
+-------------+
| Alice Smith |
+-------------+
1 row in set (0.00 sec)

mysql> 
```

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



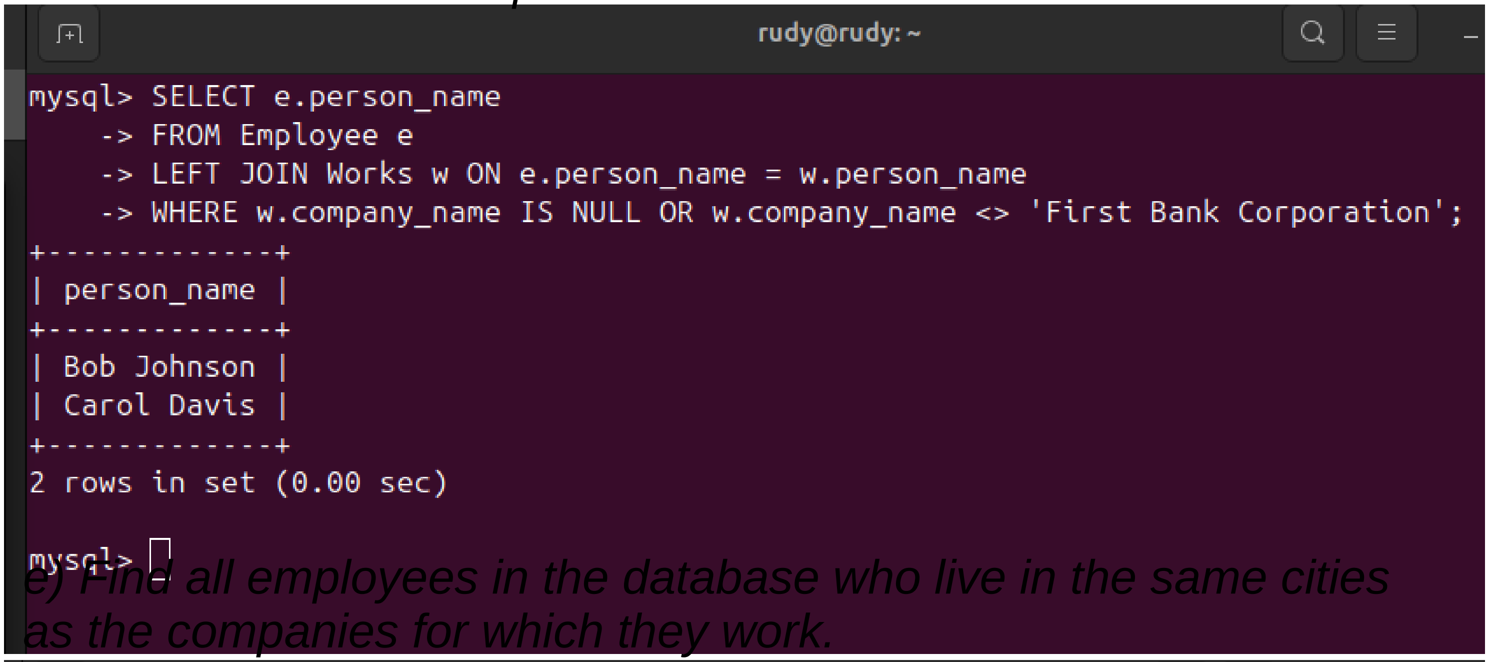
```
mysql> SELECT e.person_name, e.city
    -> FROM Employee e
    -> JOIN Works w ON e.person_name = w.person_name
    -> WHERE w.company_name = 'First Bank Corporation';
+-------------+-------------+
| person_name | city        |
+-------------+-------------+
| Alice Smith | Springfield |
+-------------+-------------+
1 row in set (0.00 sec)
```

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.



```
mysql> 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        |
+-------------+------------+-------------+
| Alice Smith | 123 Elm St | Springfield |
+-------------+------------+-------------+
1 row in set (0.00 sec)
```

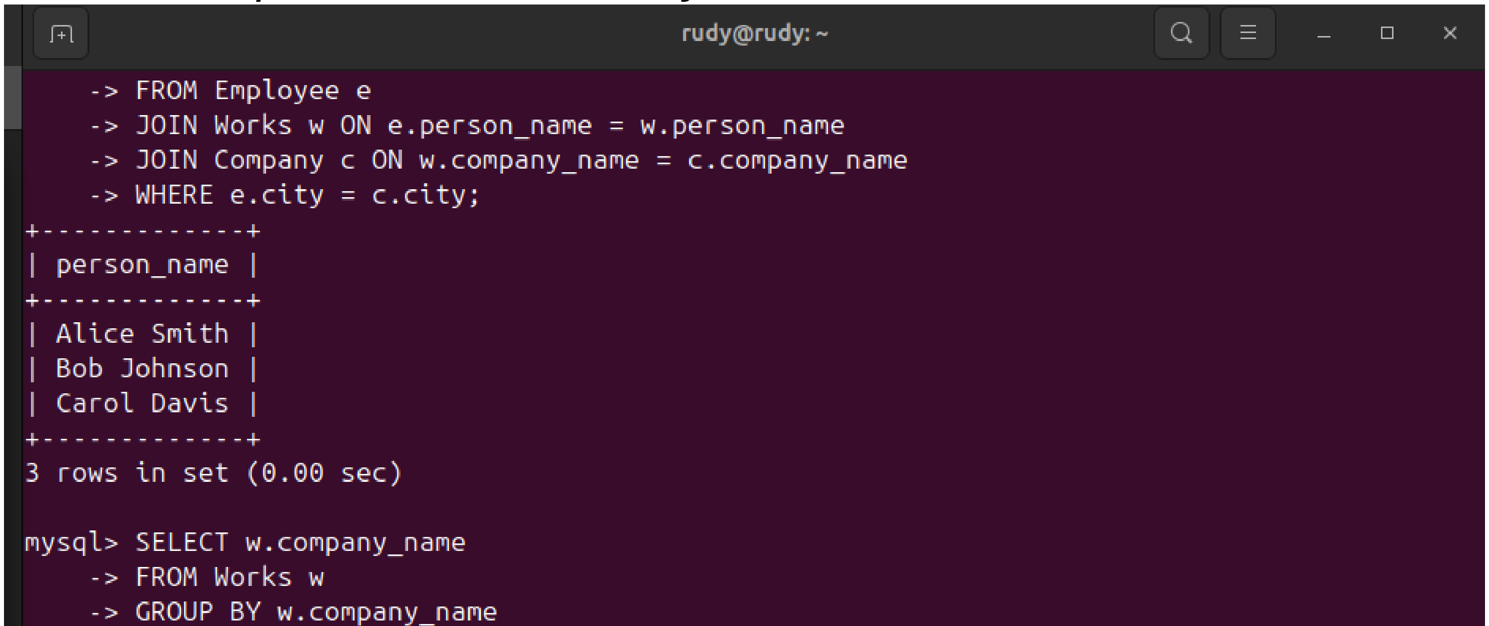
d) Find the names of all employees in this database who do not work for First Bank Corporation.



rudy@rudy: ~

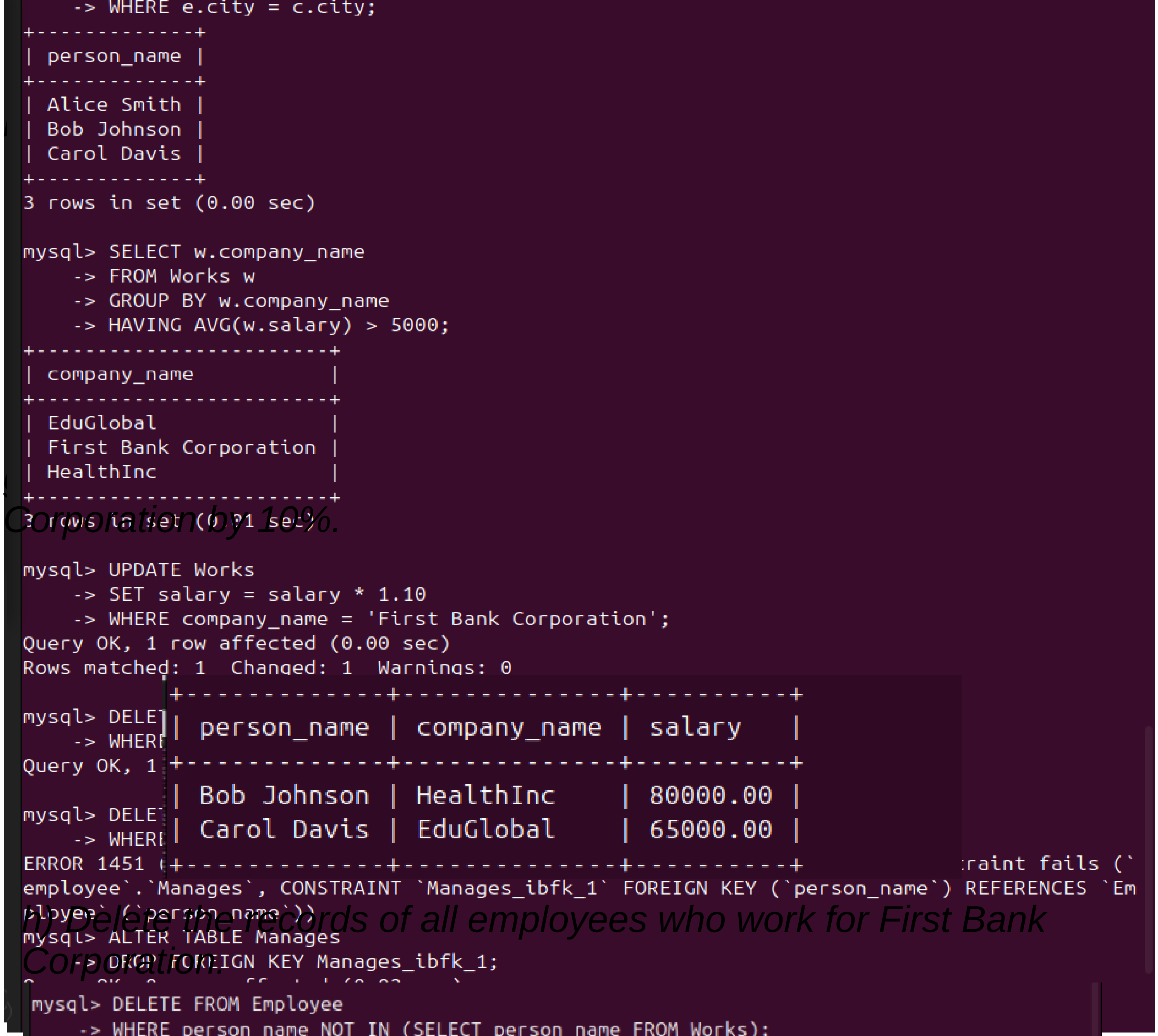
```
mysql> SELECT e.person_name
    -> FROM Employee e
    -> LEFT JOIN Works w ON e.person_name = w.person_name
    -> WHERE w.company_name IS NULL OR w.company_name <> 'First Bank Corporation';
+-------------+
| person_name |
+-------------+
| Bob Johnson |
| Carol Davis |
+-------------+
2 rows in set (0.00 sec)

mysql> 
```

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



rudy@rudy: ~

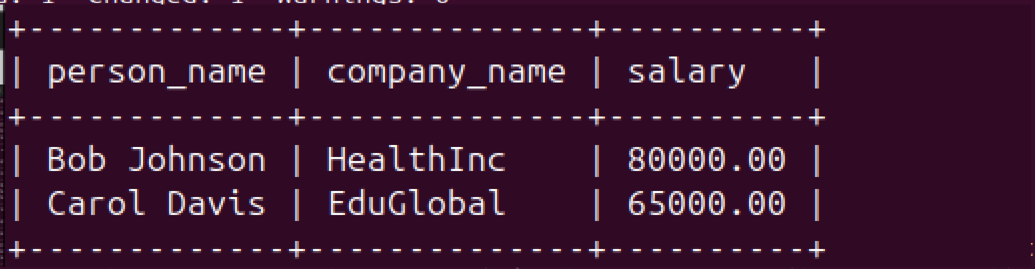
```
    -> FROM Employee e
    -> JOIN Works w ON e.person_name = w.person_name
    -> JOIN Company c ON w.company_name = c.company_name
    -> WHERE e.city = c.city;
+-------------+
| person_name |
+-------------+
| Alice Smith |
| Bob Johnson |
| Carol Davis |
+-------------+
3 rows in set (0.00 sec)

mysql> SELECT w.company_name
    -> FROM Works w
    -> GROUP BY w.company_name
```



```
    -> WHERE e.city = c.city;
+-------------+
| person_name |
+-------------+
| Alice Smith |
| Bob Johnson |
| Carol Davis |
+-------------+
3 rows in set (0.00 sec)

mysql> SELECT w.company_name
    -> FROM Works w
    -> GROUP BY w.company_name
    -> HAVING AVG(w.salary) > 5000;
+------------------------+
| company_name           |
+------------------------+
| EduGlobal              |
| First Bank Corporation |
| HealthInc              |
+------------------------+
3 rows in set (0.01 sec)
```

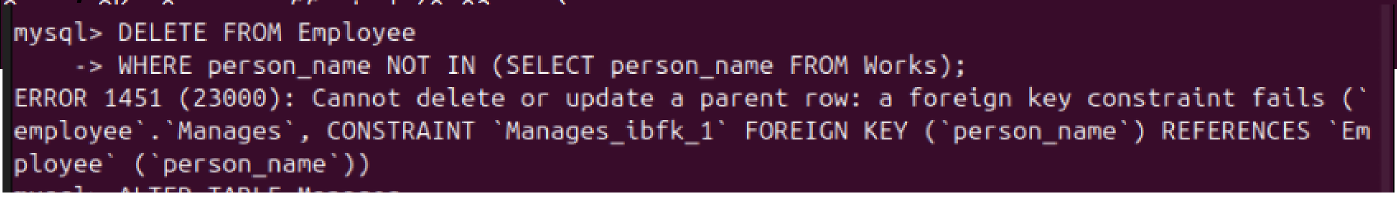
Corporation by 10%.

```
mysql> UPDATE Works
    -> SET salary = salary * 1.10
    -> WHERE company_name = 'First Bank Corporation';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> DELET
    -> WHERE
Query OK, 1

mysql> DELET
    -> WHERE
ERROR 1451 (                                                    traint fails (`
employee`.`Manages`, CONSTRAINT `Manages_ibfk_1` FOREIGN KEY (`person_name`) REFERENCES `Em
ployee` (`person_name`))
mysql> ALTER TABLE Manages
    -> DROP FOREIGN KEY Manages_ibfk_1;
```

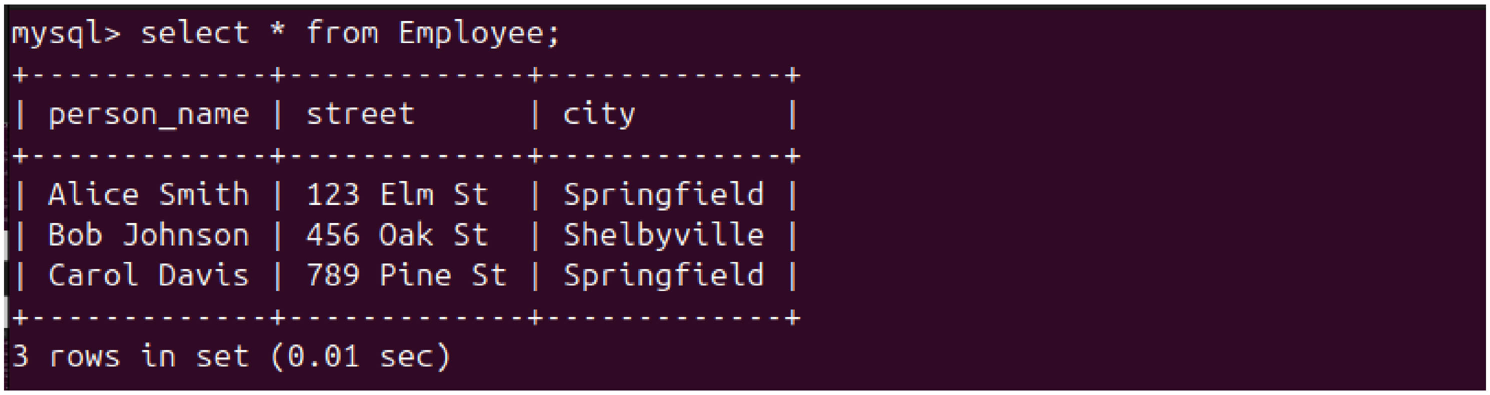


```
+-------------+--------------+----------+
| person_name | company_name | salary   |
+-------------+--------------+----------+
| Bob Johnson | HealthInc    | 80000.00 |
| Carol Davis | EduGlobal    | 65000.00 |
+-------------+--------------+----------+
```

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

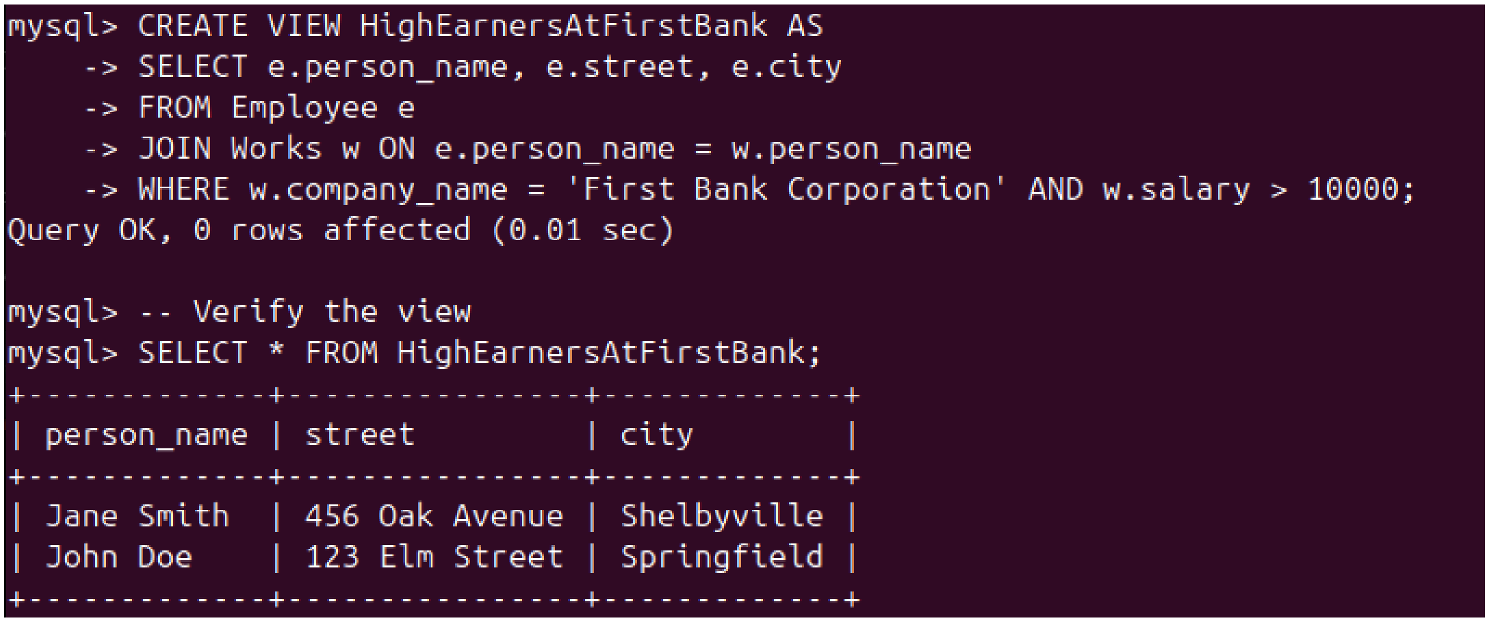


```
mysql> DELETE FROM Employee
    -> WHERE person_name NOT IN (SELECT person_name FROM Works);
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`
employee`.`Manages`, CONSTRAINT `Manages_ibfk_1` FOREIGN KEY (`person_name`) REFERENCES `Em
ployee` (`person_name`))
```



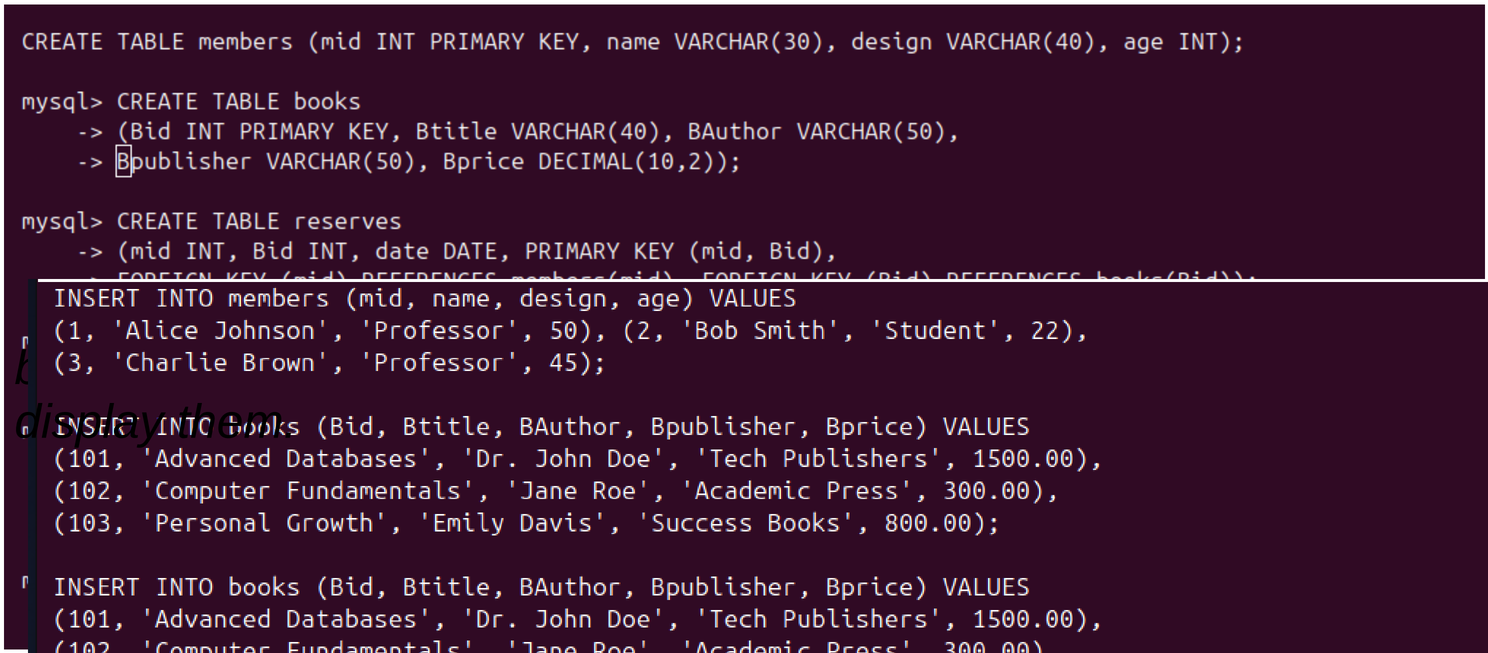
```
mysql> select * from Employee;
+-------------+------------+-------------+
| person_name | street     | city        |
+-------------+------------+-------------+
| Alice Smith | 123 Elm St | Springfield |
| Bob Johnson | 456 Oak St | Shelbyville |
| Carol Davis | 789 Pine St | Springfield |
+-------------+------------+-------------+
3 rows in set (0.01 sec)
```

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.



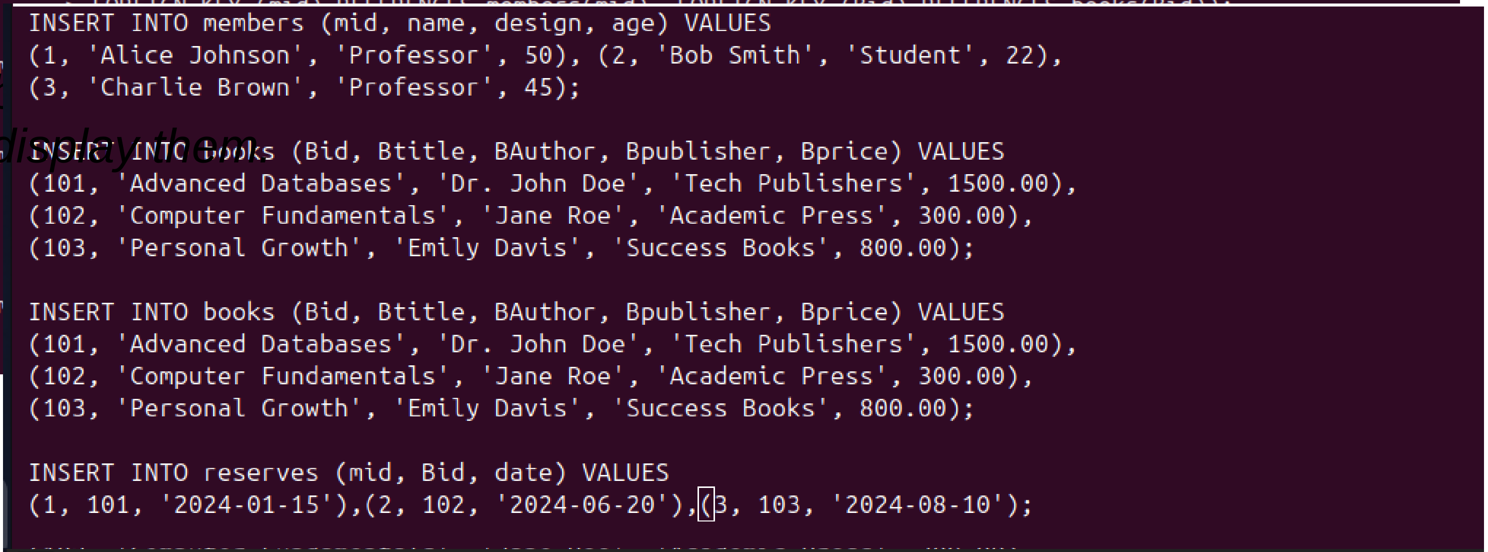
```
mysql> CREATE VIEW HighEarnersAtFirstBank AS
    -> SELECT e.person_name, e.street, e.city
    -> FROM Employee e
    -> JOIN Works w ON e.person_name = w.person_name
    -> WHERE w.company_name = 'First Bank Corporation' AND w.salary > 10000;
Query OK, 0 rows affected (0.01 sec)

mysql> -- Verify the view
mysql> SELECT * FROM HighEarnersAtFirstBank;
+-------------+---------------+-------------+
| person_name | street        | city        |
+-------------+---------------+-------------+
| Jane Smith  | 456 Oak Avenue | Shelbyville |
| John Doe    | 123 Elm Street | Springfield |
+-------------+---------------+-------------+
```

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.*



```
CREATE TABLE members (mid INT PRIMARY KEY, name VARCHAR(30), design VARCHAR(40), age INT);

mysql> CREATE TABLE books
    -> (Bid INT PRIMARY KEY, Btitle VARCHAR(40), BAuthor VARCHAR(50),
    -> Bpublisher VARCHAR(50), Bprice DECIMAL(10,2));

mysql> 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));
```

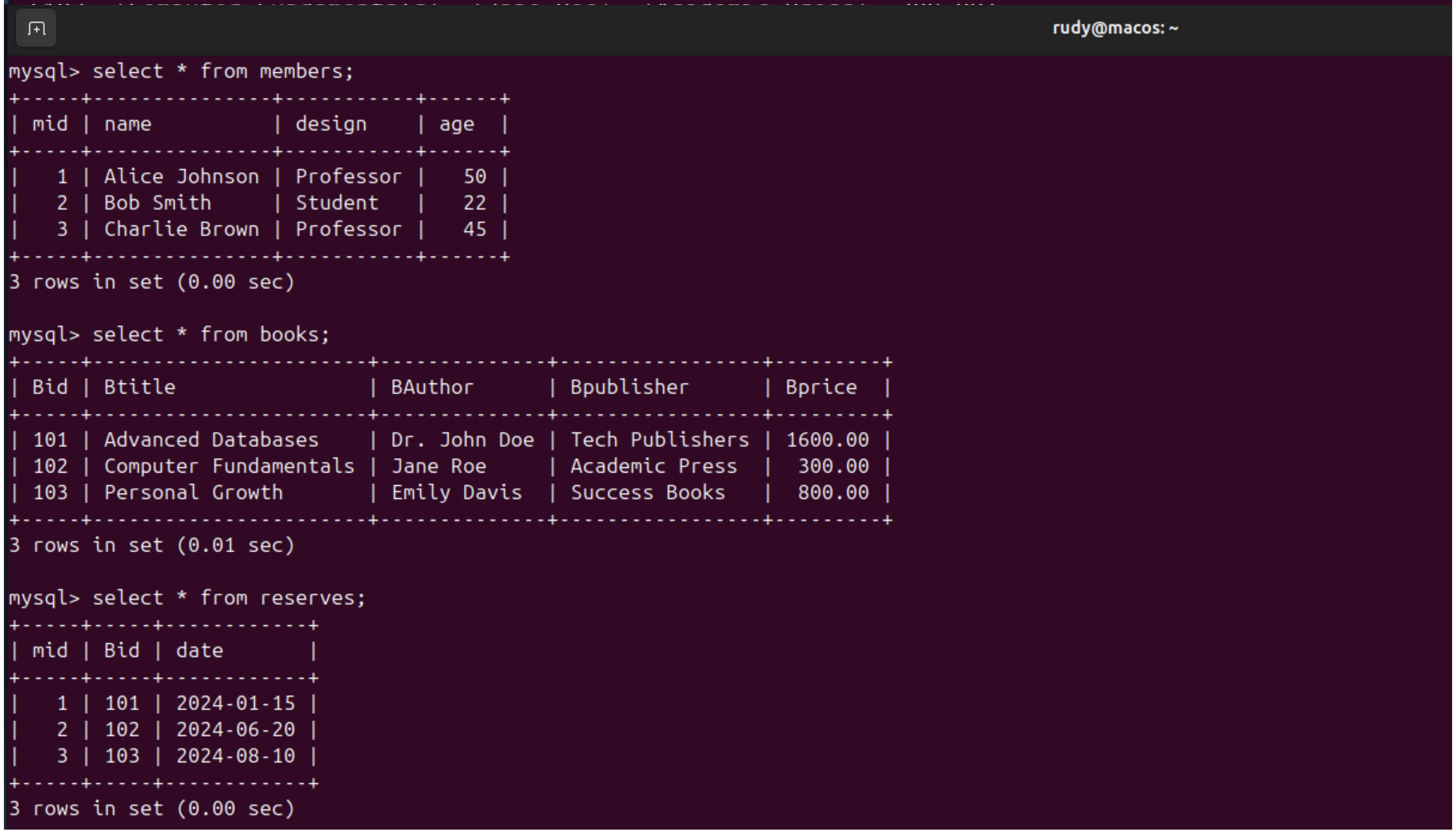
display them



```
INSERT INTO members (mid, name, design, age) VALUES
(1, 'Alice Johnson', 'Professor', 50), (2, 'Bob Smith', 'Student', 22),
(3, 'Charlie Brown', 'Professor', 45);

INSERT INTO books (Bid, Btitle, BAuthor, Bpublisher, Bprice) VALUES
(101, 'Advanced Databases', 'Dr. John Doe', 'Tech Publishers', 1500.00),
(102, 'Computer Fundamentals', 'Jane Roe', 'Academic Press', 300.00),
(103, 'Personal Growth', 'Emily Davis', 'Success Books', 800.00);
```

```
INSERT INTO books (Bid, Btitle, BAuthor, Bpublisher, Bprice) VALUES
(101, 'Advanced Databases', 'Dr. John Doe', 'Tech Publishers', 1500.00),
(102, 'Computer Fundamentals', 'Jane Roe', 'Academic Press', 300.00),
(103, 'Personal Growth', 'Emily Davis', 'Success Books', 800.00);
```

```
INSERT INTO reserves (mid, Bid, date) VALUES
(1, 101, '2024-01-15'),(2, 102, '2024-06-20'),(3, 103, '2024-08-10');
```



```
mysql> select * from members;
+-----+---------------+-----------+------+
| mid | name          | design    | age  |
+-----+---------------+-----------+------+
|   1 | Alice Johnson | Professor |   50 |
|   2 | Bob Smith     | Student   |   22 |
|   3 | Charlie Brown | Professor |   45 |
+-----+---------------+-----------+------+
3 rows in set (0.00 sec)

mysql> select * from books;
+-----+-----------------------+--------------+-----------------+---------+
| Bid | Btitle                | BAuthor      | Bpublisher      | Bprice  |
+-----+-----------------------+--------------+-----------------+---------+
| 101 | Advanced Databases    | Dr. John Doe | Tech Publishers | 1600.00 |
| 102 | Computer Fundamentals | Jane Roe     | Academic Press  |  300.00 |
| 103 | Personal Growth       | Emily Davis  | Success Books   |  800.00 |
+-----+-----------------------+--------------+-----------------+---------+
3 rows in set (0.01 sec)

mysql> select * from reserves;
+-----+-----+------------+
| mid | Bid | date       |
+-----+-----+------------+
|   1 | 101 | 2024-01-15 |
|   2 | 102 | 2024-06-20 |
|   3 | 103 | 2024-08-10 |
+-----+-----+------------+
3 rows in set (0.00 sec)
```

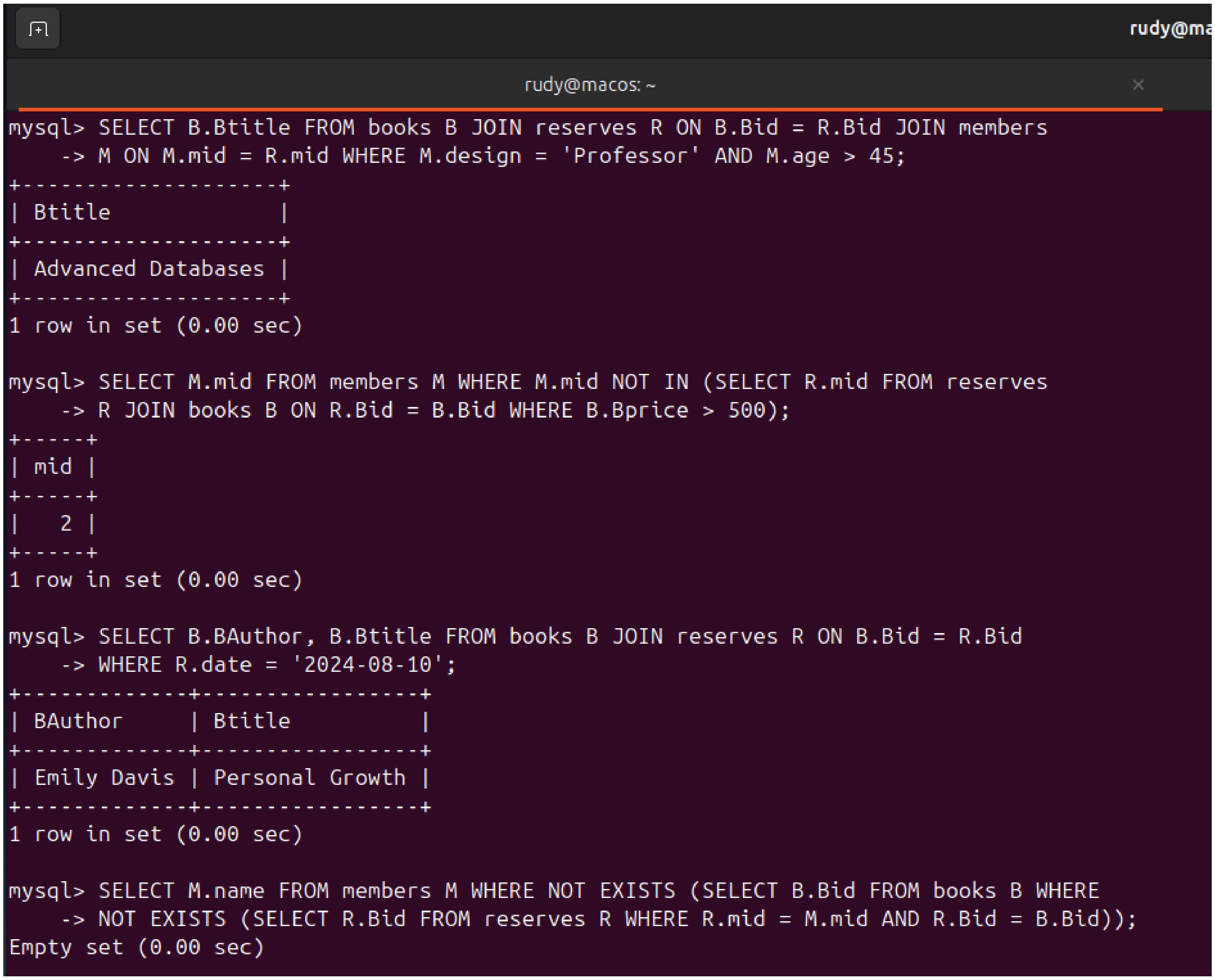
C.

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

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

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

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

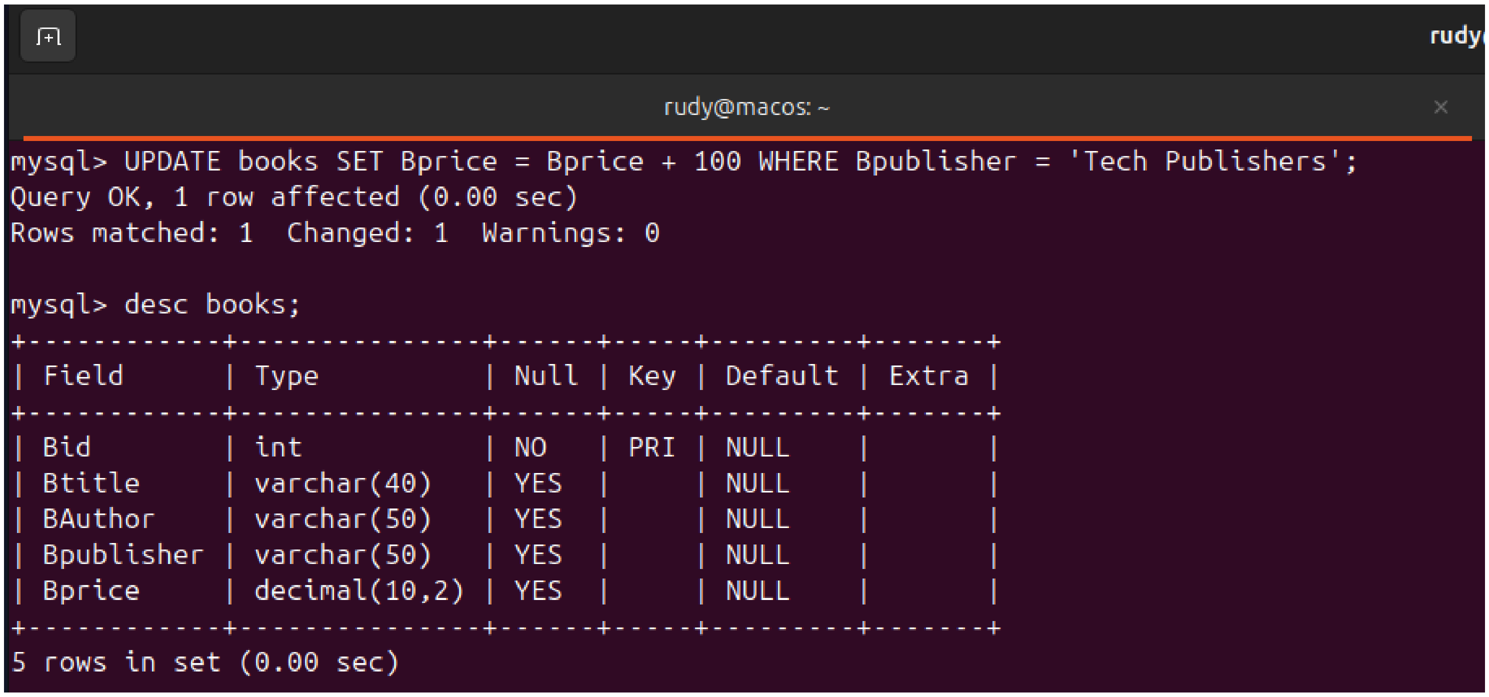


rudy@ma

rudy@macos: ~

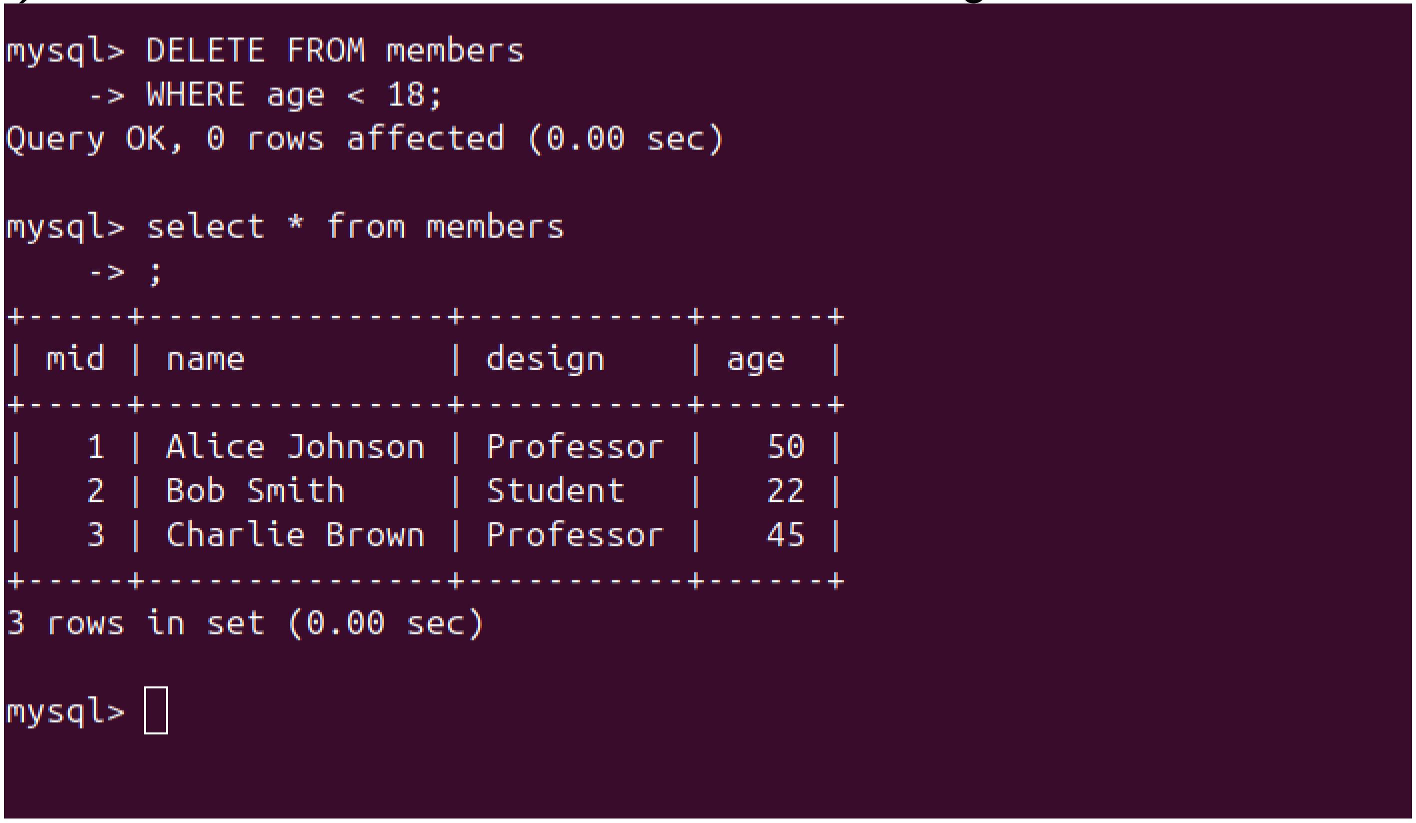
```
mysql> SELECT B.Btitle FROM books B JOIN reserves R ON B.Bid = R.Bid JOIN members
    -> M ON M.mid = R.mid WHERE M.design = 'Professor' AND M.age > 45;
+--------------------+
| Btitle             |
+--------------------+
| Advanced Databases |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT 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);
+-----+
| mid |
+-----+
|   2 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT B.BAuthor, B.Btitle FROM books B JOIN reserves R ON B.Bid = R.Bid
    -> WHERE R.date = '2024-08-10';
+-------------+-----------------+
| BAuthor     | Btitle          |
+-------------+-----------------+
| Emily Davis | Personal Growth |
+-------------+-----------------+
1 row in set (0.00 sec)

mysql> SELECT M.name FROM members M WHERE NOT EXISTS (SELECT B.Bid FROM books B WHERE
    -> NOT EXISTS (SELECT R.Bid FROM reserves R WHERE R.mid = M.mid AND R.Bid = B.Bid));
Empty set (0.00 sec)
```

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



```
mysql> UPDATE books SET Bprice = Bprice + 100 WHERE Bpublisher = 'Tech Publishers';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> desc books;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| Bid        | int           | NO   | PRI | NULL    |       |
| Btitle     | varchar(40)   | YES  |     | NULL    |       |
| BAuthor    | varchar(50)   | YES  |     | NULL    |       |
| Bpublisher | varchar(50)   | YES  |     | NULL    |       |
| Bprice     | decimal(10,2) | YES  |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

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

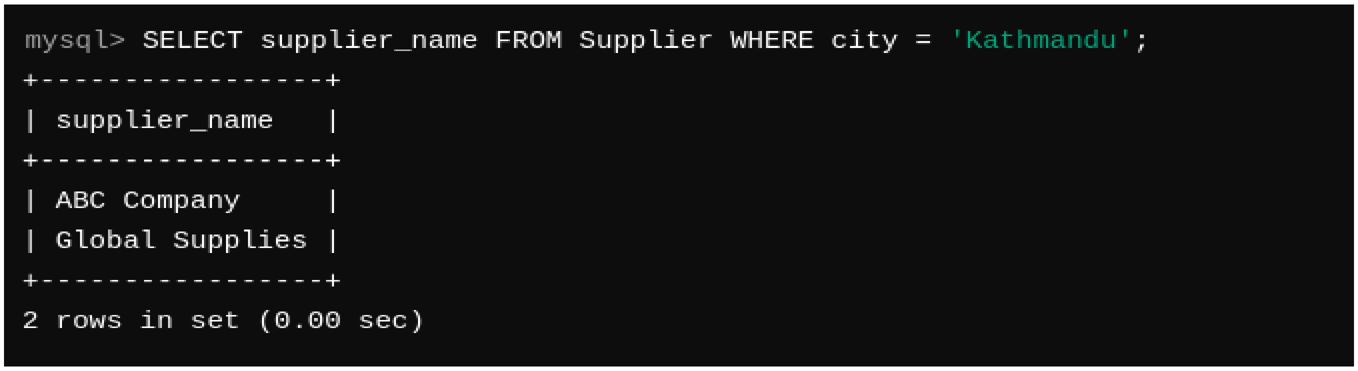


```
mysql> DELETE FROM members
    -> WHERE age < 18;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from members
    -> ;
+-----+---------------+-----------+------+
| mid | name          | design    | age  |
+-----+---------------+-----------+------+
|   1 | Alice Johnson | Professor |   50 |
|   2 | Bob Smith     | Student   |   22 |
|   3 | Charlie Brown | Professor |   45 |
+-----+---------------+-----------+------+
3 rows in set (0.00 sec)

mysql> 
```

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

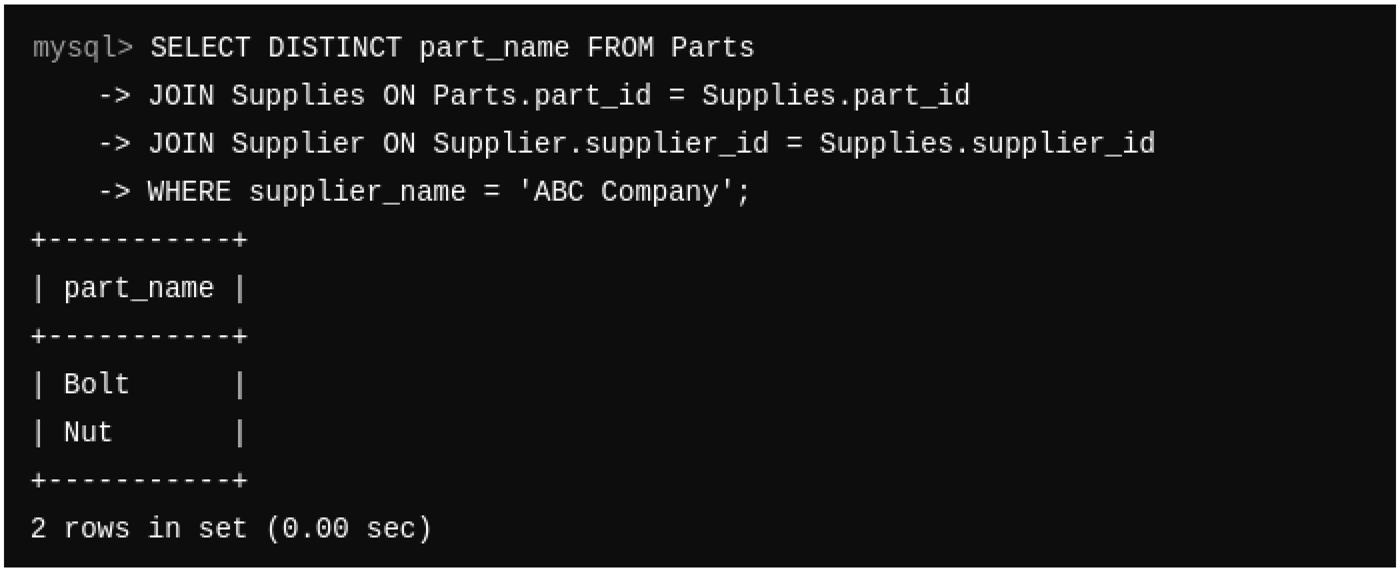
Supplier (supplier-id, supplier-name, city)
Supplies (supplier-id, part-id, quantity)
Parts (part-id, part-name, color, weight)

a) Find the name of all suppliers located in the city "Kathmandu".
Relational Algebra: $\pi_{supplier}$ name(σ_{city}='Kathmandu'(Supplier))



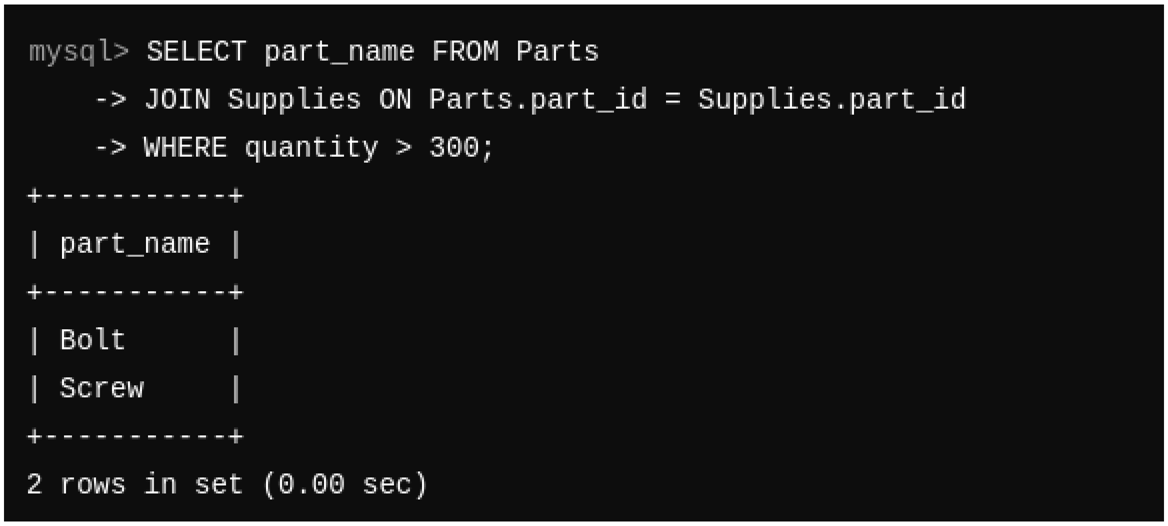
```
mysql> SELECT supplier_name FROM Supplier WHERE city = 'Kathmandu';
+-----------------+
| supplier_name   |
+-----------------+
| ABC Company     |
| Global Supplies |
+-----------------+
2 rows in set (0.00 sec)
```

b) Find the name of all parts supplied by "ABC Company".Relational Algebra: πpart-name(σsupplier-name='ABC Company'(Supplier ⋈ Supplies ⋈ Parts))



```
mysql> SELECT DISTINCT part_name FROM Parts
    -> JOIN Supplies ON Parts.part_id = Supplies.part_id
    -> JOIN Supplier ON Supplier.supplier_id = Supplies.supplier_id
    -> WHERE supplier_name = 'ABC Company';
+-----------+
| part_name |
+-----------+
| Bolt      |
| Nut       |
+-----------+
2 rows in set (0.00 sec)
```

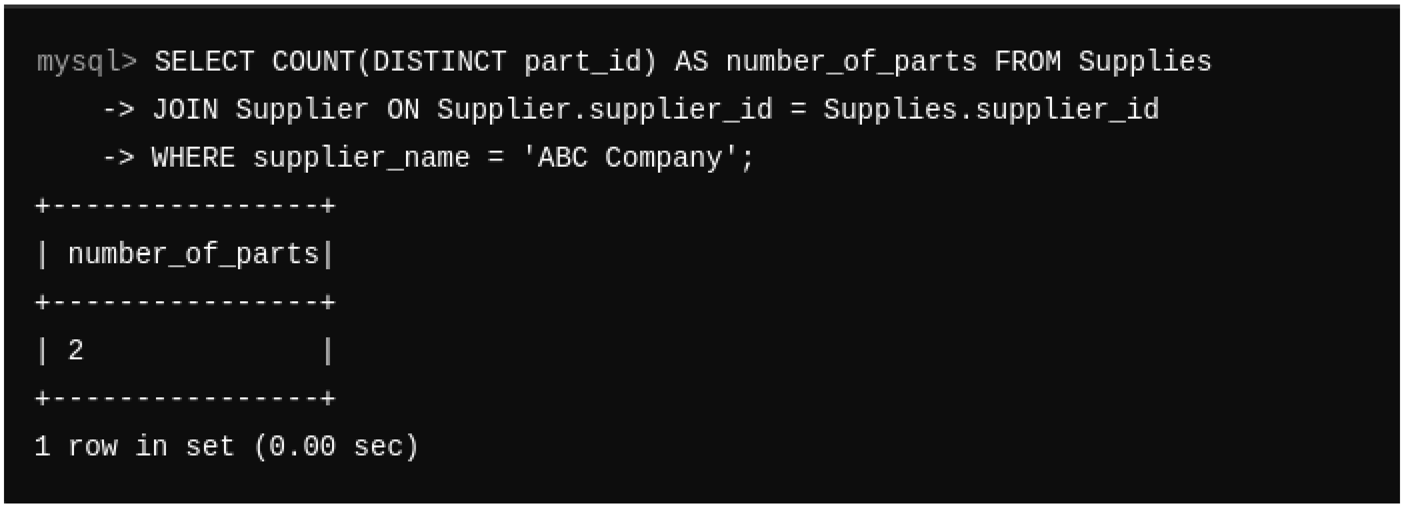
c) Find the name of all parts that are supplied in quantities greater than 300. Relational Algebra: πpart-name(σquantity>300(Supplies ⋈ Parts))



```
mysql> SELECT part_name FROM Parts
    -> JOIN Supplies ON Parts.part_id = Supplies.part_id
    -> WHERE quantity > 300;
+-----------+
| part_name |
+-----------+
| Bolt      |
| Screw     |
+-----------+
2 rows in set (0.00 sec)
```

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

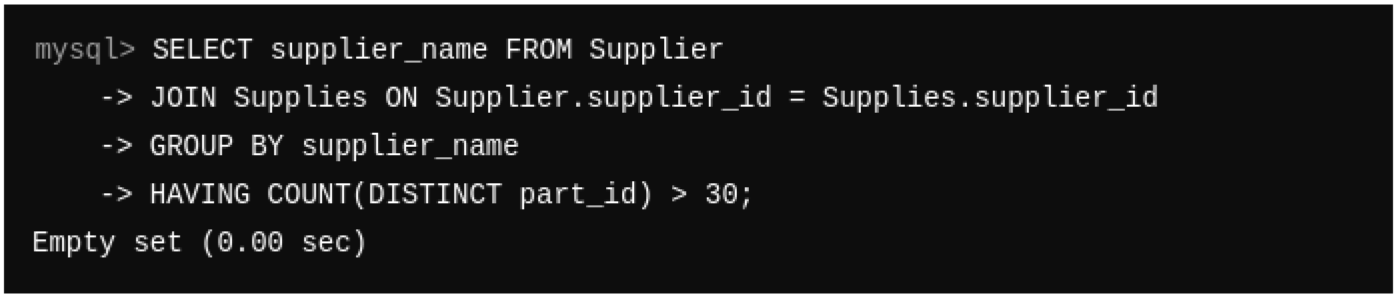
Relational Algebra: $\boldsymbol{\sigma}_{\textbf{supplier-name='ABC Company'}}(\boldsymbol{\gamma}_{\textbf{count(part-id)}}(\textbf{Supplier} \bowtie \textbf{Supplies}))$



```
mysql> SELECT COUNT(DISTINCT part_id) AS number_of_parts FROM Supplies
    -> JOIN Supplier ON Supplier.supplier_id = Supplies.supplier_id
    -> WHERE supplier_name = 'ABC Company';
+----------------+
| number_of_parts|
+----------------+
| 2              |
+----------------+
1 row in set (0.00 sec)
```

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

Relational Algebra: $\boldsymbol{\pi}_{\textbf{supplier-name}}(\boldsymbol{\sigma}_{\textbf{count(part-id)>30}}(\boldsymbol{\gamma}_{\textbf{supplier-id,count(part-id)}}(\textbf{Supplies}) \bowtie \textbf{Supplier}))$



```
mysql> SELECT supplier_name FROM Supplier
    -> JOIN Supplies ON Supplier.supplier_id = Supplies.supplier_id
    -> GROUP BY supplier_name
    -> HAVING COUNT(DISTINCT part_id) > 30;
Empty set (0.00 sec)
```

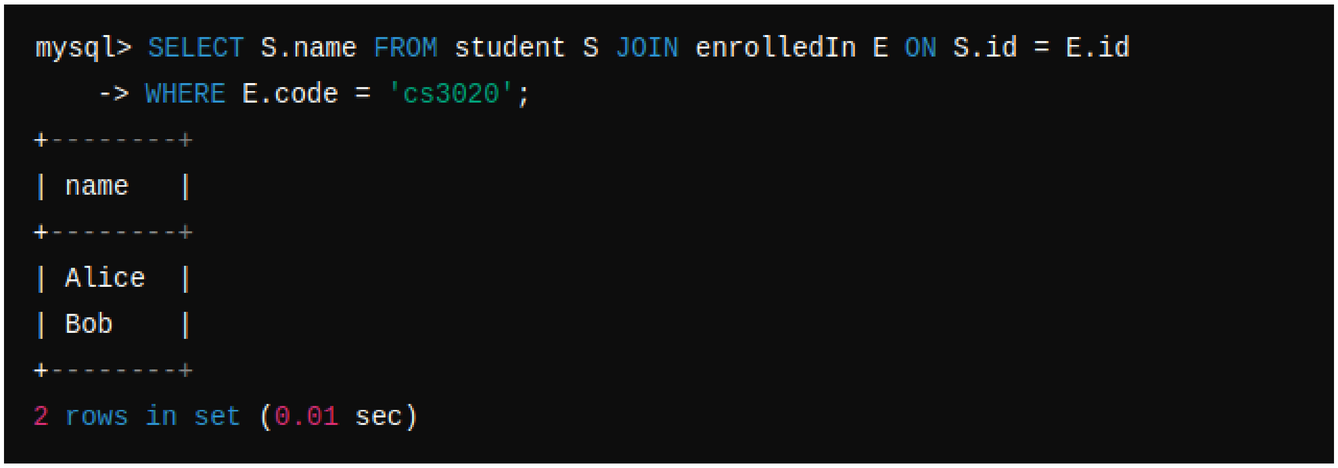
4) Consider the following relational schema and write the SQL for the following.

student(id, name)

enrolledIn(id, code)

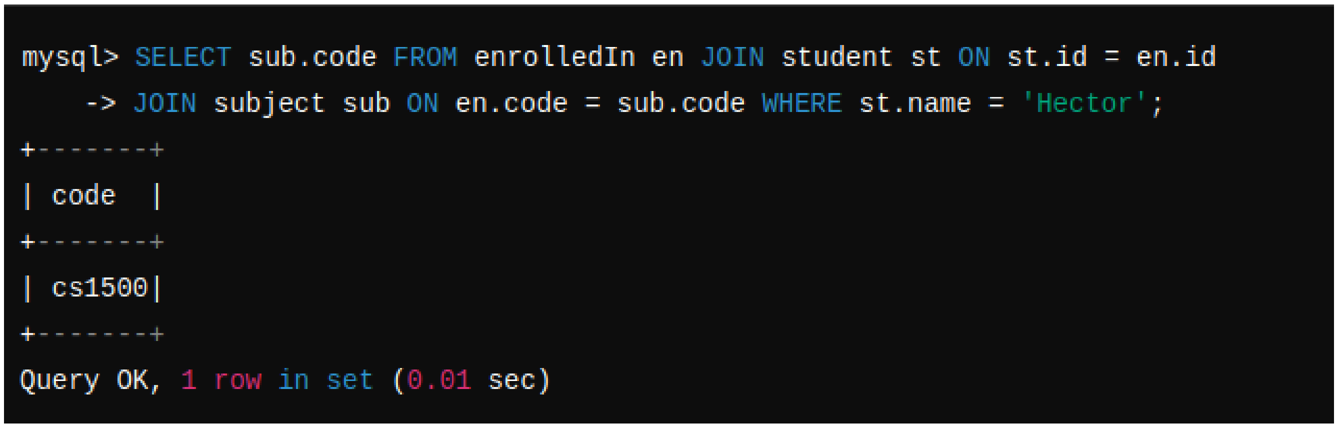
subject(code, lecturer)

I. What are the names of students enrolled in cs3020?



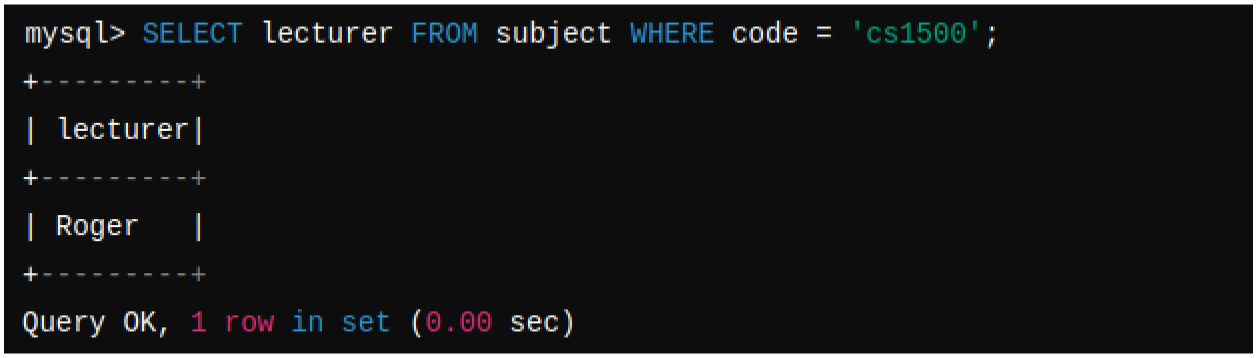
```
mysql> SELECT S.name FROM student S JOIN enrolledIn E ON S.id = E.id
    -> WHERE E.code = 'cs3020';
+--------+
| name   |
+--------+
| Alice  |
| Bob    |
+--------+
2 rows in set (0.01 sec)
```

II. Which subjects is Hector taking?



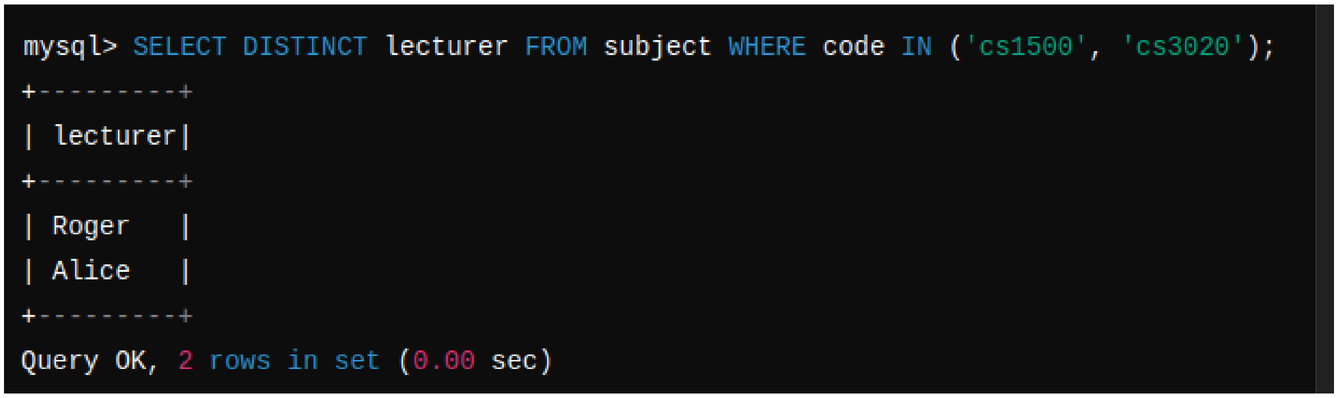
```
mysql> SELECT sub.code FROM enrolledIn en JOIN student st ON st.id = en.id
    -> JOIN subject sub ON en.code = sub.code WHERE st.name = 'Hector';
+-------+
| code  |
+-------+
| cs1500|
+-------+
Query OK, 1 row in set (0.01 sec)
```

III. Who teaches cs1500?



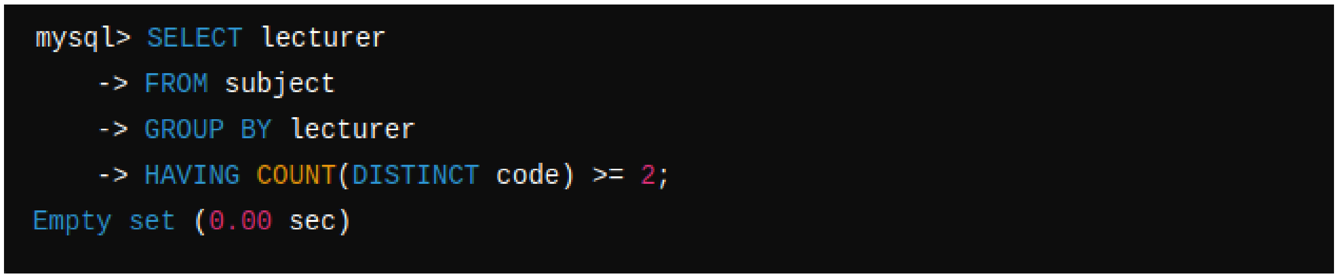
```
mysql> SELECT lecturer FROM subject WHERE code = 'cs1500';
+---------+
| lecturer|
+---------+
| Roger   |
+---------+
Query OK, 1 row in set (0.00 sec)
```

IV. Who teaches cs1500 or cs3020?



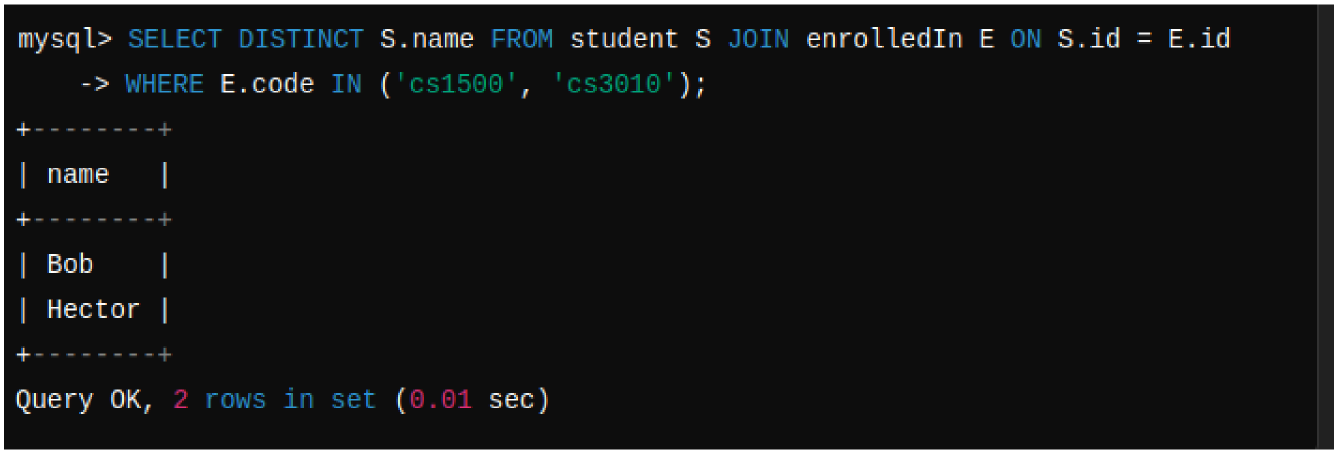
```
mysql> SELECT DISTINCT lecturer FROM subject WHERE code IN ('cs1500', 'cs3020');
+---------+
| lecturer|
+---------+
| Roger   |
| Alice   |
+---------+
Query OK, 2 rows in set (0.00 sec)
```

V. Who teaches at least two different subjects?



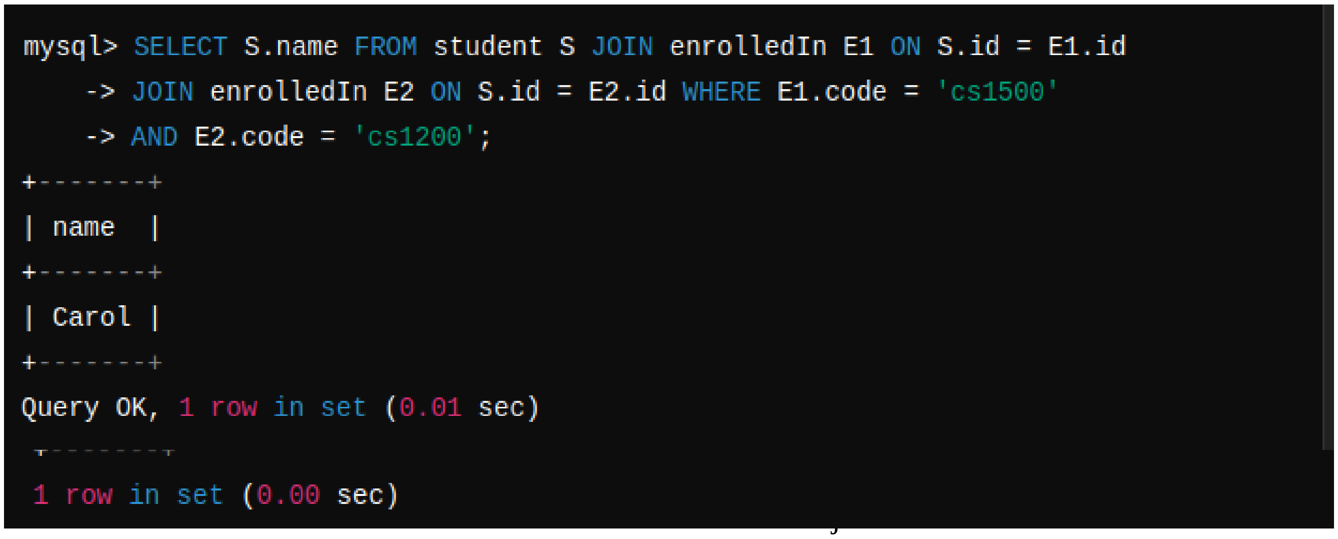
```
 mysql> SELECT lecturer
     -> FROM subject
     -> GROUP BY lecturer
     -> HAVING COUNT(DISTINCT code) >= 2;
 Empty set (0.00 sec)
```

VI. What are the names of students in cs1500 or cs3010?

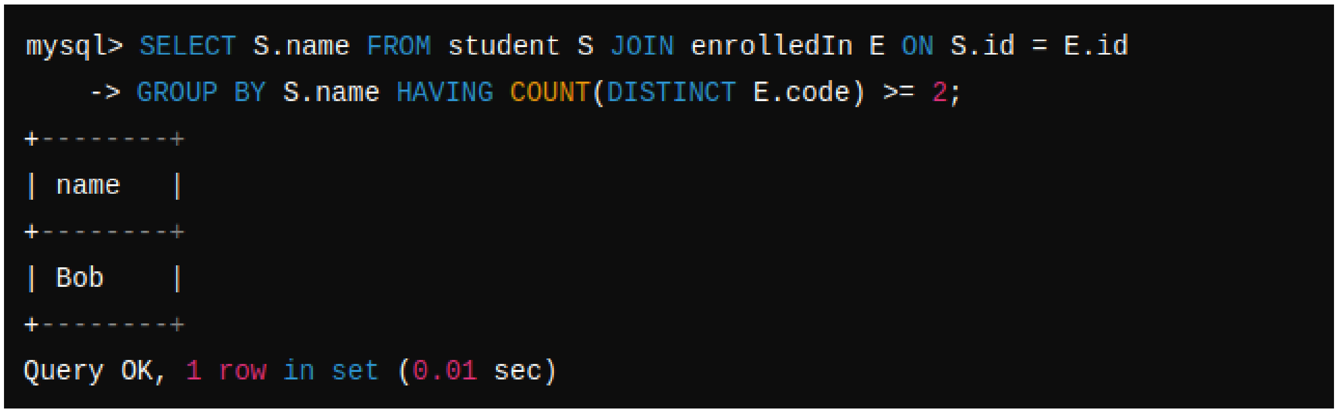


```
mysql> SELECT DISTINCT S.name FROM student S JOIN enrolledIn E ON S.id = E.id
    -> WHERE E.code IN ('cs1500', 'cs3010');
+--------+
| name   |
+--------+
| Bob    |
| Hector |
+--------+
Query OK, 2 rows in set (0.01 sec)
```

VII. What are the names of students in both cs1500 and cs1200?

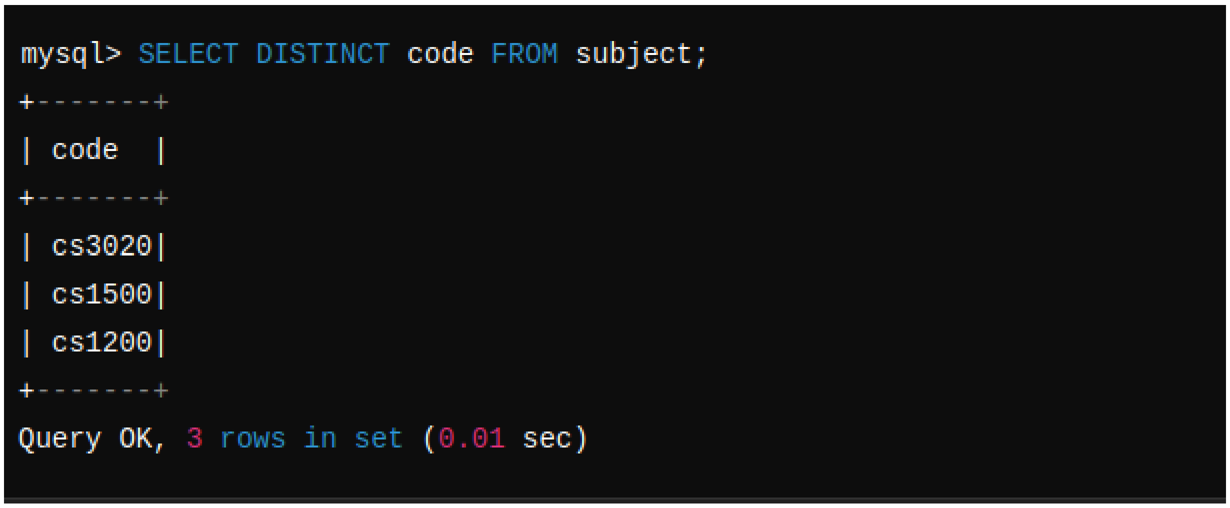


```
mysql> SELECT S.name FROM student S JOIN enrolledIn E1 ON S.id = E1.id
    -> JOIN enrolledIn E2 ON S.id = E2.id WHERE E1.code = 'cs1500'
    -> AND E2.code = 'cs1200';
+-------+
| name  |
+-------+
| Carol |
+-------+
Query OK, 1 row in set (0.01 sec)
 +--------+
 1 row in set (0.00 sec)
```



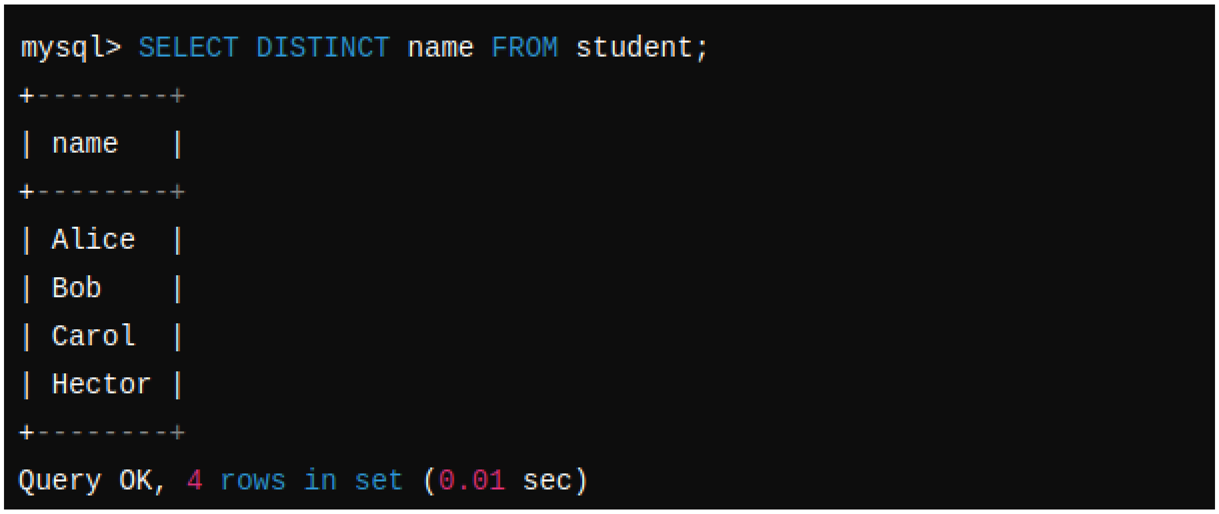
```
mysql> SELECT S.name FROM student S JOIN enrolledIn E ON S.id = E.id
    -> GROUP BY S.name HAVING COUNT(DISTINCT E.code) >= 2;
+--------+
| name   |
+--------+
| Bob    |
+--------+
Query OK, 1 row in set (0.01 sec)
```

IX. What are the codes of all the subjects taught?



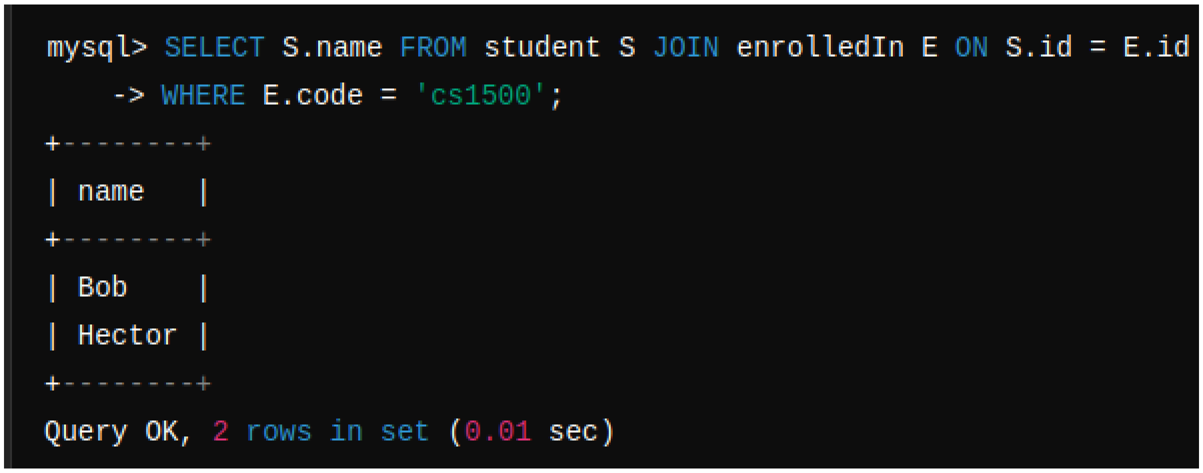
```
mysql> SELECT DISTINCT code FROM subject;
+-------+
| code  |
+-------+
| cs3020|
| cs1500|
| cs1200|
+-------+
Query OK, 3 rows in set (0.01 sec)
```

X. What are the names of all the students?



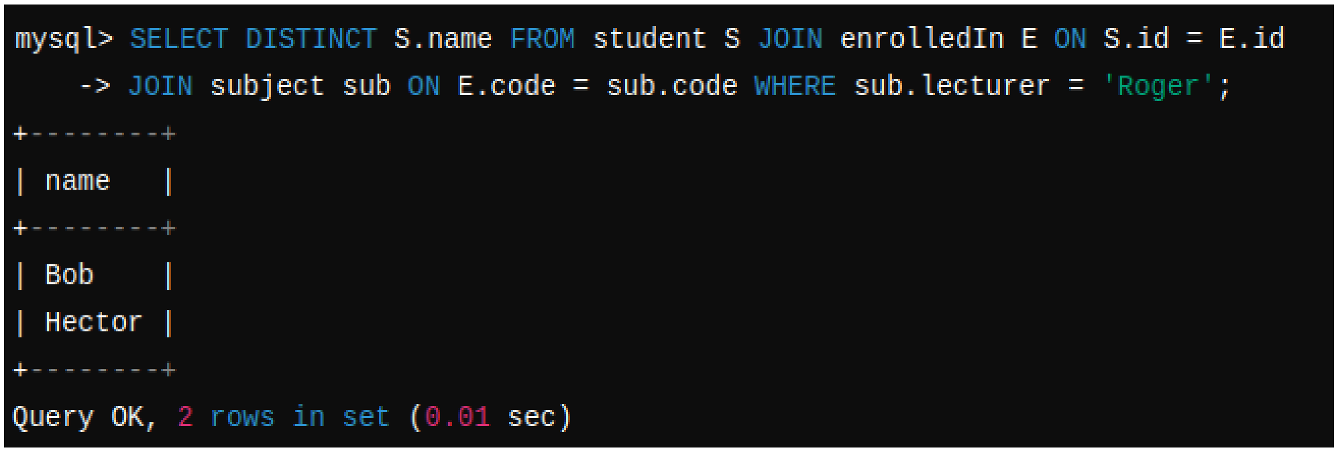
```
mysql> SELECT DISTINCT name FROM student;
+--------+
| name   |
+--------+
| Alice  |
| Bob    |
| Carol  |
| Hector |
+--------+
Query OK, 4 rows in set (0.01 sec)
```

XI. What are the names of all the students in cs1500?



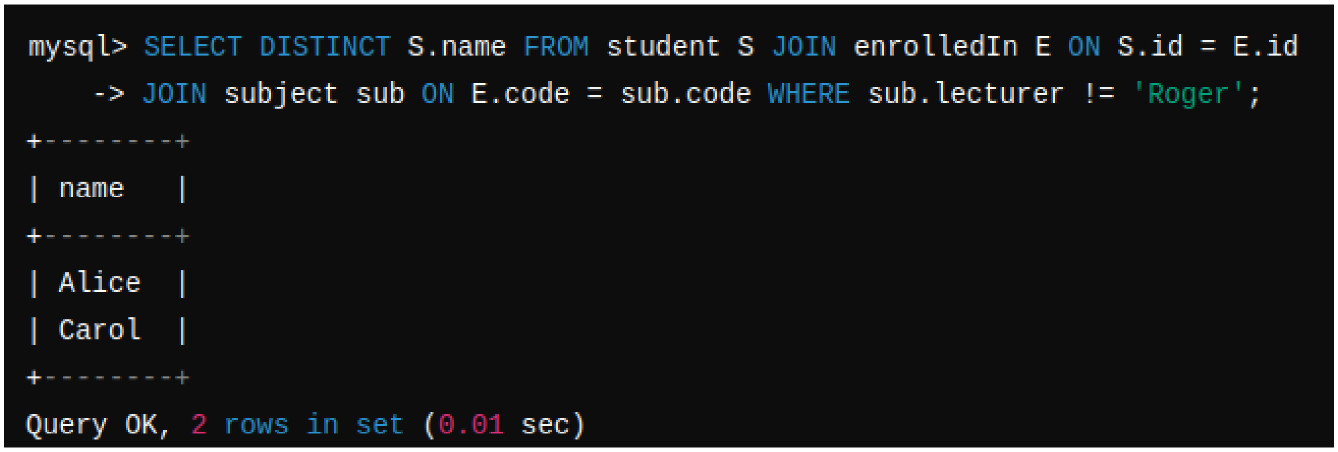
```
mysql> SELECT S.name FROM student S JOIN enrolledIn E ON S.id = E.id
    -> WHERE E.code = 'cs1500';
+--------+
| name   |
+--------+
| Bob    |
| Hector |
+--------+
Query OK, 2 rows in set (0.01 sec)
```

XII. What are the names of students taking a subject taught by Roger?



```
mysql> SELECT DISTINCT S.name FROM student S JOIN enrolledIn E ON S.id = E.id
    -> JOIN subject sub ON E.code = sub.code WHERE sub.lecturer = 'Roger';
+--------+
| name   |
+--------+
| Bob    |
| Hector |
+--------+
Query OK, 2 rows in set (0.01 sec)
```

XIII. What are the names of students who are taking a subject not taught by Roger?



```
mysql> SELECT DISTINCT S.name FROM student S JOIN enrolledIn E ON S.id = E.id
    -> JOIN subject sub ON E.code = sub.code WHERE sub.lecturer != 'Roger';
+--------+
| name   |
+--------+
| Alice  |
| Carol  |
+--------+
Query OK, 2 rows in set (0.01 sec)
```

5) Consider the following relational database.

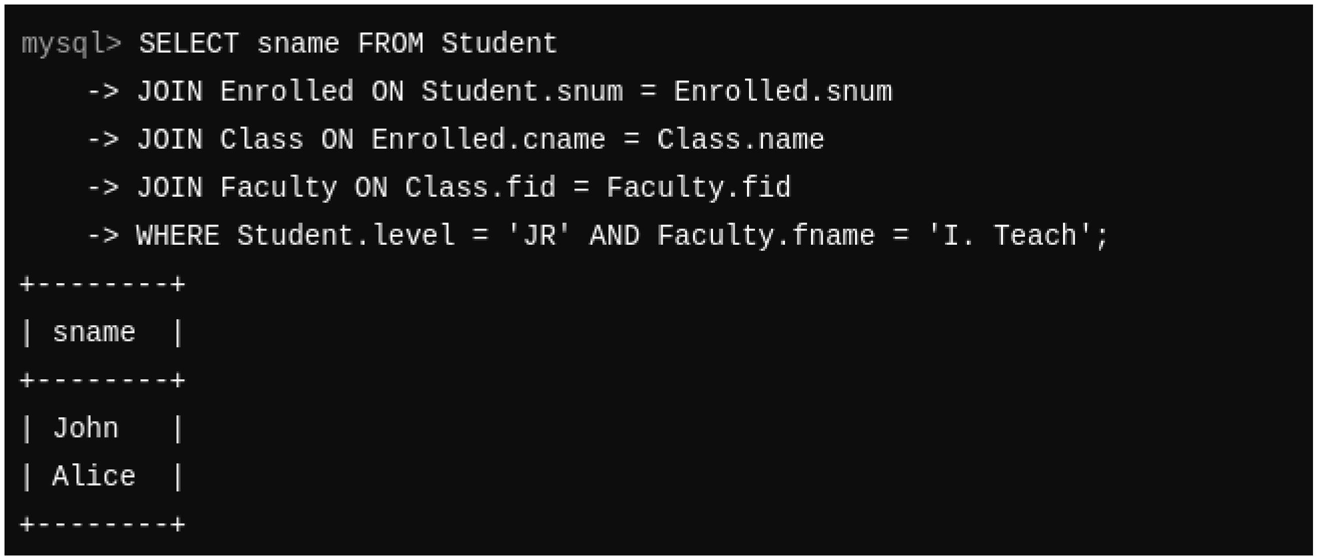
Student(snum: integer, sname: string, major: string, level: string, age: integer)

Class(name: string, meets at: time, room: string, fid: integer)

Enrolled(snum: integer, cname: string)

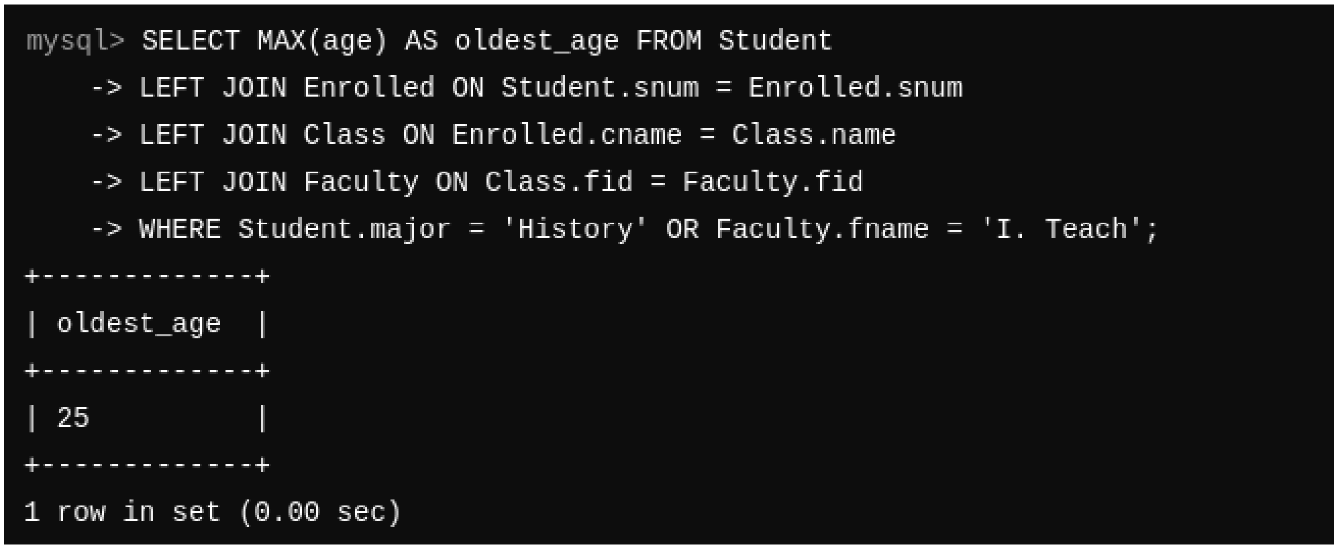
Faculty(fid: integer, fname: string, deptid: integer)

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



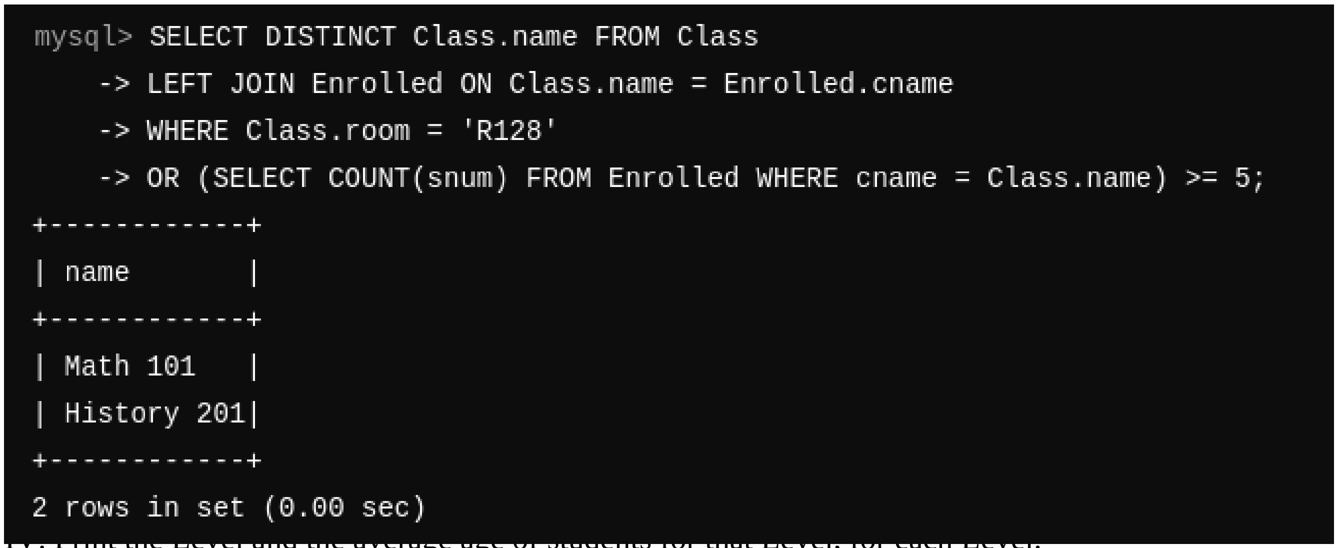
```
mysql> SELECT sname FROM Student
    -> JOIN Enrolled ON Student.snum = Enrolled.snum
    -> JOIN Class ON Enrolled.cname = Class.name
    -> JOIN Faculty ON Class.fid = Faculty.fid
    -> WHERE Student.level = 'JR' AND Faculty.fname = 'I. Teach';
+--------+
| sname  |
+--------+
| John   |
| Alice  |
+--------+
```

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



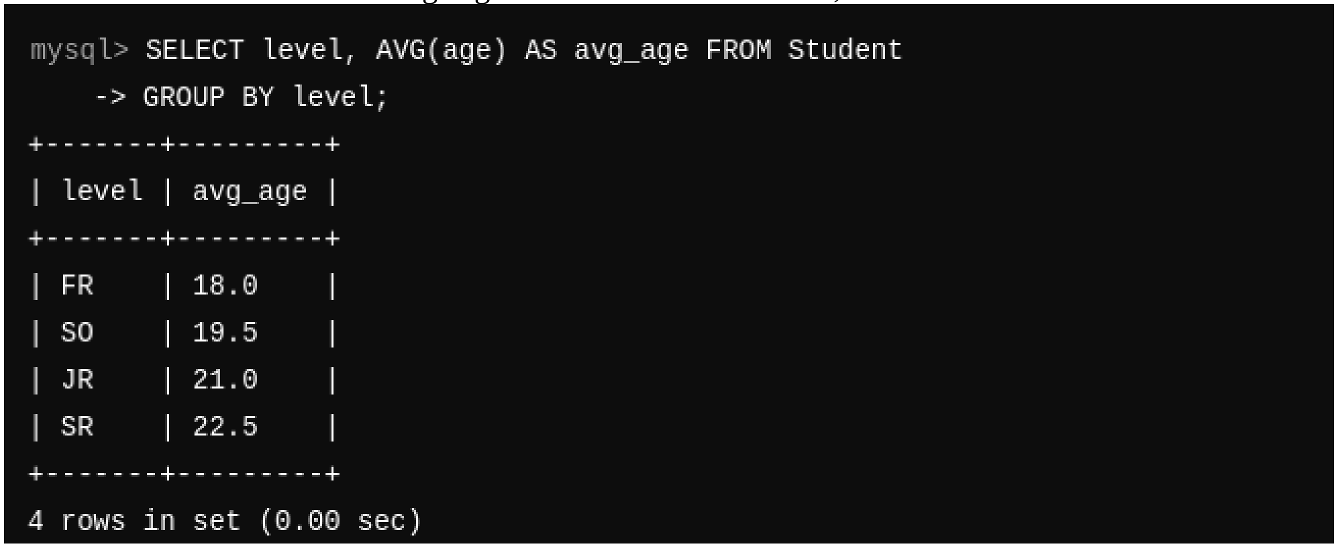
```
mysql> SELECT MAX(age) AS oldest_age FROM Student
    -> LEFT JOIN Enrolled ON Student.snum = Enrolled.snum
    -> LEFT JOIN Class ON Enrolled.cname = Class.name
    -> LEFT JOIN Faculty ON Class.fid = Faculty.fid
    -> WHERE Student.major = 'History' OR Faculty.fname = 'I. Teach';
+-------------+
| oldest_age  |
+-------------+
| 25          |
+-------------+
1 row in set (0.00 sec)
```

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



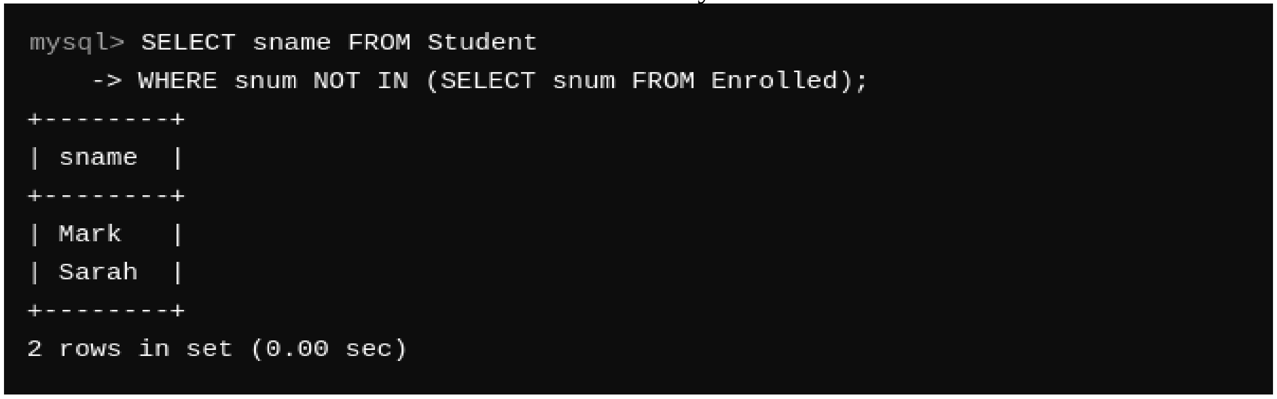
```
mysql> SELECT DISTINCT Class.name FROM Class
    -> LEFT JOIN Enrolled ON Class.name = Enrolled.cname
    -> WHERE Class.room = 'R128'
    -> OR (SELECT COUNT(snum) FROM Enrolled WHERE cname = Class.name) >= 5;
+------------+
| name       |
+------------+
| Math 101   |
| History 201|
+------------+
2 rows in set (0.00 sec)
```

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



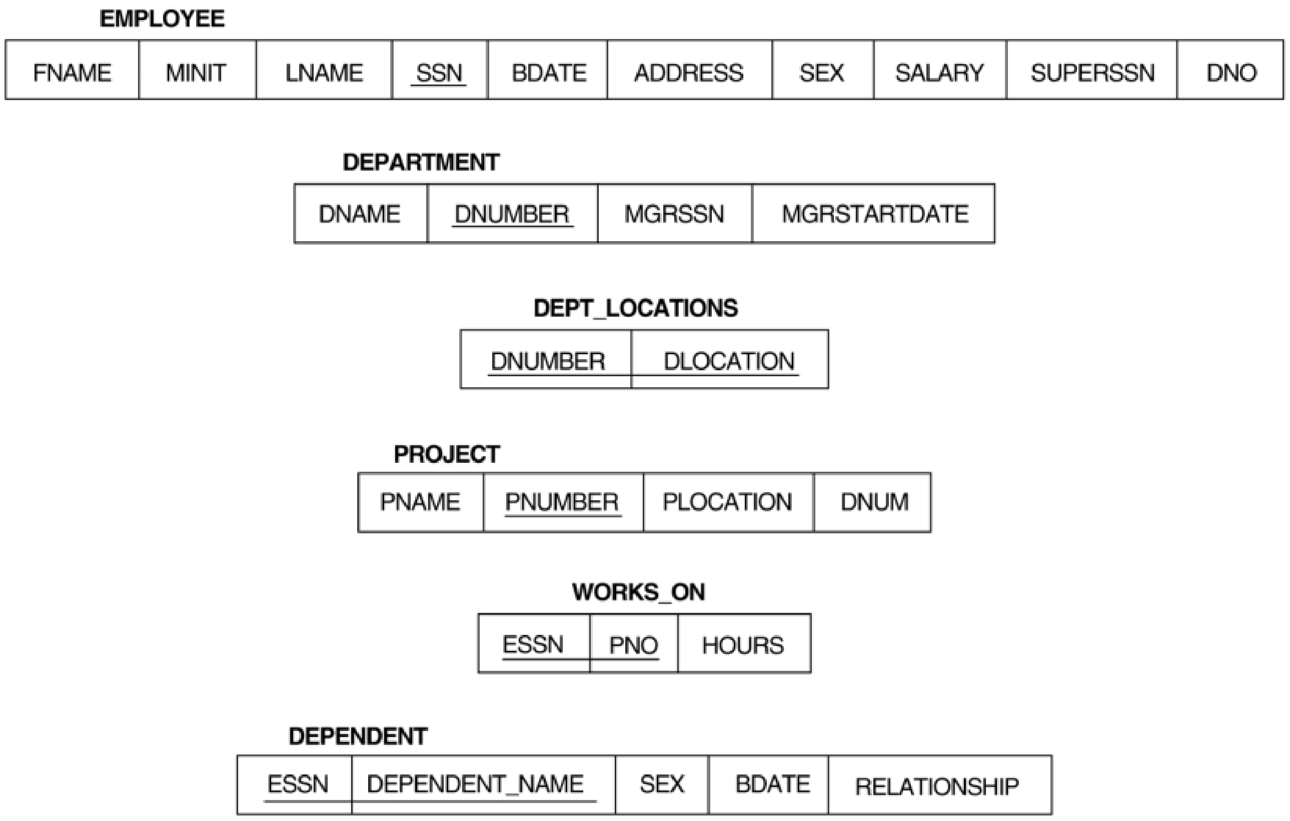
```
mysql> SELECT level, AVG(age) AS avg_age FROM Student
    -> GROUP BY level;
+-------+---------+
| level | avg_age |
+-------+---------+
| FR    | 18.0    |
| SO    | 19.5    |
| JR    | 21.0    |
| SR    | 22.5    |
+-------+---------+
4 rows in set (0.00 sec)
```

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



```
mysql> SELECT sname FROM Student
    -> WHERE snum NOT IN (SELECT snum FROM Enrolled);
+--------+
| sname  |
+--------+
| Mark   |
| Sarah  |
+--------+
2 rows in set (0.00 sec)
```

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.



EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

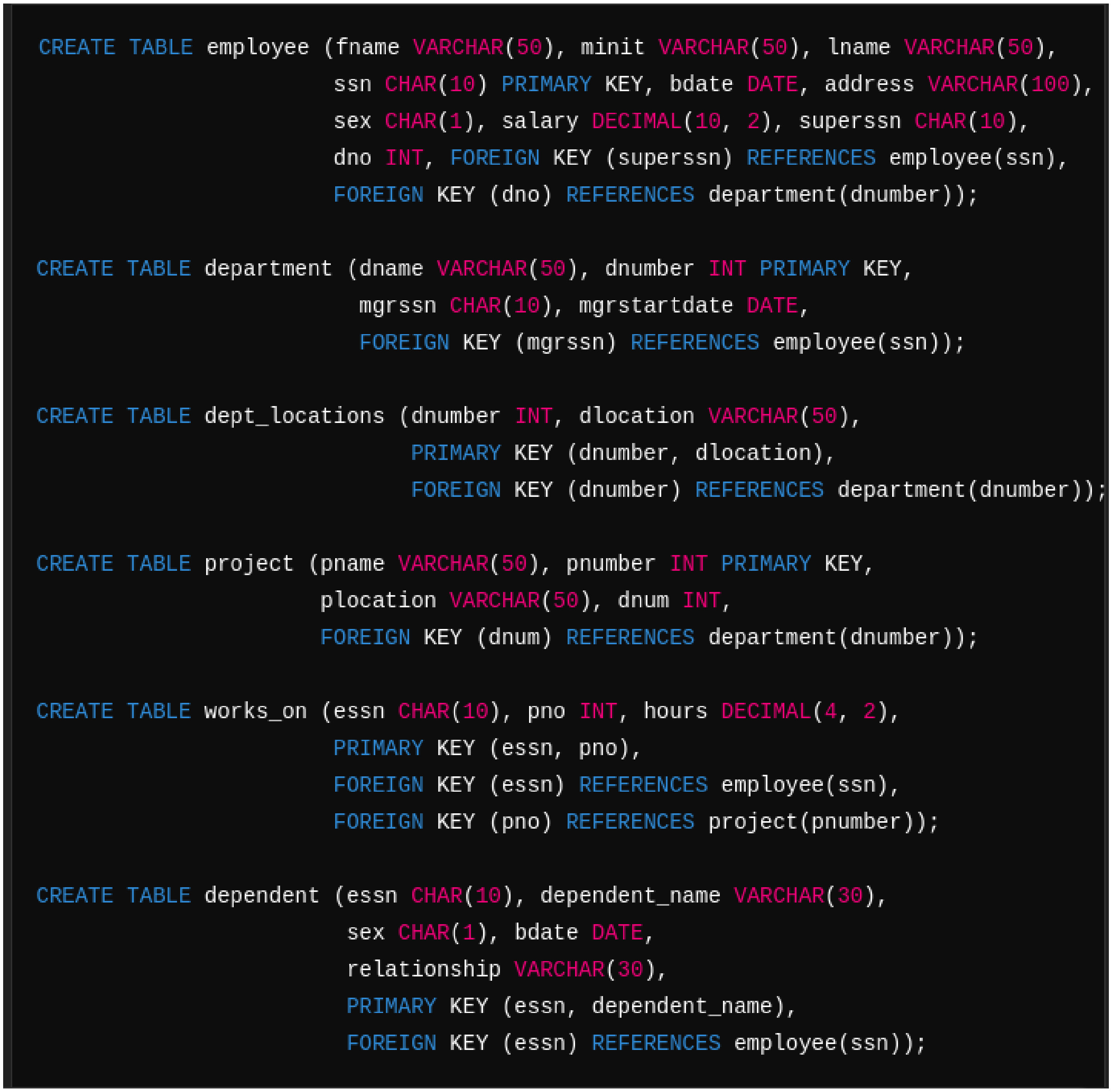
WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

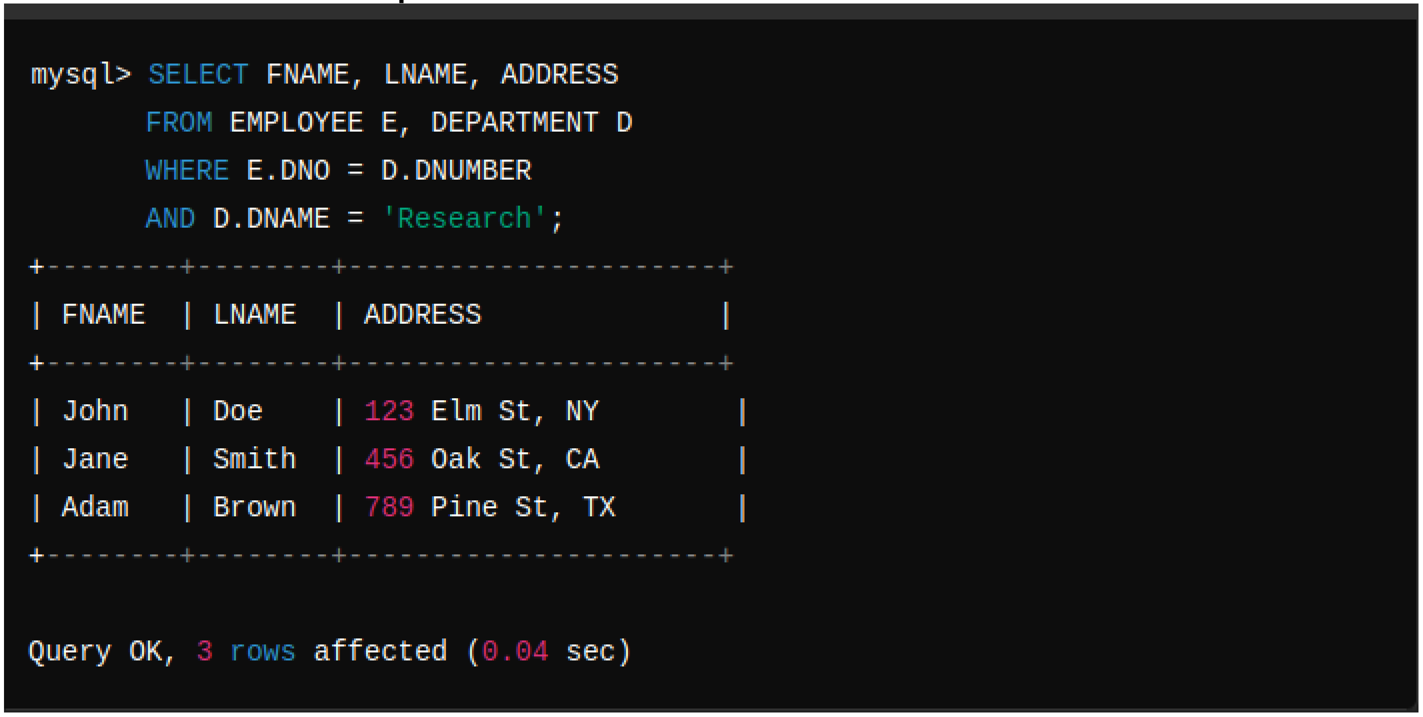
ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

DDL:



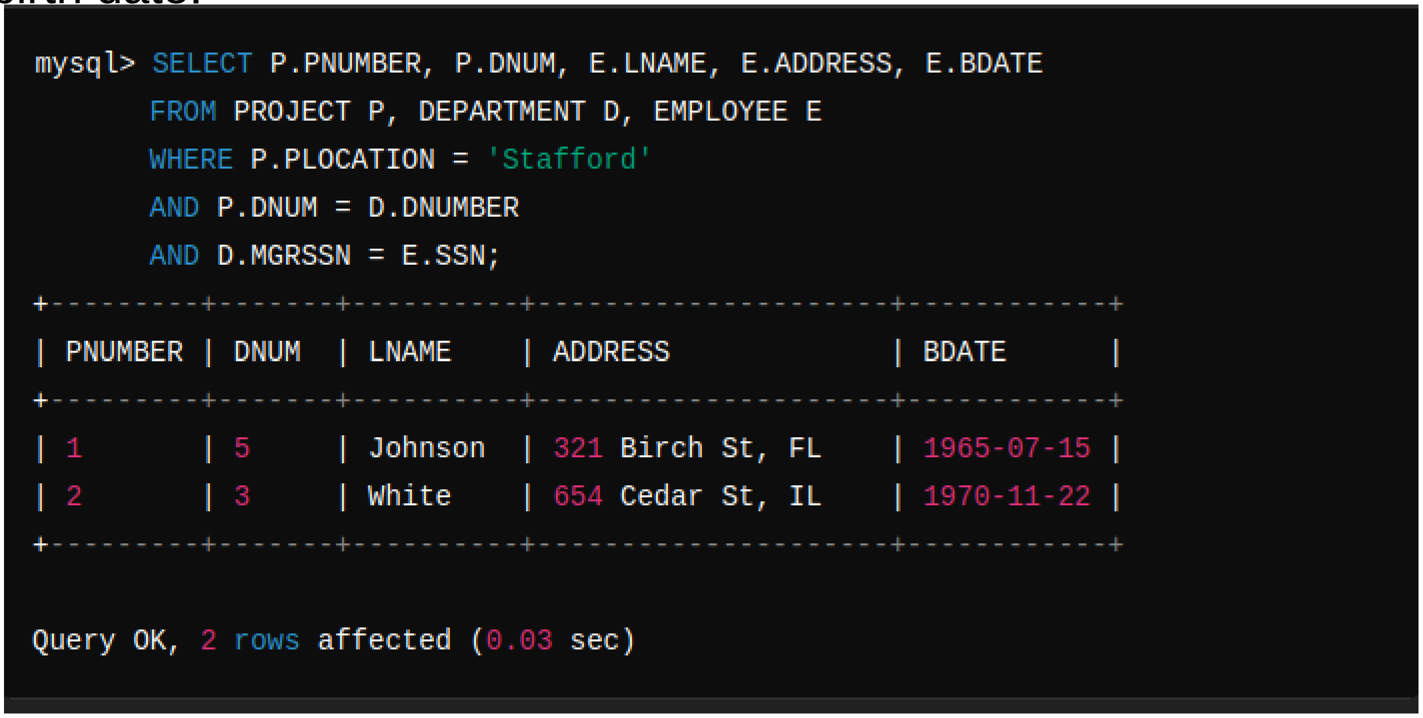
```sql
CREATE TABLE employee (fname VARCHAR(50), minit VARCHAR(50), lname VARCHAR(50),
                       ssn CHAR(10) PRIMARY KEY, bdate DATE, address VARCHAR(100),
                       sex CHAR(1), salary DECIMAL(10, 2), superssn CHAR(10),
                       dno INT, FOREIGN KEY (superssn) REFERENCES employee(ssn),
                       FOREIGN KEY (dno) REFERENCES department(dnumber));

CREATE TABLE department (dname VARCHAR(50), dnumber INT PRIMARY KEY,
                         mgrssn CHAR(10), mgrstartdate DATE,
                         FOREIGN KEY (mgrssn) REFERENCES employee(ssn));

CREATE TABLE dept_locations (dnumber INT, dlocation VARCHAR(50),
                             PRIMARY KEY (dnumber, dlocation),
                             FOREIGN KEY (dnumber) REFERENCES department(dnumber));

CREATE TABLE project (pname VARCHAR(50), pnumber INT PRIMARY KEY,
                      plocation VARCHAR(50), dnum INT,
                      FOREIGN KEY (dnum) REFERENCES department(dnumber));

CREATE TABLE works_on (essn CHAR(10), pno INT, hours DECIMAL(4, 2),
                       PRIMARY KEY (essn, pno),
                       FOREIGN KEY (essn) REFERENCES employee(ssn),
                       FOREIGN KEY (pno) REFERENCES project(pnumber));

CREATE TABLE dependent (essn CHAR(10), dependent_name VARCHAR(30),
                        sex CHAR(1), bdate DATE,
                        relationship VARCHAR(30),
                        PRIMARY KEY (essn, dependent_name),
                        FOREIGN KEY (essn) REFERENCES employee(ssn));
```

I) Retrieve the name and address of all employees who work for the ‘Research’ department.



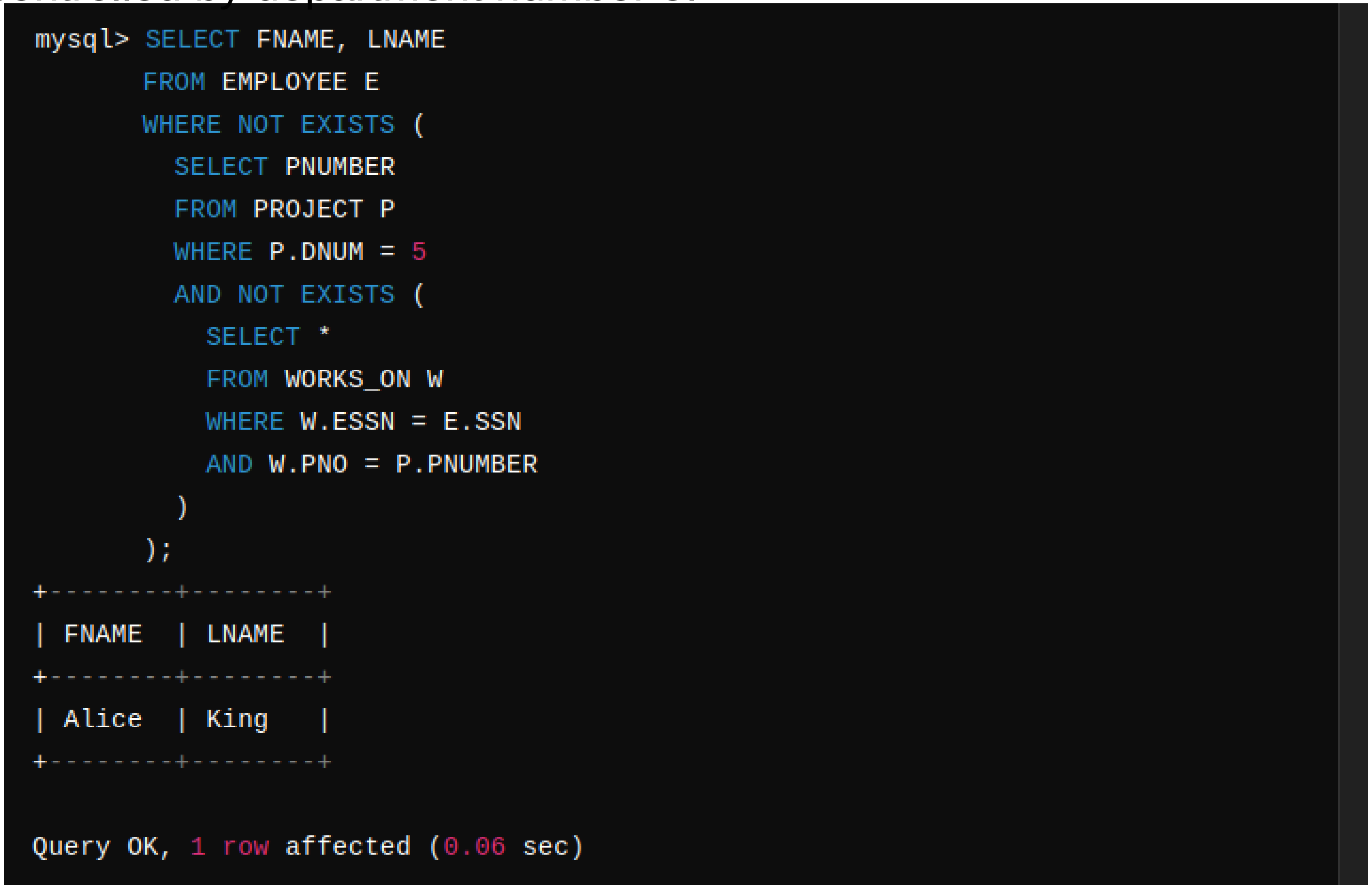
```
mysql> SELECT FNAME, LNAME, ADDRESS
       FROM EMPLOYEE E, DEPARTMENT D
       WHERE E.DNO = D.DNUMBER
       AND D.DNAME = 'Research';
+--------+--------+----------------------+
| FNAME  | LNAME  | ADDRESS              |
+--------+--------+----------------------+
| John   | Doe    | 123 Elm St, NY        |
| Jane   | Smith  | 456 Oak St, CA        |
| Adam   | Brown  | 789 Pine St, TX       |
+--------+--------+----------------------+

Query OK, 3 rows affected (0.04 sec)
```

II. 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.



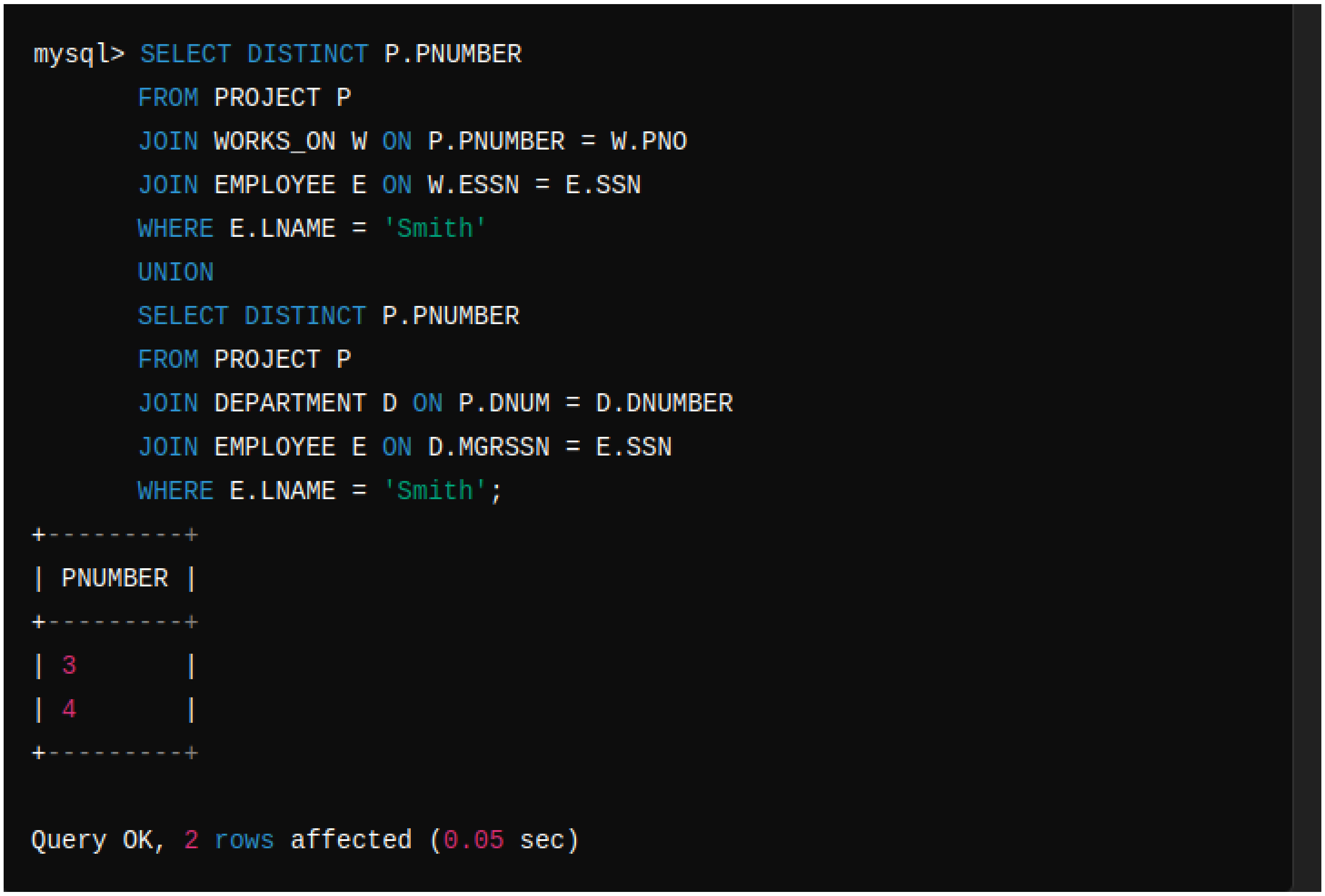
```
mysql> SELECT P.PNUMBER, P.DNUM, E.LNAME, E.ADDRESS, E.BDATE
       FROM PROJECT P, DEPARTMENT D, EMPLOYEE E
       WHERE P.PLOCATION = 'Stafford'
       AND P.DNUM = D.DNUMBER
       AND D.MGRSSN = E.SSN;
+---------+-------+----------+---------------------+------------+
| PNUMBER | DNUM  | LNAME    | ADDRESS             | BDATE      |
+---------+-------+----------+---------------------+------------+
| 1       | 5     | Johnson  | 321 Birch St, FL    | 1965-07-15 |
| 2       | 3     | White    | 654 Cedar St, IL    | 1970-11-22 |
+---------+-------+----------+---------------------+------------+

Query OK, 2 rows affected (0.03 sec)
```

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



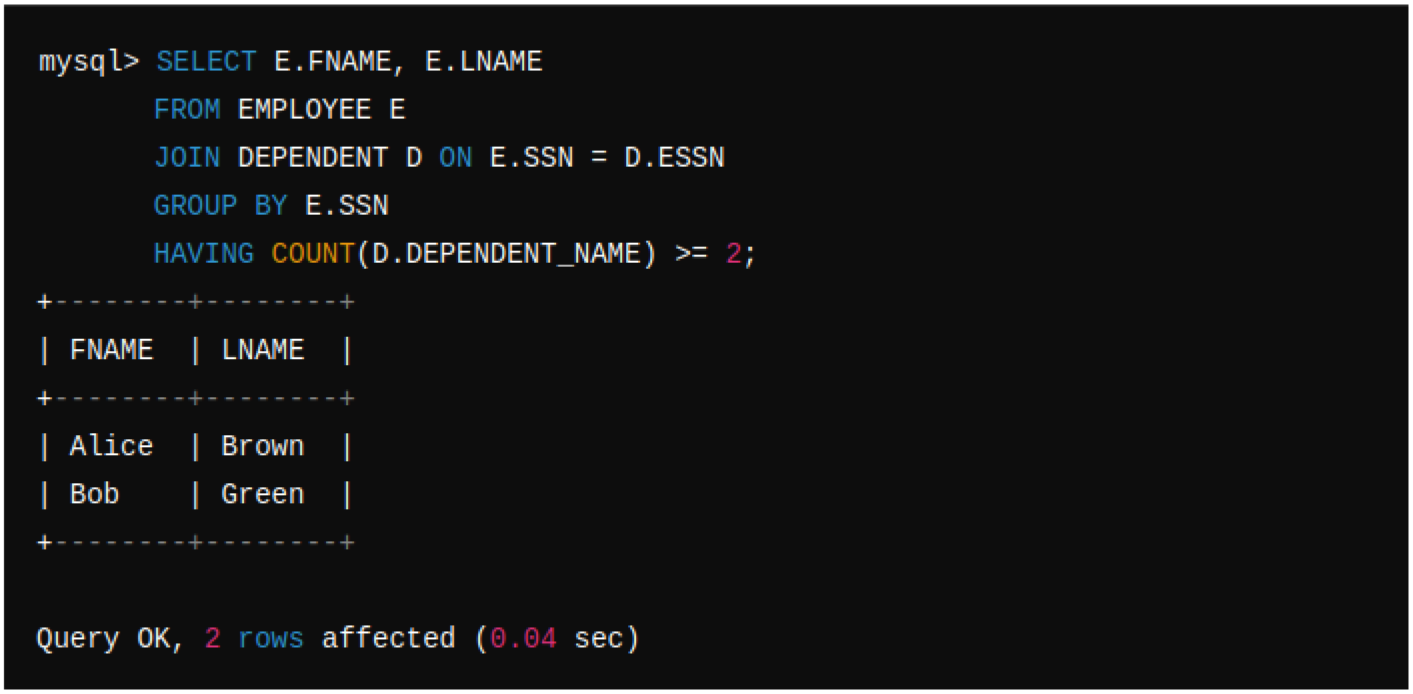
```
mysql> SELECT FNAME, LNAME
       FROM EMPLOYEE E
       WHERE NOT EXISTS (
         SELECT PNUMBER
         FROM PROJECT P
         WHERE P.DNUM = 5
         AND NOT EXISTS (
           SELECT *
           FROM WORKS_ON W
           WHERE W.ESSN = E.SSN
           AND W.PNO = P.PNUMBER
         )
       );
+--------+--------+
| FNAME  | LNAME  |
+--------+--------+
| Alice  | King   |
+--------+--------+

Query OK, 1 row affected (0.06 sec)
```

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



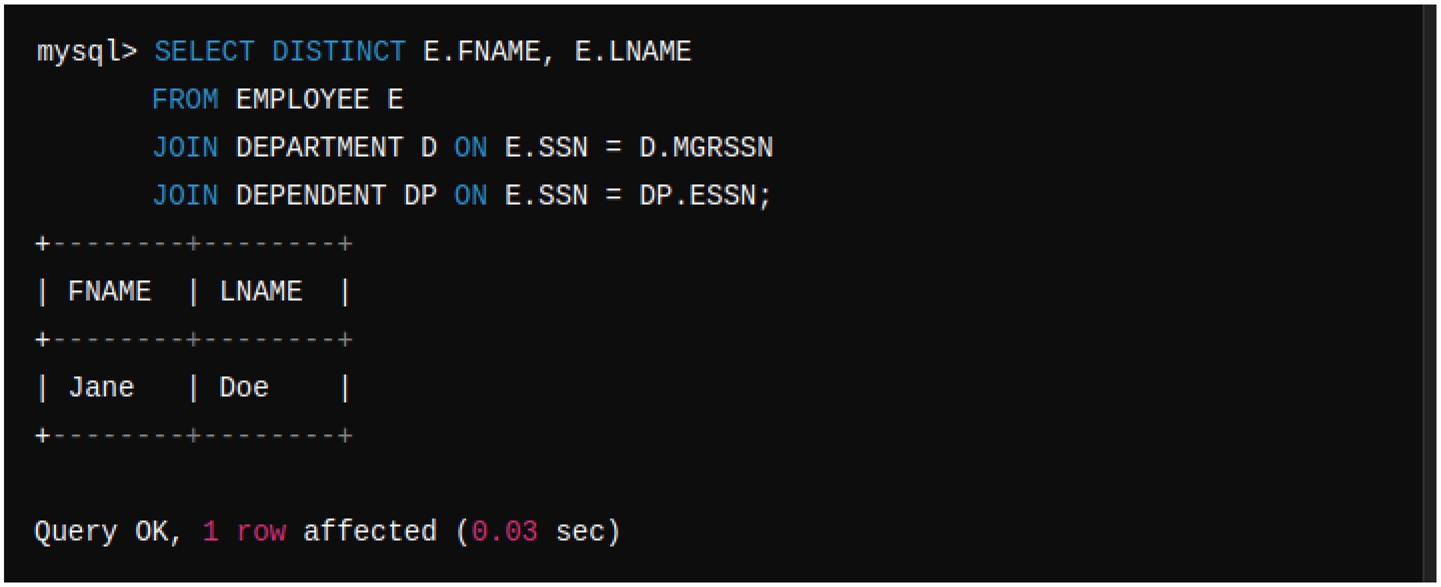
```
mysql> SELECT DISTINCT P.PNUMBER
       FROM PROJECT P
       JOIN WORKS_ON W ON P.PNUMBER = W.PNO
       JOIN EMPLOYEE E ON W.ESSN = E.SSN
       WHERE E.LNAME = 'Smith'
       UNION
       SELECT DISTINCT P.PNUMBER
       FROM PROJECT P
       JOIN DEPARTMENT D ON P.DNUM = D.DNUMBER
       JOIN EMPLOYEE E ON D.MGRSSN = E.SSN
       WHERE E.LNAME = 'Smith';
+---------+
| PNUMBER |
+---------+
| 3       |
| 4       |
+---------+

Query OK, 2 rows affected (0.05 sec)
```

V. List the names of all employees with two or more dependent



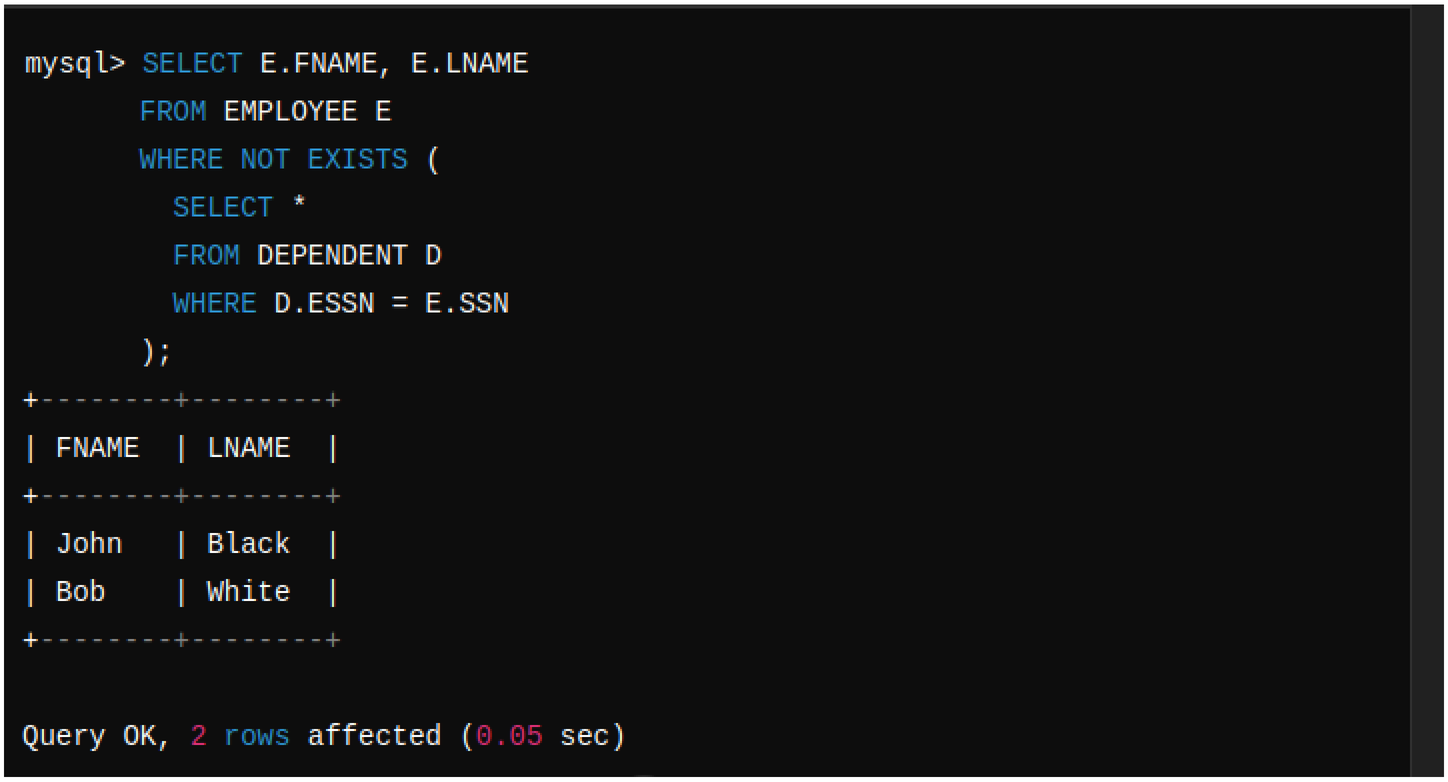
```
mysql> SELECT E.FNAME, E.LNAME
       FROM EMPLOYEE E
       JOIN DEPENDENT D ON E.SSN = D.ESSN
       GROUP BY E.SSN
       HAVING COUNT(D.DEPENDENT_NAME) >= 2;
+--------+--------+
| FNAME  | LNAME  |
+--------+--------+
| Alice  | Brown  |
| Bob    | Green  |
+--------+--------+

Query OK, 2 rows affected (0.04 sec)
```

VI. List the names of managers who have at least one dependent



```
mysql> SELECT DISTINCT E.FNAME, E.LNAME
       FROM EMPLOYEE E
       JOIN DEPARTMENT D ON E.SSN = D.MGRSSN
       JOIN DEPENDENT DP ON E.SSN = DP.ESSN;
+--------+--------+
| FNAME  | LNAME  |
+--------+--------+
| Jane   | Doe    |
+--------+--------+

Query OK, 1 row affected (0.03 sec)
```

VII. Retrieve the names of employees who have no dependents.



```
mysql> SELECT E.FNAME, E.LNAME
       FROM EMPLOYEE E
       WHERE NOT EXISTS (
         SELECT *
         FROM DEPENDENT D
         WHERE D.ESSN = E.SSN
       );
+--------+--------+
| FNAME  | LNAME  |
+--------+--------+
| John   | Black  |
| Bob    | White  |
+--------+--------+

Query OK, 2 rows affected (0.05 sec)
```

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

A view in SQL is a virtual table that is created by a SELECT query on one or more tables. It doesn't store the data itself but instead shows the result of a predefined query every time the view is accessed. The view can be used like a regular table for querying but does not physically store data.

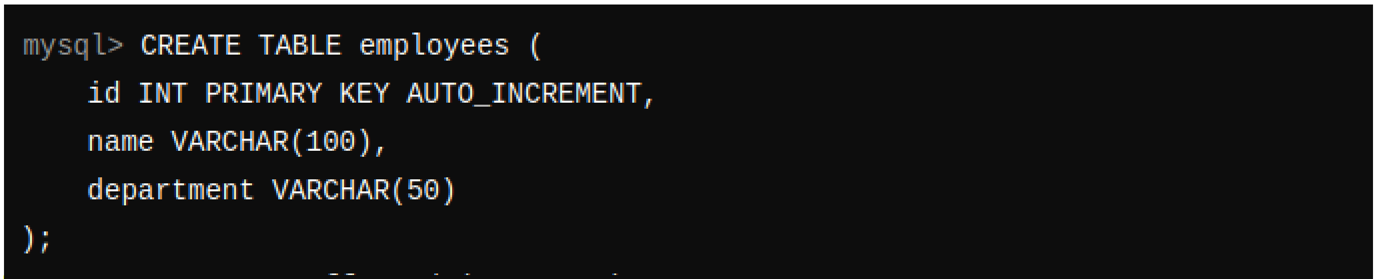
Advantages of Views:

1. Simplicity: Views simplify complex queries by encapsulating them within a single object. Users can query the view without needing to know the underlying tables or the complexity of the original query.
2. Security: Views can be used to limit access to specific columns or rows in a table. For example, sensitive information can be hidden from certain users by creating a view that only displays non-sensitive data.
3. Data Abstraction: Views provide a layer of abstraction over the physical schema. If the underlying schema changes (e.g., columns are added or renamed), the view can remain the same, offering stability for applications that depend on it.
4. Reusability: Views allow you to reuse complex SQL logic across multiple queries. Once a view is defined, it can be accessed in multiple queries without repeating the complex logic.
5. Maintenance: Views can simplify database maintenance by reducing redundancy in query logic, ensuring that changes to businss rules need to be updated only once (within the view) instead of in multiple queries.

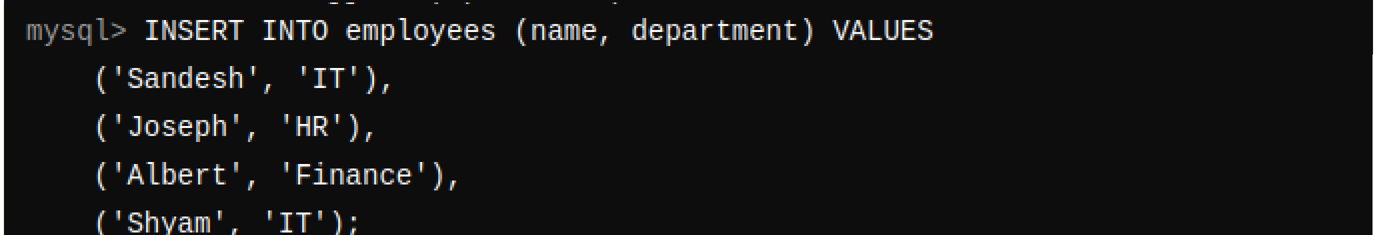
Creating a View for Employees in the IT Department

Employee (id, name, department, salary)

→ Create table and insert data

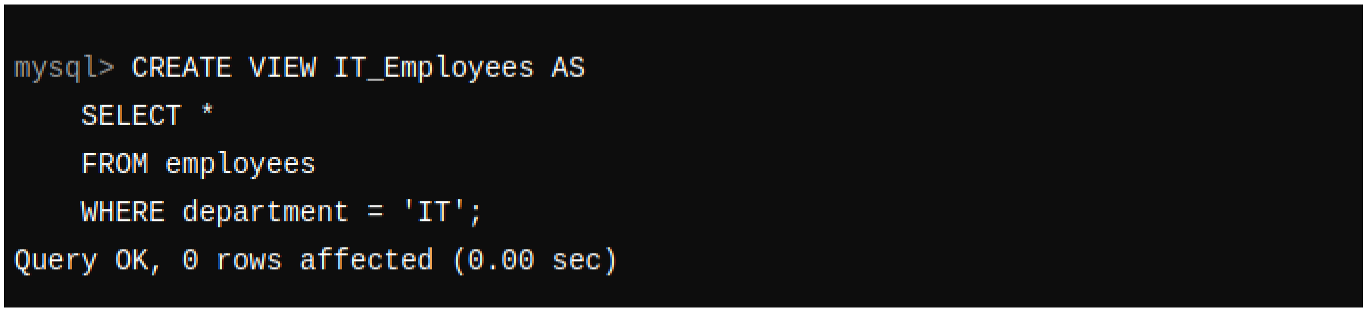


```
mysql> CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    department VARCHAR(50)
);
```



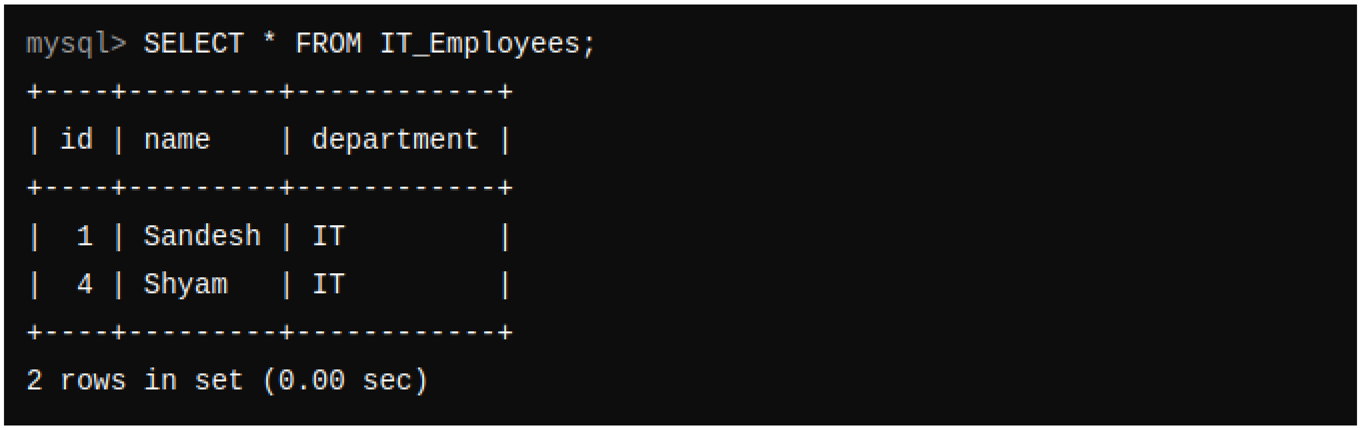
```
mysql> INSERT INTO employees (name, department) VALUES
    ('Sandesh', 'IT'),
    ('Joseph', 'HR'),
    ('Albert', 'Finance'),
    ('Shyam', 'IT');
```

3) Create a view



```
mysql> CREATE VIEW IT_Employees AS
    SELECT *
    FROM employees
    WHERE department = 'IT';
Query OK, 0 rows affected (0.00 sec)
```

4) Query the view



```
mysql> SELECT * FROM IT_Employees;
+----+---------+------------+
| id | name    | department |
+----+---------+------------+
|  1 | Sandesh | IT         |
|  4 | Shyam   | IT         |
+----+---------+------------+
2 rows in set (0.00 sec)
```