4/12/2024

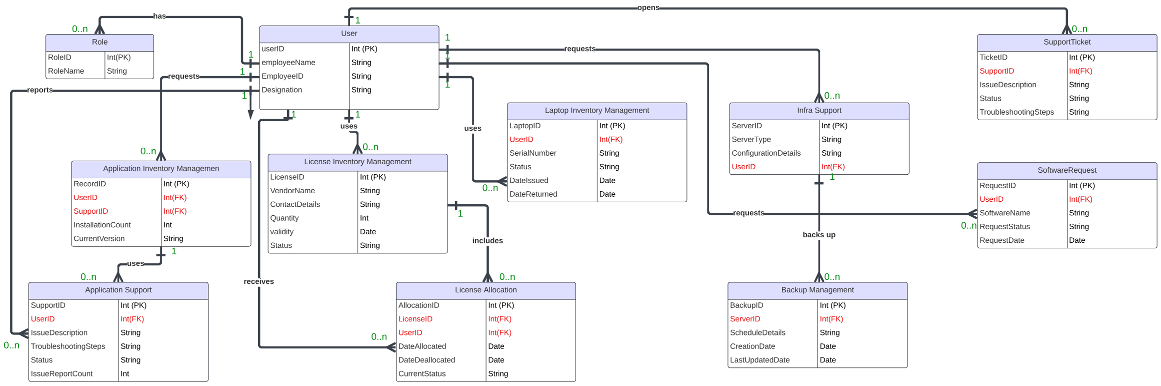
# Team 2 Project Phase 2 Submission

Maria Issac, Maha Mansoor & Siva Alaparthi
UNIVERISY OF HOUSTON DOWNTOWN

# Complete E/R diagram

Base on your mission objectives and use cases, create an E/R diagram for your database. Clearly list the attributes and the primary key for each entity, and each relationship with its multiplicities. ALL ENTITIES MUST BE DIRECTLY OR INDIRECTLY RELATED. EACH ENTITY HAS AT MOST THREE RELAIONSHIPS.

IT Applications Database system



# Relational Model

Using the method for translating an E/R diagram to relations, produce a set of tables for your database. For each table, specify the attributes, their domains, required data constraints, default values, primary key, candidate keys, foreign keys and the tables which the foreign keys are referencing. Use the CREATE TABLE statement to describe the above information for every table.

## Tables:

| **User** | | | | | | |
|---|---|---|---|---|---|---|
| **Attribute** | **Domain** | **Constraints** | **Defaul t** | **Primary Key** | **Foreign Keys** | **References** |
| UserID | INT | NOT NULL | None | Yes | None | None |
| EmployeeName | VARCHAR(255) | NOT NULL | None | No | None | None |
| EmployeeID | VARCHAR(255) | NOT NULL, UNIQUE | None | No | None | None |
| Designation | VARCHAR(255) | | None | No | None | None |

| **Roles** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Attribute** | **Domain** | **Constraints** | **Default** | **Primary Key** | **Candidate Keys** | **Foreign Keys** | **References** |
| RoleID | INT | NOT NULL | | Yes | RoleID | | |
| RoleName | VARCHAR(255) | NOT NULL | | | RoleName | | |

**Application Inventory Management Table**

| Attribute | Domain | Constraints | Default | Primary Key | Foreign Keys | References |
|---|---|---|---|---|---|---|
| RecordID | INT | NOT NULL | None | Yes | None | None |
| UserID | INT |  | None | No | UserID | Users(UserID) |
| SupportID | INT |  | None | No | SupportID | Application_Support(SupportID) |
| InstallationCount | INT |  | 0 | No | None | None |
| CurrentVersion | VARCHAR(50) |  | None | No | None | None |

**Application Support Table**

| Attribute | Domain | Constraints | Default | Primary Key | Foreign Keys | References |
|---|---|---|---|---|---|---|
| **SupportID** | **INT** | **NOT NULL** | **None** | **Yes** | **None** | **None** |
| **UserID** | **INT** |  | **None** | **No** | **UserID** | **Users(UserID)** |
| **IssueDescription** | **TEXT** |  | **None** | **No** | **None** | **None** |
| **TroubleshootingSteps** | **TEXT** |  | **None** | **No** | **None** | **None** |
| **Status** | **VARCHAR(50)** |  | **None** | **No** | **None** | **None** |
| **IssueReportCount** | **INT** |  | **0** | **No** | **None** | **None** |

**License Inventory Management Table**

| Attribute | Domain | Constraints | Default | Primary Key | Foreign Keys | References |
|---|---|---|---|---|---|---|
| **LicenseID** | **INT** | **NOT NULL** | **None** | **Yes** | **None** | **None** |
| **LicenseName** | **VARCHAR(255)** | **NOT NULL** | **None** | **No** | **None** | **None** |
| **VendorName** | **VARCHAR(255)** |  | **None** | **No** | **None** | **None** |
| **ContactDetails** | **VARCHAR(255)** |  | **None** | **No** | **None** | **None** |
| **Quantity** | **INT** | **CHECK (Quantity >= 0)** | **None** | **No** | **None** | **None** |
| **Validity** | **DATE** |  | **None** | **No** | **None** | **None** |
| **Status** | **VARCHAR(50)** |  | **None** | **No** | **None** | **None** |

**License Allocation Table**

| Attribute | Domain | Constraints | Default | Primary Key | Foreign Keys | References |
|---|---|---|---|---|---|---|
| AllocationID | INT | NOT NULL | None | Yes | None | None |
| LicenseID | INT | | None | No | LicenseID | License_Inventory_Management(LicenseID) |
| UserID | INT | | None | No | UserID | Users(UserID) |
| DateAllocated | DATE | | None | No | None | None |
| DateDeallocated | DATE | | None | No | None | None |
| CurrentStatus | VARCHAR(50) | | None | No | None | None |

**Laptop Inventory Management Table**

| Attribute | Domain | Constraints | Default | Primary Key | Foreign Keys | References |
|---|---|---|---|---|---|---|
| LaptopID | INT | NOT NULL | None | Yes | None | None |
| UserID | INT | | None | No | UserID | Users(UserID) |
| SerialNumber | VARCHAR(255) | NOT NULL, UNIQUE | None | No | None | None |
| Status | VARCHAR(50) | | None | No | None | None |
| DateIssued | DATE | | None | No | None | None |
| DateReturned | DATE | | None | No | None | None |

**Infra Support Table**

| Attribute | Domain | Constraints | Default | Primary Key | Foreign Keys | References |
|---|---|---|---|---|---|---|
| ServerID | INT | NOT NULL | None | Yes | None | None |
| ServerType | VARCHAR(255) | | None | No | None | None |
| ConfigurationDetails | TEXT | | None | No | None | None |
| UserID | INT | | None | No | UserID | Users(UserID) |

**Backup Management Table**

| Attribute | Domain | Constraints | Default | Primary Key | Foreign Keys | References |
|---|---|---|---|---|---|---|
| BackupID | INT | NOT NULL | None | Yes | None | None |
| ServerID | INT | | None | No | ServerID | Infra_Support(ServerID) |
| ScheduleDetails | TEXT | | None | No | None | None |
| CreationDate | DATE | | None | No | None | None |
| LastUpdatedDate | DATE | | None | No | None | None |

| Support Ticket Table | | | | | | |
|---|---|---|---|---|---|---|
| **Attribute** | **Domain** | **Constraints** | **Default** | **Primary Key** | **Foreign Keys** | **References** |
| **TicketID** | **INT** | **NOT NULL** | **None** | **Yes** | **None** | **None** |
| **SupportID** | **INT** | | **None** | **No** | **SupportID** | **Application_Support(SupportID)** |
| **IssueDescription** | **TEXT** | | **None** | **No** | **None** | **None** |
| **Status** | **VARCHAR(50)** | | **None** | **No** | **None** | **None** |
| **TroubleshootingSteps** | **TEXT** | | **None** | **No** | **None** | **None** |

| Software Request Table | | | | | | |
|---|---|---|---|---|---|---|
| **Attribute** | **Domain** | **Constraints** | **Default** | **Primary Key** | **Foreign Keys** | **References** |
| **RequestID** | **INT** | **NOT NULL** | **None** | **Yes** | **None** | **None** |
| **UserID** | **INT** | | **None** | **No** | **UserID** | **Users(UserID)** |
| **SoftwareName** | **VARCHAR(255)** | | **None** | **No** | **None** | **None** |
| **RequestStatus** | **VARCHAR(50)** | | **None** | **No** | **None** | **None** |
| **RequestDate** | **DATE** | | **None** | **No** | **None** | **None** |

## Queries to create the tables.

### User table

```
CREATE TABLE Users (
    UserID INT NOT NULL,
    EmployeeName VARCHAR(255) NOT NULL,
    EmployeeID VARCHAR(255) NOT NULL UNIQUE,
    Designation VARCHAR(255),
    PRIMARY KEY (UserID)
);
```

### Roles table

```
CREATE TABLE Roles (
    RoleID INT NOT NULL,
    RoleName VARCHAR(255) NOT NULL,
    PRIMARY KEY (RoleID)
);
```

### Application Inventory Management table

```
CREATE TABLE Application_Inventory_Management (
    RecordID INT NOT NULL,
    UserID INT,
    SupportID INT,
    InstallationCount INT DEFAULT 0,
    CurrentVersion VARCHAR(50),
    PRIMARY KEY (RecordID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (SupportID) REFERENCES Application_Support(SupportID)
);
```

### Application Support table

```
CREATE TABLE Application_Support (
    SupportID INT NOT NULL,
    UserID INT,
    IssueDescription TEXT,
    TroubleshootingSteps TEXT,
    Status VARCHAR(50),
    IssueReportCount INT DEFAULT 0,
    PRIMARY KEY (SupportID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

### License Inventory Management table

```
CREATE TABLE License_Inventory_Management (
    LicenseID INT NOT NULL,
    LicenseName VARCHAR(255) NOT NULL,
    VendorName VARCHAR(255),
    ContactDetails VARCHAR(255),
    Quantity INT CHECK (Quantity >= 0),
    Validity DATE,
    Status VARCHAR(50),
    PRIMARY KEY (LicenseID)
);
```

### License allocation table

```
CREATE TABLE License_Allocation (
    AllocationID INT NOT NULL,
    LicenseID INT,
    UserID INT,
    DateAllocated DATE,
    DateDeallocated DATE,
    CurrentStatus VARCHAR(50),
    PRIMARY KEY (AllocationID),
    FOREIGN KEY (LicenseID) REFERENCES License_Inventory_Management(LicenseID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

### Laptop Inventory Mangement table

```
CREATE TABLE Laptop_Inventory_Management (
    LaptopID INT NOT NULL,
    UserID INT,
    SerialNumber VARCHAR(255) NOT NULL UNIQUE,
    Status VARCHAR(50),
    DateIssued DATE,
    DateReturned DATE,
    PRIMARY KEY (LaptopID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

**Infra Support table**

```
CREATE TABLE Infra_Support (
    ServerID INT NOT NULL,
    ServerType VARCHAR(255),
    ConfigurationDetails TEXT,
    UserID INT,
    PRIMARY KEY (ServerID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

**Backup management table**

```
CREATE TABLE Backup_Management (
    BackupID INT NOT NULL,
    ServerID INT,
    ScheduleDetails TEXT,
    CreationDate DATE,
    LastUpdatedDate DATE,
    PRIMARY KEY (BackupID),
    FOREIGN KEY (ServerID) REFERENCES Infra_Support(ServerID)
);
```

**Support ticket table**

```
CREATE TABLE Support_Ticket (
    TicketID INT NOT NULL,
    SupportID INT,
    IssueDescription TEXT,
    Status VARCHAR(50),
    TroubleshootingSteps TEXT,
    PRIMARY KEY (TicketID),
    FOREIGN KEY (SupportID) REFERENCES Application_Support(SupportID)
);
```

**Software request table**

```
CREATE TABLE Software_Request (
    RequestID INT NOT NULL,
    UserID INT,
    SoftwareName VARCHAR(255),
    RequestStatus VARCHAR(50),
    RequestDate DATE,
    PRIMARY KEY (RequestID),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

## Complete List of Use Cases and Realization (40 points)

List all actors (i.e. users) of your database
Enhance the use cases as follows: For each entity, you must have use cases that perform at least one aggregate query, one insert operation, one delete operation, and one update operation; for each relationship, you must have use cases that perform at least one joint query. (Number your use cases. That's a minimum of 34 use cases for 7 entities and 2 person team, and 44 use cases for 9 entities and 3 person team)
Under each use case description, write down the complete SQL statement(s) needed to realize the use case.

The actors for the database are

1. HR
2. User (Business User)
3. Manager
4. Team Lead
5. Support Staff

**Use Case #1: Insert a New User**
Actor: HR
Description: Add a new user to the system.
Steps: HR inputs the user's name, employee ID, and designation and submits the form.
SQL Example: INSERT INTO Users (EmployeeName, EmployeeID, Designation) VALUES ('Siva', A123, 'Support');

**Use Case #2: Delete a User**
Actor: HR
Description: Remove a user from the system.
Steps: HR selects a user based on the employee ID and deletes the user.
SQL Example: DELETE FROM Users WHERE EmployeeID = A123;

**Use Case #3: Update User Information**
Actor: HR
Description: Update details of an existing user.
Steps: HR selects a user and updates their designation.
SQL Example: UPDATE Users SET Designation = 'Senior Support' WHERE EmployeeID = A123;

**Use Case #4: Aggregate Query on Users**
Actor: Manager
Description: Count the total number of users in each designation.
Steps: Manager requests a report on the count of users by designation.
SQL Example: SELECT Designation, COUNT(*) FROM Users GROUP BY Designation;

**Use Case #5: List All Users**
Actor: Manager
Description: Retrieve a list of all users in the system.
Steps: Manager accesses the user directory.
SQL Example: SELECT * FROM Users;

**Use Case #6: Find Users by Designation**
Actor: HR
Description: Search for users by their designation.
Steps: HR inputs the designation to search.
SQL Example: SELECT * FROM Users WHERE Designation = 'Support';

**Use Case #7: Insert a New Role**
Actor: HR
Description: Add a new role to the system.
Steps: HR inputs the role name and submits the form.
SQL Example: INSERT INTO Roles (RoleName) VALUES ('Consultant');

**Use Case #8: Delete a Role**
Actor: HR
Description: Remove a role from the system.
Steps: HR selects a role and deletes it.
SQL Example: DELETE FROM Roles WHERE RoleName = 'Consultant';

**Use Case #9: Update Role Information**
Actor: HR
Description: Update the name of an existing role.
Steps: HR selects a role and updates its name.
SQL Example: UPDATE Roles SET RoleName = 'Senior Consultant' WHERE RoleName = 'Consultant';

**Use Case #10: Aggregate Query on Roles**
Actor: Manager
Description: Count the number of users in each role.
Steps: Manager requests a report on the number of users assigned to each role.
SQL Example: SELECT RoleName, COUNT(UserID) FROM Users JOIN Roles ON Users.UserID = Roles.RoleID GROUP BY RoleName;

**Use Case #11: List All Roles**
Actor: HR
Description: Retrieve all roles available in the system.
Steps: HR accesses the roles directory.
SQL Example: SELECT * FROM Roles;

**Use Case #12: Insert a New Application Record**
Actor: Support
Description: Add a new application inventory record.
Steps: Support staff inputs details and submits.
SQL Example: INSERT INTO Application_Inventory_Management (UserID, SupportID, InstallationCount, CurrentVersion) VALUES (1, 2, 25, 'v3.0');

**Use Case #13: Delete an Application Record**
Actor: Support
Description: Remove an application inventory record.
Steps: Support staff selects a record and deletes it.
SQL Example: DELETE FROM Application_Inventory_Management WHERE RecordID = 101;

**Use Case #14: Update Application Record**
Actor: Support
Description: Update details of an existing application record.
Steps: Support staff selects a record and updates the version.
SQL Example: UPDATE Application_Inventory_Management SET CurrentVersion = 'v3.1' WHERE RecordID = 101;

**Use Case #15: Count Applications by Version**
Actor: manager
Description: Aggregate query to count applications by their versions.
Steps: manager runs a report to see distribution of application versions.
SQL Example: SELECT CurrentVersion, COUNT(*) FROM Application_Inventory_Management GROUP BY CurrentVersion;

**Use Case #16: Insert New License**
Actor: Support
Description: Add a new license to the inventory.
Steps: Support inputs license details and submits.
SQL Example: INSERT INTO License_Inventory_Management (LicenseName, VendorName, ContactDetails, Quantity, Validity, Status) VALUES ('New Software', 'NewVendor', 'contact@newvendor.com', 50, '2025-12-31', 'Active');

**Use Case #17: Update License Information**
Actor: Support
Description: Update existing license details.
Steps: Support selects a license and updates its details.
SQL Example: UPDATE License_Inventory_Management SET Quantity = 60 WHERE LicenseID = 2;

**Use Case #18: Delete a License**
Actor: support
Description: Remove a license from the inventory.
Steps: support selects a license and deletes it.
SQL Example: DELETE FROM License_Inventory_Management WHERE LicenseID = 2;

**Use Case #19: Aggregate Query for License Expiry**
Actor: support
Description: Count licenses expiring in a given year.
Steps: support queries for licenses expiring within the year.
SQL Example: SELECT COUNT(*) FROM License_Inventory_Management WHERE YEAR(Validity) = 2025;

**Use Case #20: Allocate License to User**
Actor: support
Description: Assign a license to a user.
Steps: support selects a license and user for allocation.
SQL Example: INSERT INTO License_Allocation (LicenseID, UserID, DateAllocated, CurrentStatus) VALUES (1, 1, CURRENT_DATE, 'Active');

**Use Case #21: Deallocate License from User**
Actor: support
Description: Remove a license from a user.
Steps: support selects the allocation and deallocates it.
SQL Example: UPDATE License_Allocation SET CurrentStatus = 'Inactive', DateDeallocated = CURRENT_DATE WHERE AllocationID = 1;

**Use Case #22: Update License Allocation Status**
Actor: support
Description: Update the status of a license allocation.
Steps: support updates the status of an allocated license.
SQL Example: UPDATE License_Allocation SET CurrentStatus = 'Suspended' WHERE AllocationID = 1;

**Use Case #23: Count Active Licenses**
Actor: support
Description: Count how many licenses are currently active.
Steps: support queries for active licenses.
SQL Example: SELECT COUNT(*) FROM License_Allocation WHERE CurrentStatus = 'Active';

**Use Case #24: Insert Laptop Inventory Record**
Actor: Support
Description: Add a new laptop to the inventory.
Steps: Support inputs the laptop details.
SQL Example: INSERT INTO Laptop_Inventory_Management (UserID, SerialNumber, Status, DateIssued) VALUES (1, 'SN001', 'Issued', CURRENT_DATE);

**Use Case #25: Update Laptop Status**
Actor: Support
Description: Update the status of a laptop in inventory.
Steps: Support selects a laptop and updates its status.
SQL Example: UPDATE Laptop_Inventory_Management SET Status = 'Returned', DateReturned = CURRENT_DATE WHERE LaptopID = 1;

**Use Case #26: Delete Laptop Record**
Actor: Supportf
Description: Remove a laptop from the inventory.
Steps: Support selects a laptop and removes it from inventory.
SQL Example: DELETE FROM Laptop_Inventory_Management WHERE LaptopID = 1;

**Use Case #27: Aggregate Query for Laptop Status**
Actor: support
Description: Count laptops by their status.
Steps: support requests a count of laptops based on status.
SQL Example: SELECT Status, COUNT(*) FROM Laptop_Inventory_Management GROUP BY Status;

**Use Case #28: Add New Server**
Actor: support
Description: Add a new server to the infrastructure support database.
Steps: support enters server details and submits.
SQL Example: INSERT INTO Infra_Support (ServerType, ConfigurationDetails, UserID) VALUES ('Database Server', '16 Core, 64GB RAM', 2);

**Use Case #29: Update Server Configuration**
Actor: Support
Description: Update the configuration details of an existing server.
Steps: Support selects a server and updates its configuration details.
SQL Example: UPDATE Infra_Support SET ConfigurationDetails = '32 Core, 128GB RAM' WHERE ServerID = 1;

**Use Case #30: Delete Server Record**
Actor: Support
Description: Remove a server from the infrastructure support database.
Steps: Support selects a server and deletes its record.
SQL Example: DELETE FROM Infra_Support WHERE ServerID = 1;

**Use Case #31: List All Servers**
Actor: Support
Description: Retrieve a list of all servers in the infrastructure.
Steps: Support Staff views all server records.
SQL Example: SELECT * FROM Infra_Support;

**Use Case #32: Schedule New Backup**
Actor: Support
Description: Create a new backup schedule for a server.
Steps: Support enters backup details for a server and submits.
SQL Example: INSERT INTO Backup_Management (ServerID, ScheduleDetails, CreationDate) VALUES (1, 'Weekly', CURRENT_DATE);

**Use Case #33: Update Backup Schedule**
Actor: Support
Description: Update an existing backup schedule's details.
Steps: Support selects a backup record and updates its schedule.
SQL Example: UPDATE Backup_Management SET ScheduleDetails = 'Bi-Weekly' WHERE BackupID = 1;

**Use Case #34: Delete Backup Schedule**
Actor: Support
Description: Remove a backup schedule from the database.
Steps: Support selects a backup record and deletes it.
SQL Example: DELETE FROM Backup_Management WHERE BackupID = 1;

**Use Case #35: Aggregate Query for Backup Frequency**
Actor: Support
Description: Count backup schedules by their frequency.
Steps: Support queries for the count of backup schedules based on their frequency.
SQL Example: SELECT ScheduleDetails, COUNT(*) FROM Backup_Management GROUP BY ScheduleDetails;

**Use Case #36: Create Support Ticket**
Actor: User
Description: Open a new support ticket for an issue.
Steps: User enters issue details into the support system.
SQL Example: INSERT INTO Support_Ticket (SupportID, IssueDescription, Status, TroubleshootingSteps) VALUES (1, 'Cannot connect to network', 'Open', 'Restart router');

**Use Case #37: Update Support Ticket**
Actor: Support
Description: Update the status and troubleshooting steps of an existing ticket.
Steps: Support selects a ticket and updates its details.
SQL Example: UPDATE Support_Ticket SET Status = 'Closed', TroubleshootingSteps = 'Router reset successful' WHERE TicketID = 1;

### Use Case #38: Close Support Ticket

Actor: Support
Description: Close an open support ticket after resolving the issue.
Steps: Support selects a ticket and marks it as closed.
SQL Example: UPDATE Support_Ticket SET Status = 'Closed' WHERE TicketID = 1;

### Use Case #39: Count Open Tickets

Actor: Manager
Description: Count the number of open support tickets.
Steps: Manager requests a report on the count of open tickets.
SQL Example: SELECT COUNT(*) FROM Support_Ticket WHERE Status = 'Open';

### Use Case #40: Submit Software Request

Actor: User
Description: Request new software for installation.
Steps: User fills out a form with software details and submits.
SQL Example: INSERT INTO Software_Request (UserID, SoftwareName, RequestStatus, RequestDate) VALUES (2, 'Adobe Photoshop', 'Pending', CURRENT_DATE);

### Use Case #41: Approve Software Request

Actor: Support
Description: Approve a pending software request.
Steps: Support reviews the request and approves it.
SQL Example: UPDATE Software_Request SET RequestStatus = 'Approved' WHERE RequestID = 1;

### Use Case #42: Deny Software Request

Actor: Support
Description: Deny a software request.
Steps: Support reviews the request and denies it.
SQL Example: UPDATE Software_Request SET RequestStatus = 'Denied' WHERE RequestID = 1;

### Use Case #43: Aggregate Query for Software Requests

Actor: Support
Description: Count software requests by status.
Steps: Support queries the number of software requests based on their approval status.
SQL Example: SELECT RequestStatus, COUNT(*) FROM Software_Request GROUP BY RequestStatus;

### Use Case #44: Delete Software Request

Actor: Support
Description: Remove a software request from the system.
Steps: Support selects a request and deletes it.
SQL Example: DELETE FROM Software_Request WHERE RequestID = 1;

### Use Case #45: List All Software Requests

Actor: Manager
Description: Retrieve a list of all software requests.
Steps: Manager views all software request records.
SQL Example: SELECT * FROM Software_Request;

# Use cases involving Joint operations.

**Use Case #46: Joint Query Between Users and Roles**
Actor: Manager
Description: Retrieve all users and their roles.
Steps: Manager requests a list of all users along with their respective roles.
SQL Example: SELECT Users.EmployeeName, Roles.RoleName FROM Users JOIN Roles ON Users.UserID = Roles.RoleID;

**Use Case #47: Joint Query Between Users and Application Inventory**
Actor: Support
Description: Retrieve all applications requested by a specific user.
Steps: Support selects a user and views their application requests.
SQL Example: SELECT Users.EmployeeName, Application_Inventory_Management.CurrentVersion FROM Users JOIN Application_Inventory_Management ON Users.UserID = Application_Inventory_Management.UserID WHERE Users.UserID = 1;

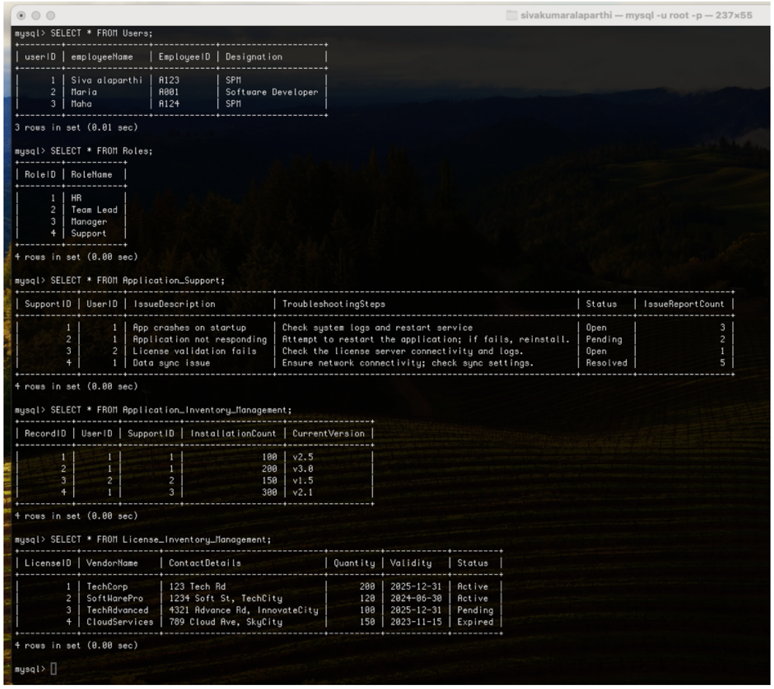
**Use Case #48: Joint Query Between Application Inventory and Support**
Actor: Support
Description: Get details of support linked with each application.
Steps: Support reviews support details for applications.
SQL Example: SELECT Application_Inventory_Management.RecordID, Application_Support.IssueDescription FROM Application_Inventory_Management JOIN Application_Support ON Application_Inventory_Management.SupportID = Application_Support.SupportID;

**Use Case #49: Joint Query Between Users and Software Requests**
Actor: Manager
Description: Get all software requests made by a specific user.
Steps: Manager selects a user and retrieves their software requests.
SQL Example: SELECT Users.EmployeeName, Software_Request.SoftwareName FROM Users JOIN Software_Request ON Users.UserID = Software_Request.UserID WHERE Users.UserID = 1;

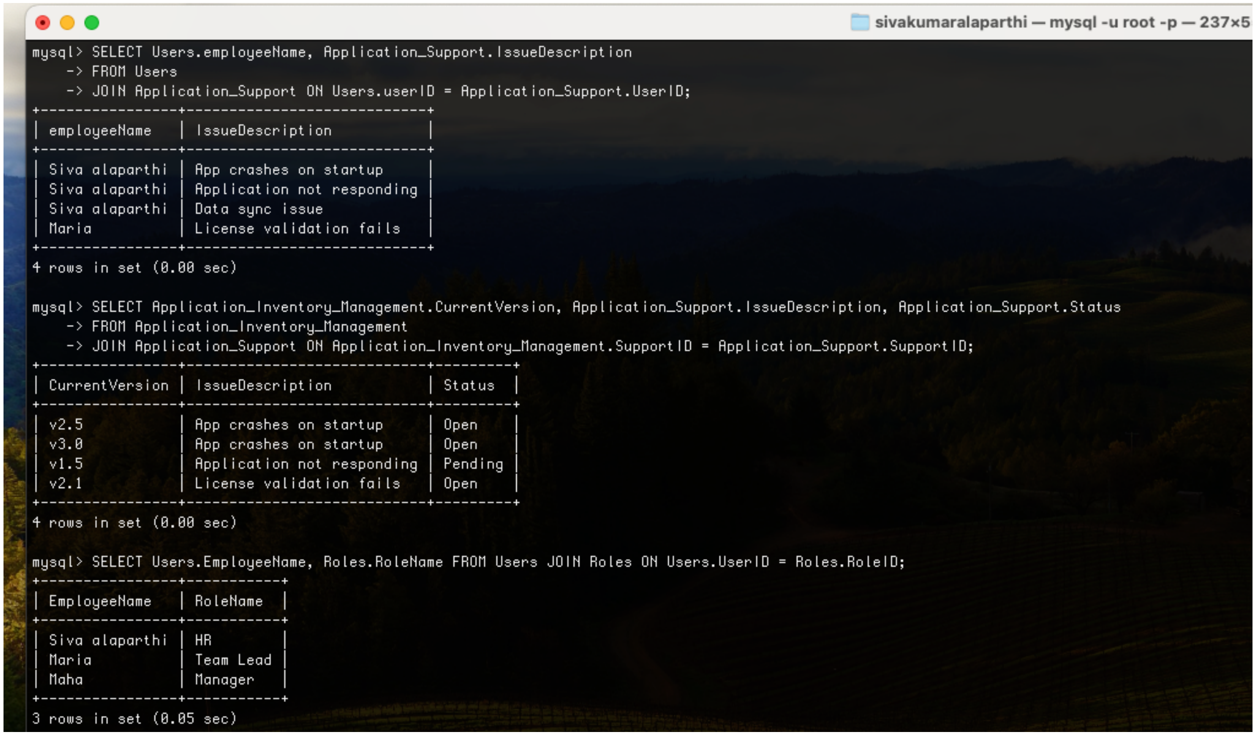
**Use Case #50: Detailed Report of Software Requests and User Information**
Actor: Support
Description: Generate a detailed report linking user profiles with their software requests.
Steps: Support queries for detailed information on all software requests, including user details.
SQL Example: SELECT Users.EmployeeName, Users.Designation, Software_Request.SoftwareName, Software_Request.RequestStatus FROM Users JOIN Software_Request ON Users.UserID = Software_Request.UserID;

Prototype
Install a multi-user DBMS of your choice, create at least three tables that are related in your relational model and insert some sample data.
Show all the data in every table, and at least two joint queries. Make a screen capture of the outputs.

The below screen shot shows the data in the users, roles, Application support, application inventory management and license inventory management tables



```
mysql> SELECT * FROM Users;
+--------+----------------+------------+--------------------+
| userID | employeeName   | EmployeeID | Designation        |
+--------+----------------+------------+--------------------+
|      1 | Siva alaparthi | A123       | SPM                |
|      2 | Maria          | A001       | Software Developer |
|      3 | Maha           | A124       | SPM                |
+--------+----------------+------------+--------------------+
3 rows in set (0.01 sec)

mysql> SELECT * FROM Roles;
+--------+-----------+
| RoleID | RoleName  |
+--------+-----------+
|      1 | HR        |
|      2 | Team Lead |
|      3 | Manager   |
|      4 | Support   |
+--------+-----------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM Application_Support;
+-----------+--------+----------------------------+----------------------------------------------------------+----------+------------------+
| SupportID | UserID | IssueDescription           | TroubleshootingSteps                                     | Status   | IssueReportCount |
+-----------+--------+----------------------------+----------------------------------------------------------+----------+------------------+
|         1 |      1 | App crashes on startup     | Check system logs and restart service                    | Open     |                3 |
|         2 |      1 | Application not responding | Attempt to restart the application; if fails, reinstall. | Pending  |                2 |
|         3 |      2 | License validation fails   | Check the license server connectivity and logs.          | Open     |                1 |
|         4 |      1 | Data sync issue            | Ensure network connectivity; check sync settings.        | Resolved |                5 |
+-----------+--------+----------------------------+----------------------------------------------------------+----------+------------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM Application_Inventory_Management;
+----------+--------+-----------+-------------------+----------------+
| RecordID | UserID | SupportID | InstallationCount | CurrentVersion |
+----------+--------+-----------+-------------------+----------------+
|        1 |      1 |         1 |               100 | v2.5           |
|        2 |      1 |         1 |               200 | v3.0           |
|        3 |      2 |         2 |               150 | v1.5           |
|        4 |      1 |         3 |               300 | v2.1           |
+----------+--------+-----------+-------------------+----------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM License_Inventory_Management;
+-----------+---------------+-------------------------------+----------+------------+---------+
| LicenseID | VendorName    | ContactDetails                | Quantity | Validity   | Status  |
+-----------+---------------+-------------------------------+----------+------------+---------+
|         1 | TechCorp      | 123 Tech Rd                   |      200 | 2025-12-31 | Active  |
|         2 | SoftWarePro   | 1234 Soft St, TechCity        |      120 | 2024-06-30 | Active  |
|         3 | TechAdvanced  | 4321 Advance Rd, InnovateCity |      100 | 2025-12-31 | Pending |
|         4 | CloudServices | 789 Cloud Ave, SkyCity        |      150 | 2023-11-15 | Expired |
+-----------+---------------+-------------------------------+----------+------------+---------+
4 rows in set (0.00 sec)

mysql> 
```



```
mysql> SELECT Users.employeeName, Application_Support.IssueDescription
    -> FROM Users
    -> JOIN Application_Support ON Users.userID = Application_Support.UserID;
+----------------+----------------------------+
| employeeName   | IssueDescription           |
+----------------+----------------------------+
| Siva alaparthi | App crashes on startup     |
| Siva alaparthi | Application not responding |
| Siva alaparthi | Data sync issue            |
| Maria          | License validation fails   |
+----------------+----------------------------+
4 rows in set (0.00 sec)

mysql> SELECT Application_Inventory_Management.CurrentVersion, Application_Support.IssueDescription, Application_Support.Status
    -> FROM Application_Inventory_Management
    -> JOIN Application_Support ON Application_Inventory_Management.SupportID = Application_Support.SupportID;
+----------------+----------------------------+---------+
| CurrentVersion | IssueDescription           | Status  |
+----------------+----------------------------+---------+
| v2.5           | App crashes on startup     | Open    |
| v3.0           | App crashes on startup     | Open    |
| v1.5           | Application not responding | Pending |
| v2.1           | License validation fails   | Open    |
+----------------+----------------------------+---------+
4 rows in set (0.00 sec)

mysql> SELECT Users.EmployeeName, Roles.RoleName FROM Users JOIN Roles ON Users.UserID = Roles.RoleID;
+----------------+-----------+
| EmployeeName   | RoleName  |
+----------------+-----------+
| Siva alaparthi | HR        |
| Maria          | Team Lead |
| Maha           | Manager   |
+----------------+-----------+
3 rows in set (0.05 sec)
```

The Joint queries are:

Query 1 - **Users and their Application Support Details**

SELECT Users.employeeName, Application_Support.IssueDescription
FROM Users
JOIN Application_Support ON Users.userID = Application_Support.UserID;

Query 2: **Application Inventory and Support Details**

SELECT Application_Inventory_Management.CurrentVersion, Application_Support.IssueDescription, Application_Support.Status
FROM Application_Inventory_Management
JOIN Application_Support ON Application_Inventory_Management.SupportID = Application_Support.SupportID;

Query 3 : **Joint Query Between Users and Roles**

SELECT Users.EmployeeName, Roles.RoleName FROM Users JOIN Roles ON Users.UserID = Roles.RoleID;