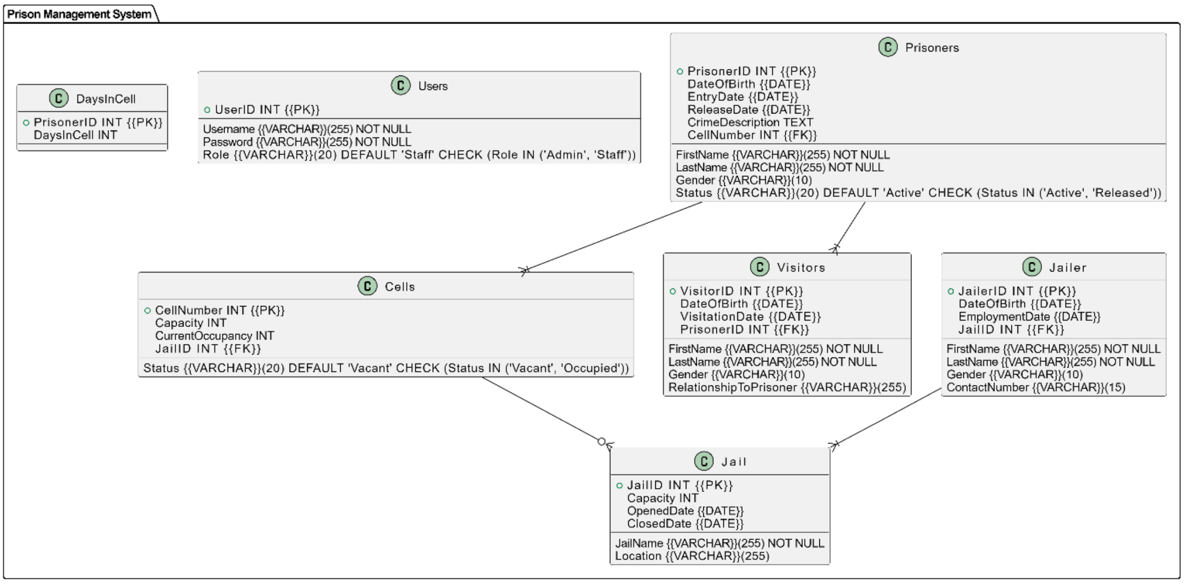
Sankey Solutions E-Internship 2024 Milestone 1

Question 1

1.1

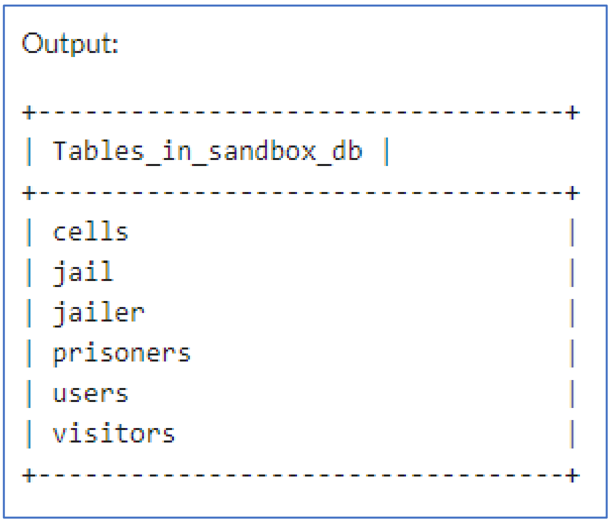
Schema: Prison Management System

Tool used: draw.io

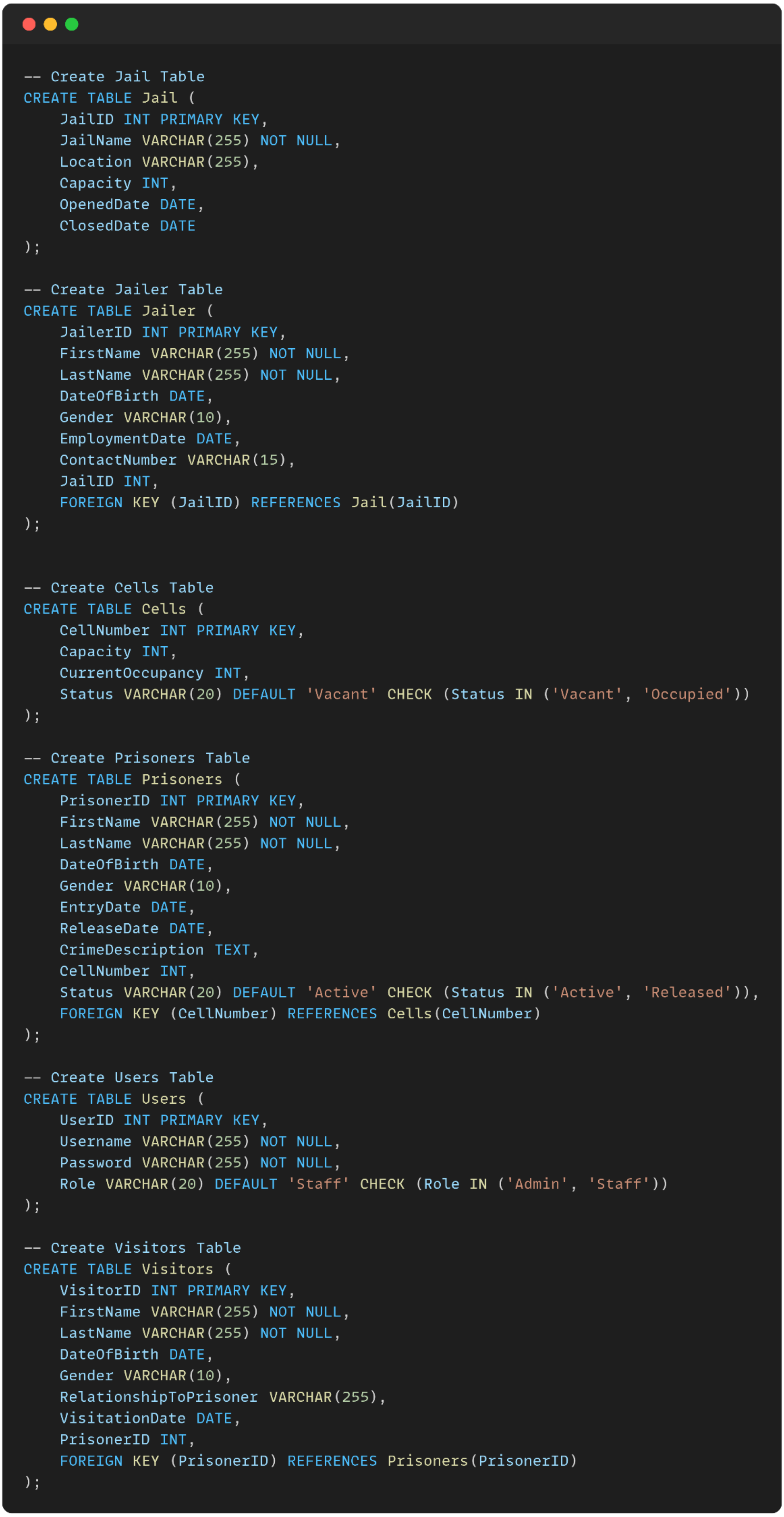

Prison Management System
DaysInCell
PrisonerID INT {{PK}}
DaysInCell INT
Users
UserID INT {{PK}}
Username {{VARCHAR}}(255) NOT NULL
Password {{VARCHAR}}(255) NOT NULL
Role {{VARCHAR}}(20) DEFAULT 'Staff' CHECK (Role IN ('Admin', 'Staff'))
Prisoners
PrisonerID INT {{PK}}
DateOfBirth {{DATE}}
EntryDate {{DATE}}
ReleaseDate {{DATE}}
CrimeDescription TEXT
CellNumber INT {{FK}}
FirstName {{VARCHAR}}(255) NOT NULL
LastName {{VARCHAR}}(255) NOT NULL
Gender {{VARCHAR}}(10)
Status {{VARCHAR}}(20) DEFAULT 'Active' CHECK (Status IN ('Active', 'Released'))
Cells
CellNumber INT {{PK}}
Capacity INT
CurrentOccupancy INT
JailID INT {{FK}}
Status {{VARCHAR}}(20) DEFAULT 'Vacant' CHECK (Status IN ('Vacant', 'Occupied'))
Visitors
VisitorID INT {{PK}}
DateOfBirth {{DATE}}
VisitationDate {{DATE}}
PrisonerID INT {{FK}}
FirstName {{VARCHAR}}(255) NOT NULL
LastName {{VARCHAR}}(255) NOT NULL
Gender {{VARCHAR}}(10)
RelationshipToPrisoner {{VARCHAR}}(255)
Jailer
JailerID INT {{PK}}
DateOfBirth {{DATE}}
EmploymentDate {{DATE}}
JailID INT {{FK}}
FirstName {{VARCHAR}}(255) NOT NULL
LastName {{VARCHAR}}(255) NOT NULL
Gender {{VARCHAR}}(10)
ContactNumber {{VARCHAR}}(15)
Jail
JailID INT {{PK}}
Capacity INT
OpenedDate {{DATE}}
ClosedDate {{DATE}}
JailName {{VARCHAR}}(255) NOT NULL
Location {{VARCHAR}}(255)

1.2 Creating the Tables

```
-- Create Jail Table
CREATE TABLE Jail (
    JailID INT PRIMARY KEY,
    JailName VARCHAR(255) NOT NULL,
    Location VARCHAR(255),
    Capacity INT,
    OpenedDate DATE,
    ClosedDate DATE
);

-- Create Jailer Table
CREATE TABLE Jailer (
    JailerID INT PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    DateOfBirth DATE,
    Gender VARCHAR(10),
    EmploymentDate DATE,
    ContactNumber VARCHAR(15),
    JailID INT,
    FOREIGN KEY (JailID) REFERENCES Jail(JailID)
);


-- Create Cells Table
CREATE TABLE Cells (
    CellNumber INT PRIMARY KEY,
    Capacity INT,
    CurrentOccupancy INT,
    Status VARCHAR(20) DEFAULT 'Vacant' CHECK (Status IN ('Vacant', 'Occupied'))
);

-- Create Prisoners Table
CREATE TABLE Prisoners (
    PrisonerID INT PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    DateOfBirth DATE,
    Gender VARCHAR(10),
    EntryDate DATE,
    ReleaseDate DATE,
    CrimeDescription TEXT,
    CellNumber INT,
    Status VARCHAR(20) DEFAULT 'Active' CHECK (Status IN ('Active', 'Released')),
    FOREIGN KEY (CellNumber) REFERENCES Cells(CellNumber)
);

-- Create Users Table
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Username VARCHAR(255) NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Role VARCHAR(20) DEFAULT 'Staff' CHECK (Role IN ('Admin', 'Staff'))
);

-- Create Visitors Table
CREATE TABLE Visitors (
    VisitorID INT PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    DateOfBirth DATE,
    Gender VARCHAR(10),
    RelationshipToPrisoner VARCHAR(255),
    VisitationDate DATE,
    PrisonerID INT,
    FOREIGN KEY (PrisonerID) REFERENCES Prisoners(PrisonerID)
);
```



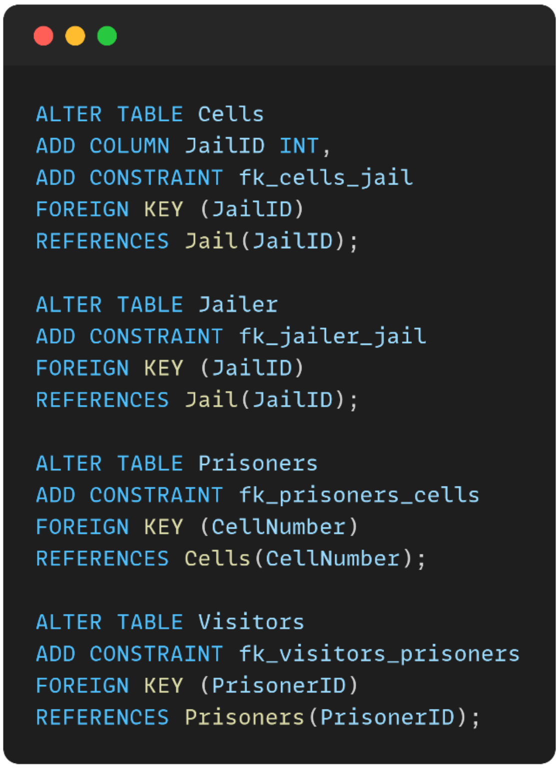
```
Output:

+-----------------------------------+
| Tables_in_sandbox_db |
+-----------------------------------+
| cells                             |
| jail                              |
| jailer                            |
| prisoners                         |
| users                             |
| visitors                          |
+-----------------------------------+
```



```sql
-- Create Jail Table
CREATE TABLE Jail (
    JailID INT PRIMARY KEY,
    JailName VARCHAR(255) NOT NULL,
    Location VARCHAR(255),
    Capacity INT,
    OpenedDate DATE,
    ClosedDate DATE
);

-- Create Jailer Table
CREATE TABLE Jailer (
    JailerID INT PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    DateOfBirth DATE,
    Gender VARCHAR(10),
    EmploymentDate DATE,
    ContactNumber VARCHAR(15),
    JailID INT,
    FOREIGN KEY (JailID) REFERENCES Jail(JailID)
);


-- Create Cells Table
CREATE TABLE Cells (
    CellNumber INT PRIMARY KEY,
    Capacity INT,
    CurrentOccupancy INT,
    Status VARCHAR(20) DEFAULT 'Vacant' CHECK (Status IN ('Vacant', 'Occupied'))
);

-- Create Prisoners Table
CREATE TABLE Prisoners (
    PrisonerID INT PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    DateOfBirth DATE,
    Gender VARCHAR(10),
    EntryDate DATE,
    ReleaseDate DATE,
    CrimeDescription TEXT,
    CellNumber INT,
    Status VARCHAR(20) DEFAULT 'Active' CHECK (Status IN ('Active', 'Released')),
    FOREIGN KEY (CellNumber) REFERENCES Cells(CellNumber)
);

-- Create Users Table
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Username VARCHAR(255) NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Role VARCHAR(20) DEFAULT 'Staff' CHECK (Role IN ('Admin', 'Staff'))
);

-- Create Visitors Table
CREATE TABLE Visitors (
    VisitorID INT PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    DateOfBirth DATE,
    Gender VARCHAR(10),
    RelationshipToPrisoner VARCHAR(255),
    VisitationDate DATE,
    PrisonerID INT,
    FOREIGN KEY (PrisonerID) REFERENCES Prisoners(PrisonerID)
);
```

1.3 Implementing relationships and Foreign keys

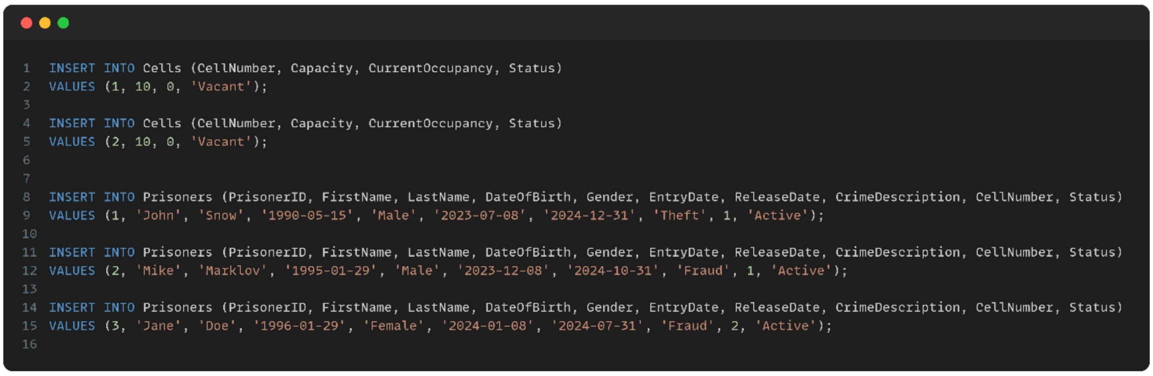
```
ALTER TABLE Cells
ADD COLUMN JailID INT,
ADD CONSTRAINT fk_cells_jail
FOREIGN KEY (JailID)
REFERENCES Jail(JailID);

ALTER TABLE Jailer
ADD CONSTRAINT fk_jailer_jail
FOREIGN KEY (JailID)
REFERENCES Jail(JailID);

ALTER TABLE Prisoners
ADD CONSTRAINT fk_prisoners_cells
FOREIGN KEY (CellNumber)
REFERENCES Cells(CellNumber);

ALTER TABLE Visitors
ADD CONSTRAINT fk_visitors_prisoners
FOREIGN KEY (PrisonerID)
REFERENCES Prisoners(PrisonerID);
```



```
ALTER TABLE Cells
ADD COLUMN JailID INT,
ADD CONSTRAINT fk_cells_jail
FOREIGN KEY (JailID)
REFERENCES Jail(JailID);

ALTER TABLE Jailer
ADD CONSTRAINT fk_jailer_jail
FOREIGN KEY (JailID)
REFERENCES Jail(JailID);

ALTER TABLE Prisoners
ADD CONSTRAINT fk_prisoners_cells
FOREIGN KEY (CellNumber)
REFERENCES Cells(CellNumber);

ALTER TABLE Visitors
ADD CONSTRAINT fk_visitors_prisoners
FOREIGN KEY (PrisonerID)
REFERENCES Prisoners(PrisonerID);
```

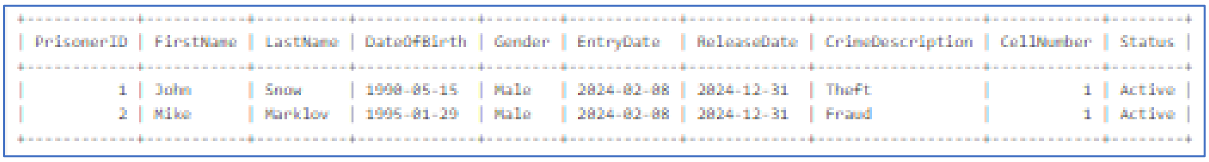
1.4 Query to add new prisoners.

```
INSERT INTO Cells (CellNumber, Capacity, CurrentOccupancy, Status)
VALUES (1, 10, 0, 'Vacant');

INSERT INTO Cells (CellNumber, Capacity, CurrentOccupancy, Status)
VALUES (2, 10, 0, 'Vacant');


INSERT INTO Prisoners (PrisonerID, FirstName, LastName, DateOfBirth, Gender,
EntryDate, ReleaseDate, CrimeDescription, CellNumber, Status)
VALUES (1, 'John', 'Snow', '1990-05-15', 'Male', '2023-07-08', '2024-12-31',
'Theft', 1, 'Active');

INSERT INTO Prisoners (PrisonerID, FirstName, LastName, DateOfBirth, Gender,
EntryDate, ReleaseDate, CrimeDescription, CellNumber, Status)
VALUES (2, 'Mike', 'Marklov', '1995-01-29', 'Male', '2023-12-08', '2024-10-31',
'Fraud', 1, 'Active');

INSERT INTO Prisoners (PrisonerID, FirstName, LastName, DateOfBirth, Gender,
EntryDate, ReleaseDate, CrimeDescription, CellNumber, Status)
VALUES (3, 'Jane', 'Doe', '1996-01-29', 'Female', '2024-01-08', '2024-07-31',
'Fraud', 2, 'Active');
```



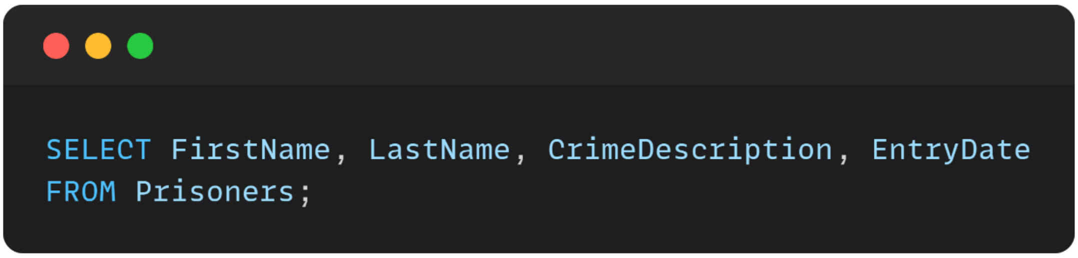
```
INSERT INTO Cells (CellNumber, Capacity, CurrentOccupancy, Status)
VALUES (1, 10, 0, 'Vacant');

INSERT INTO Cells (CellNumber, Capacity, CurrentOccupancy, Status)
VALUES (2, 10, 0, 'Vacant');


INSERT INTO Prisoners (PrisonerID, FirstName, LastName, DateOfBirth, Gender, EntryDate, ReleaseDate, CrimeDescription, CellNumber, Status)
VALUES (1, 'John', 'Snow', '1990-05-15', 'Male', '2023-07-08', '2024-12-31', 'Theft', 1, 'Active');

INSERT INTO Prisoners (PrisonerID, FirstName, LastName, DateOfBirth, Gender, EntryDate, ReleaseDate, CrimeDescription, CellNumber, Status)
VALUES (2, 'Mike', 'Marklov', '1995-01-29', 'Male', '2023-12-08', '2024-10-31', 'Fraud', 1, 'Active');

INSERT INTO Prisoners (PrisonerID, FirstName, LastName, DateOfBirth, Gender, EntryDate, ReleaseDate, CrimeDescription, CellNumber, Status)
VALUES (3, 'Jane', 'Doe', '1996-01-29', 'Female', '2024-01-08', '2024-07-31', 'Fraud', 2, 'Active');

```

codesnap.dev

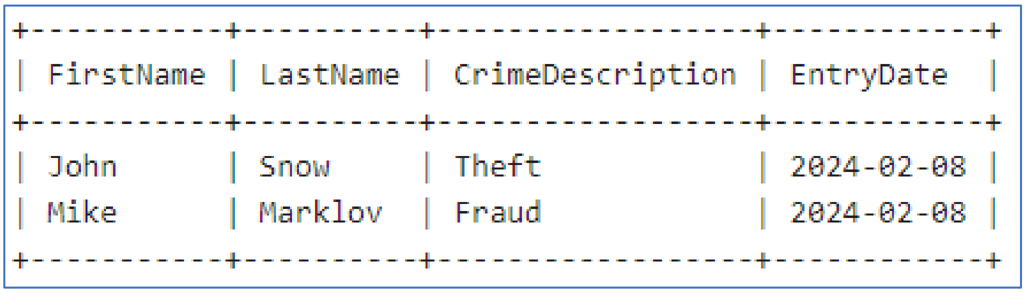


```
+------------+-----------+----------+-------------+--------+------------+-------------+------------------+------------+--------+
| PrisonerID | FirstName | LastName | DateOfBirth | Gender | EntryDate  | ReleaseDate | CrimeDescription | CellNumber | Status |
+------------+-----------+----------+-------------+--------+------------+-------------+------------------+------------+--------+
|          1 | John      | Snow     | 1990-05-15  | Male   | 2024-02-08 | 2024-12-31  | Theft            |          1 | Active |
|          2 | Mike      | Marklov  | 1995-01-29  | Male   | 2024-02-08 | 2024-12-31  | Fraud            |          1 | Active |
+------------+-----------+----------+-------------+--------+------------+-------------+------------------+------------+--------+
```

1.5 Query to show all convicts with their Crime tagged with entry date



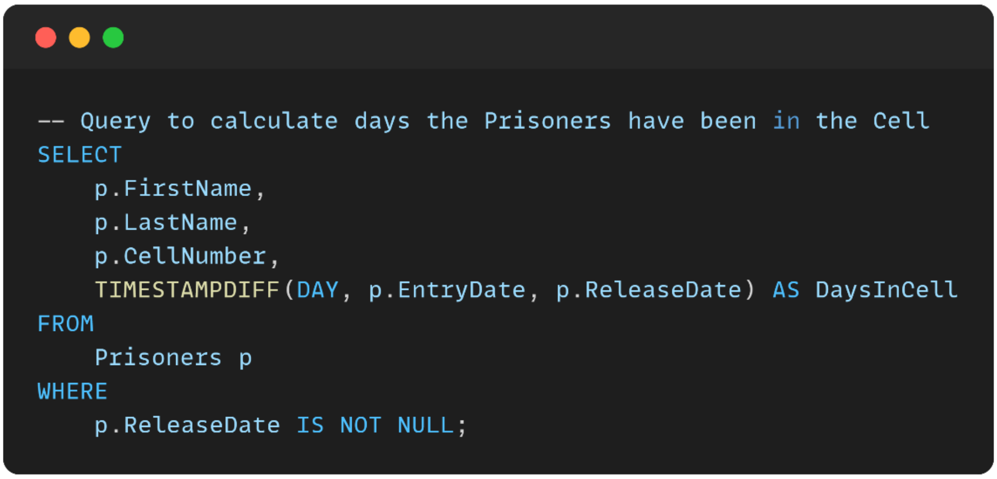
```
SELECT FirstName, LastName, CrimeDescription, EntryDate
FROM Prisoners;
```



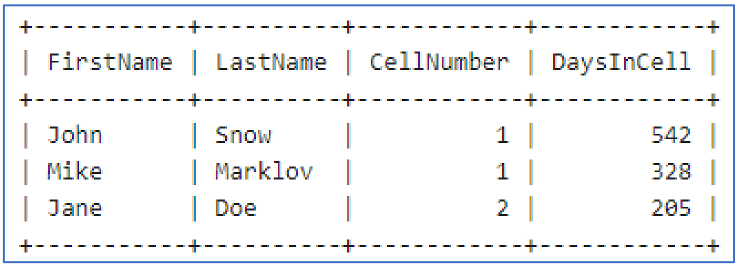
```
+-----------+----------+------------------+------------+
| FirstName | LastName | CrimeDescription | EntryDate  |
+-----------+----------+------------------+------------+
| John      | Snow     | Theft            | 2024-02-08 |
| Mike      | Marklov  | Fraud            | 2024-02-08 |
+-----------+----------+------------------+------------+
```

1.6 Query to calculate days the Prisoners have been in the Cell.

```
SELECT
    p.FirstName,
    p.LastName,
    p.CellNumber,
    TIMESTAMPDIFF(DAY, p.EntryDate, p.ReleaseDate) AS DaysInCell
FROM
    Prisoners p
WHERE
    p.ReleaseDate IS NOT NULL;
```



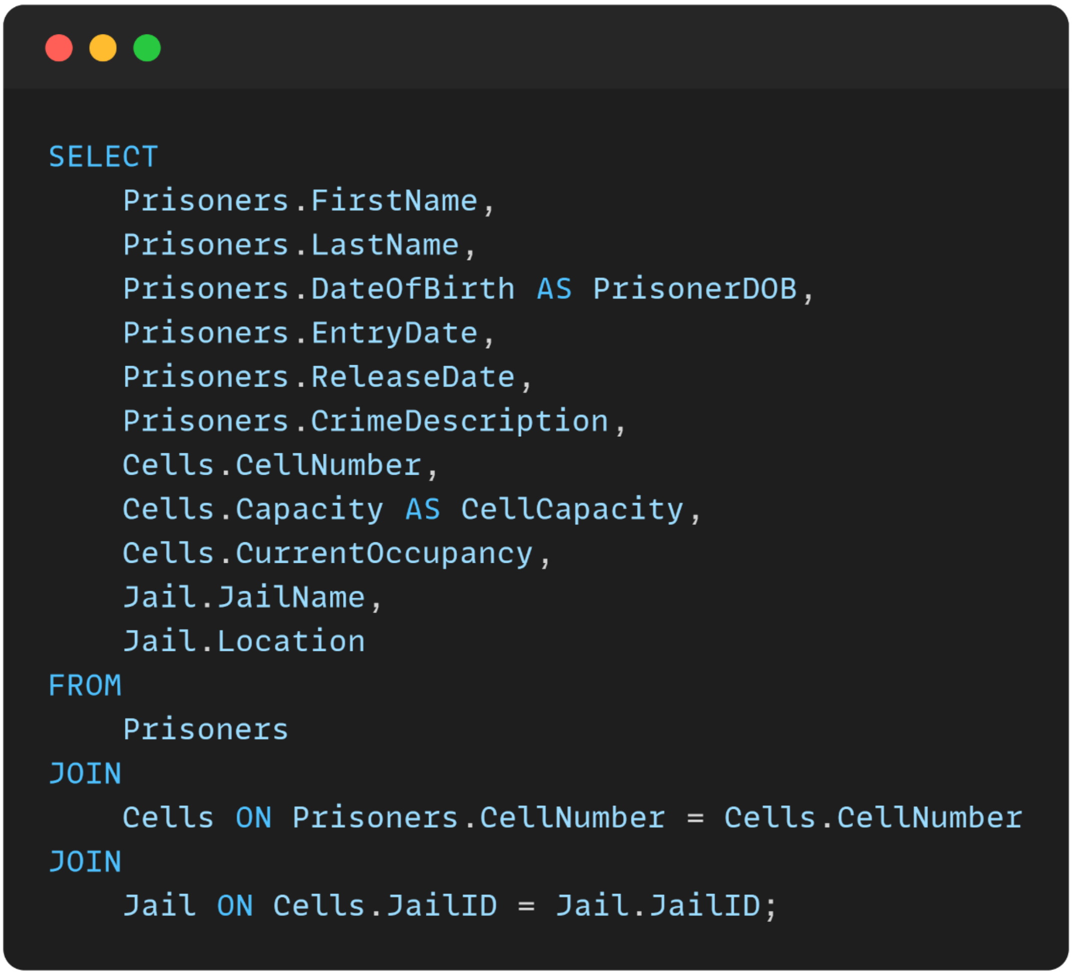
```
-- Query to calculate days the Prisoners have been in the Cell
SELECT
    p.FirstName,
    p.LastName,
    p.CellNumber,
    TIMESTAMPDIFF(DAY, p.EntryDate, p.ReleaseDate) AS DaysInCell
FROM
    Prisoners p
WHERE
    p.ReleaseDate IS NOT NULL;
```



```
+-----------+----------+------------+------------+
| FirstName | LastName | CellNumber | DaysInCell |
+-----------+----------+------------+------------+
| John      | Snow     |          1 |        542 |
| Mike      | Marklov  |          1 |        328 |
| Jane      | Doe      |          2 |        205 |
+-----------+----------+------------+------------+
```

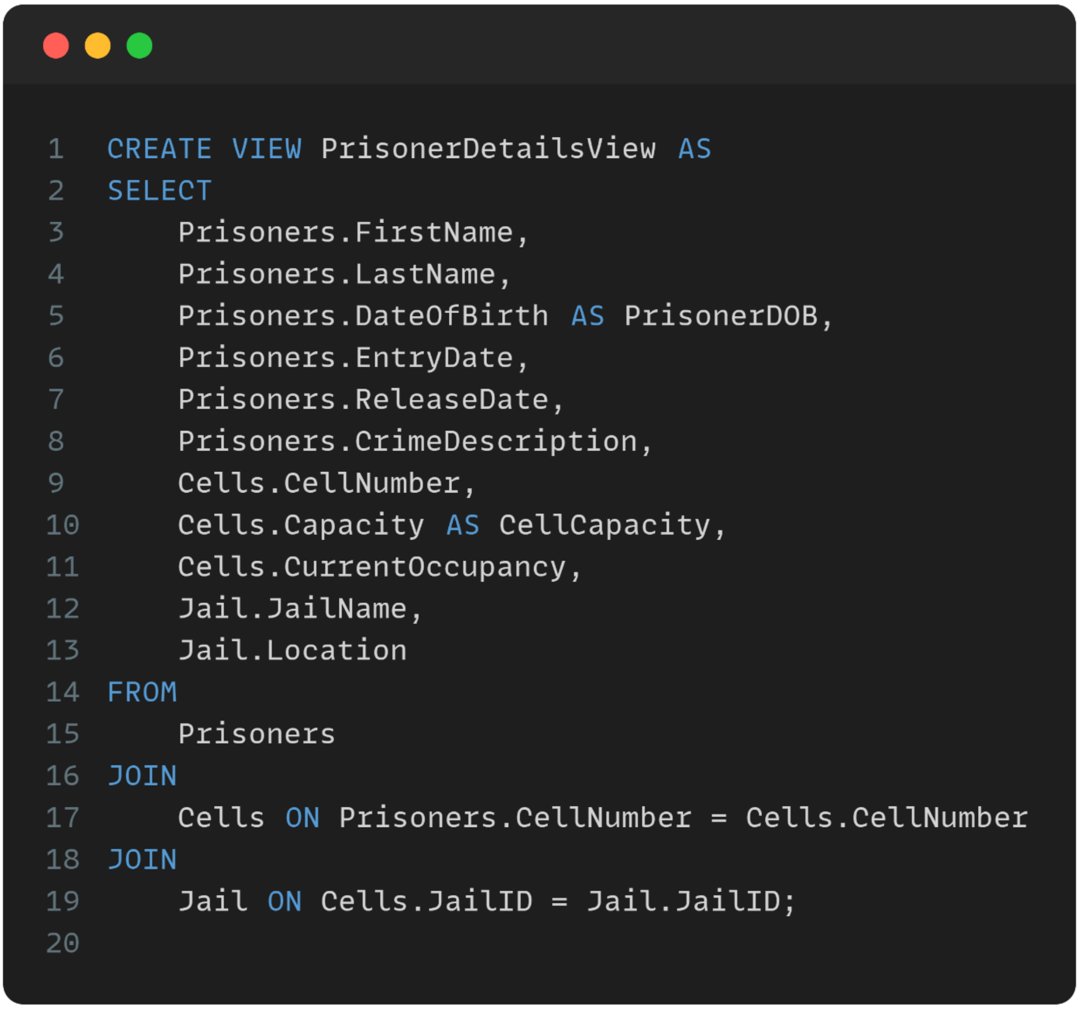
1.7 Query to fetch and show data from various related tables.

```
SELECT
    Prisoners.FirstName,
    Prisoners.LastName,
    Prisoners.DateOfBirth AS PrisonerDOB,
    Prisoners.EntryDate,
    Prisoners.ReleaseDate,
    Prisoners.CrimeDescription,
    Cells.CellNumber,
    Cells.Capacity AS CellCapacity,
    Cells.CurrentOccupancy,
    Jail.JailName,
    Jail.Location
FROM
    Prisoners
JOIN
    Cells ON Prisoners.CellNumber = Cells.CellNumber
JOIN
    Jail ON Cells.JailID = Jail.JailID;
```



```
SELECT
    Prisoners.FirstName,
    Prisoners.LastName,
    Prisoners.DateOfBirth AS PrisonerDOB,
    Prisoners.EntryDate,
    Prisoners.ReleaseDate,
    Prisoners.CrimeDescription,
    Cells.CellNumber,
    Cells.Capacity AS CellCapacity,
    Cells.CurrentOccupancy,
    Jail.JailName,
    Jail.Location
FROM
    Prisoners
JOIN
    Cells ON Prisoners.CellNumber = Cells.CellNumber
JOIN
    Jail ON Cells.JailID = Jail.JailID;
```

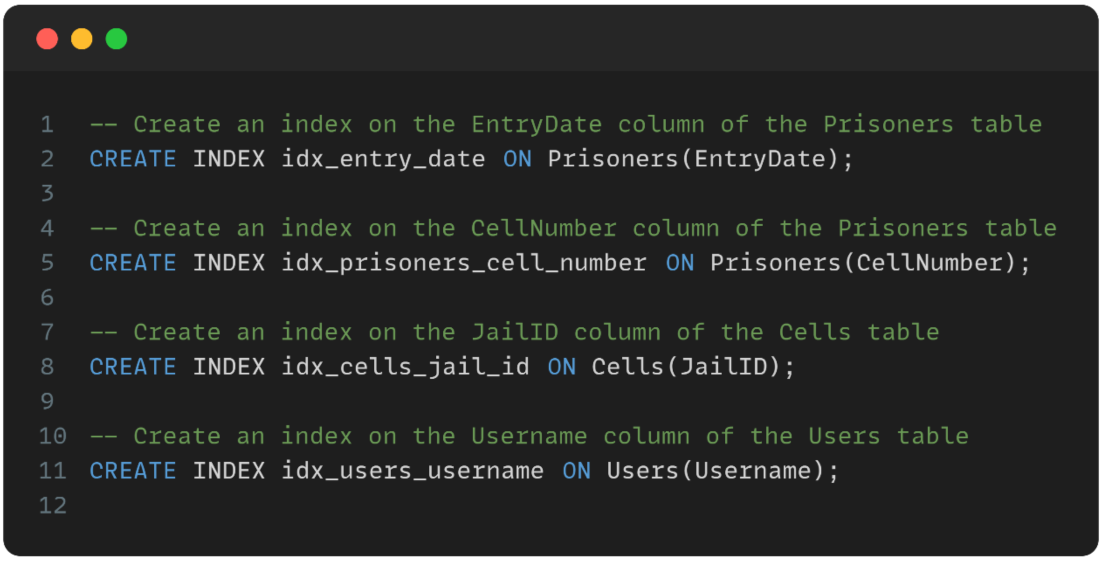
1.8 Optimize repeated read operations using views/materialized views.



```sql
CREATE VIEW PrisonerDetailsView AS
SELECT
    Prisoners.FirstName,
    Prisoners.LastName,
    Prisoners.DateOfBirth AS PrisonerDOB,
    Prisoners.EntryDate,
    Prisoners.ReleaseDate,
    Prisoners.CrimeDescription,
    Cells.CellNumber,
    Cells.Capacity AS CellCapacity,
    Cells.CurrentOccupancy,
    Jail.JailName,
    Jail.Location
FROM
    Prisoners
JOIN
    Cells ON Prisoners.CellNumber = Cells.CellNumber
JOIN
    Jail ON Cells.JailID = Jail.JailID;

```

1.9 Optimize read operations using indexing wherever required.

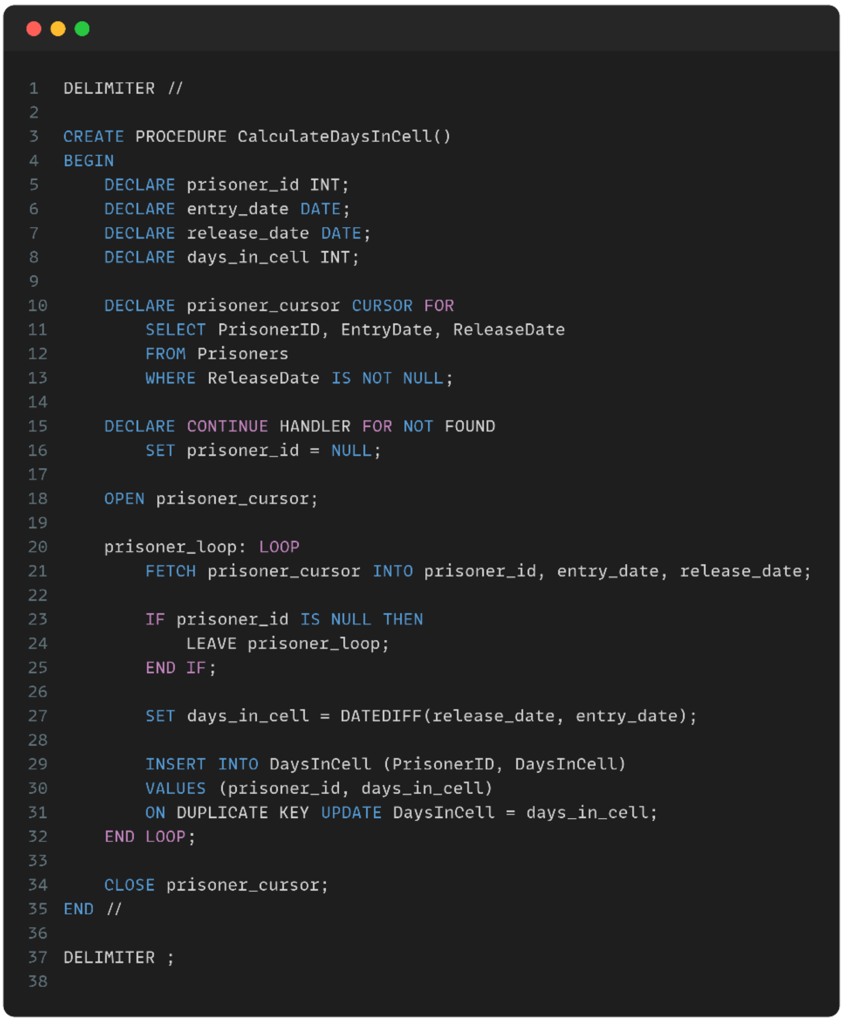
```
-- Index on the EntryDate column of the Prisoners table
CREATE INDEX idx_entry_date ON Prisoners(EntryDate);

-- Index on the CellNumber column of the Prisoners table
CREATE INDEX idx_prisoners_cell_number ON Prisoners(CellNumber);

-- Index on the JailID column of the Cells table
CREATE INDEX idx_cells_jail_id ON Cells(JailID);

-- Index on the Username column of the Users table
CREATE INDEX idx_users_username ON Users(Username);
```



```
-- Create an index on the EntryDate column of the Prisoners table
CREATE INDEX idx_entry_date ON Prisoners(EntryDate);

-- Create an index on the CellNumber column of the Prisoners table
CREATE INDEX idx_prisoners_cell_number ON Prisoners(CellNumber);

-- Create an index on the JailID column of the Cells table
CREATE INDEX idx_cells_jail_id ON Cells(JailID);

-- Create an index on the Username column of the Users table
CREATE INDEX idx_users_username ON Users(Username);

```

1.10 Optimizing calculating days in cell using stored procedures.

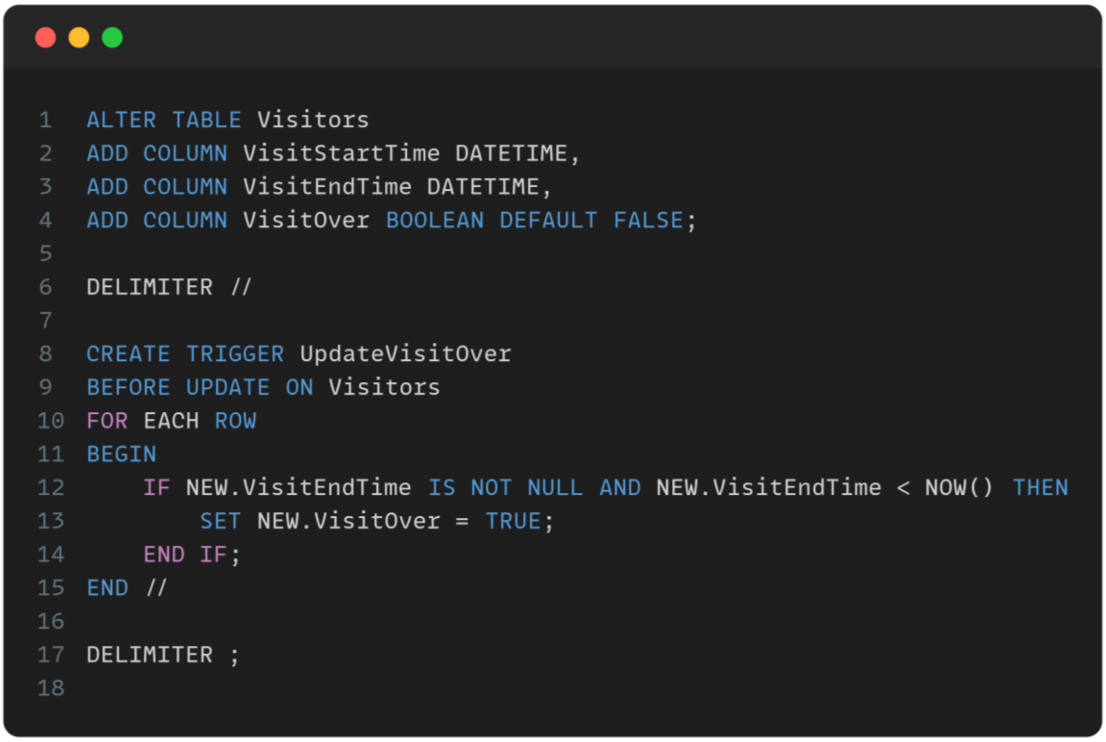
```
DELIMITER //
CREATE PROCEDURE CalculateDaysInCell()n
BEGIN
    DECLARE prisoner_id INT;
    DECLARE entry_date DATE;
    DECLARE release_date DATE;
    DECLARE days_in_cell INT;
    DECLARE prisoner_cursor CURSOR FOR
        SELECT PrisonerID, EntryDate, ReleaseDate
        FROM Prisoners
        WHERE ReleaseDate IS NOT NULL;
    DECLARE CONTINUE HANDLER FOR NOT FOUND
        SET prisoner_id = NULL;
    OPEN prisoner_cursor;
    prisoner_loop: LOOP
        FETCH prisoner_cursor INTO prisoner_id, entry_date, release_date;
        IF prisoner_id IS NULL THEN
            LEAVE prisoner_loop;
        END IF;
        SET days_in_cell = DATEDIFF(release_date, entry_date);
        INSERT INTO DaysInCell (PrisonerID, DaysInCell)
        VALUES (prisoner_id, days_in_cell)
        ON DUPLICATE KEY UPDATE DaysInCell = days_in_cell;
    END LOOP;
    CLOSE prisoner_cursor;
END //
DELIMITER ;
```



```
DELIMITER //

CREATE PROCEDURE CalculateDaysInCell()
BEGIN
    DECLARE prisoner_id INT;
    DECLARE entry_date DATE;
    DECLARE release_date DATE;
    DECLARE days_in_cell INT;

    DECLARE prisoner_cursor CURSOR FOR
        SELECT PrisonerID, EntryDate, ReleaseDate
        FROM Prisoners
        WHERE ReleaseDate IS NOT NULL;

    DECLARE CONTINUE HANDLER FOR NOT FOUND
        SET prisoner_id = NULL;

    OPEN prisoner_cursor;

    prisoner_loop: LOOP
        FETCH prisoner_cursor INTO prisoner_id, entry_date, release_date;

        IF prisoner_id IS NULL THEN
            LEAVE prisoner_loop;
        END IF;

        SET days_in_cell = DATEDIFF(release_date, entry_date);

        INSERT INTO DaysInCell (PrisonerID, DaysInCell)
        VALUES (prisoner_id, days_in_cell)
        ON DUPLICATE KEY UPDATE DaysInCell = days_in_cell;
    END LOOP;

    CLOSE prisoner_cursor;
END //

DELIMITER ;
```

1.11 Add necessary triggers to indicate when Visitors' visiting time limit is over.

```
ALTER TABLE Visitors
ADD COLUMN VisitStartTime DATETIME,
ADD COLUMN VisitEndTime DATETIME,
ADD COLUMN VisitOver BOOLEAN DEFAULT FALSE;

DELIMITER //

CREATE TRIGGER UpdateVisitOver
BEFORE UPDATE ON Visitors
FOR EACH ROW
BEGIN
    IF NEW.VisitEndTime IS NOT NULL AND NEW.VisitEndTime < NOW() THEN
        SET NEW.VisitOver = TRUE;
    END IF;
END //

DELIMITER ;
```



```
ALTER TABLE Visitors
ADD COLUMN VisitStartTime DATETIME,
ADD COLUMN VisitEndTime DATETIME,
ADD COLUMN VisitOver BOOLEAN DEFAULT FALSE;

DELIMITER //

CREATE TRIGGER UpdateVisitOver
BEFORE UPDATE ON Visitors
FOR EACH ROW
BEGIN
    IF NEW.VisitEndTime IS NOT NULL AND NEW.VisitEndTime < NOW()  THEN
        SET NEW.VisitOver = TRUE;
    END IF;
END //

DELIMITER ;
```