Sri Lanka Institute of Information Technology

B.Sc. (Hons) Information Technology-

Cyber security

Y2 S1

Database Management Systems or Security – IE 2042

Group Number 23

Member Details

IT Number	Name
IT 23 1843 12	AMANTHA M. A
IT 23 1705 20	RAJASOORIYA D.G.C.H
IT 23 1878 32	PERAMUNUGAMA M.R. A
IT 23 1697 08	PERERA P.A.J.M

Contents

Assumptions for the EER

a. Every Users can transact many Copies of items, but a unique copy of item cannot be borrowed by multiple users at the same time.

b. The relation TRANSACTS indicates borrowing or returning a copy.

c. The cardinality between the aggregation (USERS+COPY) and TRANSACTION- One specific User Copy pair can generate multiple transactions over time.

d. Each item can have multiple copies, and these copies may be located in different parts of the library.

e. Every borrowing transaction generates a unique Transac_id for tracking purposes and returning will update the borrowed row.

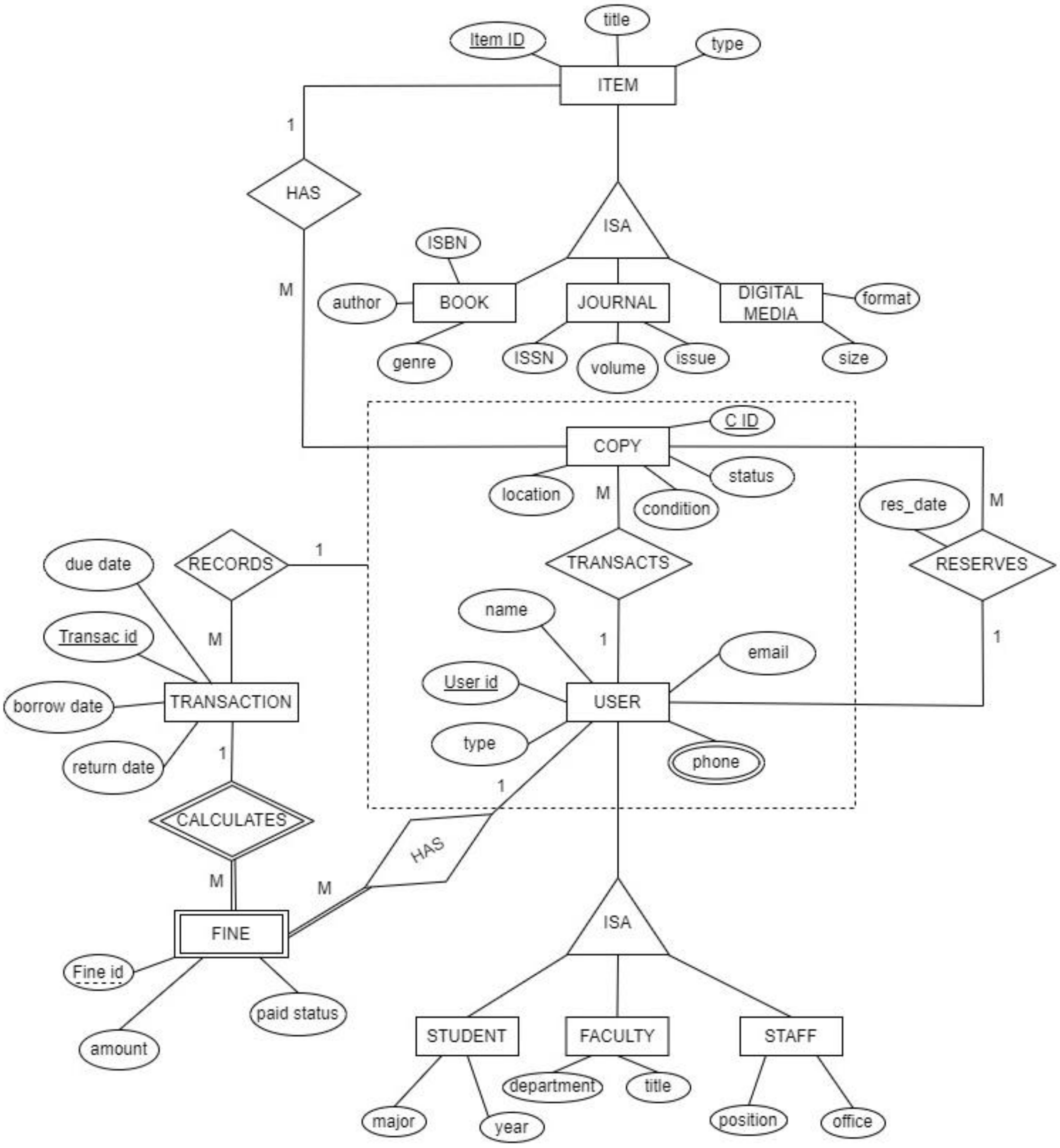
f. Users may or may not have multiple fines per transaction but each fines must have users relevant for that.

g. When a transaction is created (borrowing), the item's status is automatically updated to 'borrowed', and when a return is processed, the status changes back to 'available'. (Handles by triggers)

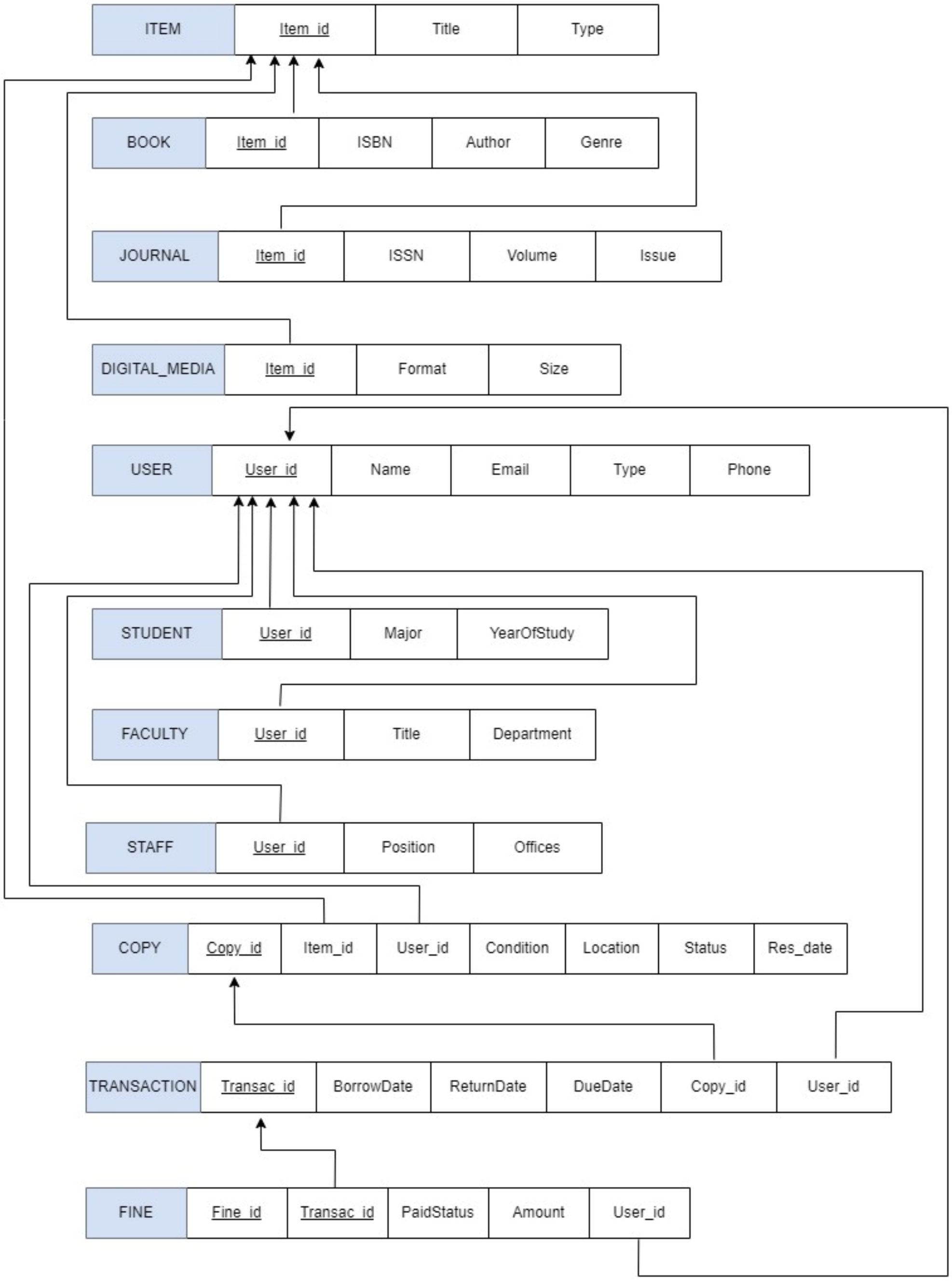
h. Every borrowing transaction must have a valid user and a valid copy of an item. If either the **`User_id`** or **`C_ID`** is invalid, the transaction fails.

i. The Copies which are reserved by the User cannot be borrowed, those copies are reserved only for the reference purposes.

1. Enhanced Entity Relationship Diagram



2. Schema of the Database according to the Relational Data Model



ITEM | Item_id | Title | Type

BOOK | Item_id | ISBN | Author | Genre

JOURNAL | Item_id | ISSN | Volume | Issue

DIGITAL_MEDIA | Item_id | Format | Size

USER | User_id | Name | Email | Type | Phone

STUDENT | User_id | Major | YearOfStudy

FACULTY | User_id | Title | Department

STAFF | User_id | Position | Offices

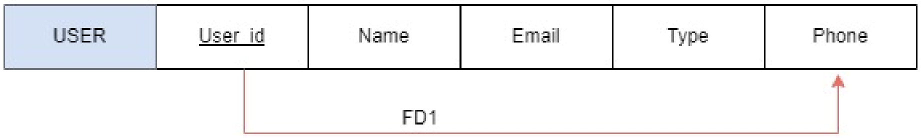
COPY | Copy_id | Item_id | User_id | Condition | Location | Status | Res_date

TRANSACTION | Transac_id | BorrowDate | ReturnDate | DueDate | Copy_id | User_id

FINE | Fine_id | Transac_id | PaidStatus | Amount | User_id

3. Normalized tables using functional Dependencies

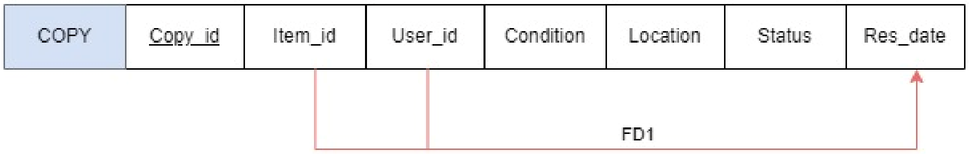
Remove composite attributes



CONTACT	User_id	Phone

USER	User_id	Name	Email	Type

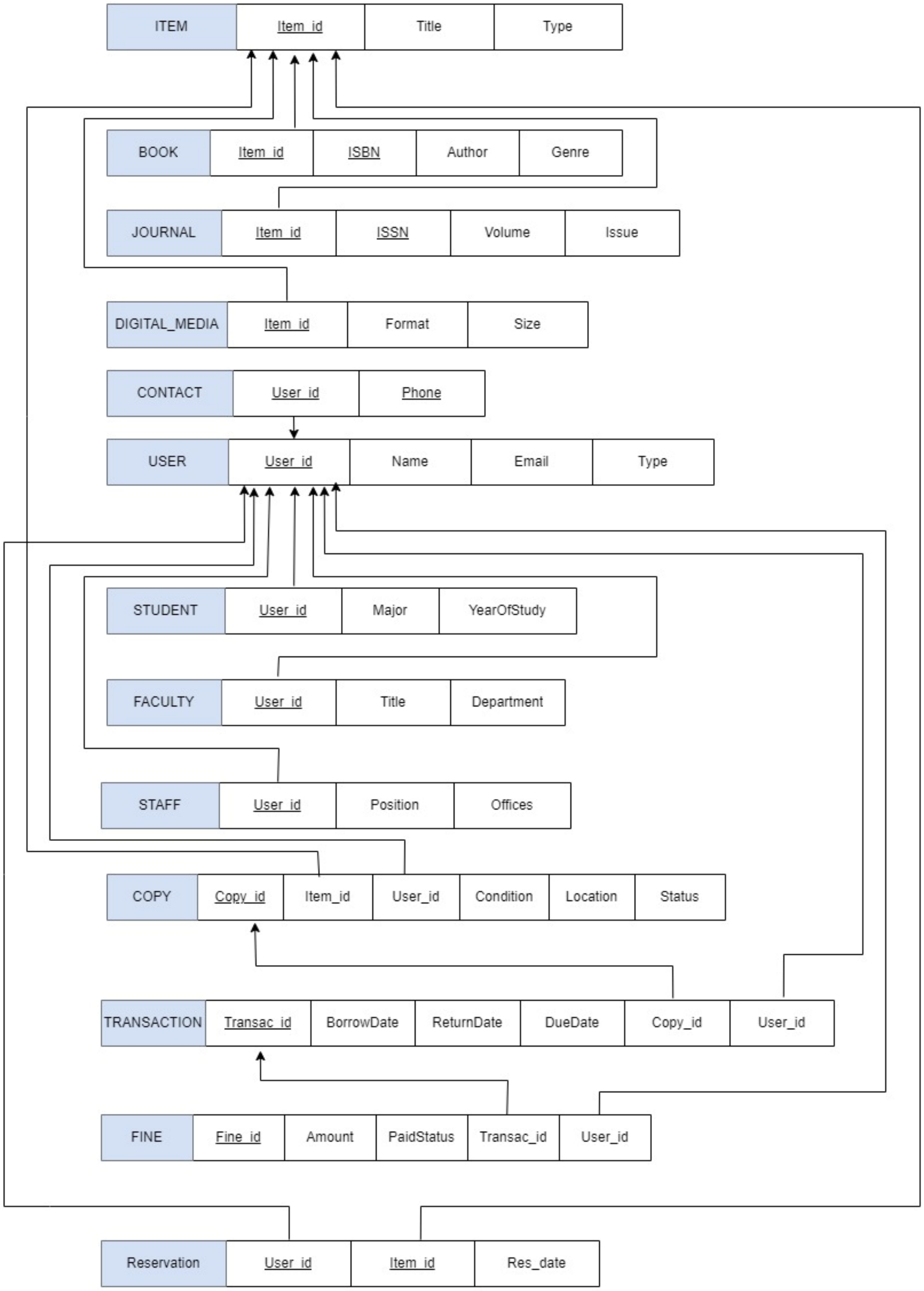
Remove partial Dependencies



Reservation	Item_id	User_id	Res_date

COPY	Copy_id	Item_id	User_id	Condition	Location	Status

4. Schema of the Database after Normalization



ITEM | Item_id | Title | Type

BOOK | Item_id | ISBN | Author | Genre

JOURNAL | Item_id | ISSN | Volume | Issue

DIGITAL_MEDIA | Item_id | Format | Size

CONTACT | User_id | Phone

USER | User_id | Name | Email | Type

STUDENT | User_id | Major | YearOfStudy

FACULTY | User_id | Title | Department

STAFF | User_id | Position | Offices

COPY | Copy_id | Item_id | User_id | Condition | Location | Status

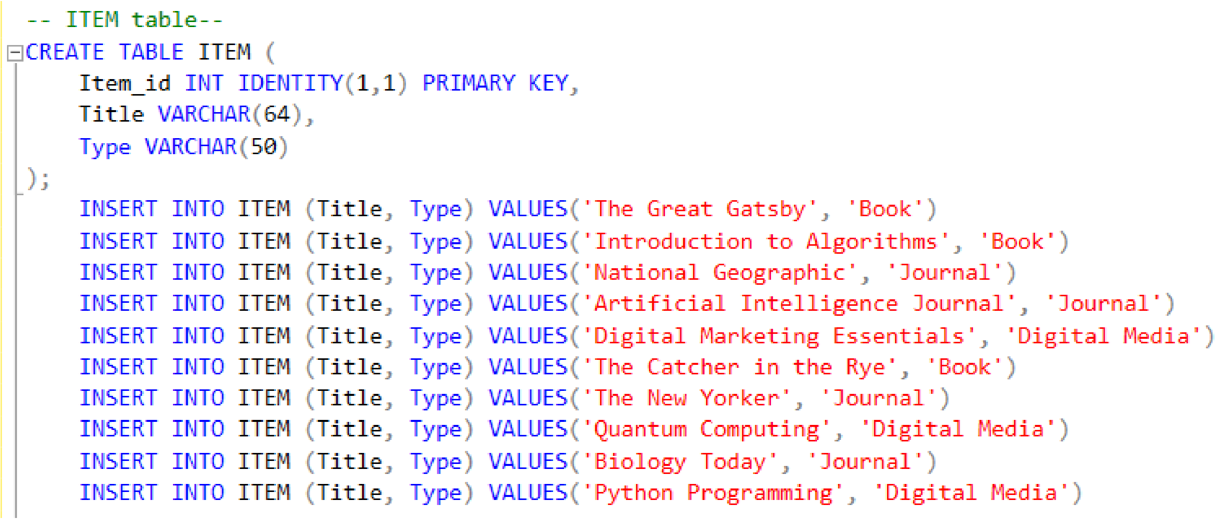
TRANSACTION | Transac_id | BorrowDate | ReturnDate | DueDate | Copy_id | User_id

FINE | Fine_id | Amount | PaidStatus | Transac_id | User_id

Reservation | User_id | Item_id | Res_date

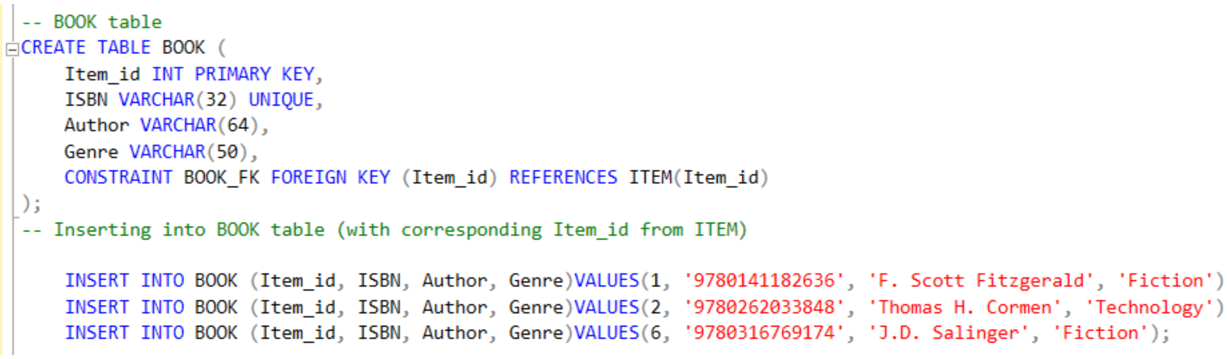
5. Table Implementations and Constraints Implementation

ITEM TABLE



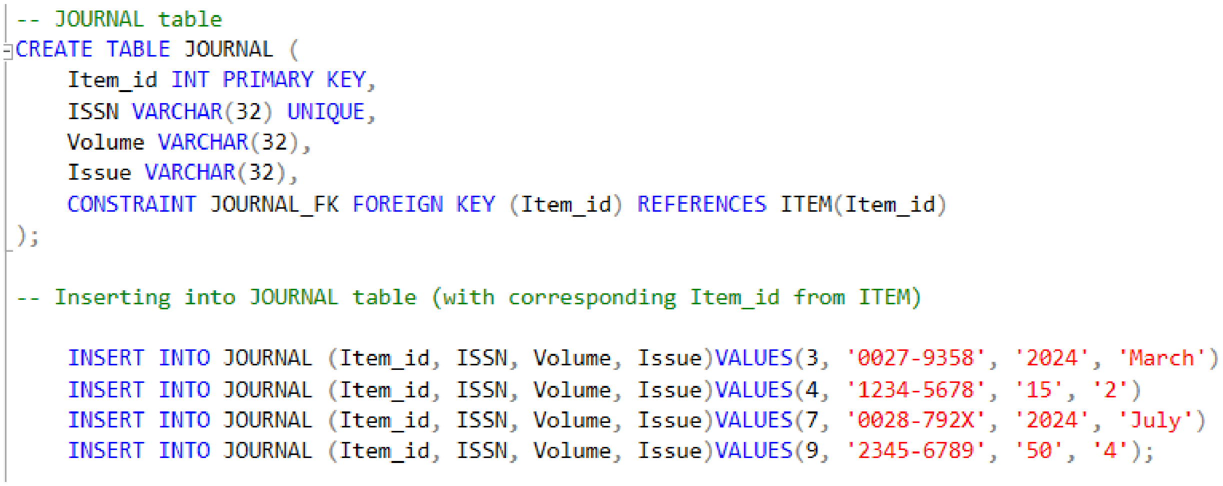
```
 -- ITEM table--
CREATE TABLE ITEM (
    Item_id INT IDENTITY(1,1) PRIMARY KEY,
    Title VARCHAR(64),
    Type VARCHAR(50)
);
    INSERT INTO ITEM (Title, Type) VALUES('The Great Gatsby', 'Book')
    INSERT INTO ITEM (Title, Type) VALUES('Introduction to Algorithms', 'Book')
    INSERT INTO ITEM (Title, Type) VALUES('National Geographic', 'Journal')
    INSERT INTO ITEM (Title, Type) VALUES('Artificial Intelligence Journal', 'Journal')
    INSERT INTO ITEM (Title, Type) VALUES('Digital Marketing Essentials', 'Digital Media')
    INSERT INTO ITEM (Title, Type) VALUES('The Catcher in the Rye', 'Book')
    INSERT INTO ITEM (Title, Type) VALUES('The New Yorker', 'Journal')
    INSERT INTO ITEM (Title, Type) VALUES('Quantum Computing', 'Digital Media')
    INSERT INTO ITEM (Title, Type) VALUES('Biology Today', 'Journal')
    INSERT INTO ITEM (Title, Type) VALUES('Python Programming', 'Digital Media')
```

BOOK TABLE



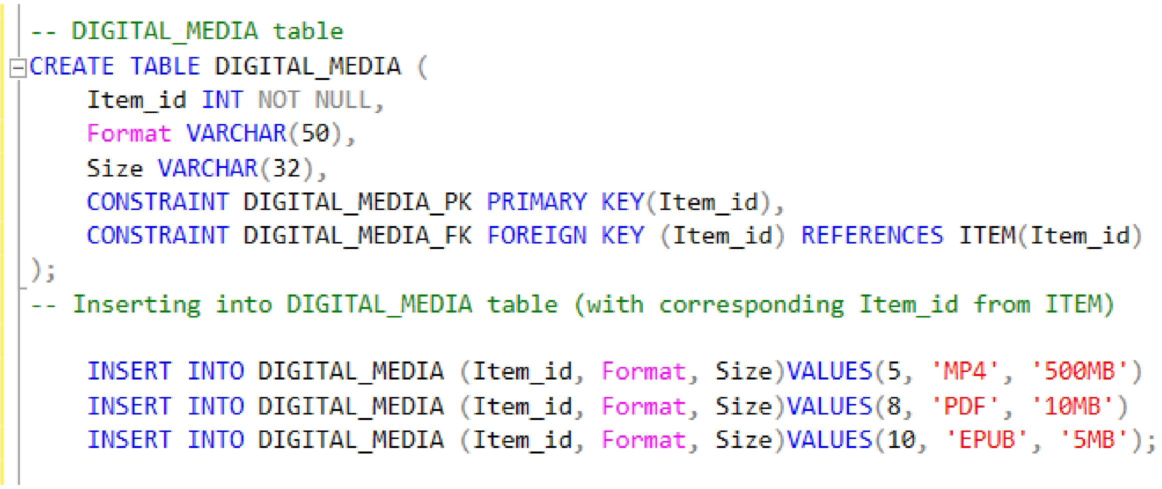
```
-- BOOK table
CREATE TABLE BOOK (
    Item_id INT PRIMARY KEY,
    ISBN VARCHAR(32) UNIQUE,
    Author VARCHAR(64),
    Genre VARCHAR(50),
    CONSTRAINT BOOK_FK FOREIGN KEY (Item_id) REFERENCES ITEM(Item_id)
);
-- Inserting into BOOK table (with corresponding Item_id from ITEM)

    INSERT INTO BOOK (Item_id, ISBN, Author, Genre)VALUES(1, '9780141182636', 'F. Scott Fitzgerald', 'Fiction')
    INSERT INTO BOOK (Item_id, ISBN, Author, Genre)VALUES(2, '9780262033848', 'Thomas H. Cormen', 'Technology')
    INSERT INTO BOOK (Item_id, ISBN, Author, Genre)VALUES(6, '9780316769174', 'J.D. Salinger', 'Fiction');
```

JOURNAL TABLE



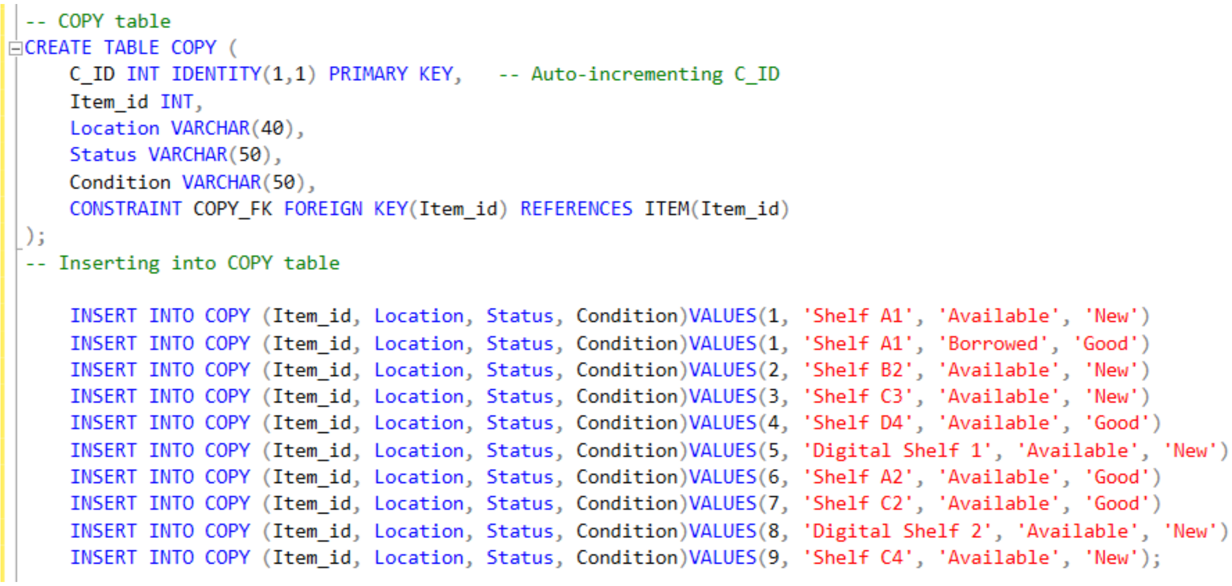
```
-- JOURNAL table
CREATE TABLE JOURNAL (
    Item_id INT PRIMARY KEY,
    ISSN VARCHAR(32) UNIQUE,
    Volume VARCHAR(32),
    Issue VARCHAR(32),
    CONSTRAINT JOURNAL_FK FOREIGN KEY (Item_id) REFERENCES ITEM(Item_id)
);

-- Inserting into JOURNAL table (with corresponding Item_id from ITEM)

    INSERT INTO JOURNAL (Item_id, ISSN, Volume, Issue)VALUES(3, '0027-9358', '2024', 'March')
    INSERT INTO JOURNAL (Item_id, ISSN, Volume, Issue)VALUES(4, '1234-5678', '15', '2')
    INSERT INTO JOURNAL (Item_id, ISSN, Volume, Issue)VALUES(7, '0028-792X', '2024', 'July')
    INSERT INTO JOURNAL (Item_id, ISSN, Volume, Issue)VALUES(9, '2345-6789', '50', '4');
```

DIGITAL MEDIA TABLE



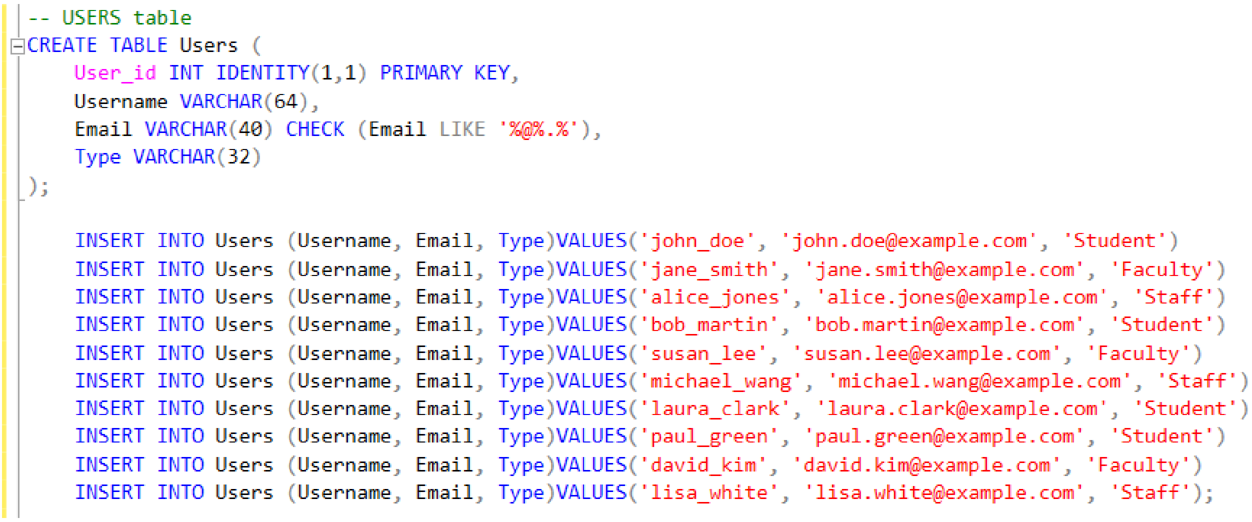
```sql
-- DIGITAL_MEDIA table
CREATE TABLE DIGITAL_MEDIA (
    Item_id INT NOT NULL,
    Format VARCHAR(50),
    Size VARCHAR(32),
    CONSTRAINT DIGITAL_MEDIA_PK PRIMARY KEY(Item_id),
    CONSTRAINT DIGITAL_MEDIA_FK FOREIGN KEY (Item_id) REFERENCES ITEM(Item_id)
);
-- Inserting into DIGITAL_MEDIA table (with corresponding Item_id from ITEM)

    INSERT INTO DIGITAL_MEDIA (Item_id, Format, Size)VALUES(5, 'MP4', '500MB')
    INSERT INTO DIGITAL_MEDIA (Item_id, Format, Size)VALUES(8, 'PDF', '10MB')
    INSERT INTO DIGITAL_MEDIA (Item_id, Format, Size)VALUES(10, 'EPUB', '5MB');
```

COPY TABLE



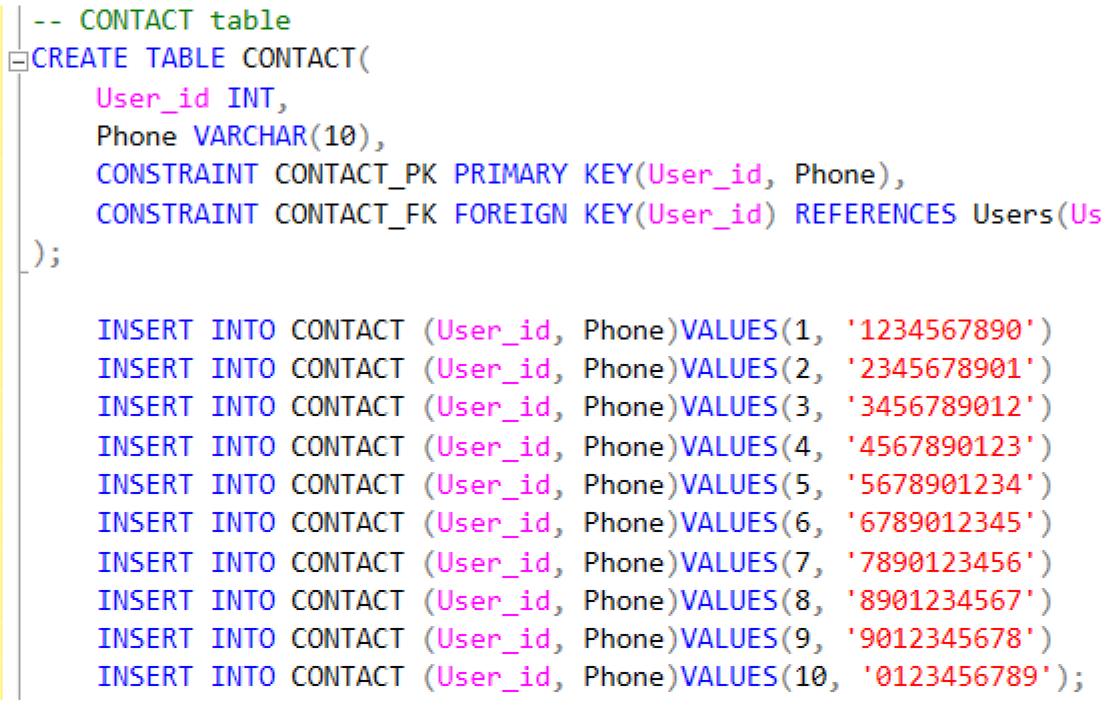
```sql
-- COPY table
CREATE TABLE COPY (
    C_ID INT IDENTITY(1,1) PRIMARY KEY,   -- Auto-incrementing C_ID
    Item_id INT,
    Location VARCHAR(40),
    Status VARCHAR(50),
    Condition VARCHAR(50),
    CONSTRAINT COPY_FK FOREIGN KEY(Item_id) REFERENCES ITEM(Item_id)
);
-- Inserting into COPY table

    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(1, 'Shelf A1', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(1, 'Shelf A1', 'Borrowed', 'Good')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(2, 'Shelf B2', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(3, 'Shelf C3', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(4, 'Shelf D4', 'Available', 'Good')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(5, 'Digital Shelf 1', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(6, 'Shelf A2', 'Available', 'Good')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(7, 'Shelf C2', 'Available', 'Good')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(8, 'Digital Shelf 2', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(9, 'Shelf C4', 'Available', 'New');
```

USERS TABLE



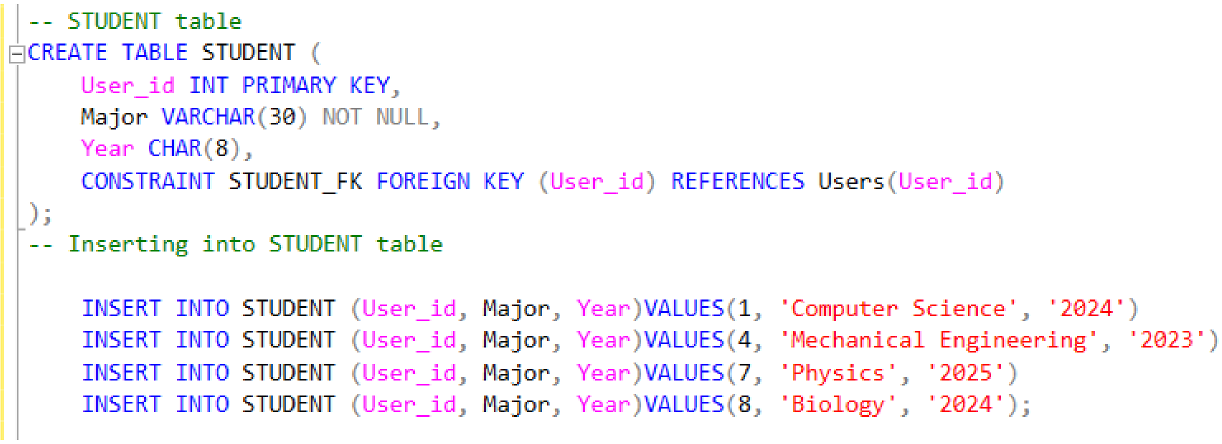
```sql
-- USERS table
CREATE TABLE Users (
    User_id INT IDENTITY(1,1) PRIMARY KEY,
    Username VARCHAR(64),
    Email VARCHAR(40) CHECK (Email LIKE '%@%.%'),
    Type VARCHAR(32)
);

    INSERT INTO Users (Username, Email, Type)VALUES('john_doe', 'john.doe@example.com', 'Student')
    INSERT INTO Users (Username, Email, Type)VALUES('jane_smith', 'jane.smith@example.com', 'Faculty')
    INSERT INTO Users (Username, Email, Type)VALUES('alice_jones', 'alice.jones@example.com', 'Staff')
    INSERT INTO Users (Username, Email, Type)VALUES('bob_martin', 'bob.martin@example.com', 'Student')
    INSERT INTO Users (Username, Email, Type)VALUES('susan_lee', 'susan.lee@example.com', 'Faculty')
    INSERT INTO Users (Username, Email, Type)VALUES('michael_wang', 'michael.wang@example.com', 'Staff')
    INSERT INTO Users (Username, Email, Type)VALUES('laura_clark', 'laura.clark@example.com', 'Student')
    INSERT INTO Users (Username, Email, Type)VALUES('paul_green', 'paul.green@example.com', 'Student')
    INSERT INTO Users (Username, Email, Type)VALUES('david_kim', 'david.kim@example.com', 'Faculty')
    INSERT INTO Users (Username, Email, Type)VALUES('lisa_white', 'lisa.white@example.com', 'Staff');
```

CONTACT TABLE



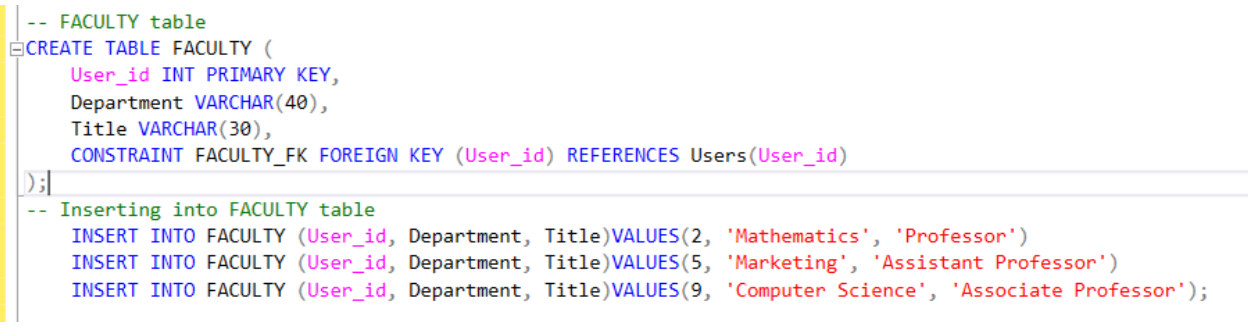
```
-- CONTACT table
CREATE TABLE CONTACT(
    User_id INT,
    Phone VARCHAR(10),
    CONSTRAINT CONTACT_PK PRIMARY KEY(User_id, Phone),
    CONSTRAINT CONTACT_FK FOREIGN KEY(User_id) REFERENCES Users(User_id)
);

    INSERT INTO CONTACT (User_id, Phone)VALUES(1, '1234567890')
    INSERT INTO CONTACT (User_id, Phone)VALUES(2, '2345678901')
    INSERT INTO CONTACT (User_id, Phone)VALUES(3, '3456789012')
    INSERT INTO CONTACT (User_id, Phone)VALUES(4, '4567890123')
    INSERT INTO CONTACT (User_id, Phone)VALUES(5, '5678901234')
    INSERT INTO CONTACT (User_id, Phone)VALUES(6, '6789012345')
    INSERT INTO CONTACT (User_id, Phone)VALUES(7, '7890123456')
    INSERT INTO CONTACT (User_id, Phone)VALUES(8, '8901234567')
    INSERT INTO CONTACT (User_id, Phone)VALUES(9, '9012345678')
    INSERT INTO CONTACT (User_id, Phone)VALUES(10, '0123456789');
```

STUDENT TABLE



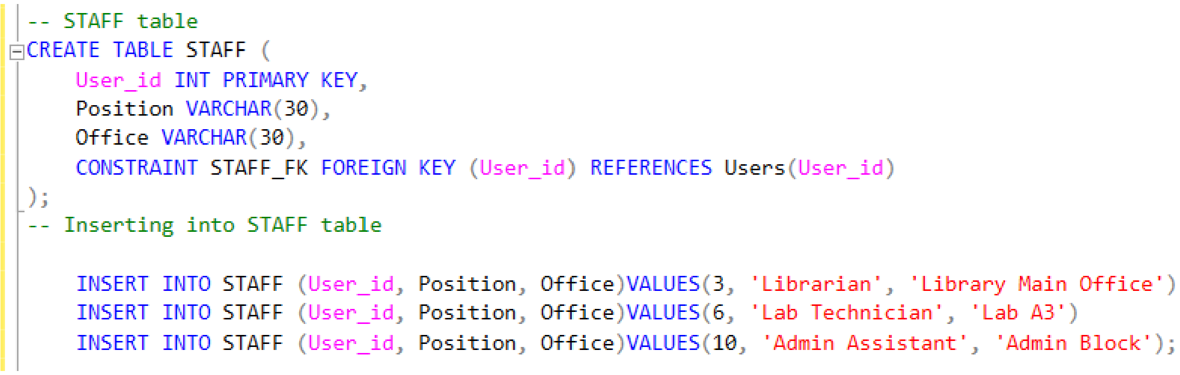
```
-- STUDENT table
CREATE TABLE STUDENT (
    User_id INT PRIMARY KEY,
    Major VARCHAR(30) NOT NULL,
    Year CHAR(8),
    CONSTRAINT STUDENT_FK FOREIGN KEY (User_id) REFERENCES Users(User_id)
);
-- Inserting into STUDENT table

    INSERT INTO STUDENT (User_id, Major, Year)VALUES(1, 'Computer Science', '2024')
    INSERT INTO STUDENT (User_id, Major, Year)VALUES(4, 'Mechanical Engineering', '2023')
    INSERT INTO STUDENT (User_id, Major, Year)VALUES(7, 'Physics', '2025')
    INSERT INTO STUDENT (User_id, Major, Year)VALUES(8, 'Biology', '2024');
```

FACULTY TABLE



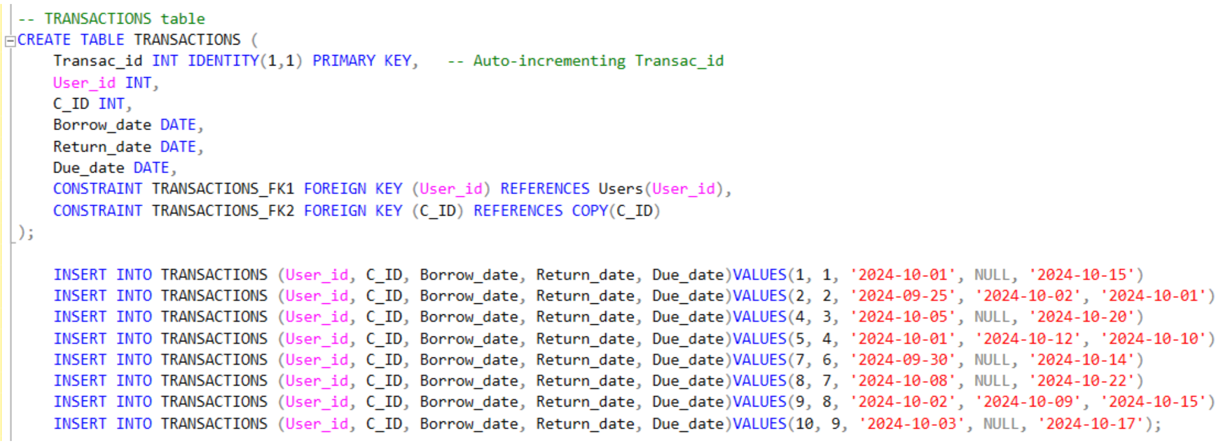
```
-- FACULTY table
CREATE TABLE FACULTY (
    User_id INT PRIMARY KEY,
    Department VARCHAR(40),
    Title VARCHAR(30),
    CONSTRAINT FACULTY_FK FOREIGN KEY (User_id) REFERENCES Users(User_id)
);
-- Inserting into FACULTY table
    INSERT INTO FACULTY (User_id, Department, Title)VALUES(2, 'Mathematics', 'Professor')
    INSERT INTO FACULTY (User_id, Department, Title)VALUES(5, 'Marketing', 'Assistant Professor')
    INSERT INTO FACULTY (User_id, Department, Title)VALUES(9, 'Computer Science', 'Associate Professor');
```

STAFF TABLE



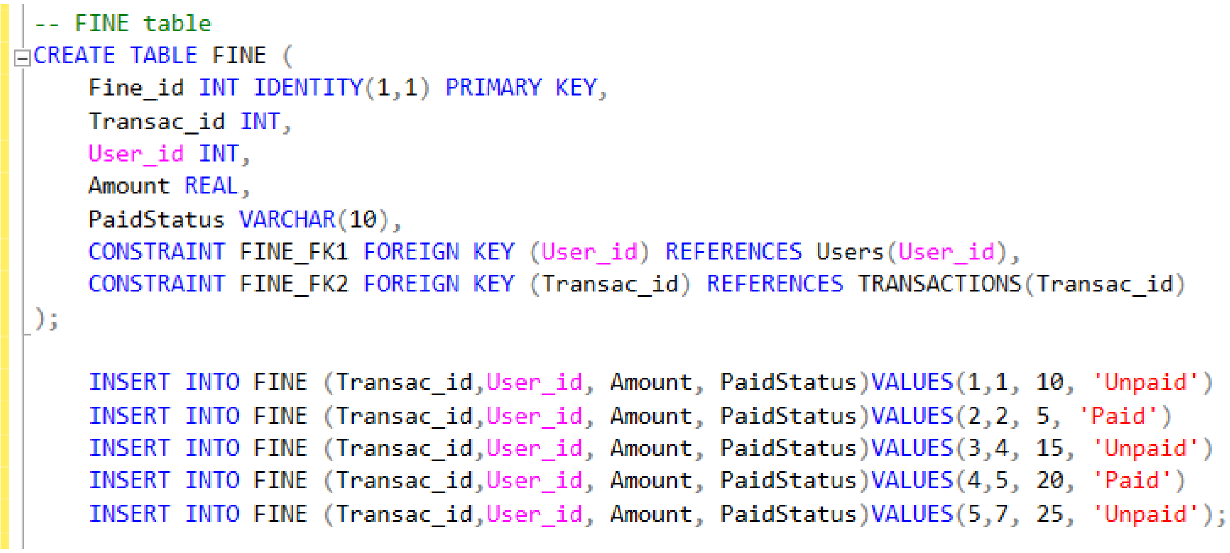
```
-- STAFF table
CREATE TABLE STAFF (
    User_id INT PRIMARY KEY,
    Position VARCHAR(30),
    Office VARCHAR(30),
    CONSTRAINT STAFF_FK FOREIGN KEY (User_id) REFERENCES Users(User_id)
);
-- Inserting into STAFF table

    INSERT INTO STAFF (User_id, Position, Office)VALUES(3, 'Librarian', 'Library Main Office')
    INSERT INTO STAFF (User_id, Position, Office)VALUES(6, 'Lab Technician', 'Lab A3')
    INSERT INTO STAFF (User_id, Position, Office)VALUES(10, 'Admin Assistant', 'Admin Block');
```

TRANSACTIONS TABLE



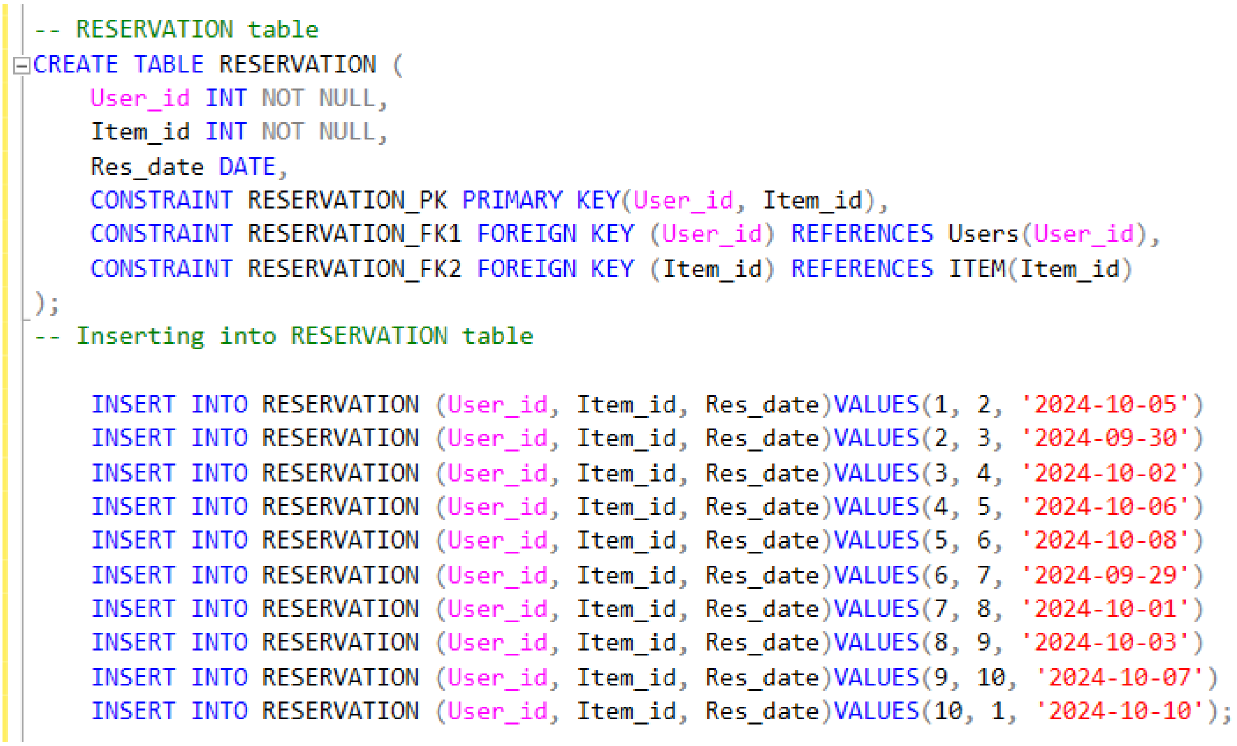
```
-- TRANSACTIONS table
CREATE TABLE TRANSACTIONS (
    Transac_id INT IDENTITY(1,1) PRIMARY KEY,   -- Auto-incrementing Transac_id
    User_id INT,
    C_ID INT,
    Borrow_date DATE,
    Return_date DATE,
    Due_date DATE,
    CONSTRAINT TRANSACTIONS_FK1 FOREIGN KEY (User_id) REFERENCES Users(User_id),
    CONSTRAINT TRANSACTIONS_FK2 FOREIGN KEY (C_ID) REFERENCES COPY(C_ID)
);

    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date, Due_date)VALUES(1, 1, '2024-10-01', NULL, '2024-10-15')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date, Due_date)VALUES(2, 2, '2024-09-25', '2024-10-02', '2024-10-01')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date, Due_date)VALUES(4, 3, '2024-10-05', NULL, '2024-10-20')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date, Due_date)VALUES(5, 4, '2024-10-01', '2024-10-12', '2024-10-10')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date, Due_date)VALUES(7, 6, '2024-09-30', NULL, '2024-10-14')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date, Due_date)VALUES(8, 7, '2024-10-08', NULL, '2024-10-22')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date, Due_date)VALUES(9, 8, '2024-10-02', '2024-10-09', '2024-10-15')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date, Due_date)VALUES(10, 9, '2024-10-03', NULL, '2024-10-17');
```

FINE TABLE



```
-- FINE table
CREATE TABLE FINE (
    Fine_id INT IDENTITY(1,1) PRIMARY KEY,
    Transac_id INT,
    User_id INT,
    Amount REAL,
    PaidStatus VARCHAR(10),
    CONSTRAINT FINE_FK1 FOREIGN KEY (User_id) REFERENCES Users(User_id),
    CONSTRAINT FINE_FK2 FOREIGN KEY (Transac_id) REFERENCES TRANSACTIONS(Transac_id)
);

    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(1,1, 10, 'Unpaid')
    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(2,2, 5, 'Paid')
    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(3,4, 15, 'Unpaid')
    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(4,5, 20, 'Paid')
    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(5,7, 25, 'Unpaid');
```

RESERVATION TABLE



```
-- RESERVATION table
CREATE TABLE RESERVATION (
    User_id INT NOT NULL,
    Item_id INT NOT NULL,
    Res_date DATE,
    CONSTRAINT RESERVATION_PK PRIMARY KEY(User_id, Item_id),
    CONSTRAINT RESERVATION_FK1 FOREIGN KEY (User_id) REFERENCES Users(User_id),
    CONSTRAINT RESERVATION_FK2 FOREIGN KEY (Item_id) REFERENCES ITEM(Item_id)
);
-- Inserting into RESERVATION table

    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(1, 2, '2024-10-05')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(2, 3, '2024-09-30')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(3, 4, '2024-10-02')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(4, 5, '2024-10-06')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(5, 6, '2024-10-08')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(6, 7, '2024-09-29')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(7, 8, '2024-10-01')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(8, 9, '2024-10-03')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(9, 10, '2024-10-07')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(10, 1, '2024-10-10');
```

Script of the database

```
-- ITEM table
CREATE TABLE ITEM (
    Item_id INT IDENTITY(1,1) PRIMARY KEY,   -- Auto-incrementing Item_id
    Title VARCHAR(64),
    Type VARCHAR(50)
);
      INSERT INTO ITEM (Title, Type) VALUES('The Great Gatsby', 'Book')
    INSERT INTO ITEM (Title, Type) VALUES('Introduction to Algorithms',
'Book')
    INSERT INTO ITEM (Title, Type) VALUES('National Geographic', 'Journal')
    INSERT INTO ITEM (Title, Type) VALUES('Artificial Intelligence Journal',
'Journal')
    INSERT INTO ITEM (Title, Type) VALUES('Digital Marketing Essentials',
'Digital Media')
    INSERT INTO ITEM (Title, Type) VALUES('The Catcher in the Rye', 'Book')
    INSERT INTO ITEM (Title, Type) VALUES('The New Yorker', 'Journal')
    INSERT INTO ITEM (Title, Type) VALUES('Quantum Computing', 'Digital
Media')
    INSERT INTO ITEM (Title, Type) VALUES('Biology Today', 'Journal')
    INSERT INTO ITEM (Title, Type) VALUES('Python Programming', 'Digital
Media')
```

```
-- BOOK table
CREATE TABLE BOOK (
    Item_id INT PRIMARY KEY,
    ISBN VARCHAR(32) UNIQUE,
    Author VARCHAR(64),
    Genre VARCHAR(50),
    CONSTRAINT BOOK_FK FOREIGN KEY (Item_id) REFERENCES ITEM(Item_id)
);
-- Inserting into BOOK table (with corresponding Item_id from ITEM)

    INSERT INTO BOOK (Item_id, ISBN, Author, Genre)VALUES(1, '9780141182636',
'F. Scott Fitzgerald', 'Fiction')
    INSERT INTO BOOK (Item_id, ISBN, Author, Genre)VALUES(2, '9780262033848',
'Thomas H. Cormen', 'Technology')
    INSERT INTO BOOK (Item_id, ISBN, Author, Genre)VALUES(6, '9780316769174',
'J.D. Salinger', 'Fiction');
```

```
-- JOURNAL table
CREATE TABLE JOURNAL (
    Item_id INT PRIMARY KEY,
    ISSN VARCHAR(32) UNIQUE,
    Volume VARCHAR(32),
    Issue VARCHAR(32),
    CONSTRAINT JOURNAL_FK FOREIGN KEY (Item_id) REFERENCES ITEM(Item_id)
);



-- Inserting into JOURNAL table (with corresponding Item_id from ITEM)

    INSERT INTO JOURNAL (Item_id, ISSN, Volume, Issue)VALUES(3, '0027-9358', 
'2024', 'March')
    INSERT INTO JOURNAL (Item_id, ISSN, Volume, Issue)VALUES(4, '1234-5678', 
'15', '2')
    INSERT INTO JOURNAL (Item_id, ISSN, Volume, Issue)VALUES(7, '0028-792X', 
'2024', 'July')
    INSERT INTO JOURNAL (Item_id, ISSN, Volume, Issue)VALUES(9, '2345-6789', 
'50', '4');





-- DIGITAL_MEDIA table
CREATE TABLE DIGITAL_MEDIA (
    Item_id INT NOT NULL,
    Format VARCHAR(50),
    Size VARCHAR(32),
    CONSTRAINT DIGITAL_MEDIA_PK PRIMARY KEY(Item_id),
    CONSTRAINT DIGITAL_MEDIA_FK FOREIGN KEY (Item_id) REFERENCES 
ITEM(Item_id)
);
-- Inserting into DIGITAL_MEDIA table (with corresponding Item_id from ITEM)

    INSERT INTO DIGITAL_MEDIA (Item_id, Format, Size)VALUES(5, 'MP4', 
'500MB')
    INSERT INTO DIGITAL_MEDIA (Item_id, Format, Size)VALUES(8, 'PDF', '10MB')
    INSERT INTO DIGITAL_MEDIA (Item_id, Format, Size)VALUES(10, 'EPUB', 
'5MB');
```

```
-- COPY table
CREATE TABLE COPY (
    C_ID INT IDENTITY(1,1) PRIMARY KEY,   -- Auto-incrementing C_ID
    Item_id INT,
    Location VARCHAR(40),
    Status VARCHAR(50),
    Condition VARCHAR(50),
    CONSTRAINT COPY_FK FOREIGN KEY(Item_id) REFERENCES ITEM(Item_id)
);
```

```
-- Inserting into COPY table

    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(1, 'Shelf A1', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(1, 'Shelf A1', 'Borrowed', 'Good')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(2, 'Shelf B2', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(3, 'Shelf C3', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(4, 'Shelf D4', 'Available', 'Good')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(5, 'Digital Shelf 1', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(6, 'Shelf A2', 'Available', 'Good')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(7, 'Shelf C2', 'Available', 'Good')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(8, 'Digital Shelf 2', 'Available', 'New')
    INSERT INTO COPY (Item_id, Location, Status, Condition)VALUES(9, 'Shelf C4', 'Available', 'New');
```

```sql
-- USERS table
CREATE TABLE Users (
    User_id INT IDENTITY(1,1) PRIMARY KEY,   -- Auto-incrementing User_id
    Username VARCHAR(64),
    Email VARCHAR(40) CHECK (Email LIKE '%@%.%'),
    Type VARCHAR(32)
);

    INSERT INTO Users (Username, Email, Type)VALUES('john_doe',
'john.doe@example.com', 'Student')
    INSERT INTO Users (Username, Email, Type)VALUES('jane_smith',
'jane.smith@example.com', 'Faculty')
    INSERT INTO Users (Username, Email, Type)VALUES('alice_jones',
'alice.jones@example.com', 'Staff')
    INSERT INTO Users (Username, Email, Type)VALUES('bob_martin',
'bob.martin@example.com', 'Student')
    INSERT INTO Users (Username, Email, Type)VALUES('susan_lee',
'susan.lee@example.com', 'Faculty')
    INSERT INTO Users (Username, Email, Type)VALUES('michael_wang',
'michael.wang@example.com', 'Staff')
    INSERT INTO Users (Username, Email, Type)VALUES('laura_clark',
'laura.clark@example.com', 'Student')
    INSERT INTO Users (Username, Email, Type)VALUES('paul_green',
'paul.green@example.com', 'Student')
    INSERT INTO Users (Username, Email, Type)VALUES('david_kim',
'david.kim@example.com', 'Faculty')
    INSERT INTO Users (Username, Email, Type)VALUES('lisa_white',
'lisa.white@example.com', 'Staff');


-- CONTACT table
CREATE TABLE CONTACT(
    User_id INT,
    Phone VARCHAR(10),
    CONSTRAINT CONTACT_PK PRIMARY KEY(User_id, Phone),
    CONSTRAINT CONTACT_FK FOREIGN KEY(User_id) REFERENCES Users(User_id)
);

    INSERT INTO CONTACT (User_id, Phone)VALUES(1, '1234567890')
    INSERT INTO CONTACT (User_id, Phone)VALUES(2, '2345678901')
    INSERT INTO CONTACT (User_id, Phone)VALUES(3, '3456789012')
    INSERT INTO CONTACT (User_id, Phone)VALUES(4, '4567890123')
    INSERT INTO CONTACT (User_id, Phone)VALUES(5, '5678901234')
    INSERT INTO CONTACT (User_id, Phone)VALUES(6, '6789012345')
    INSERT INTO CONTACT (User_id, Phone)VALUES(7, '7890123456')
    INSERT INTO CONTACT (User_id, Phone)VALUES(8, '8901234567')
    INSERT INTO CONTACT (User_id, Phone)VALUES(9, '9012345678')
    INSERT INTO CONTACT (User_id, Phone)VALUES(10, '0123456789');
```

```
-- STUDENT table

CREATE TABLE STUDENT (
    User_id INT PRIMARY KEY,
    Major VARCHAR(30) NOT NULL,
    Year CHAR(8),
    CONSTRAINT STUDENT_FK FOREIGN KEY (User_id) REFERENCES Users(User_id)
);
-- Inserting into STUDENT table

    INSERT INTO STUDENT (User_id, Major, Year)VALUES(1, 'Computer Science',
'2024')
    INSERT INTO STUDENT (User_id, Major, Year)VALUES(4, 'Mechanical
Engineering', '2023')
    INSERT INTO STUDENT (User_id, Major, Year)VALUES(7, 'Physics', '2025')
    INSERT INTO STUDENT (User_id, Major, Year)VALUES(8, 'Biology', '2024');



-- FACULTY table
CREATE TABLE FACULTY (
    User_id INT PRIMARY KEY,
    Department VARCHAR(40),
    Title VARCHAR(30),
    CONSTRAINT FACULTY_FK FOREIGN KEY (User_id) REFERENCES Users(User_id)
);

-- Inserting into FACULTY table

    INSERT INTO FACULTY (User_id, Department, Title)VALUES(2, 'Mathematics',
'Professor')
    INSERT INTO FACULTY (User_id, Department, Title)VALUES(5, 'Marketing',
'Assistant Professor')
    INSERT INTO FACULTY (User_id, Department, Title)VALUES(9, 'Computer
Science', 'Associate Professor');
```

```sql
-- STAFF table
CREATE TABLE STAFF (
    User_id INT PRIMARY KEY,
    Position VARCHAR(30),
    Office VARCHAR(30),
    CONSTRAINT STAFF_FK FOREIGN KEY (User_id) REFERENCES Users(User_id)
);
-- Inserting into STAFF table

    INSERT INTO STAFF (User_id, Position, Office)VALUES(3, 'Librarian',
'Library Main Office')
    INSERT INTO STAFF (User_id, Position, Office)VALUES(6, 'Lab Technician',
'Lab A3')
    INSERT INTO STAFF (User_id, Position, Office)VALUES(10, 'Admin
Assistant', 'Admin Block');



-- TRANSACTIONS table
CREATE TABLE TRANSACTIONS (
    Transac_id INT IDENTITY(1,1) PRIMARY KEY,    -- Auto-incrementing
Transac_id
    User_id INT,
    C_ID INT,
    Borrow_date DATE,
    Return_date DATE,
    Due_date DATE,
    CONSTRAINT TRANSACTIONS_FK1 FOREIGN KEY (User_id) REFERENCES
Users(User_id),
    CONSTRAINT TRANSACTIONS_FK2 FOREIGN KEY (C_ID) REFERENCES COPY(C_ID)
);

    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date,
Due_date)VALUES(1, 1, '2024-10-01', NULL, '2024-10-15')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date,
Due_date)VALUES(2, 2, '2024-09-25', '2024-10-02', '2024-10-01')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date,
Due_date)VALUES(4, 3, '2024-10-05', NULL, '2024-10-20')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date,
Due_date)VALUES(5, 4, '2024-10-01', '2024-10-12', '2024-10-10')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date,
Due_date)VALUES(7, 6, '2024-09-30', NULL, '2024-10-14')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date,
Due_date)VALUES(8, 7, '2024-10-08', NULL, '2024-10-22')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date,
Due_date)VALUES(9, 8, '2024-10-02', '2024-10-09', '2024-10-15')
    INSERT INTO TRANSACTIONS (User_id, C_ID, Borrow_date, Return_date,
Due_date)VALUES(10, 9, '2024-10-03', NULL, '2024-10-17');
```

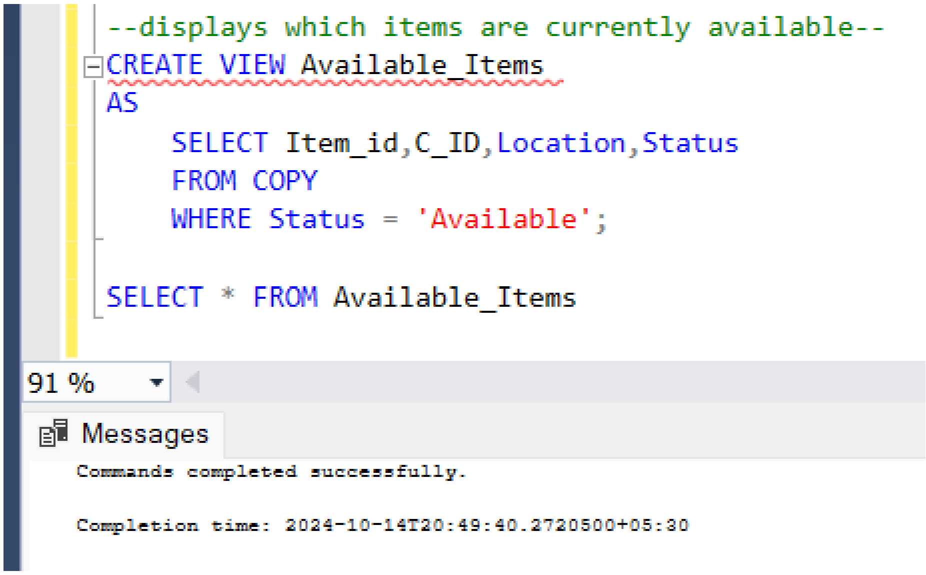
```sql
-- FINE table
CREATE TABLE FINE (
    Fine_id INT IDENTITY(1,1) PRIMARY KEY,   -- Auto-incrementing Fine_id
    Transac_id INT,
    User_id INT,
    Amount REAL,
    PaidStatus VARCHAR(10),
    CONSTRAINT FINE_FK1 FOREIGN KEY (User_id) REFERENCES Users(User_id),
    CONSTRAINT FINE_FK2 FOREIGN KEY (Transac_id) REFERENCES
TRANSACTIONS(Transac_id)
);


    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(1,1, 10,
'Unpaid')
    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(2,2, 5,
'Paid')
    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(3,4, 15,
'Unpaid')
    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(4,5, 20,
'Paid')
    INSERT INTO FINE (Transac_id,User_id, Amount, PaidStatus)VALUES(5,7, 25,
'Unpaid');
```

```sql
-- RESERVATION table
CREATE TABLE RESERVATION (
    User_id INT NOT NULL,
    Item_id INT NOT NULL,
    Res_date DATE,
    CONSTRAINT RESERVATION_PK PRIMARY KEY(User_id, Item_id),
    CONSTRAINT RESERVATION_FK1 FOREIGN KEY (User_id) REFERENCES
Users(User_id),
    CONSTRAINT RESERVATION_FK2 FOREIGN KEY (Item_id) REFERENCES ITEM(Item_id)
);
```

```sql
-- Inserting into RESERVATION table

    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(1, 2, '2024-10-05')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(2, 3, '2024-09-30')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(3, 4, '2024-10-02')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(4, 5, '2024-10-06')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(5, 6, '2024-10-08')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(6, 7, '2024-09-29')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(7, 8, '2024-10-01')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(8, 9, '2024-10-03')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(9, 10, '2024-10-07')
    INSERT INTO RESERVATION (User_id, Item_id, Res_date)VALUES(10, 1, '2024-10-10');
```

6. Create Views

I. Displays which Items are currently available

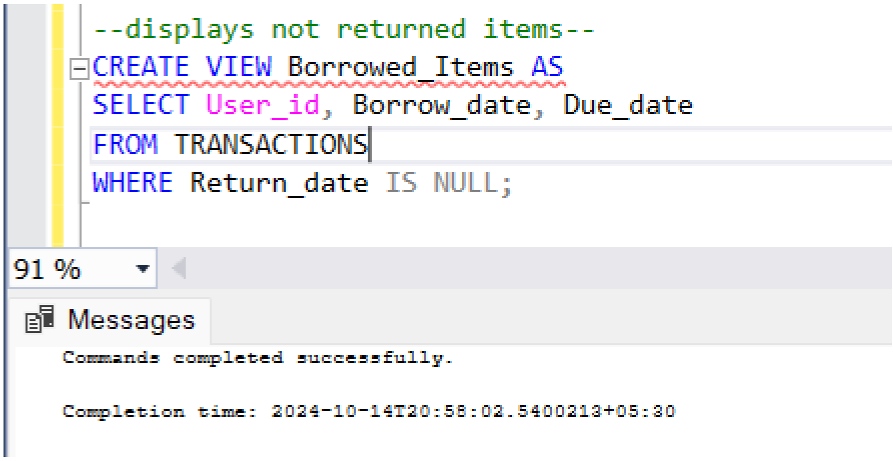


```
--displays which items are currently available--
CREATE VIEW Available_Items
AS
    SELECT Item_id,C_ID,Location,Status
    FROM COPY
    WHERE Status = 'Available';

SELECT * FROM Available_Items
```

```
Commands completed successfully.

Completion time: 2024-10-14T20:49:40.2720500+05:30
```

```
CREATE VIEW Available_Items AS
SELECT Item_id,C_ID,Location
FROM COPY
WHERE Status = 'Available';

SELECT * FROM Available_Items;
```

II. Displays information about the users of the items that have been borrowed but not yet returned



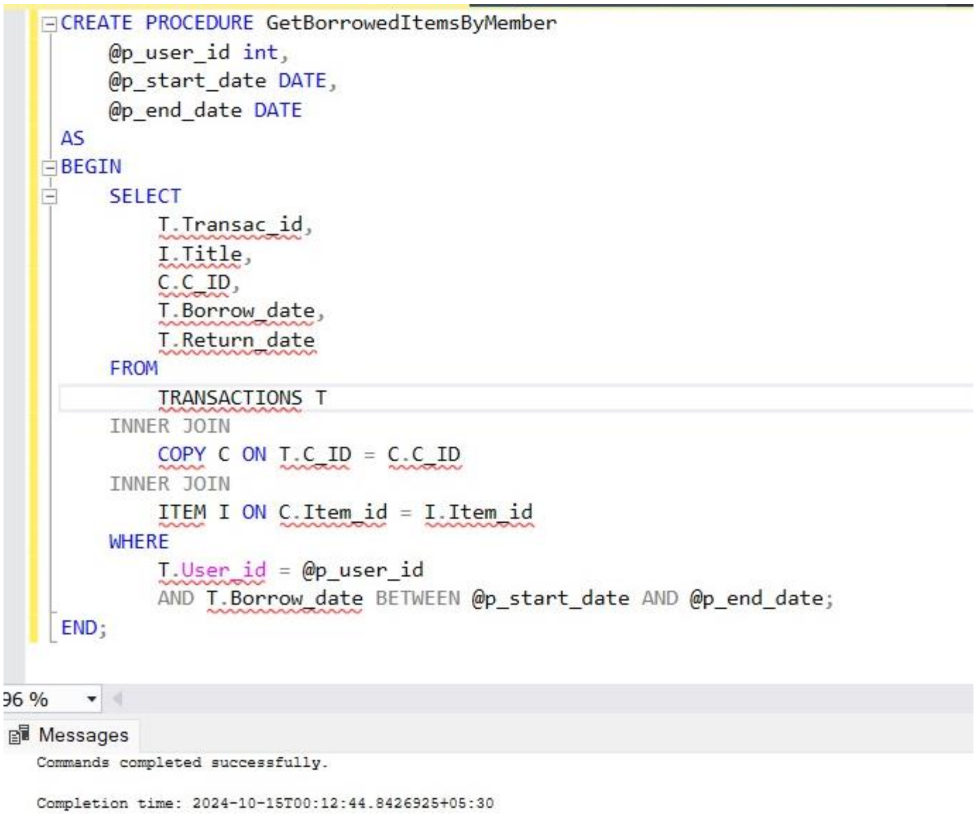
```
--displays not returned items--
CREATE VIEW Borrowed_Items AS
SELECT User_id, Borrow_date, Due_date
FROM TRANSACTIONS
WHERE Return_date IS NULL;
```

```
Commands completed successfully.

Completion time: 2024-10-14T20:58:02.5400213+05:30
```

```
CREATE VIEW Borrowed_Items AS
SELECT User_id, Borrow_date, Due_date
FROM TRANSACTIONS
WHERE Return_date IS NULL;

SELECT * FROM Borrowed_Items;
```

7. Stored Procedures

1) Retrieve the details of all the items borrowed by a given member within a given period.

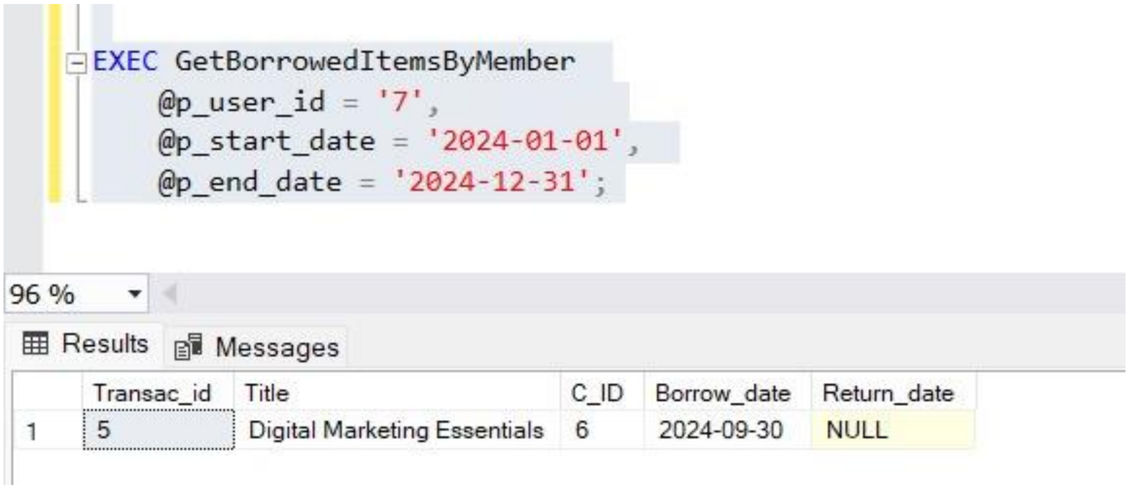


```
CREATE PROCEDURE GetBorrowedItemsByMember
    @p_user_id int,
    @p_start_date DATE,
    @p_end_date DATE
AS
BEGIN
    SELECT
        T.Transac_id,
        I.Title,
        C.C_ID,
        T.Borrow_date,
        T.Return_date
    FROM
        TRANSACTIONS T
    INNER JOIN
        COPY C ON T.C_ID = C.C_ID
    INNER JOIN
        ITEM I ON C.Item_id = I.Item_id
    WHERE
        T.User_id = @p_user_id
        AND T.Borrow_date BETWEEN @p_start_date AND @p_end_date;
END;
```

```
Commands completed successfully.

Completion time: 2024-10-15T00:12:44.8426925+05:30
```

```
CREATE PROCEDURE GetBorrowedItemsByMember
    @p_user_id int,
    @p_start_date DATE,
    @p_end_date DATE
AS
BEGIN
    SELECT
        T.Transac_id,
        I.Title,
        C.C_ID,
        T.Borrow_date,
        T.Return_date
    FROM
        TRANSACTIONS T
    INNER JOIN
        COPY C ON T.C_ID = C.C_ID
    INNER JOIN
        ITEM I ON C.Item_id = I.Item_id
    WHERE
        T.User_id = @p_user_id
        AND T.Borrow_date BETWEEN @p_start_date AND @p_end_date;
END;
```

Execution of the procedure.

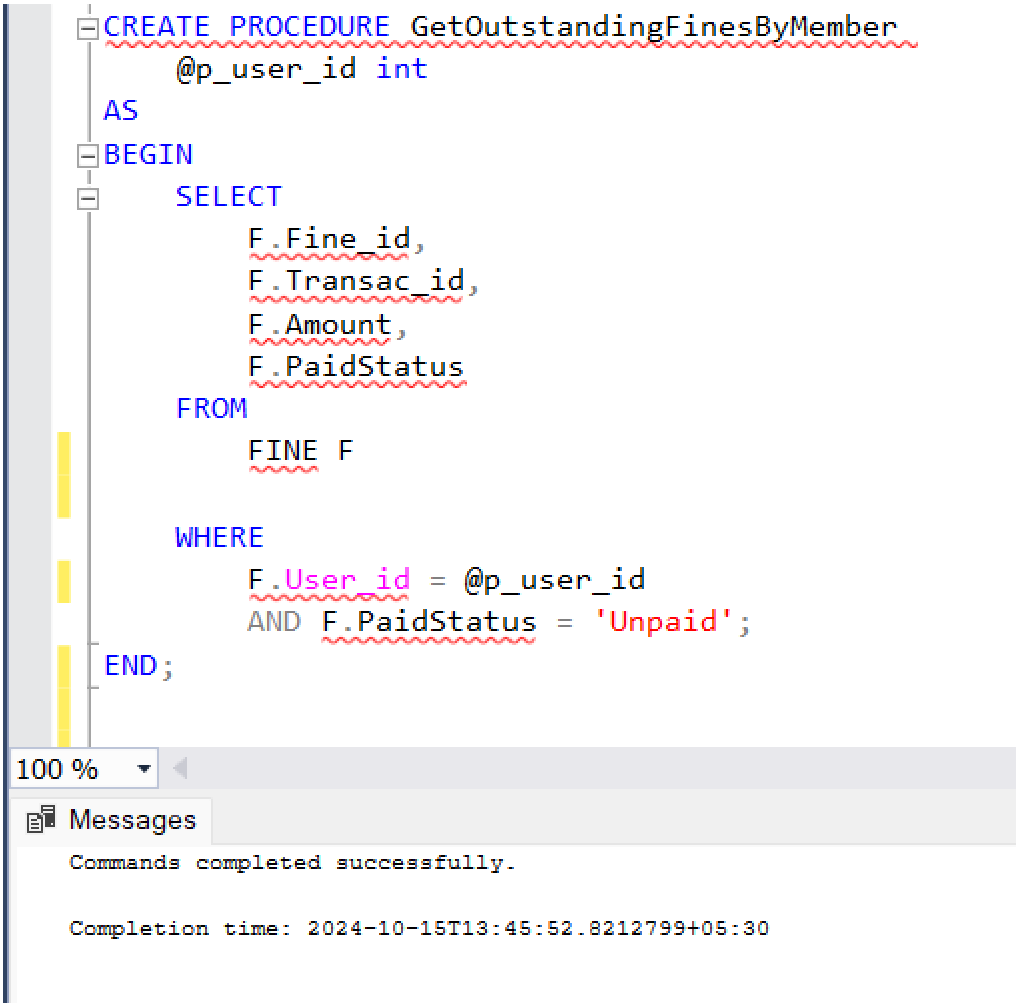


```
EXEC GetBorrowedItemsByMember
    @p_user_id = '7',
    @p_start_date = '2024-01-01',
    @p_end_date = '2024-12-31';
```

	Transac_id	Title	C_ID	Borrow_date	Return_date
1	5	Digital Marketing Essentials	6	2024-09-30	NULL

```
EXEC GetBorrowedItemsByMember
    @p_user_id = '7',
    @p_start_date = '2024-01-01',
    @p_end_date = '2024-12-31';
```

2) Retrieve the outstanding fines for a given member.

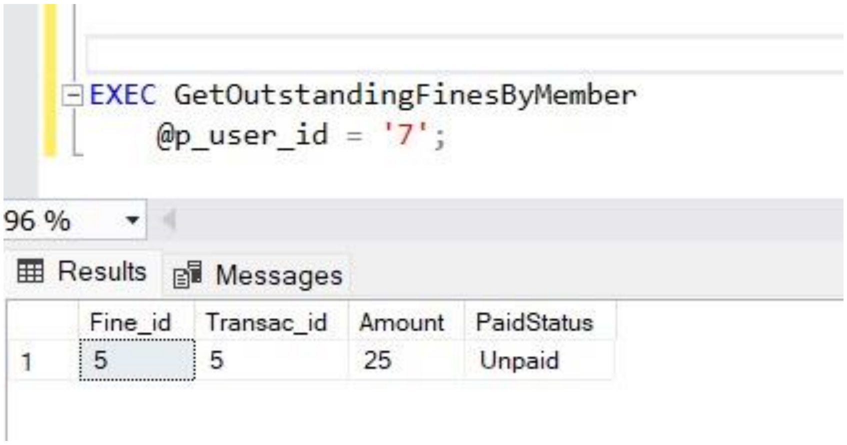


```sql
CREATE PROCEDURE GetOutstandingFinesByMember
    @p_user_id int
AS
BEGIN
    SELECT
        F.Fine_id,
        F.Transac_id,
        F.Amount,
        F.PaidStatus
    FROM
        FINE F

    WHERE
        F.User_id = @p_user_id
        AND F.PaidStatus = 'Unpaid';
END;
```

```
Commands completed successfully.

Completion time: 2024-10-15T13:45:52.8212799+05:30
```

```sql
CREATE PROCEDURE GetOutstandingFinesByMember
      @p_user_id int
AS
BEGIN
    SELECT
        F.Fine_id,
        F.Transac_id,
        F.Amount,
        F.PaidStatus
    FROM
        FINE F

    WHERE
        F.User_id = @p_user_id
        AND F.PaidStatus = 'Unpaid';
END;
```

Execution of the procedure



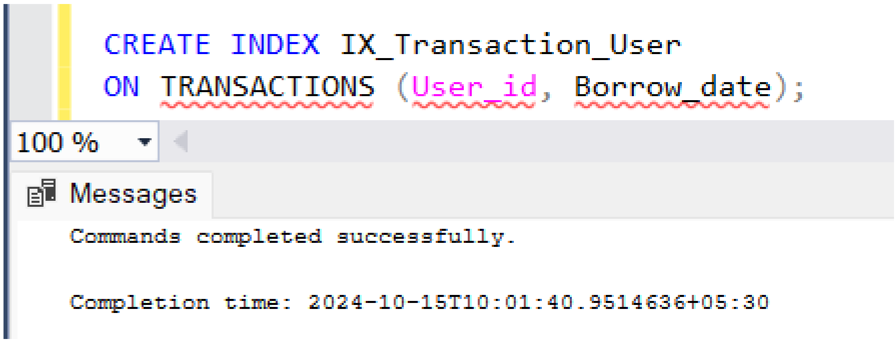
```
EXEC GetOutstandingFinesByMember
    @p_user_id = '7';
```

	Fine_id	Transac_id	Amount	PaidStatus
1	5	5	25	Unpaid

```
EXEC GetOutstandingFinesByMember
    @p_user_id = '7';
```

8. Create Indexes

I. Creates an index named **IX_Transaction_User** on the **TRANSACTIONS** table, which includes the columns **User_id** and **Borrow_date**

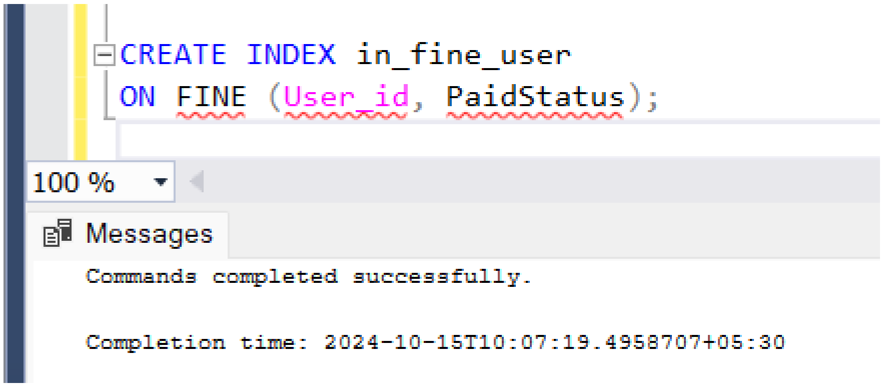


```
CREATE INDEX IX_Transaction_User
ON TRANSACTIONS (User_id, Borrow_date);
```

```
Commands completed successfully.

Completion time: 2024-10-15T10:01:40.9514636+05:30
```

```
CREATE INDEX IX_Transaction_User
ON TRANSACTIONS (User_id, Borrow_date);
```

II. Creates an index named **in_fine_user** on the **FINE** table, specifically on the columns **User_id** and **PaidStatus**



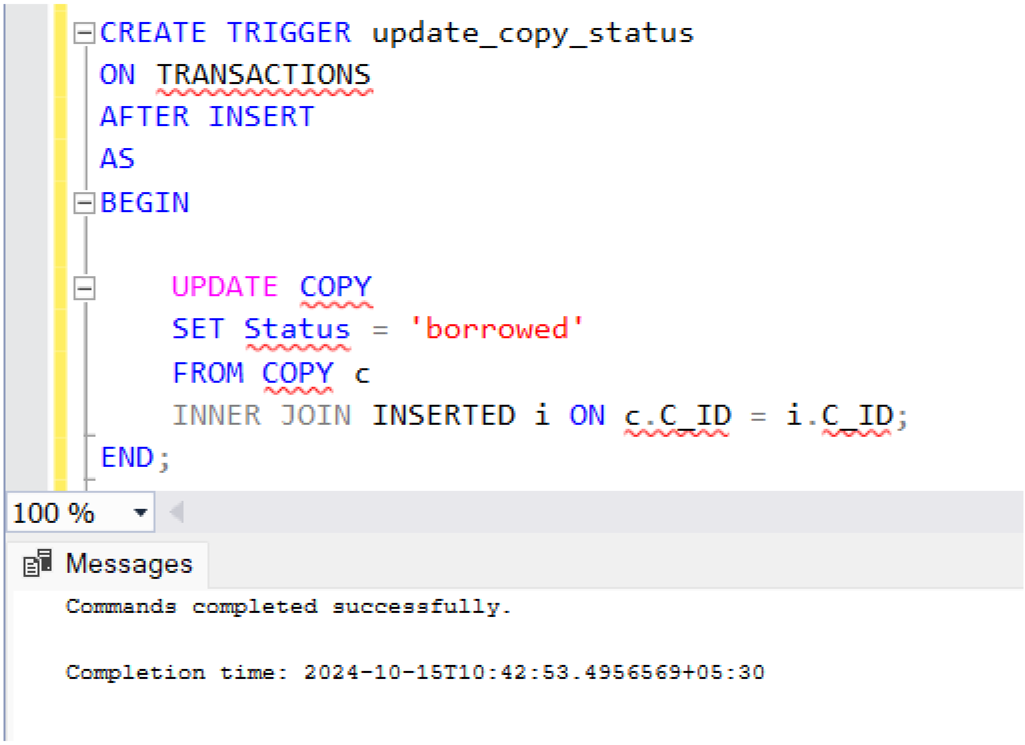
```
CREATE INDEX in_fine_user
ON FINE (User_id, PaidStatus);
```

```
Commands completed successfully.

Completion time: 2024-10-15T10:07:19.4958707+05:30
```

```
CREATE INDEX in_fine_user
ON FINE (User_id, PaidStatus);
```

9. Create Triggers

I. The trigger **update_copy_status** updates the status of an item copy when a new transaction is inserted (when an item is borrowed).

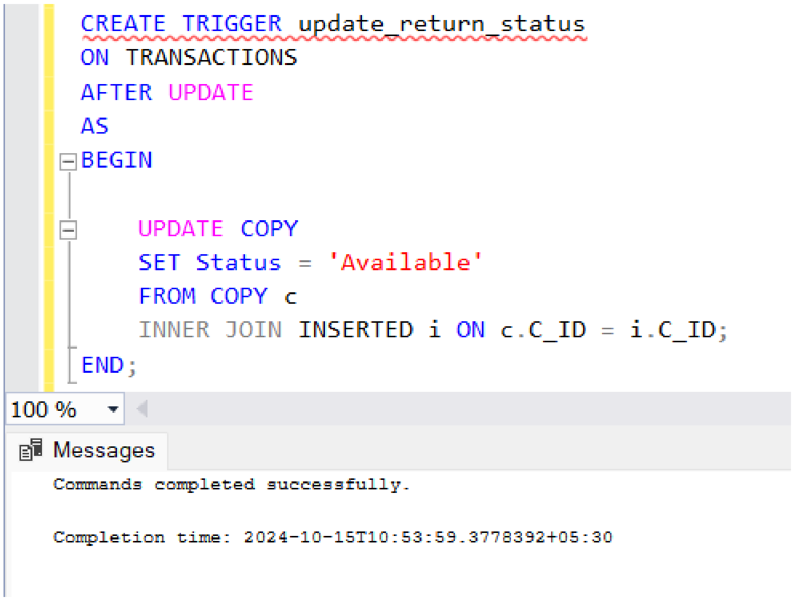


```sql
CREATE TRIGGER update_copy_status
ON TRANSACTIONS
AFTER INSERT
AS
BEGIN

    UPDATE COPY
    SET Status = 'borrowed'
    FROM COPY c
    INNER JOIN INSERTED i ON c.C_ID = i.C_ID;
END;
```

```
Commands completed successfully.

Completion time: 2024-10-15T10:42:53.4956569+05:30
```

```sql
CREATE TRIGGER update_copy_status
ON TRANSACTIONS
AFTER INSERT
AS
BEGIN

    UPDATE COPY
    SET Status = 'borrowed'
    FROM COPY c
    INNER JOIN INSERTED i ON c.C_ID = i.C_ID;
END;
```

II. The trigger **update_return_status** updates the status of an item copy when the transaction table is updated/when the return date updated (when an item is returned).



```
CREATE TRIGGER update_return_status
ON TRANSACTIONS
AFTER UPDATE
AS
BEGIN

    UPDATE COPY
    SET Status = 'Available'
    FROM COPY c
    INNER JOIN INSERTED i ON c.C_ID = i.C_ID;
END;
```

```
Commands completed successfully.

Completion time: 2024-10-15T10:53:59.3778392+05:30
```

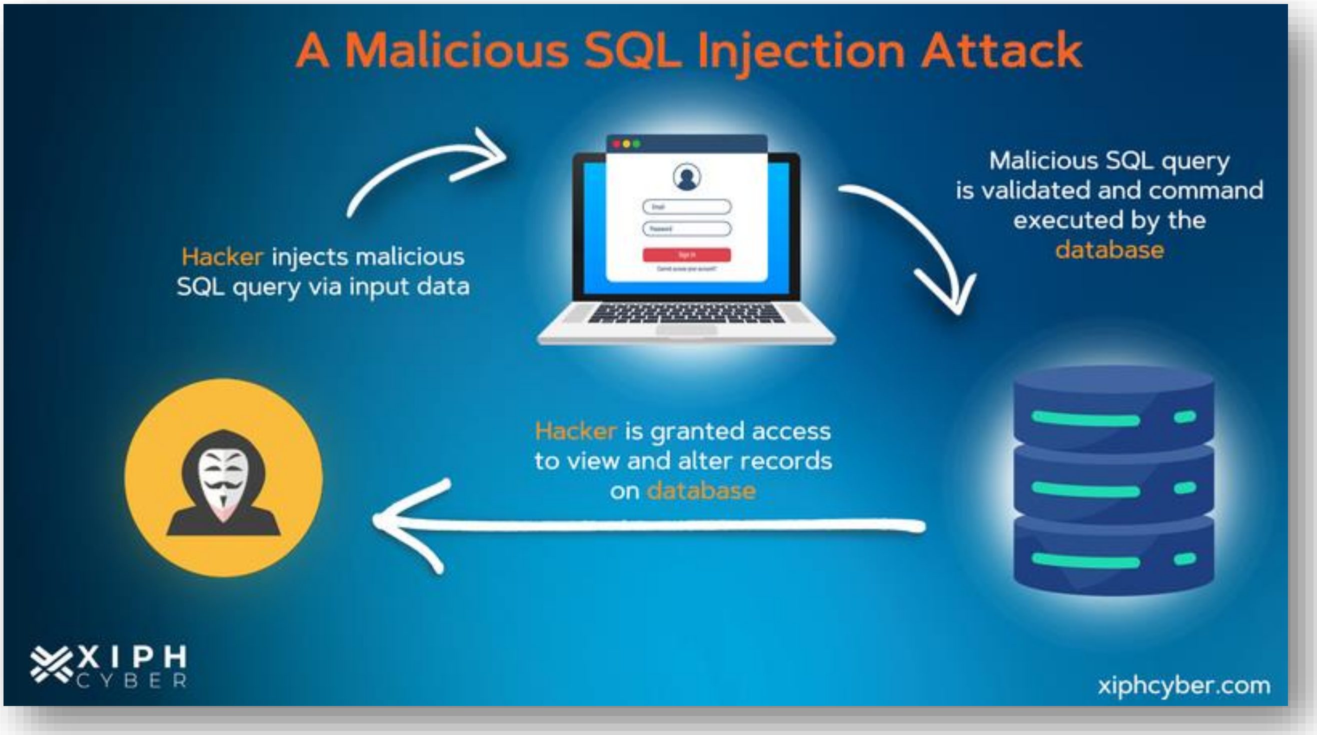
```
CREATE TRIGGER update_return_status
ON TRANSACTIONS
AFTER UPDATE
AS
BEGIN

    UPDATE COPY
    SET Status = 'Available'
    FROM COPY c
    INNER JOIN INSERTED i ON c.C_ID = i.C_ID;
END;
```

10. Data base Vulnerabilities

SQL Injections (SQLi)

SQLi or SQL injections is the most common and dangerous vulnerability that can find in most of the vulnerable databases. It usually happens when an intruder or an attacker can manipulate a poorly validate SQL query by injecting malicious SQL codes into input fields.

- Here is a sample figure that we can understand how SQLi works.


A Malicious SQL Injection Attack
Hacker injects malicious SQL query via input data
Malicious SQL query is validated and command executed by the database
Hacker is granted access to view and alter records on database
XIPH CYBER
xiphcyber.com

Impacts of an SQL injection attack

- Unauthorized access to sensitive data

If a database is affected with an SQLi it will violates the CIA which stands for,

I. Confidentiality

II. Integrity

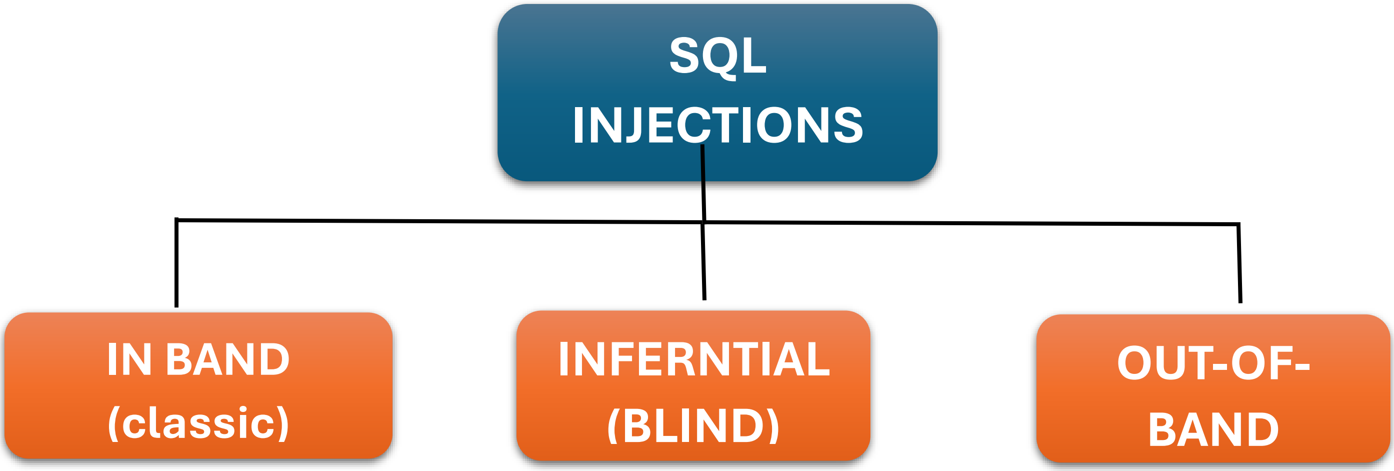
III. Availability

- Remote code execution on the OS

Hacking Code Execution (RCE) through SQL injections raises security concerns as this is a skill that enables an unauthorized user deploys code on a remote server**.**

Types of SQL Injections

There are several types of SQL injections attacks below diagram clearly depicts those types.



In-Band SQL Injections

- In- Band SQLi usually happens when the attacker uses the same communication channel to both launch the attack and gather the result of the attack.
- This type of attack is easiest to make exploit comparing to other SQLi attack types.
- There are two types of in-bands SQLi

a) Error-based SQLi
b) Union-based SQLi

Inferential (Blind) SQL Injection

- This type of SQLi vulnerability where there is no actual transfer of data via the web application.
- This type of attack has the same risk level as the in-band SQL injection
- The only demerit of this kind of attack is that it takes longer to exploit comparing to in-band SQL injection
- There are two types of blind SQLi's

    a) Boolean-based SQLi
    b) Time-based SQLi

Out-of-Band (OAST) SQL Injections

- Out-of-Band is a vulnerability that consist of that executes an out-of-band network connection to a system that we control
-
- This type of attack is very rare, and it uses variety of protocols like (ex: DNS, HTTP)

Mitigations to SQL Injection attacks

There are several countermeasures we can apply to prevent SQLi attacks:

- **Input Validation**: All the inputs should be validated against expected formats, as an example if the input should be an integer, ensure that only integer values are allowed.

- **Parameterized Queries:** using parameterized queries will ensure that the inputted values are treated as data.

  Ex: SELECT * FROM users WHERE username = ?;

- **Escaping Input:** When using special characters in user inputs, such as quotes, should be properly escaped before incorporating them to SQL.

- **Use of ORM (Object oriented Mapping):** This framework abstract SQL queries and helps to prevent injection by automatically parameterizing input.

- **Least Privilege Principle:** This ensures that the database account have minimum privileges. Even if an attacker manages to exploit SQLi, this limits the damage they can cause [1].

Privilege Escalation

Overview:

Privilege escalation is a kind of attack where an individual raises their access levels more than is permitted. It can be either horizontal (cross access of other user's data) or vertical (gain rights to administer other users' accounts). Lateral movement can be enabled due to poorly designed system settings or deficiencies that exist in a database management system (DBMS).

Common Techniques for privilege escalation

- **Exploiting stored procedures:**

  Some databases consist of stored procedures that runs with very high-level privileges. If these privileges are not in a securely manner, or if they are not used any cryptographic methods the attackers or intruders can manipulate those procedures to perform unauthorized actions.

- **Weak permission configuration:**

  If a database has given access to overly permissive user role, the attackers or insiders can gain access to the areas of the database that should be critically secured.

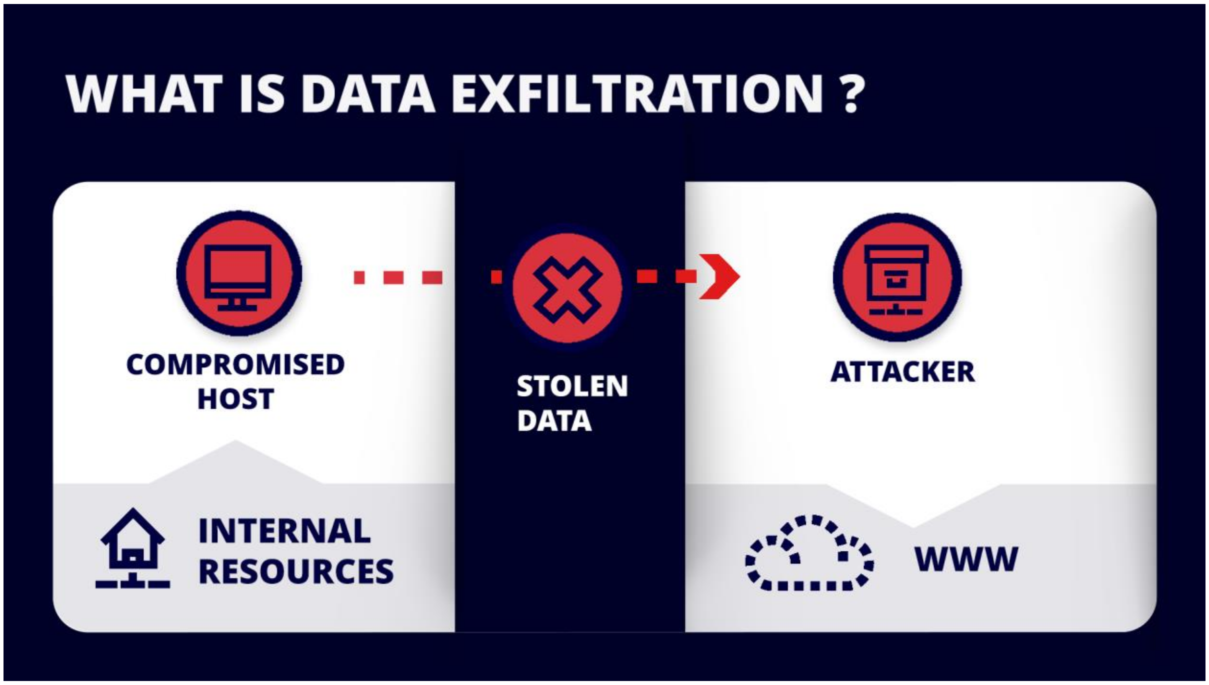
- **Exploiting software vulnerabilities:**

  Sometimes the DBMS consists of vulnerabilities that ca be exploited for privilege escalation.

  Ex: Sometimes an attacker can be exploiting a buffer overflow vulnerability in the database to execute arbitrary code with elevated privileges.

Impacts of privilege escalation

a) Data exfiltration



Above picture clearly depicts what is a data exfiltration. Data exfiltration basically means an attacker, or an intruder can gain access to unauthorized areas which can contain critical or very sensitive data like personal information, encryption keys, and financial records, bank details etc.

a) Database Server Control

After gaining the access of the administrative controls for the database, attackers can perform different tasks. They can create new accounts, disable security controls and manipulate or delete critical data.

b) Persistence

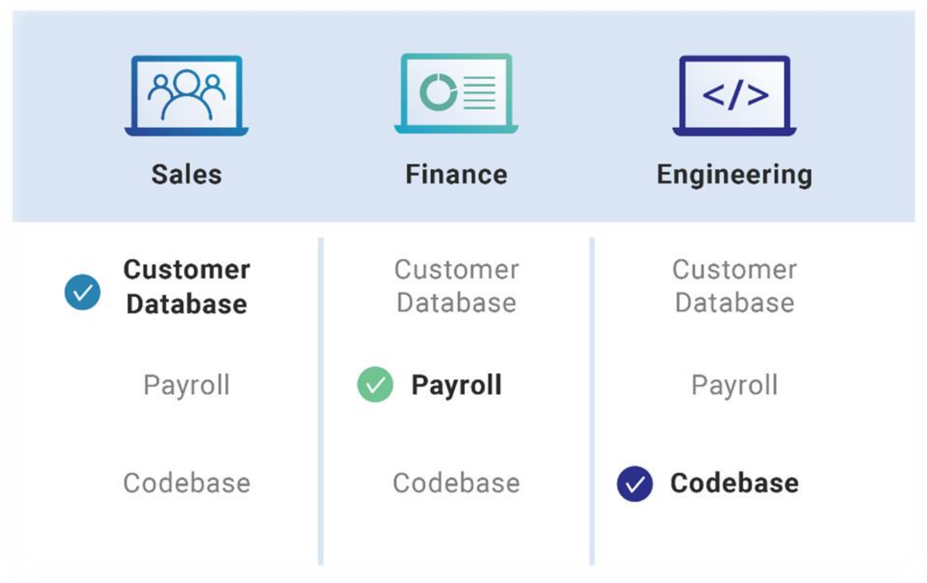
Attacker may leave backdoors or create new privileged accounts to maintain access to the system after the initial breach has been mitigated [2].

Mitigations and preventive controls

There are several preventive controls and mitigations related to privilege escalation:

I. Role-based access control (RBAC):

RBAC means set up strict policies to ensure only the authorized person has access to the DB to perform operations according to their respective role. In a database its very important to set up administrative actions to a minimal set of administrators.



Sales	Finance	Engineering
✓ Customer Database	Customer Database	Customer Database
Payroll	✓ Payroll	Payroll
Codebase	Codebase	✓ Codebase

As an example, the above image depicts how the RBAC works. In customer database only the authorized users from the sales have the access and they are not authorized for other databases. The same theory goes to other databases according to their respective department.

II. Regular Security Audits

This is a very important when preventing privilege escalation. It is essential to conduct frequent security audits to identify security vulnerabilities and understand unusual behaviours in databases like unauthorized access.

III. Patch Management

When preventing from privilege escalation patch management is also a very important mitigate. Keeping the DBMS software up to date with the latest security patches to face unknown vulnerabilities that could cause for privilege escalation.

IV. Stored procedure security

This ensures that stored procedures are securely written and run with least amount of privilege necessary. Procedures that require high level access should be thoroughly reviewed and monitored for suspicious behaviour [4].

11. References

[1] J. Clarke, "Defending Against SQL Injection Attacks," IEEE Security & Privacy, vol. 18, no. 3, pp. 12-20, May 2022. A Reclosing Scheme of Hybrid DC Circuit Breaker for MMC-HVdc Systems | IEEE Journals & Magazine | IEEE Xplore

[2] A. Shah, "Database Security and Privilege Escalation," IEEE Transactions on Information Forensics and Security, vol. 17, no. 8, pp. 1234-1245, August 2021. Accurate and Robust Malware Detection: Running XGBoost on Runtime Data From Performance Counters | IEEE Journals & Magazine | IEEE Xplore

[3] M. Morgan, "Best Practices for Database Security: A Comprehensive Guide," IEEE Computer Society Press, 2020. https://www.computer.org/publications/digital-library

Sri Lanka Institute of Information Technology

BSc Honors in Information Technology Specializing in Cyber Security

IE2042- Database Management Systems for Security

July 2024

Group Assignment - WBS

Database Design, Implementation and Security

	Student ID 1	Student ID 2	Student ID 3	Student ID 4
Task 1				
Properly Documented Assumptions	IT23184312	IT23170520	IT23187832	IT23169708
ERD and Logical Model	IT23184312	IT23170520	IT23187832	IT23169708
Normalization		IT23170520		IT23169708
Table Implementations	IT23184312			
Constraints Implementation		IT23170520		
2 Triggers	IT23184312			
2 Views			IT23187832	
2 Indexes		IT23170520		
2 Stored Procedures				IT23169708
Task 2				
Description and analysis of 2 database vulnerabilities	IT23184312	IT23170520	IT23187832	IT23169708
Mitigation and countermeasure suggestions	IT23184312	IT23170520	IT23187832	IT23169708
Total				

IT 23 1843 12	**IT 23 1705 20**	**IT 23 1878 32**	**IT 23 1697 08**
AMANTHA M A	**RAJASOORIYA D G C H**	**PERAMUNUGAMA M R A**	**PERERA P A J M**