CineFlick Documentation

Created By: Gabriel Henderson

Course: Dr. Atluri CS 488 Introduction to Databases Fall 2025

Table of Contents

- Introduction 3
- Database Requirements..... 4
- Technology Used.....5
- Conceptual Schema (ER-Diagram) 6
- Relational Schema..... 7
- Normalized Relational Schema.....8
- Data Dictionary..... 12
- SQL Commands.....22

Introduction

This project is the implementation of a database for a movie theatre chain named CineFlick. The database was designed to manage and handle the day-to-day ins and outs of a real life functional database in the most efficient way possible. To ensure efficiency for this database I utilized database design concepts and methodologies I learned from the course CS 488 Introduction to Databases i.e. Normalization, conceptual schema to visualize and design everything before writing any code, Data Dictionaries to properly structure my data for best case implementation. This project is an amalgamation of everything I learned in this course applied to solve a real world issue in the best ways possible.

Database Requirements

-**Staff and Payroll Management:** The system must efficiently track employee and payroll data with information like what theater that employee works at what role they are in tenure with the company and for Payroll the system must be able to manage salaries, hourly wages, deductions, pay period beginnings and end.

- **Concessions Inventory Management:** The system must manage stock levels and pricing for items in concessions at each theater. Having this functionality should allow management to run reports and Id stock items.

- **Customer and Loyalty Tracking:** The database will support customer retention via the CineFlick membership program. Loyalty data will be located here allowing the theater to track tier levels and accumulated points.

- **Transaction Reporting:** Each booking will be recorded in the reservation table with details on number of tickets, seat numbers and crucial data such as payment status which will help the theater maximize profits based on attendance and scheduling data.

- **Scheduling:** the system will help with scheduling films for peak efficiency and to maximize getting the most in demand films on as many screens as possible to draw in as many customers that we can. The database will track movie release dates, ratings, duration and as well as what films are showing in which theaters specifically.

Technology Used

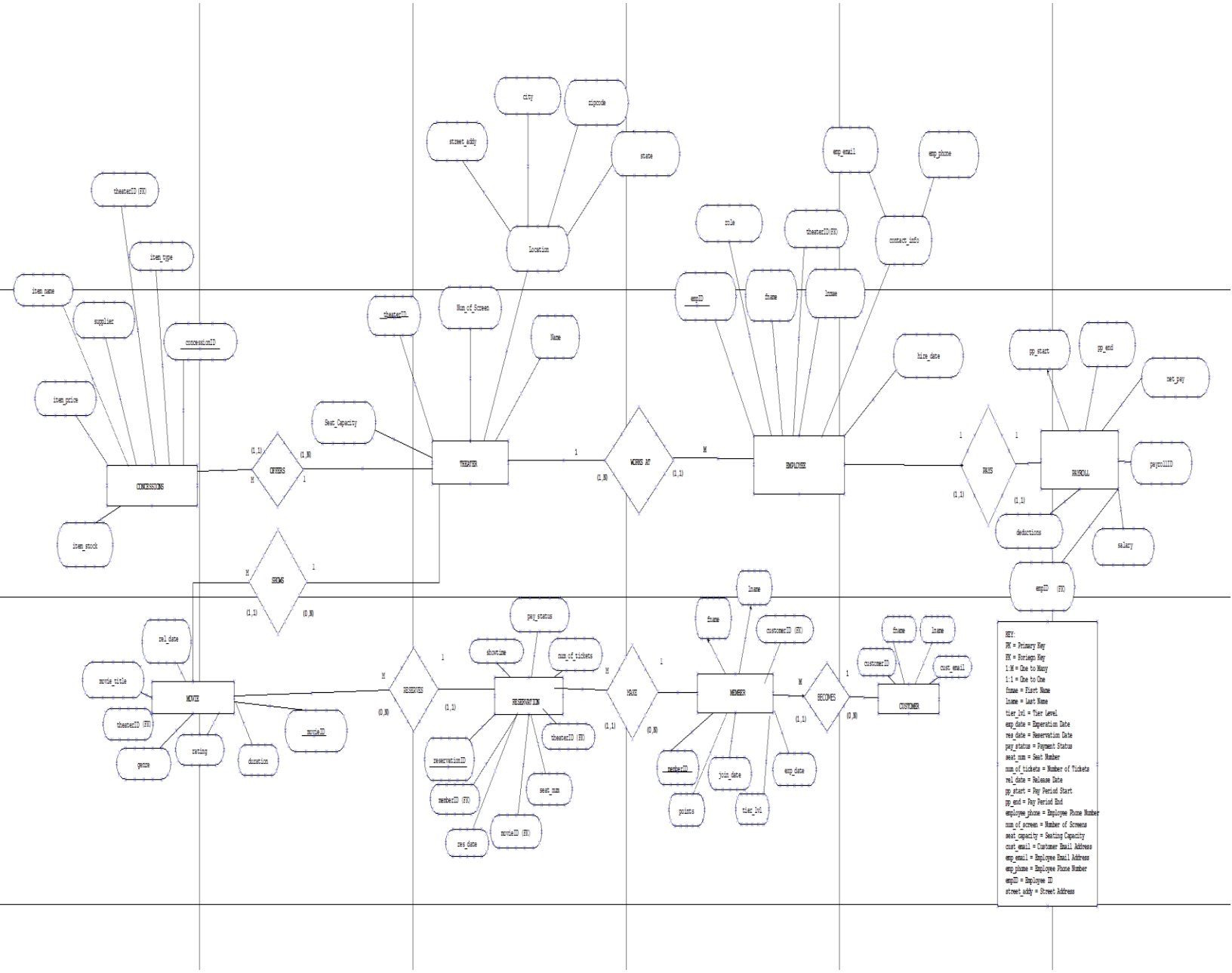
Database Management System (DBMS)

- The database for Cineflick was implemented by utilizing Microsoft Access. I chose Access because it was simple to use and has everything you need to implement a database in one place which in turn allowed for rapid development of tables, SELECT and INSERT INTO commands per the projects requirements.

Language and Design Tools

- Structured Query Language (SQL): SQL was used to create the database in Access via DDL and DML commands.
- Conceptual Design: I utilized Dia Diagram Editor to create my Entity Relationship diagram which helped with mapping out all the entities and their relationships prior to database implementation.

Conceptual Schema (ER-Diagram)



Relational Schema

Theatre (theaterID, name, num_of_screen, seat_capacity, street, city, zip, state)

- The composite attribute is replaced by its broken down components location --> street_addy, city, state, zipcode

Customer (customerID, fname, lname, cust_email)

Membership (memberID, tier_lvl, points, join_date, exp_date, customerID)

- customerID is an FK from the Customer Table

Employee (empID, role, fname, lname, theatreID, emp_email, emp_phone, hire date)

- theaterID is the FK from the theatre table,

- The composite attribute is replaced by its broken down components contact_info -- > emp_email and emp_phone

Payroll (payrollID, pp_start, pp_end, salary, net_pay, deductions empID)

- EmpID is the FK from the Employee table

Concessions (concessionID, item_name, item_type, item_price, supplier, theaterID, item_stock)

- TheaterID is the FK from the Theater table

Movie (movieID, title, genre, duration, rating, rel_date, theaterID)

- TheaterID is the FK referencing the Theater table

Reservation (resID, showtime, num_of_tickets, seat_num, res_date, pay_status, memberID, movieID, theaterID)

- Where memberID is an FK referencing Membership, movieID is an FK referencing Movie, and theaterID is an FK referencing Theatre

Normalized Relational Schema

- First Normal Form (1NF): Table format, no repeating groups and pk identified
- Second Normal Form (2NF): 1NF and no partial dependencies
- Third Normal Form (3NF): 2NF and no transitive dependencies

I. First Normal Form (1NF)

Rule for 1NF: A table must have no repeating groups of data, and all attributes must contain atomic (single, indivisible) values.

All tables in the final CineFlick design satisfy **1NF** because the initial design phase avoided multi-valued attributes and composite columns.

1NF Verification Example: MOVIE Table

Attribute	Definition	Atomic?
movieID	Unique identifier for a film.	Yes
title	Single text value for the movie name.	Yes
duration	Single integer represents length.	Yes

- **Conclusion:** Since no table stores lists of values or multi-valued attributes, all entities, including **THEATRE**, **CUSTOMER**, **MOVIE**, and **RESERVATION**, are successfully in **1NF**.

II. Second Normal Form (2NF)

Rule for 2NF: The table must be in 1NF **AND** all non-key attributes must be fully dependent on the **entire Composite Primary Key (PK)**.

This check applies specifically to tables with Composite Keys. In this schema, the **INVENTORY** table uses a Composite Key and requires this check.

Violation and Solution: INVENTORY --> CONCESSIONS Split

Unnormalized Table State (Violates 2NF)

If item details were kept at inventory levels, Partial **Dependency** would exist:

Table	Composite Primary Key	Non-Key Attribute	Partial Dependency

INVENTORY (Initial)	(**theaterID**, **concessionsID**)	unit_price	**concessionsID** --> unit_price

The unit_price depends only on the item itself (**concessionsID**), not the location (**theaterID**). This is a violation.

Normalized Solution (Achieves 2NF)

The partially dependent attributes are moved to a new table where the partial key becomes the new Primary Key.

Final 2NF Table	Key Type	Attributes (Keys Bolded)	Dependency Proof
INVENTORY	Composite PK	**theaterID**, **concessionsID**, stock_level	stock_level requires *both* keys for definition (fully dependent).
CONCESSIONS	Simple PK	**concessionsID**, item_name, unit_price, supplier	All attributes are fully dependent on **concessionsID**.

- **Conclusion:** By separating the item definition from the location-specific stock level, the schema is confirmed to be in **2NF**.

III. Third Normal Form (3NF)

Rule for 3NF: The table must be in 2NF **AND** there must be no **Transitive Dependencies**.

A **Transitive Dependency** exists when one non-key attribute determines another non-key attribute (e.g., A --> B --> C, where B and C are non-key).

3NF Proof 1: Employee and Payroll Isolation

This addresses the risk of redundancy caused by common job roles having the same salary data.

Unnormalized Dependency (Violates 3NF)

If the rate were stored in **EMPLOYEE**, the following transitive chain exists:

empID --> role --> hourly_rate (Transitive Dependency)

Normalized Solution (Achieves 3NF)

The pay data is isolated into the **PAYROLL** entity.

Final 3NF Table	Role	Attributes (Keys Bolded)	Dependency Proof
EMPLOYEE	Determinant	**empID**, theaterID, fname, lname, role, hire_date	All non-key attributes are fully and directly dependent on **empID**.
PAYROLL	Dependent Data	**payrollID**, empID (FK), gross_salary, hourly_rate, pay_freq	This table isolates the pay data. It is directly dependent on its PK (**payrollID**) and links back via empID, breaking the transitive chain originating from role.

3NF Proof 2: Customer and Membership Isolation

This addresses the risk of storing volatile loyalty data (points, tier) with stable personal data (name, email).

Unnormalized Dependency (Violates 3NF)

If all attributes were in one table:

customerID --> tier_lvl --> points (Transitive Dependency)

Normalized Solution (Achieves 3NF)

Final 3NF Table	Role	Attributes (Keys Bolded)	Dependency Proof
CUSTOMER	Stable Data	**customerID**, fname, lname, email	Contains only stable contact information directly dependent on **customerID**.
MEMBERSHIP	Volatile Data	**memberID**, customerID (FK), tier_lvl, points	Contains attributes (like points) that change frequently, isolated from the core customer record and directly dependent on **memberID**.

Summary of All Remaining Tables

The following simple entities are confirmed to be in **3NF** as they contain only simple primary keys and no transitive dependencies:

Table	Primary Key	Key Attributes
THEATRE	**theaterID**	All attributes depend only on **theaterID**.
MOVIE	**movieID**	All attributes depend only on **movieID**.
RESERVATION	**reservationID**	All attributes depend only on **reservationID**.

- **Final Conclusion:** By satisfying **1NF**, **2NF** (via the **INVENTORY** split), and **3NF** (via the **EMPLOYEE/PAYROLL** and **CUSTOMER/MEMBERSHIP** splits), the CineFlick database schema is fully compliant with **Third Normal Form (3NF)**.

Data Dictionary

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK	FK	FK REFERENCED TABLE
THEATRE	TheatreID	Unique identifier for the theatre location.	CHAR(5)	XXXXX	A1001-Z9999	Y	PK		
	Name	Official name of the theatre.	VARCHAR(50)	X(50)	Alphanumeric	Y			
	Location	Physical address or district.	VARCHAR(255)	X(255)	Full Address	Y			
	Number_Of_Screens	Count of auditoriums in the theatre.	INT	99	1-20	Y			

	Seating_Capacity	Total seating capacity of the theatre.	INT	99999	50-5000	Y			
MOVIE	MovieID	Unique identifier for the movie title.	CHAR(5)	XXXXX	A0001-Z9999	Y	PK		
	Title	Full title of the movie.	VARCHAR(255)	X(255)	Alphanumeric	Y			
	Genre	Primary genre classification.	VARCHAR(50)	X(50)	e.g., Action, Comedy	Y			
	Duration	Runtime of the movie in minutes.	INT	999	60-300 min	Y			
	Rating	MPAA or equivalent film rating.	CHAR(5)	XXXXX	e.g., PG, R, 18+	Y			
	Release_Date	Official release date of the movie.	DATE	DD/MM/YYYY	N/A	Y			
CUSTOMER	CustomerID	Unique identifier for a	CHAR(10)	X(10)	N/A	Y	PK		

		customer.							
	First_Name	Customer's first name.	VARCHAR(50)	X(50)	N/A	Y			
	Last_Name	Customer's last name.	VARCHAR(50)	X(50)	N/A	Y			
	Email	Customer's contact email address.	VARCHAR(255)	X(255)	Valid Email	Y			
MEMBERSHIP	MembershipID	Unique ID for the loyalty membership record.	CHAR(10)	X(10)	N/A	Y	PK		
	CustomerID	Foreign Key to link to the customer.	CHAR(10)	X(10)	N/A	Y		FK	**CUSTOMER**
	Tier_Level	The current membership level.	VARCHAR(20)	X(20)	e.g., Gold, Platinum	Y			
	Points_Balance	Accumulated	INT	9999999	0-9,999,999	Y			

		loyalty points.							
	Join_Date	Date the membership started.	DATE	DD/MM/YYYY	N/A	Y			
	Expiration_Date	Date the membership expires.	DATE	DD/MM/YYYY	Future Date	N			
EMPLOYEE	EmployeeID	Unique ID for an employee.	CHAR(10)	X(10)	N/A	Y	PK		
	TheatreID	Foreign Key for the theatre they work at.	CHAR(5)	XXXXX	N/A	Y		FK	**THEATRE**
	First_Name	Employee's first name.	VARCHAR(50)	X(50)	N/A	Y			
	Last_Name	Employee's last name.	VARCHAR(50)	X(50)	N/A	Y			
	Role	Employee's job title/role.	VARCHAR(50)	X(50)	e.g., Manager, Usher, Cleaner	Y			
	Hire_Date	Date the employe	DATE	DD/MM/YYYY	N/A	Y			

		e was hired.							
	Contact_Info	Employee's phone or secondary contact.	VARCHAR(100)	X(100)	N/A	Y			
PAYROLL	PayrollID	Unique ID for a payroll transaction record.	CHAR(10)	X(10)	N/A	Y	PK		
	EmployeeID	Foreign Key to link to the employee.	CHAR(10)	X(10)	N/A	Y		FK	**EMPLOYEE**
	PayPeriod_Start	Start date of the pay period.	DATE	DD/MM/YYYY	N/A	Y			
	PayPeriod_End	End date of the pay period.	DATE	DD/MM/YYYY	N/A	Y			
	Salary_Amount	Gross salary for the pay period.	INT	9999999.99	>= 0.00	Y			

	Deductions	Total deductions amount.	INT	9999999.99	>= 0.00	Y			
	NetPay	Final amount paid to the employee.	INT	9999999.99	>= 0.00	Y			
CONCESSIONS	ConcessionID	Unique ID for an item in the menu.	CHAR(5)	XXXXX	N/A	Y	PK		
	Name	Name of the concession item.	VARCHAR(50)	X(50)	e.g., Large Popcorn	Y			
	Type	Category of the concession item.	VARCHAR(50)	X(50)	e.g., Food, Drink, Candy	Y			
	Price	Selling price of the item.	NUMBER	999.99	>= 0.01	Y			
	Stock_Quantity	Current quantity in inventory.	INT	99999	>= 0	Y			
	Supplier	Vendor who	VARCHAR(100)	X(100)	N/A	Y			

		supplies the item.							
RESERVATION	ReservationID	Unique ID for a single customer booking.	CHAR(10)	X(10)	N/A	Y	PK		
	CustomerID	FK for the customer who made the reservation.	CHAR(10)	X(10)	N/A	Y		FK	**CUSTOMER**
	MovieID	FK for the movie being viewed.	CHAR(5)	XXXXX	N/A	Y		FK	**MOVIE**
	TheatreID	FK for the theatre hosting the showing.	CHAR(5)	XXXXX	N/A	Y		FK	**THEATRE**
	Showtime	Scheduled start time of the movie.	TIME	HH:MM:SS	N/A	Y			

	Number_Of_Tickets	Count of tickets in the reservation.	INT	99	1-10	Y			
	Reservation_Date	Date the reservation was placed.	DATE	DD/MM/YYYY	N/A	Y			
	Payment_Status	Status of the payment for the booking.	VARCHAR(20)	X(20)	Paid, Pending, Cancelled	Y			
CONCESSION_ORDER	ReservationID	Composite Primary Key and FK to Reservation.	CHAR(10)	X(10)	N/A	Y	PK	FK	**RESERVATION**
	ConcessionID	Composite Primary Key and FK to Concessions.	CHAR(5)	XXXXX	N/A	Y	PK	FK	**CONCESSIONS**
	Quantity	Number of this specific item purchased.	INT	99	1-99	Y			

	Total_Price	Total cost for this item quantity.	INT	99999.99	>= 0.00	Y			
RESERVED_SEAT	ReservationID	Composite Primary Key and FK to Reservation.	CHAR(10)	X(10)	N/A	Y	PK	FK	**RESERVATION**
	Seat_Number	Composite Primary Key for the seat number (e.g., A1, F12).	VARCHAR(5)	XXXXX	N/A	Y	PK		

KEY:

Abbreviation	Meaning
PK	Primary Key — uniquely identifies each record in a table.
FK	Foreign Key — references a primary key in another table to establish a relationship.
CHAR(n)	Fixed-length character data type; stores exactly *n* characters.

VARCHAR(n)	Variable-length character data type; stores up to *n* characters.
NUMBER(p,s)	Numeric data type with precision *p* (total digits) and scale *s* (digits after the decimal).
DATE	Stores calendar dates (day, month, year).
TIME	Stores time values (hours, minutes, seconds).
DD/MM/YYYY	Date format: Day / Month / Year.
HH:MM:SS	Time format: Hours / Minutes / Seconds (24-hour format).
X(n)	Placeholder notation for alphanumeric fields of length *n*.
XXXXX	Placeholder pattern used for 5 character alphanumeric IDs.
N/A	Not Applicable
Y	Yes (field is required or true for that attribute).
Future Date	Indicates the value must be a date later than the current date.
>= 0.00	Value must be greater than or equal to zero.
Alphanumeric	Combination of letters (A–Z) and numbers (0–9).
Composite Primary Key	A primary key made up of two or more columns together uniquely identifying a record.

SQL Commands

Create Table Commands DDL:

1) Theater:

```
CREATE TABLE THEATRE (
    theaterID CHAR(5) PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    num_of_screen INTEGER NOT NULL,
    seat_capacity INTEGER NOT NULL,
    street VARCHAR(100) NOT NULL,
    city VARCHAR(50) NOT NULL,
    zip CHAR(10) NOT NULL,
    state CHAR(2) NOT NULL
);
```

2) Customer:

```
CREATE TABLE CUSTOMER (
        customerID CHAR(10) PRIMARY KEY
```

```
    fname VARCHAR(50) NOT NULL,
    lname VARCHAR(50) NOT NULL,
    cust_email VARCHAR(255) UNIQUE NOT NULL
);
```

3) Employee:

```
CREATE TABLE EMPLOYEE (
    empID CHAR(10) PRIMARY KEY,
    theaterID CHAR(5) NOT NULL,
    ROLE VARCHAR(50) NOT NULL,
    fname VARCHAR(50) NOT NULL,
    lname VARCHAR(50) NOT NULL,
    emp_email VARCHAR(100) NOT NULL,
    emp_phone VARCHAR(20) NOT NULL,
    hire_date DATE NOT NULL,
    FOREIGN KEY (theaterID) REFERENCES THEATRE (theaterID)
);
```

4) Payroll:

```
CREATE TABLE PAYROLL (
    payrollID CHAR(10) PRIMARY KEY,
    empID CHAR(10) NOT NULL,
    pp_start DATE NOT NULL,
    pp_end DATE NOT NULL,
    salary CURRENCY NOT NULL,
    deductions CURRENCY NOT NULL,
    net_pay CURRENCY NOT NULL,
    FOREIGN KEY (empID) REFERENCES EMPLOYEE (empID)
);
```

5) Membership:

```
CREATE TABLE MEMBERSHIP (
    memberID CHAR(10) PRIMARY KEY,
    customerID CHAR(10) NOT NULL,
    tier_lvl VARCHAR(20) NOT NULL,
    points INTEGER,
    join_date DATE NOT NULL,
```

```
    exp_date DATE,
    FOREIGN KEY (customerID) REFERENCES CUSTOMER (customerID)
);
```

6) Concessions:

```
CREATE TABLE CONCESSIONS (
    concessionID CHAR(5) PRIMARY KEY,
    theaterID CHAR(5) NOT NULL,
    item_name VARCHAR(50) NOT NULL,
    item_type VARCHAR(50) NOT NULL,
    item_price CURRENCY NOT NULL,
    item_stock INTEGER NOT NULL,
    supplier VARCHAR(100) NOT NULL,
    FOREIGN KEY (theaterID) REFERENCES THEATRE (theaterID)
);
```

7) Movie:

```
CREATE TABLE MOVIE (
```

```
    movieID CHAR(5) PRIMARY KEY,
    theaterID CHAR(5) NOT NULL,
    title VARCHAR(255) NOT NULL,
    genre VARCHAR(50) NOT NULL,
    duration INTEGER NOT NULL,
    rating CHAR(5) NOT NULL,
    rel_date DATE NOT NULL,
    FOREIGN KEY (theaterID) REFERENCES THEATRE (theaterID)
);
```

8) Reservation:

```
CREATE TABLE RESERVATION (
    resID CHAR(10) PRIMARY KEY,
    memberID CHAR(10) NOT NULL,
    movieID CHAR(5) NOT NULL,
    theaterID CHAR(5) NOT NULL,
    showtime DATETIME NOT NULL,
    num_of_tickets INTEGER NOT NULL,
```

```
    seat_num VARCHAR(5) NOT NULL,
    res_date DATE NOT NULL,
    pay_status VARCHAR(20) NOT NULL,
    FOREIGN KEY (memberID) REFERENCES MEMBERSHIP (memberID),
    FOREIGN KEY (movieID) REFERENCES MOVIE (movieID),
    FOREIGN KEY (theaterID) REFERENCES THEATRE (theaterID)
);
```

INSERT INTO Commands DML:

1) Theater:

```
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip,
state)
VALUES ('T0001', 'CineFlick Grand  Plaza', 10, 2200, '176 Segers rd', 'Atlantis', '90210', 'VT');
```

```
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip,
state)
```

VALUES('T0002', 'CineFlick Central', 8, 1800, '456 Markdown Blvd', 'Metroville', '10001', 'NY');

2) Customer:

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email)

VALUES('CUST000001' , 'Terry' , 'Lamar' , 'tlamar@outlook.com');

INSERT INTO CUSTOMER(customerID, fname , lname ,cust_email)

VALUES('CUST000002', 'Roc' , 'Marciano' , 'jjflash@yahoo.com');

3) Employee:

INSERT INTO EMPLOYEE(empID, theaterID, role, fname, lname, emp_email, emp_phone, hire_date)

VALUES('EMP0000001', 'T0001', 'Manager', 'Marcus', 'White', 'marcus@grandview.com', '322-555-1212', '2023-11-01')

INSERT INTO EMPLOYEE(empID, theaterID, role, fname, lname, emp_email, emp_phone, hire_date)

VALUES('EMP0000002', 'T0002', 'Gneral Manager', 'Billy', 'Woods', 'billy@grandview.com', '604-271-5958', '2011-07-11')

// run a update command for general

4) Payroll:

INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay)

VALUES ('PAY0000001', 'EMP0000001', '2025-11-01', '2025-11-15', 2307.69, 576.92, 1730.77);

INSERT INTO PAYROLL

VALUES ('PAY0000002', 'EMP0000002', '2025-11-01', '2025-11-15', 3461.54, 865.38, 2596.16);

5) Membership:

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date)

VALUES ('MEMB000001', 'CUST000001', 'Platinum', 5500, '2022-05-15', '2027-05-15');

```
INSERT INTO MEMBERSHIP

VALUES ('MEMB000002', 'CUST000002', 'Gold', 2100, '2023-01-20', '2028-01-20');
```

6) Concessions:

```
INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type, item_price,
item_stock, supplier)

VALUES ('CONC1', 'T0001', 'Large Popcorn', 'Popcorn', 10.50, 150, 'Kernel Foods Inc.');


INSERT INTO CONCESSIONS

    VALUES ('CONC2', 'T0002', 'XL Soda', 'Drink', 6.00, 200, 'Fizz Drinks Co.');
```

7) Movie:

```
INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date)

    VALUES ('MOV01', 'T0001', 'Space Rider', 'Sci-Fi', 145, 'PG-13', '2025-11-11');


INSERT INTO MOVIE

    VALUES ('MOV02', 'T0002', 'Normalization Nightmare', 'Horror', 95, 'R', '2025-11-11');
```

8) Reservation:

```
INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime,
num_of_tickets, seat_num, res_date, pay_status)
   VALUES ('RES0000001', 'MEMB000001', 'MOV01', 'T0001', '2024-12-01 19:00:00', 4, 'H1-
H4', '2024-11-07', 'Paid');

INSERT INTO RESERVATION
   VALUES ('RES0000002', 'MEMB000002', 'MOV02', 'T0002', '2024-12-05 21:30:00', 2, 'C10-
C11', '2024-11-07', 'Pending');
```

Fill in the other 10 Records for Each table:

Theatre:

```
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip,
state)
   VALUES ('T0005', 'CineFlick Southside', 6, 1100, '55 Elm St', 'Suburbia', '40111', 'KY');
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip,
state)
   VALUES ('T0006', 'CineFlick Highland', 2, 500, '10 Hill Top Rd', 'Hillside', '70012', 'OH');
```

```
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip, state)
    VALUES ('T0007', 'CineFlick Coastal', 8, 1700, '1 Ocean Blvd', 'Seaside', '30202', 'FL');
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip, state)
    VALUES ('T0008', 'CineFlick Mountain View', 4, 800, '21 Peak Trail', 'Valleydale', '80808', 'CO');
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip, state)
    VALUES ('T0009', 'CineFlick Tech Plaza ', 7, 1500, '101 Data Way', 'Innovation Hub', '60606', 'IL');
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip, state)
    VALUES ('T0010', 'CineFlick Old Town', 3, 750, '5 Heritage Ln', 'Historic City', '11111', 'MA');
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip, state)
    VALUES ('T0011', 'CineFlick West End', 9, 2000, '77 Grand Blvd', 'West City', '40004', 'KY');
INSERT INTO THEATRE (theaterID, name, num_of_screen, seat_capacity, street, city, zip, state)
```

VALUES ('T0012', 'CineFlick East Gate', 6, 1300, '12 Sunrise Pkwy', 'East City', '20202', 'VA');

Customer:

```
INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000003', 'Charles', 'Davis', 'charles.d@test.com');

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000004', 'Diana', 'Evans', 'diana.e@test.com');

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000005', 'Edward', 'Foster', 'edward.f@test.com');

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000006', 'Fiona', 'Green', 'fiona.g@test.com');

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000007', 'George', 'Harris', 'george.h@test.com');

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000008', 'Hannah', 'Irwin', 'hannah.i@test.com');

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000009', 'Ian', 'Jackson', 'ian.j@test.com');
```

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000010', 'Julia', 'King', 'julia.k@test.com');

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000011', 'Kevin', 'Lee', 'kevin.l@test.com');

INSERT INTO CUSTOMER (customerID, fname, lname, cust_email) VALUES ('CUST000012', 'Laura', 'Miller', 'laura.m@test.com');

Employee:

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP0000003', 'T0003', 'Shift Lead', 'Noah', 'Peters', 'noah.p@test.com', '700-555-1003', #2022-03-10#);

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP0000004', 'T0004', 'Projectionist', 'Olivia', 'Quinn', 'olivia.q@test.com', '700-555-1004', #2021-08-15#);

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP000005', 'T0005', 'Concessions', 'Patrick', 'Ross', 'patrick.r@test.com', '700-555-1005', #2023-01-25#);

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP000006', 'T0006', 'Usher', 'Quincy', 'Scott', 'quincy.s@test.com', '700-555-1006', #2023-05-18#);

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP000007', 'T0007', 'Shift Lead', 'Rachel', 'Thomas', 'rachel.t@test.com', '700-555-1007', #2022-11-20#);

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP000008', 'T0008', 'Projectionist', 'Sam', 'Upton', 'sam.u@test.com', '700-555-1008', #2021-06-01#);

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP000009', 'T0009', 'Concessions', 'Tina', 'Vance', 'tina.v@test.com', '700-555-1009', #2023-09-05#);

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP000010', 'T0010', 'Usher', 'Victor', 'Wells', 'victor.w@test.com', '700-555-1010', #2024-02-14#);

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP000011', 'T0011', 'Manager', 'Wendy', 'Xavier', 'wendy.x@test.com', '700-555-1011', #2020-04-12#);

INSERT INTO EMPLOYEE (empID, theaterID, ROLE, fname, lname, emp_email, emp_phone, hire_date) VALUES ('EMP000012', 'T0012', 'Shift Lead', 'Yara', 'Zane', 'yara.z@test.com', '700-555-1012', #2022-07-30#);

Payroll:

INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay) VALUES ('PAY0000003', 'EMP0000003', #2024-11-01#, #2024-11-15#, 1846.15, 461.54, 1384.61);

INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay) VALUES ('PAY0000004', 'EMP0000004', #2024-11-01#, #2024-11-15#, 1923.08, 480.77, 1442.31);

INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay) VALUES ('PAY000005', 'EMP000005', #2024-11-01#, #2024-11-15#, 1200.00, 300.00, 900.00);

```
INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay)
VALUES ('PAY000006', 'EMP000006', #2024-11-01#, #2024-11-15#, 1150.00, 287.50,
862.50);

INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay)
VALUES ('PAY000007', 'EMP000007', #2024-11-01#, #2024-11-15#, 1846.15, 461.54,
1384.61);


INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay)
VALUES ('PAY000008', 'EMP000008', #2024-11-01#, #2024-11-15#, 1923.08, 480.77,
1442.31);


INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay)
VALUES ('PAY000009', 'EMP000009', #2024-11-01#, #2024-11-15#, 1200.00, 300.00,
900.00);


INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay)
VALUES ('PAY000010', 'EMP000010', #2024-11-01#, #2024-11-15#, 1150.00, 287.50,
862.50);
```

INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay) VALUES ('PAY000011', 'EMP000011', #2024-11-01#, #2024-11-15#, 2692.31, 673.08, 2019.23);

INSERT INTO PAYROLL (payrollID, empID, pp_start, pp_end, salary, deductions, net_pay) VALUES ('PAY000012', 'EMP000012', #2024-11-01#, #2024-11-15#, 1846.15, 461.54, 1384.61);

Membership:

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000003', 'CUST000003', 'Silver', 800, #2023-10-01#, #2025-10-01#);

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000004', 'CUST000004', 'Bronze', 300, #2024-01-15#, #2025-01-15#);

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000005', 'CUST000005', 'Platinum', 7200, #2021-08-20#, #2024-08-20#);

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000006', 'CUST000006', 'Gold', 3500, #2022-12-05#, #2025-12-05#);

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000007', 'CUST000007', 'Silver', 950, #2024-03-01#, #2025-03-01#);

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000008', 'CUST000008', 'Bronze', 100, #2024-06-10#, #2025-06-10#);

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000009', 'CUST000009', 'Platinum', 6100, #2022-04-01#, #2025-04-01#);

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000010', 'CUST000010', 'Gold', 4000, #2023-04-25#, #2026-04-25#);

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000011', 'CUST000011', 'Silver', 1200, #2023-07-07#, #2024-07-07#);

INSERT INTO MEMBERSHIP (memberID, customerID, tier_lvl, points, join_date, exp_date) VALUES ('MEMB000012', 'CUST000012', 'Bronze', 500, #2024-05-11#, #2025-05-11#);

Concessions:

INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type, item_price, item_stock, supplier) VALUES ('CONC3', 'T0003', 'Hot Dog', 'Food', 5.50, 80, 'Frankie Meats');

INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type, item_price, item_stock, supplier) VALUES ('CONC4', 'T0004', 'Nachos Supreme', 'Snack', 12.00, 70, 'Mexi-Foods');

INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type, item_price, item_stock, supplier) VALUES ('CONC5', 'T0005', 'Small Popcorn', 'Snack', 7.00, 250, 'Kernel Foods Inc.');

```
INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type,
item_price, item_stock, supplier) VALUES ('CONC6', 'T0006', 'Water Bottle', 'Drink', 3.00,
300, 'Aqua Source');

INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type,
item_price, item_stock, supplier) VALUES ('CONC7', 'T0007', 'Candy Bar', 'Snack', 4.00,
400, 'Sweet Tooth Co.');

INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type,
item_price, item_stock, supplier) VALUES ('CONC8', 'T0008', 'Pretzel', 'Snack', 6.50, 100,
'Bakery Goods');

INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type,
item_price, item_stock, supplier) VALUES ('CONC9', 'T0009', 'Coffee', 'Drink', 4.50, 120,
'Brewmasters');

INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type,
item_price, item_stock, supplier) VALUES ('CONC10', 'T0010', 'Iced Tea', 'Drink', 5.00, 150,
'Fizz Drinks Co.');
```

INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type, item_price, item_stock, supplier) VALUES ('CONC11', 'T0011', 'Large Soda', 'Drink', 7.50, 180, 'Fizz Drinks Co.');

INSERT INTO CONCESSIONS (concessionID, theaterID, item_name, item_type, item_price, item_stock, supplier) VALUES ('CONC12', 'T0012', 'Small Nachos', 'Snack', 8.00, 90, 'Mexi-Foods');

Reservation:

INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime, num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000003', 'MEMB000003', 'M003', 'T0003', #2024-12-10 18:30:00#, 2, 'G12, G13', #2024-12-08#, 'Paid');

INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime, num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000004', 'MEMB000004', 'M004', 'T0004', #2024-12-11 20:00:00#, 1, 'C05', #2024-12-09#, 'Pending');

INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime, num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000005', 'MEMB000005', 'M005', 'T0005', #2024-12-12 15:45:00#, 4, 'H01, H02, H03, H04', #2024-12-10#, 'Paid');

```
INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime,
num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000006', 'MEMB000006',
'M006', 'T0006', #2024-12-13 19:15:00#, 3, 'A01, A02, A03', #2024-12-11#, 'Paid');

INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime,
num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000007', 'MEMB000007',
'M007', 'T0007', #2024-12-14 17:00:00#, 2, 'K08, K09', #2024-12-12#, 'Pending');

INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime,
num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000008', 'MEMB000008',
'M008', 'T0008', #2024-12-15 13:00:00#, 1, 'B04', #2024-12-13#, 'Paid');

INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime,
num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000009', 'MEMB000009',
'M009', 'T0009', #2024-12-16 21:30:00#, 5, 'J10, J11, J12, J13, J14', #2024-12-14#, 'Paid');

INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime,
num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000010', 'MEMB000010',
'M010', 'T0010', #2024-12-17 14:00:00#, 2, 'D07, D08', #2024-12-15#, 'Pending');
```

INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime, num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000011', 'MEMB000011', 'M011', 'T0011', #2024-12-18 18:00:00#, 3, 'F05, F06, F07', #2024-12-16#, 'Paid');

INSERT INTO RESERVATION (resID, memberID, movieID, theaterID, showtime, num_of_tickets, seat_num, res_date, pay_status) VALUES ('RES000012', 'MEMB000012', 'M012', 'T0012', #2024-12-19 22:00:00#, 1, 'E01', #2024-12-17#, 'Paid');

Movie:

INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date) VALUES ('M003', 'T0003', 'The Crimson Tide', 'Action', 145, 'R', #2024-11-15#);

INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date) VALUES ('M004', 'T0004', 'A Star\'s Journey', 'Drama', 120, 'PG-13', #2024-10-01#);

INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date) VALUES ('M005', 'T0005', 'The Laugh Riot', 'Comedy', 95, 'PG', #2024-12-05#);

INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date) VALUES ('M006', 'T0006', 'Secrets of the Deep', 'Documentary', 88, 'G', #2025-01-20#);

```
INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date)
VALUES ('M007', 'T0007', 'Midnight Caller', 'Horror', 105, 'R', #2024-11-01#);

INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date)
VALUES ('M008', 'T0008', 'The Last Frontier', 'Sci-Fi', 155, 'PG-13', #2024-12-25#);

INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date)
VALUES ('M009', 'T0009', 'Romantic Haze', 'Romance', 110, 'PG', #2025-02-14#);

INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date)
VALUES ('M010', 'T0010', 'Epic Battle', 'Action', 135, 'PG-13', #2024-10-28#);

INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date)
VALUES ('M011', 'T0011', 'The Silent Witness', 'Thriller', 98, 'R', #2025-03-01#);

INSERT INTO MOVIE (movieID, theaterID, title, genre, duration, rating, rel_date)
VALUES ('M012', 'T0012', 'Family Fun Day', 'Animation', 92, 'G', #2024-12-01#);
```