CSC/BIF375: Database Management Systems

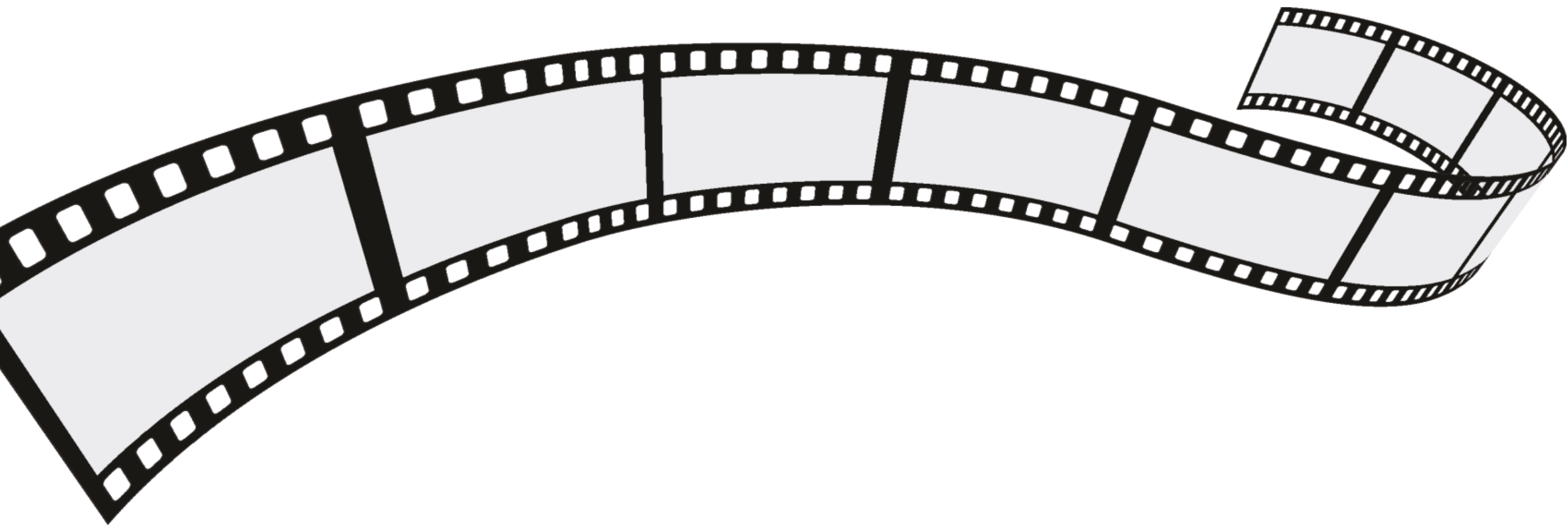
Dr. Ramzi A. HARATY

April 30, 2025

DATABASE APPLICATION DESIGN FOR A MOVIE THEATRE

by The Primary Keys

Joudy ALLAM, Haya MAZYAD, Lynn OUEIDAT, Nour SAAD, Alaa SAKR



This page is intentionally left blank.

Table of Contents

Introduction

Welcome to “222” theatre read as Triple 2 – where entertainment collides with organization. Inspired by the powerful numerological meaning of 222, our theatre is built around the values of harmony, balance, and excellence, ensuring that every show, every seat, and every customer interaction runs smoothly. 222 is often associated with being "in the right place, at the right time," which is not only our slogan but a reflection of our commitment to enhancing your experience. To bring this vision to life, we are developing a comprehensive database designed to control theatre operations and enhance customer satisfaction. From movie scheduling and seat reservations to ticketing, memberships, suppliers, and payments, our system ensures that every aspect of theatre management is efficient, organized, and connected.

System Description and Constraints

Movies have the power to transport audiences to different worlds, evoke emotions, and provide unforgettable experiences. Triple 2 Movie Theatre (222) is dedicated to offering a premium cinematic journey with its advanced audiovisual technology, comfortable seating, and diverse movie selections. From the latest blockbusters to timeless classics and exclusive screenings, the theatre ensures that every moviegoer finds something to enjoy. With multiple screening halls equipped with state-of-the-art projectors and immersive sound systems, Triple 2 caters to a variety of preferences, including 3D, IMAX, and VIP private screenings.

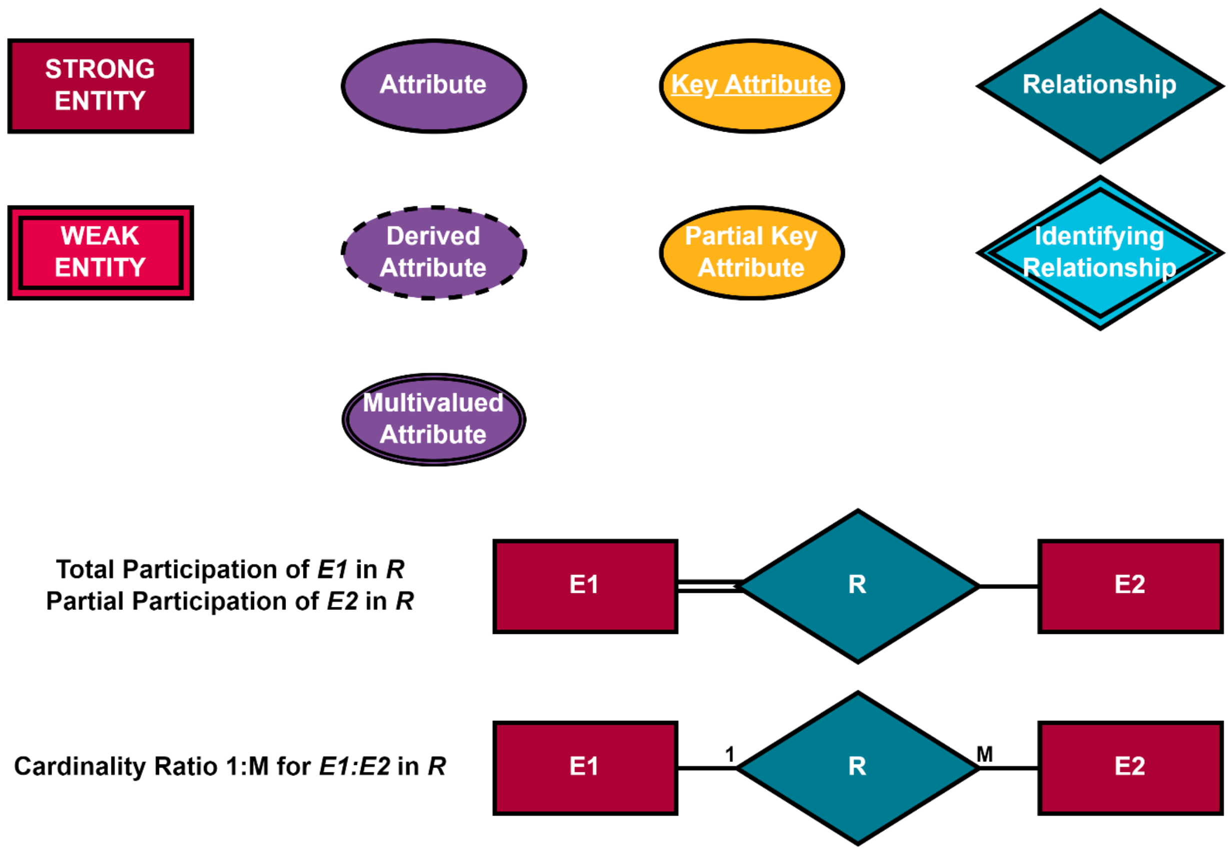
The theatre’s operations revolve around a well-integrated system that manages ticket reservations, movie scheduling, customer memberships, and in-theatre services such as concessions and VIP lounges. Customers can book tickets online or at the theatre, selecting their preferred seats in real time to avoid overbooking and ensure a seamless experience. The theatre’s dynamic scheduling system optimizes screening times to accommodate peak and off-peak hours while maintaining operational efficiency. Frequent visitors can benefit from loyalty programs that provide discounts, priority bookings, and exclusive rewards. Additionally, Triple 2 ensures smooth internal management, handling employee schedules, cashier operations, and maintenance services to guarantee an optimal moviegoing experience.

Despite its seamless operation, Triple 2 Movie Theatre faces various challenges and constraints. Each screening hall has a limited seating capacity, requiring a well-regulated booking and waiting-list system to manage high-demand movies. Scheduling multiple movies across different halls can lead to conflicts, necessitating precise coordination to prevent delays and overlapping showtimes. Technical malfunctions, such as projector failures or sound system issues, can disrupt screenings and require immediate attention to minimize inconvenience. Customer cancellations and refund policies must strike a balance between flexibility and business profitability, ensuring customer satisfaction while maintaining financial stability. Additionally, the theatre must prioritize data security and financial transaction integrity, safeguarding customer information and payment details against potential breaches. Compliance with entertainment industry regulations, safety protocols, and age restrictions further ensures that all screenings adhere to legal and ethical standards.

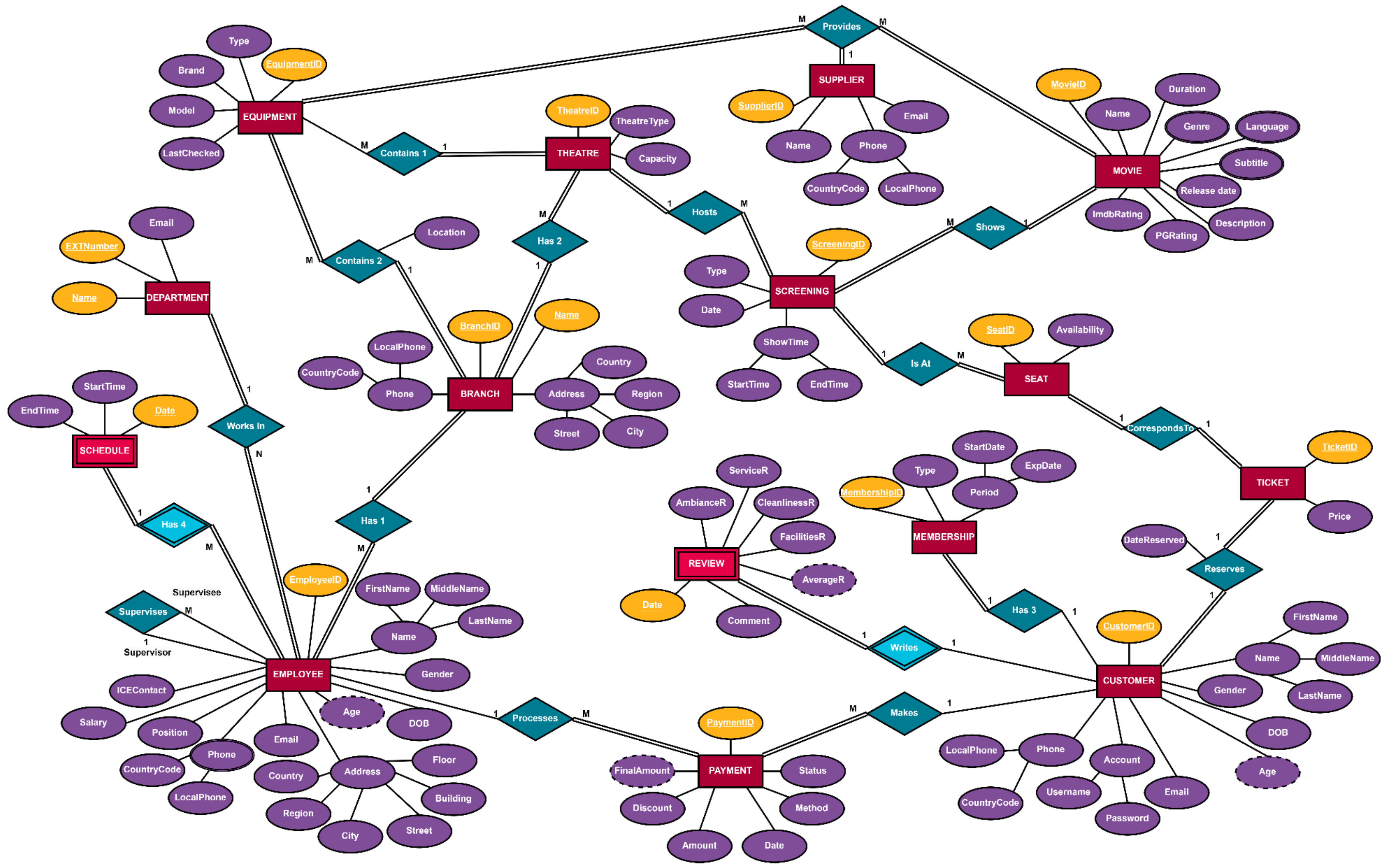
Triple 2 Movie Theatre remains committed to delivering a world-class entertainment experience, combining innovation, customer convenience, and operational excellence. By continuously improving its services, addressing challenges proactively, and adapting to technological advancements, the theatre strives to be the premier destination for movie lovers seeking an unparalleled cinematic experience.

ER Diagram

Entity Diagram Symbols

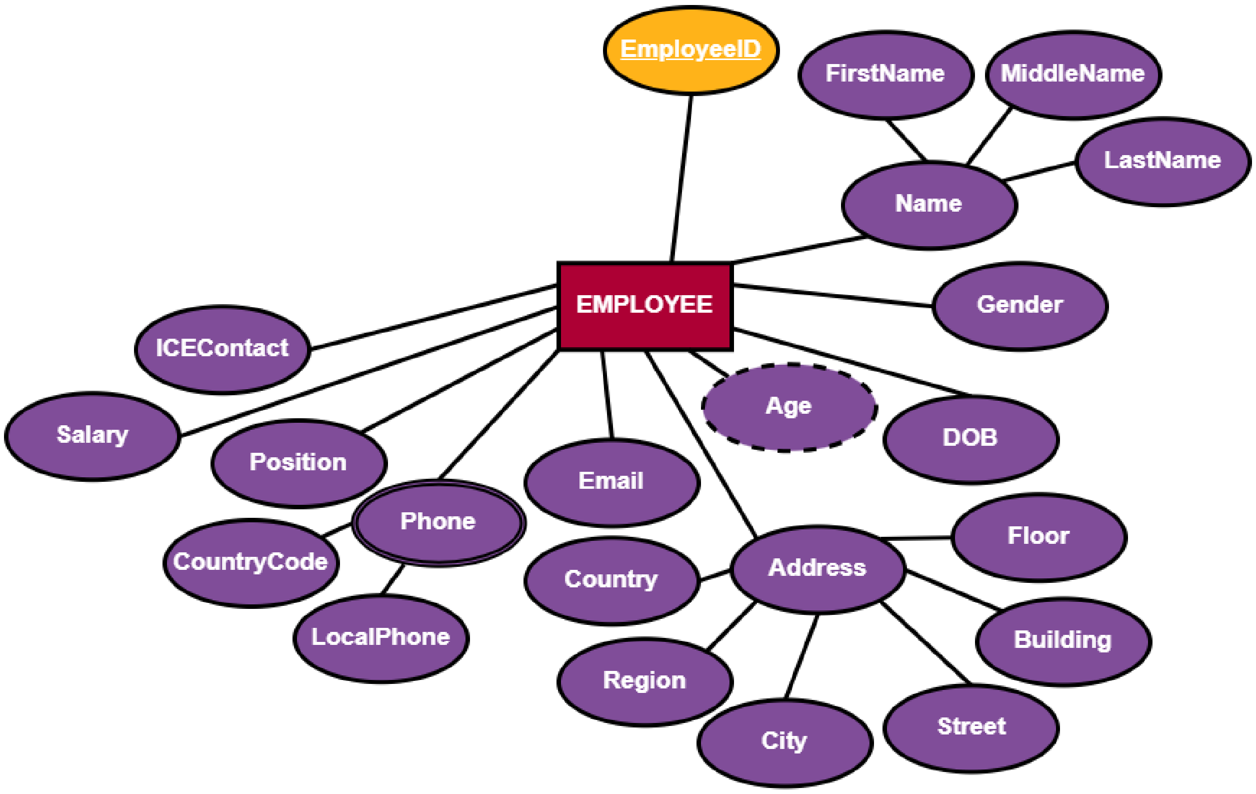


ER Diagram for 222



Entity Data Types

1. EMPLOYEE



The **EMPLOYEE** entity type represents all individuals working in the movie theatre “222” across all the different departments and branches. It stores essential personal details, job-related information, and salary details for each employee.

Below is a detailed breakdown of its attributes:

- **EmployeeID** *(Key Attribute)*: A unique identifier for each employee. It ensures that all employee records remain distinct and easily accessible.
- **Name** *(Composite Attribute)*: Represents the employee’s full name. It consists of:
    - **First Name**: The employee’s first name.
    - **Middle Name**: The employee’s middle name (if applicable).
    - **Last Name**: The employee’s last name.
- **Gender**: Specifies the employee’s gender (male or female). It is used for HR records and demographic analysis.

- **DOB** (date of birth): Represents the employee’s birth date in the format DD/MM/YYYY. It is essential for verifying age eligibility for employment and calculating benefits.
- **Age** *(Derived Attribute)*: The employee’s age, calculated based on the date of birth.

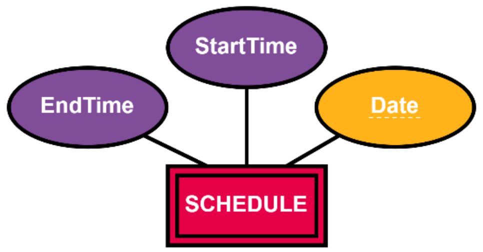
$$Age = Current\,Year - Year\ of\ Birth$$

- **Address** *(Composite Attribute)*: Stores the employee’s complete address. It consists of:
    - **Country**: The country of residence.
    - **Region**: The region where the employee is located.
    - **City**: The city in which the employee lives.
    - **Street**: The specific street name where the employee resides.
    - **Building**: The name of the building where the employee resides.
    - **Floor**: The floor number of the building.
- **Email**: Stores the employee’s official email address, used for communication and work-related notifications.
- **Phone** *(Composite & Multivalued Attribute)*: Represents the employee’s contact number(s). It consists of:
    - **Country Code**: The country’s dialing code.
    - **Local Phone Number**: The employee’s personal or work phone number.

  Since an employee may have multiple phone numbers, this attribute is multivalued. It ensure better accessibility and flexible communication.

- **Position**: Specifies the employee’s job title or role within the movie theatre “222”.
- **Salary**: Represents the employee’s earnings and is used for payroll management and financial records.
- **ICEContact** (In Case of Emergency Contact): Stores the phone number of an emergency contact person designated by the employee. This is crucial for safety measures.

2. SCHEDULE

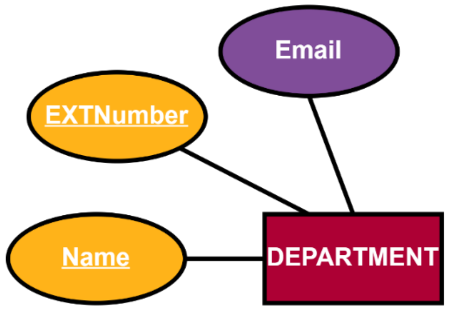


SCHEDULE is a **weak entity data type** that depends on the **EMPLOYEE** entity type, meaning it cannot exist without being associated with an employee. It represents the work shifts assigned to employees, ensuring proper time management and coordination of staffing within the movie theatre 222 and avoiding schedule or time conflicts.

Below is a detailed breakdown of its attributes:

- **Date** *(Partial Key)*: Represents the specific day for which the schedule applies, uniquely identifying different shifts for the same employee.
- **StartTime**: The time when an employee's work shift begins. This helps in organizing staff schedules and ensuring smooth operations and time management.
- **EndTime**: The time when an employee's work shift ends. It ensures proper workforce planning and prevents scheduling conflicts.

3. DEPARTMENT

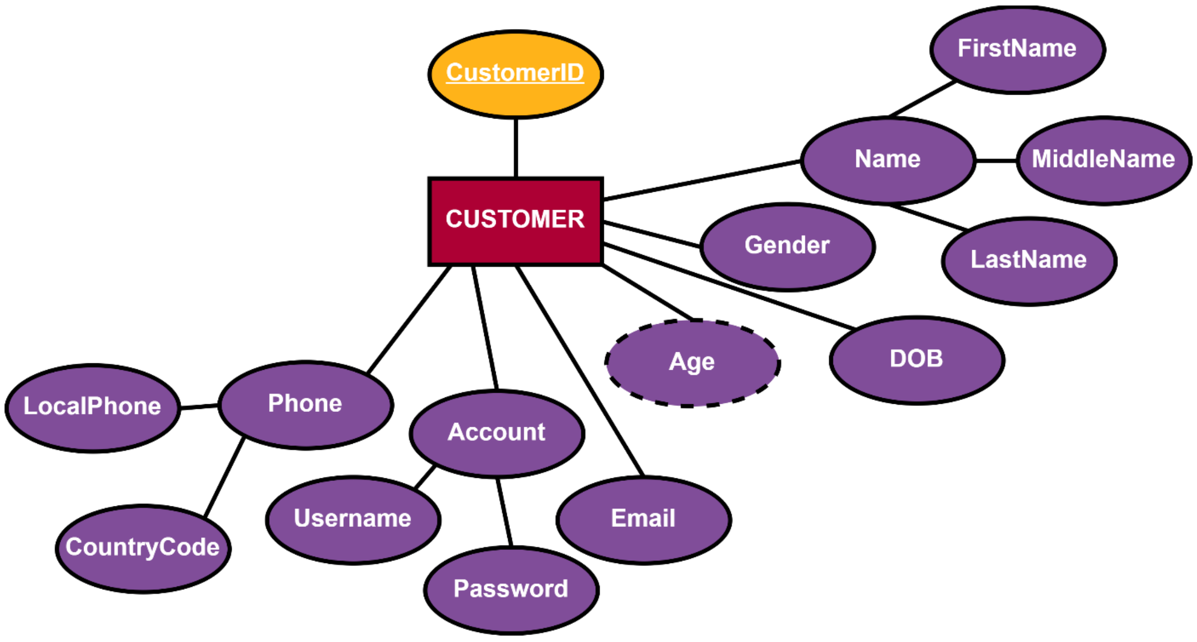


The **DEPARTMENT** entity type represents the various departments within the movie theatre 222, each managing specific functions such as customer service, ticketing, maintenance, and management… It helps organize the theatre's workforce and ensures efficient communication between departments. This entity type stores the key details of each department.

Below is a detailed breakdown of its attributes:

- **Name** *(Key Attribute)*: The official name of the department. It ensures proper identification and distinguishes different departments within the theatre.
- **EXTNumber** *(Key Attribute)*: The unique extension number assigned to the department. It is used for internal communication within the theatre.
- **Email**: Stores the department's official email address, used for communication and coordination with employees and other departments.

4. CUSTOMER



The **CUSTOMER** entity type represents individuals who purchase movie tickets and utilize the services at 222. It plays a crucial role in managing customer interactions, ensuring a seamless booking experience, and facilitating loyalty programs.

Below is a detailed breakdown of its attributes:

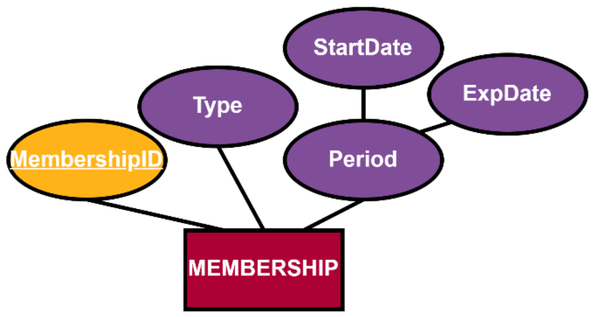
- **CustomerID** *(Key Attribute)*: A unique identifier assigned to each customer. It is used for tracking customer transactions and interactions within the system.
- **Name** *(Composite Attribute)*: It represents the full name of the customer. This attribute is further divided into:
  - **FirstName**: Stores the first name of the customer.

    - **MiddleName**: Stores the middle name of the customer.
    - **LastName**: Stores the last name of the customer.
- **Gender**: Gender of the customer (e.g., Male, Female). It helps in demographic analysis and targeted marketing.
- **DOB (Date of Birth)**: It stores the birth date of the customer in a standard format (e.g. DD-MM-YYYY). It is used for age verification and customer categorization.
- **Age** *(Derived Attribute)*: A derived attribute calculated from the **DOB**.

$$Age = Current\ Date - DOB$$

- **Phone** *(Composite Attribute)*: It stores the customer's phone number details. It consists of:
    - **CountryCode**: Represents the international calling code (e.g., +1 for the USA).
    - **LocalPhoneNb**: Stores the local phone number of the customer.
- **Email**: It contains the customer's email address. It is used for communication, ticket confirmations, and promotional campaigns.
- **Account** *(Composite Attribute)*: It represents the customer's login credentials for the online booking system. It consists of:
    - **Username**: The chosen identifier for customer login.
    - **Password**: A secure credential required for authentication.

5. MEMBERSHIP

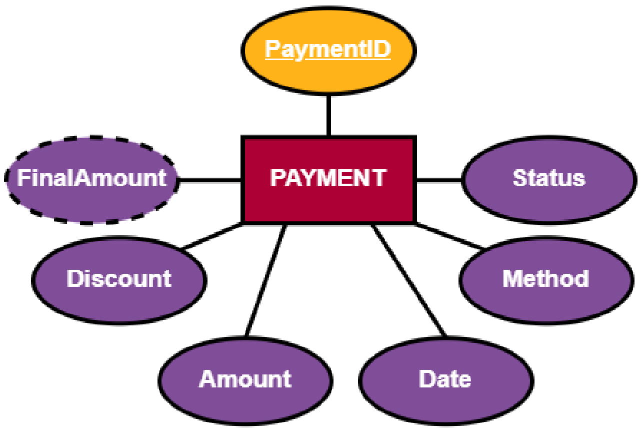


222 offers exclusive membership benefits designed to enhance the movie-going experience for its loyal customers. The **MEMBERSHIP** entity type plays a key role in fostering customer retention by providing perks such as discounts, priority booking, and special rewards. Customers who enroll in a membership program gain access to premium benefits based on their membership type. This entity type helps in tracking membership details, ensuring timely renewals, and managing different membership tiers to deliver a personalized and rewarding cinema experience.

Below is a detailed breakdown of its attributes:

- **MembershipID** *(Key Attribute)*: A unique identifier assigned to each membership, ensuring accurate tracking, management, and differentiation of memberships within the system.
- **Type**: It defines the category of the membership (e.g., Silver, Gold, Platinum). It determines the benefits and privileges associated with the membership.
- **Period** *(Composite Attribute)*: Duration of the membership. It consists of:
  - **StartDate**: The date when the membership becomes active.
  - **ExpDate**: The expiration date when the membership ends or requires renewal.

6. PAYMENT



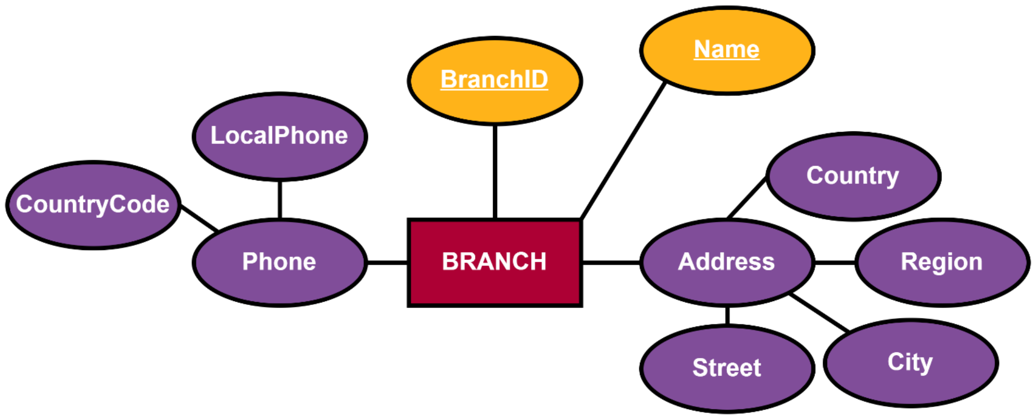
The **PAYMENT** entity type stores details about ticket transactions within the movie theatre. It tracks ticket purchases, applied discounts, and payment statuses to ensure efficient financial management. Below is a detailed breakdown of its attributes:

- **PaymentID** *(Key Attribute)*: A unique identifier assigned to each ticket payment. It ensures that every transaction is recorded and traceable.
- **Amount**: The original ticket(s) price(s) before any discounts are applied.
- **Discount**: The percentage of discount applied to the ticket, if applicable, from promotions, loyalty rewards, or special offers.
- **FinalAmount** *(Derived Attribute)*: The actual final amount paid by the costumer after applying the discount. It is derived from the original amount and the discount and calculated using the formula:

$$FinalAmount = Amount - \frac{Discount}{100} \times Amount$$

- **Date**: The date on which the ticket payment was made. This helps in tracking sales over time.
- **Method**: The mode of payment used for purchasing the ticket. It includes cash, credit card, debit card or digital wallet.
- **Status**: Indicates the current state of the payment, whether it was paid, unpaid, cancelled, refunded or declined. This ensures proper transaction tracking.

7. BRANCH

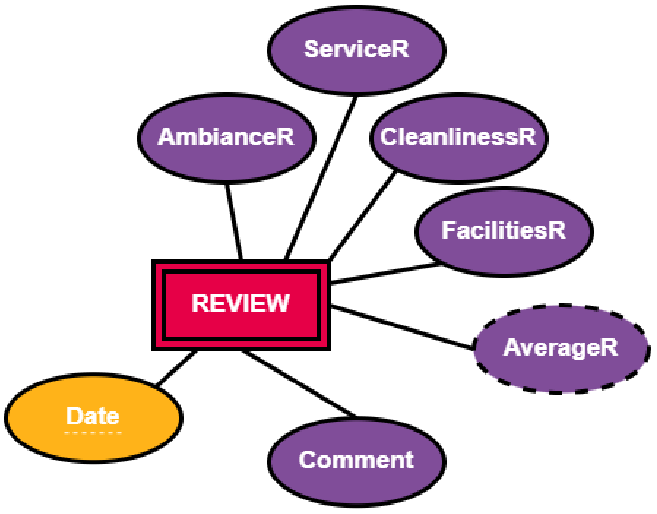


The **BRANCH** entity type represents the information of the **222** branches. Each branch operates independently but is part of the global franchise network. It holds important identifying and location-related details.

Below is a detailed breakdown of its attributes:

- **BranchID** (*Key Attribute*): A unique identifier to each branch. It ensures that all data related to a specific branch remains distinct.
- **Name** (*Key Attribute*): Stores the official name of the branch (222 *City*, e.g. 222 Paris). It helps with branding and customer recognition.
- **Address** *(Composite Attribute)*: Stores the detailed location information of the branch. It consists of:
  - **Country**: The country in which the branch exists.
  - **Region**: The state or region where the branch is situated.
  - **City**: The city in which the branch operates.
  - **Street**: The specific street name and number where the branch is located.
- **Phone** *(Composite Attribute)*: Stores the branch's phone number details. It consists of:
  - **CountryCode**: The international dialing code for the branch's country.
  - **LocalPhone**: The phone number specific to the branch within the country.

8. REVIEW



Reviews play a crucial role in maintaining the high standards of 222 by collecting and analyzing customer feedback. **REVIEW** is a **weak entity type** dependent on the **CUSTOMER** and **BRANCH** entity types. It allows customers to rate their experiences across multiple aspects, helping the branch to continuously improve.

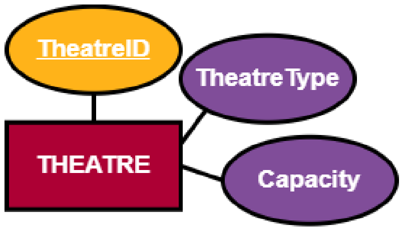
Below is a detailed breakdown of its attributes:

- **Date** *(Partial Key)*: Represents the submission date of the review. It ensures uniqueness when combined with **CustomerID** and **BranchID**.
- **AmbianceR**: Rates the atmosphere and comfort of the branch on a scale from **1 to 5**.
- **ServiceR**: Rates the quality of customer service received on a scale from **1 to 5**.
- **CleanlinessR**: Rates the hygiene and cleanliness of the theater on a scale from **1 to 5**.
- **FacilitiesR:** Rates the quality and maintenance of theater facilities, including seating and restrooms, on a scale from **1 to 5**.
- **AverageR** *(Derived Attribute)*: Represents the overall review score, calculated as the average of all individual ratings.

$$AverageR = \frac{AmbianceR + ServiceR + CleanlinessR + FacilitiesR}{4}$$

- **Comment:** Stores additional feedback provided by the customer.

9. THEATRE

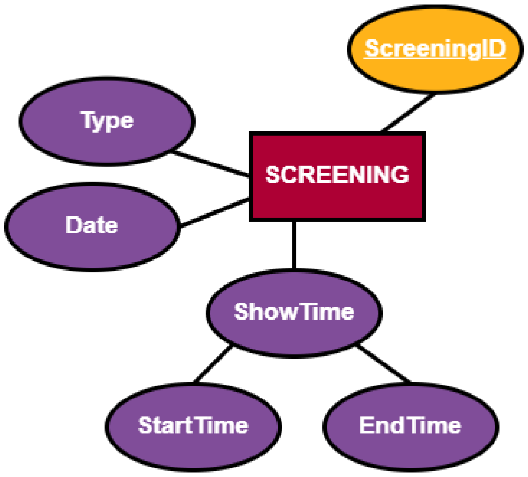


The **THEATRE** entity type represents the different screening rooms at 222. Each theater has a unique identification number and specific characteristics such as capacity and type. This entity type is essential for managing seating arrangements, scheduling movie screenings, and optimizing space utilization.

Below is a detailed breakdown of its attributes:

- **TheatreID** *(Key Attribute)*: A unique identifier assigned to each theater. It helps in distinguishing different theaters within the complex.
- **TheatreType**: It defines the type of theater (e.g., Standard, Premium, VIP). It determines the ticket pricing and experience level offered.
- **Capacity**: Maximum number of seats available in the theater. It helps in managing bookings and avoiding overbooking situations.

10. SCREENING

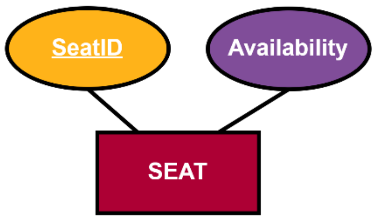


The **SCREENING** entity type represents the specific showings of movies at the movie theatre “222.” It is pivotal for scheduling and managing when movies are shown, which also affects ticket availability.

Below is a detailed description of its attributes:

- **ScreeningID** *(Key Attribute)*: A unique identifier assigned to each individual screening of a movie. It is crucial for tracking and managing different screenings within the system.
- **Type**: Specifies the type of screening, such as regular, 3D, IMAX, or private events. It helps in organizing the logistical aspects of movie showings and ticket pricing.
- **Date**: The specific date on which the screening is held. It is important for planning and allows customers to select showings based on their availability.
- **ShowTime** *(Composite Attribute)*: Contains details about the start and end times of the screening. It consists of:
    - **StartTime**: The time when the screening begins.
    - **EndTime**: The time when the screening concludes.

11. SEAT

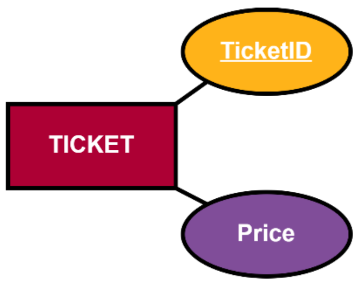


The **SEAT** entity type represents the individual seats available within a theater. Each seat is uniquely identified and has an availability status to determine whether it is booked or open for reservation. This entity type is crucial for managing seating arrangements, preventing overbooking, and ensuring a smooth movie-watching experience for customers.

Below is a detailed breakdown of its attributes:

- **SeatID** *(Key Attribute)*: A unique identifier assigned to each seat within a theater. It helps in tracking seat assignments and reservations.
- **Availability**: It indicates whether the seat is available, reserved, or occupied. It helps in managing real-time seat selection and booking systems.

12. TICKET

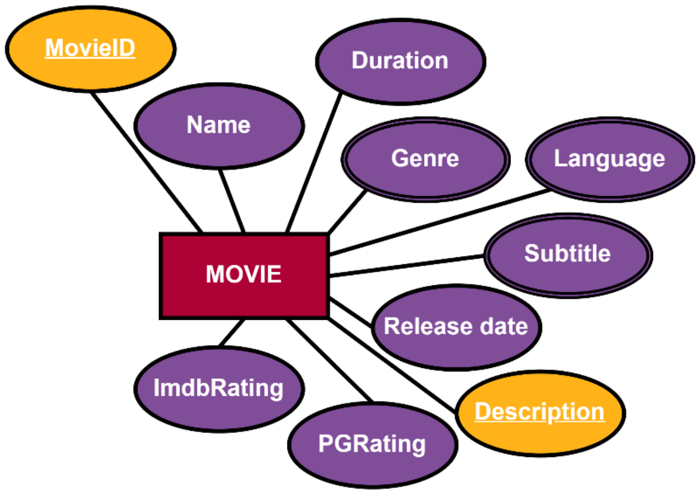


The **TICKET** are purchased by customers to watch movies at 222. Each ticket is uniquely identified and has a set price based on factors such as the movie, theater type, and time of booking. This entity type is essential for managing ticket sales, revenue tracking, and seat reservations.

Below is a detailed breakdown of its attributes:

- **TicketID** *(Key Attribute)*: A unique identifier assigned to each ticket. It helps in tracking ticket sales and validating entry to the theater.
- **Price**: The cost of the ticket. The price may vary based on factors like seating type, and time of the show.

13. MOVIE



The **MOVIE** entity type represents films available for viewing at the movie theatre "222." It is essential for cataloging the movies shown, organizing showtimes, and providing customers with information about the film offerings.

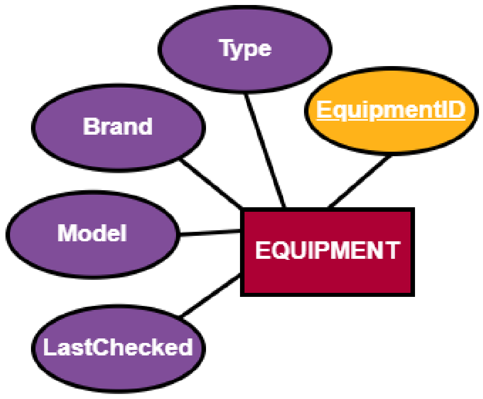
Below is a detailed breakdown of its attributes:

- **MovieID** *(Key Attribute)*: A unique identifier assigned to each movie. It is used for identifying and managing movie records within the system.
- **Description** *(Key Attribute)*: A unique synopsis or summary of the movie's plot and thematic elements. It helps customers decide whether the movie aligns with their interests.
- **Name**: The official title of the movie. It is used in promotional materials, ticket sales, and customer selections.
- **Duration**: The total runtime of the movie, measured in minutes. It is important for scheduling screenings and informing customers about the length of the movie.
- **Genre** *(Multivalued Attribute)*: The categories or genres that the movie belongs to (e.g., Action, Romance, Thriller). It helps in filtering and recommending movies based on customer preferences.

  This attribute is multivalued because a single movie may fit into multiple genres, making it necessary to store multiple genre values per movie.

- **Language** *(Multivalued Attribute)*: Specifies the languages in which the movie is available. It is essential for catering to a diverse audience and managing language-specific showings.
  Since films are often available in multiple languages, either through dubbing or original language options, this attribute is multivalued.
- **Subtitle** *(Multivalued Attribute)*: Lists available subtitle languages. It enhances accessibility for non-native speakers and the hearing impaired.
  Like Languages, subtitles can be available in multiple languages for a single movie, making it a multivalued attribute.
- **Release Date**: The official release date of the movie. It is used for promotional events and to prioritize new releases.
- **PGRating**: Indicates the audience suitability based on age or viewer discretion (e.g., PG, R…). It is critical for audience targeting and adhering to regulatory standards.
- **ImdbRating**: The movie's rating on the IMDb platform. It provides a measure of the movie's popularity and critical reception, aiding customer choice.

14. EQUIPMENT

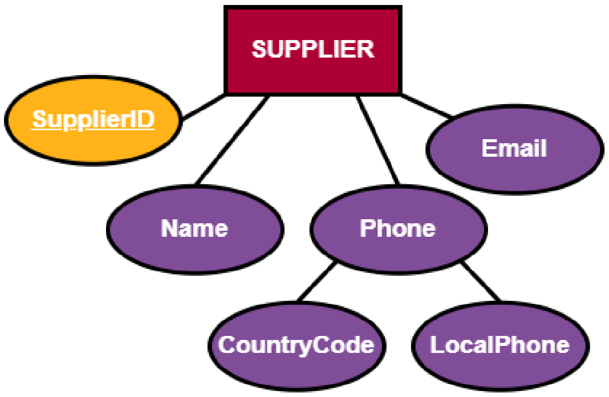


The **EQUIPMENT** entity type represents the various technical assets used in Triple 2 branches, such as projectors and sound systems. It ensures that each piece of equipment is uniquely identified, maintained, and tracked for operational efficiency.

Below is a detailed breakdown of its attributes:

- **EquipmentID** *(Key Attribute)*: A unique identifier assigned to each piece of equipment. It ensures that the equipment data remains distinct and traceable.
- **Type**: Specifies the category of the equipment (e.g., Projector, Speaker, Screen…).
- **Brand**: Stores the brand or manufacturer of the equipment (e.g., Sony, IMAX, JBL).
- **Model**: Contains the specific model name or number of the equipment.
- **LastChecked**: Stores the date when the equipment was last inspected or maintained. It helps in scheduling routine maintenance and ensuring quality performance.

15. SUPPLIER



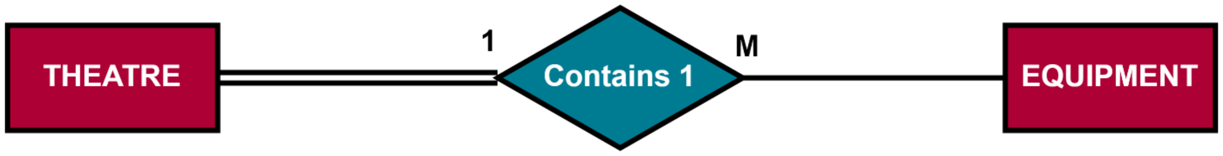
The **SUPPLIER** entity type represents the external vendors that provide movies and equipment to the theater "222". It plays a crucial role in maintaining the theater's operations by ensuring a steady supply of films and necessary equipment. Managing suppliers efficiently helps the theater maintain a diverse movie lineup and keep its facilities in top condition.

Below is a detailed description of its attributes:

- **SupplierID** *(Key Attribute)*: A unique identifier assigned to each supplier. It is essential for tracking and managing supplier relationships within the system.
- **Name:** The official name of the supplier. This helps in identifying and organizing different vendors.
- **Phone** *(Composite Attribute):* The primary contact number of the supplier, allowing quick communication regarding deliveries, agreements, and issues. It consists of:
    - **CountryCode:** The international dialing code to accommodate international suppliers.
    - **LocalPhone:** The phone number specific to the supplier within the country.
- **Email:** The official email address of the supplier used for correspondence and order confirmations.

Relationship Types

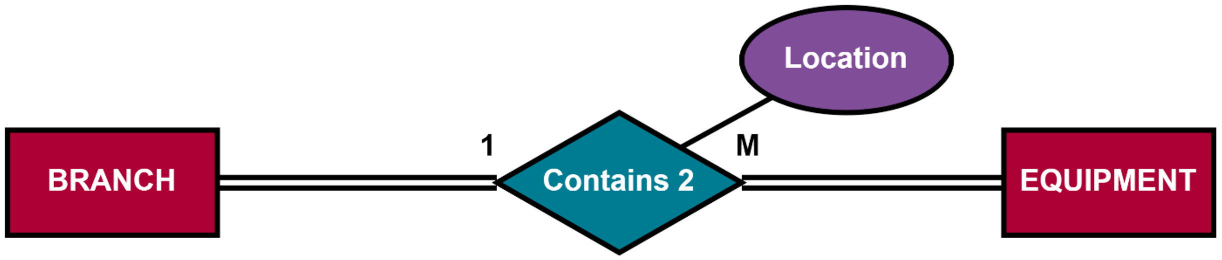
1. Contains 1



The **"CONTAINS 1"** relationship represents the association between the **THEATRE** and **EQUIPMENT** entity types, ensuring that each theater is properly equipped for movie screenings. This relationship helps in tracking the distribution and maintenance of equipment within the theater.
The participation is **total** on the **THEATRE** side, as every theatre must have at least one piece of equipment in it. However, participation on the **EQUIPMENT** side is **partial**, as not all equipment may be present inside a theatre room, some equipment may be used in the hall outside or next to the cashier.

This relationship follows a **one-to-many** (1:M) cardinality, meaning that a theater can contain multiple pieces of equipment, but each piece of equipment must be associated with exactly one theater. This structure allows for efficient equipment management, ensuring that each theater has the necessary technical resources for its screenings.

2. Contains 2

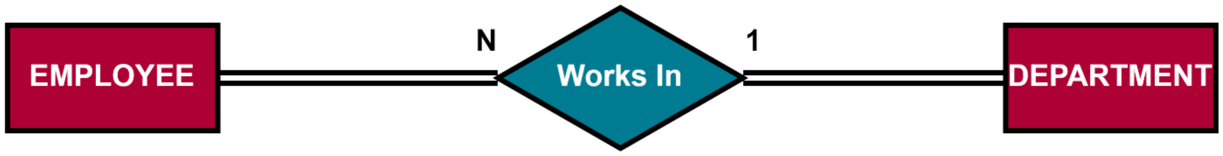


The **"CONTAINS 2"** relationship represents the connection between the **BRANCH** and **EQUIPMENT** entity types, ensuring that each branch has the necessary tools for its operations. This relationship is essential for managing resource allocation, tracking equipment distribution, and ensuring that each branch is well-equipped to provide quality service.

The participation is **total** on the **EQUIPMENT** and **BRANCH** side, meaning that every piece of equipment must be assigned to a specific branch, ensuring that no equipment exists without being located in a branch and that every branch has a piece of equipment.

This relationship follows a **one-to-many cardinality (1:M)**, meaning that a branch "Contains 2" one or more pieces of equipment, while each piece of equipment must be associated with exactly one branch. Additionally, the **location** attribute within the **BRANCH** entity type specifies where each branch is present, helping define its physical presence and aiding in operational management.

3. Works In

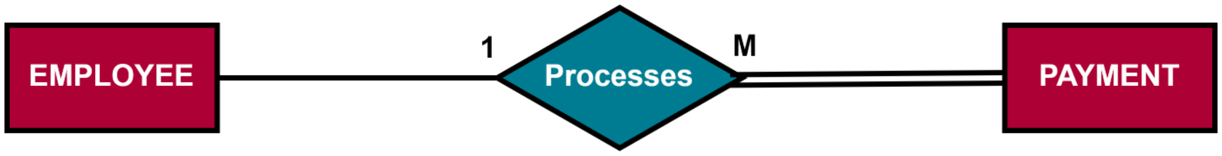


Each employee has a specific position or role in their job at the theater "222". The **"WORKS IN"** relationship represents the connection between the **EMPLOYEE** and **DEPARTMENT** entity types, ensuring that each employee is assigned to a specific department within the theater. This relationship is essential for organizing staff roles, managing department responsibilities, and maintaining an efficient workflow.

The participation is **total** on both the **EMPLOYEE** and **DEPARTMENT** side, meaning that every employee must be assigned to a department, ensuring that no employee exists in the system without a designated department and that every department has an employee working in it.

This relationship follows a **many-to-one** cardinality (1:M), meaning that multiple employees "Work In" one department, while each employee works in one specific department.

4. Processes

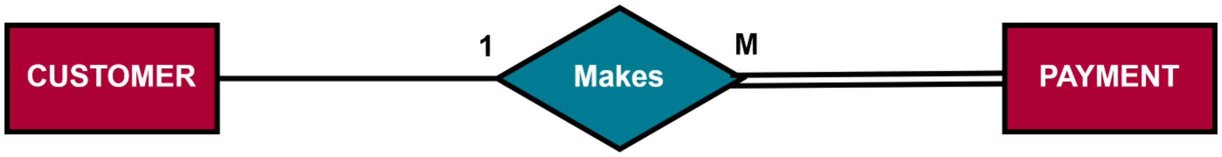


To ensure a smooth payment processing and maintain accountability we need a **"PROCESSES"** relationship that represents the connection between the **EMPLOYEE** and **PAYMENT** entity types, ensuring that employees handle and manage customer transactions within the theater.

The participation is **total** on the **PAYMENT** side, meaning that every payment must be processed by an employee, ensuring that no transaction occurs without an assigned staff member overseeing it. However, the participation on the **EMPLOYEE** side is **partial**, as not all employees may be responsible for processing payments, depending on their assigned department within the theater.

This relationship follows a **one-to-many** cardinality (1:M), meaning that one employee "Processes" multiple payments, while each payment must be processed by exactly one employee.

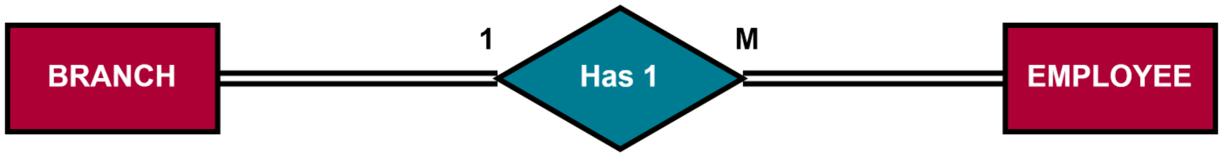
5. Makes



A customer has the option to book tickets for movie screenings at the theater. So, to manage reservations we have the **"RESERVES"** relationship that represents the connection between the **CUSTOMER** and **TICKET** entity types. This relationship is essential for managing reservations, tracking ticket availability, and providing a seamless booking experience.

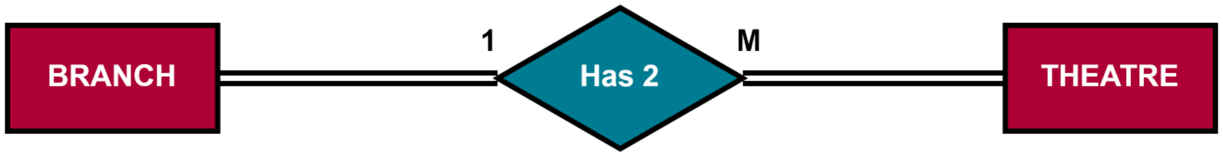
The participation is **total** on both the **CUSTOMER** and **TICKET** sides, meaning that every reservation must involve one customer and one ticket, ensuring that no ticket exists without an associated customer and no customer reservation exists without a ticket.
This relationship follows a **one-to-one cardinality (1:1)**, meaning that each customer "Reserves" exactly one ticket, and each ticket is reserved by exactly one customer. This ensures a structured and efficient reservation system, preventing multiple customers from reserving the same ticket while maintaining clear records of bookings.

6. Has 1



Each branch has to have employees working in it so the **"HAS 1"** relationship represents the connection between the **BRANCH** and **EMPLOYEE** entity types, ensuring that each branch is staffed by one or more employees. This relationship is crucial for managing theater operations, assigning employees to specific locations, and ensuring smooth workflow within each branch.
The participation is **total** on both the **BRANCH** and **EMPLOYEE** sides, this means that each branch must have at least one employee working at it.
This relationship follows a **many-to-one cardinality (M:1)**, meaning that multiple employees "Has 1" branch, while each branch must have at least one employee. This structure ensures that each branch is staffed appropriately while allowing multiple employees to be associated with the same branch.

7. Has 2

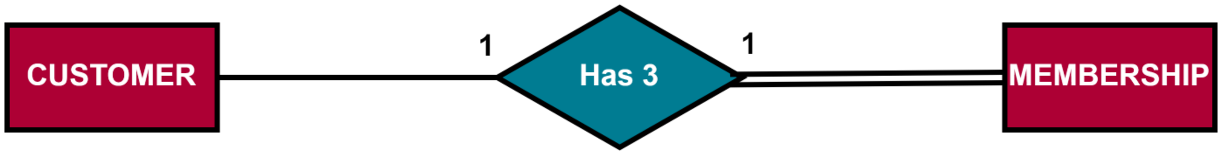


The **BRANCH** and **THEATRE** entity types have a **"HAS 2"**, relationship ensuring that each branch consists of multiple screening rooms. This relationship is essential for organizing theater operations, managing screening schedules, and optimizing space utilization within each branch.
The participation is **total** on the **THEATRE** side, meaning that every theater must be associated with a branch, ensuring that no individual theater exists without belonging to a specific branch. However, the

participation on the **BRANCH** side is **partial**, as not all branches may have multiple theaters, depending on their size and infrastructure.

This relationship follows a **one-to-many cardinality (1:M)**, meaning that one branch "Has 2" one or more theaters, while each theater must belong to exactly one branch.

8. Has 3

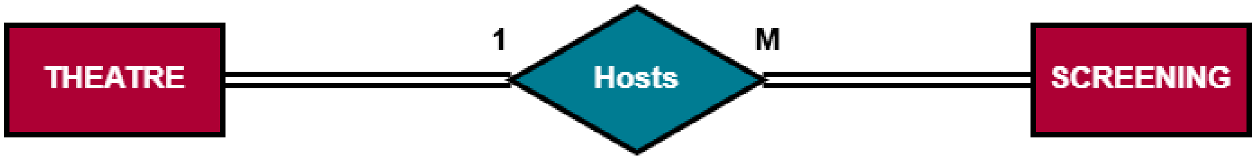


Customers have the option to enroll in a membership that offer various benefits at the “222” theater which is represented by the “**HAS 3"** relationship which is the connection between the **CUSTOMER** and **MEMBERSHIP** entity types.

The participation is **total** on the **MEMBERSHIP** side, meaning that every membership must be associated with a customer, ensuring that no membership exists without a registered customer. However, the participation on the **CUSTOMER** side is **partial**, as not all customers may have a membership.

This relationship follows a **one-to-one cardinality (1:1)**, meaning that one customer can have only one membership at a time (silver, gold or platinum…), and each membership must be associated with exactly one customer. This structure helps the theater manage customer engagement and encourage long-term loyalty through various membership tiers and benefits.

9. Hosts

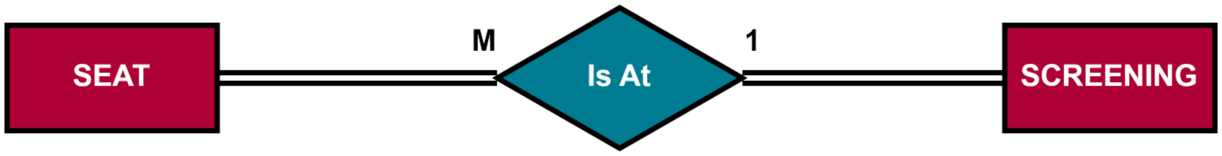


The **SCREENING** and **THEATRE** entity types have a **“HOSTS”**, relationship ensuring that each theater room hosts and is showing a specific screening. This relationship is important to organize in which theater each screening is taking place.

The participation is **total** on both the **THEATRE** and **SCREENING** side, meaning that every screening must take place in a theater room and every theatre must be showing a screening in it.

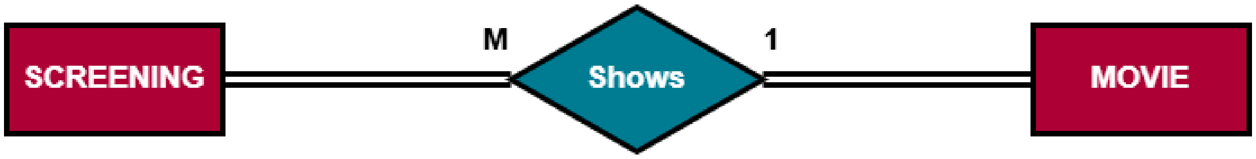
This relationship follows a **one-to-many cardinality (1:M)**, where one theater can host many screenings, but each screening happens in only one theater. This organizes the place each screening is taking place in.

10. Is At



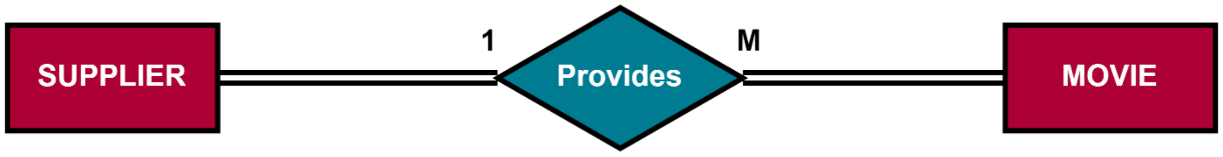
A unique seat is located at a specific screening which is shown in the "**IS AT"** relationship between the **SEAT** and **SCREENING** entity types. This is important for reservations of seats in screenings later on. The participation is **total** on both the **SEAT** and **SCREENING** side, meaning that each seat must be associated with a specific screening and cannot exist without being linked to one and that every screening must have at least one seat for the viewers to sit on.
This relationship follows a **one-to-many cardinality** (1:M), multiple seats can be assigned to a single screening, but each seat belongs to exactly one screening can't belong to multiple screenings at the same time.

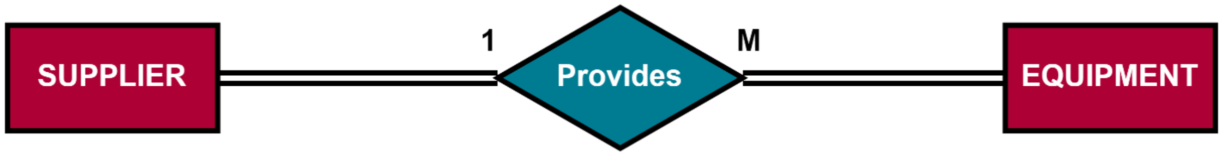
11. Shows



Each **SCREENING** will **"SHOW"** a **MOVIE**, which is the relationship that links these two entity types together. This helps identify which movie is being played at each screening and properly distribute the movies.
The participation is **total** on both the **SCREENING** and **MOVIE** side, meaning that every screening must show a specific movie and every movie must be presented in a screening.
This relationship follows a **one-to-many cardinality (1:M)**, multiple screenings can be assigned to a single movie, but each screening shows only one movie at its assigned time.

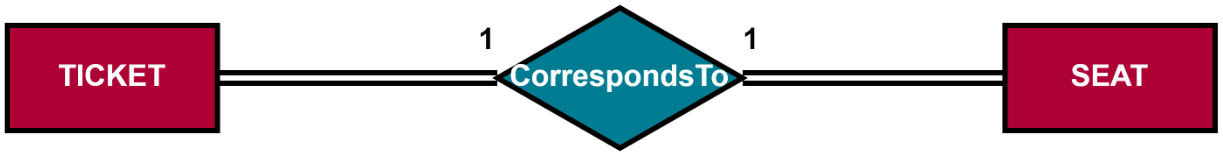
12. Provides





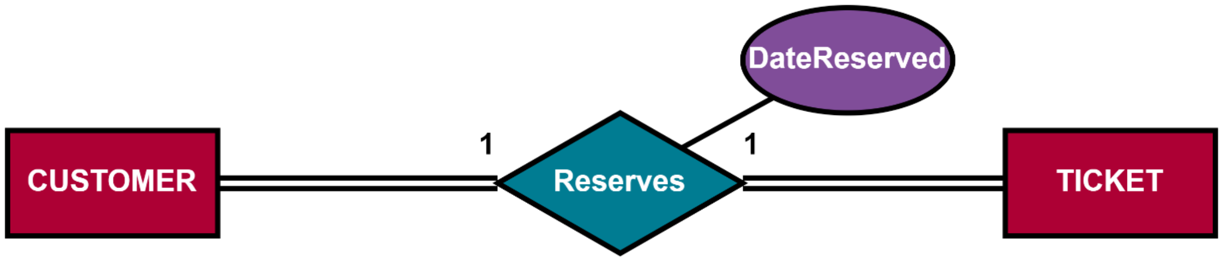
The **SUPPLIER** gives the “222” theatre **EQUIPMENT** and **MOVIES** its defined by the **“PROVIDES”** relationship. It’s important to manage what the supplier is providing and properly sort them.
The participation is **total** on both the **SUPPLIER** and **(MOVIE and EQUIPMENT)** side, meaning that each movie or equipment can’t exist or be provided without a supplier and that each supplier must provide a movie or equipment.
This relationship follows a **one-to-many** cardinality (1:M), a supplier can provide multiple movies or equipment but each of the specific movies or equipment is provided by a single supplier. For example, the supplier can be providing movies but not equipment or vice versa.

13. CorrespondsTo



The ticket that a customer buys must relate to reserve a specific seat in the theatre. This relationship is shown in a **TICKET** that **“CORRESPONDSTO”** a **SEAT**. This helps ensure that each ticket is personalized to a unique ticket and that no confusion occurs.
The participation is **total** on both the **TICKET** and **SEATS** side, meaning that every ticket must correspond to a seat and a seat can have only one ticket per screening to prevent double booking.
This relationship follows a **one-to-one** cardinality (1:1), meaning that each ticket corresponds to one seat and each seat has a ticket pointing to it. This ensures an efficient reservation system.

14. Reserves



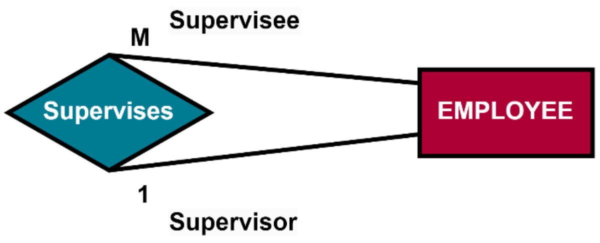
CUSTOMER can “**RESERVE” TICKET** which represents the process of booking a ticket for a specific screening. This relationship ensures that ticket reservations are properly linked to customers and that every booking is recorded for tracking purposes.

The participation is **total** on both the **CUSTOMERS** and **TICKET** side, meaning that every customer must have a ticket and **every ticket belongs to a customer.**

This relationship follows a **one-to-one cardinality (1:1)**, meaning that each ticket corresponds to one customer and each ticket is reserved by exactly one customer.

Additionally, the **DateReserved** attribute within the **RESERVED relationship** enters a specific date when the customer makes a ticket reservation.

15. Supervises

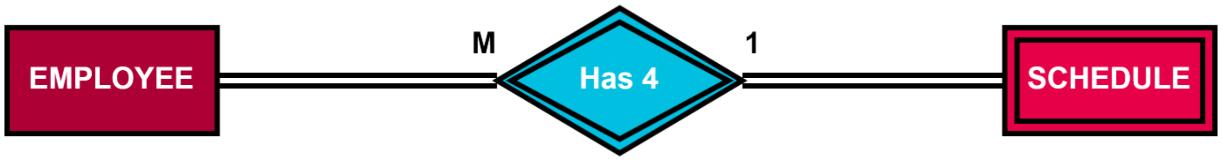


In every movie theatre, proper supervision is essential to ensure tasks are completed efficiently, operations run smoothly, and everything remains well-organized. Thus, a “**SUPERVISES**” relationship exists within the **EMPLOYEE** entity type itself, representing a hierarchical structure where some employees oversee others.

Not all employees serve as supervisors, and not all employees are assigned a supervisor. Therefore, the participation in this relationship is **partial** on both sides.

This relationship follows a **one-to-many** cardinality (1:M) cardinality, meaning that one employee can supervise multiple employees, but each employee can have only one direct supervisor.

16. Has 4

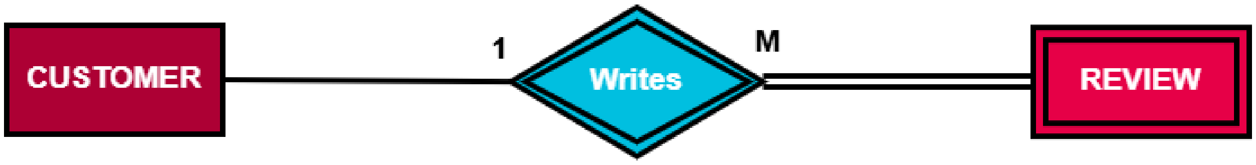


To ensure efficient time management, coordination, and fair distribution of work hours, employees are assigned schedules that they must follow. The “**HAS 4**” relationship is an identifying relationship between the **EMPLOYEE** entity type and the **SCHEDULE** entity type, which is a weak entity type dependent on the former entity type **EMPLOYEE**.

The participation in this relationship is **total** for both entity types, meaning that every employee has a schedule, and every schedule is assigned to at least one employee, ensuring that no schedule exists without an employee.

This relationship follows a **one-to-many** (1:M) cardinality, where each employee is associated with only one schedule, but a single schedule can be shared by multiple employees.

17. Writes



Customers may write multiple reviews based on their experiences, contributing valuable feedback to improve services. Thus, a "**WRITES**" relationship is created between the **CUSTOMER** entity type and the weak entity type **REVIEW**, with the relationship identifying the latter.

The participation is **total** on the **REVIEW** side, as each review must be associated with a customer, ensuring that no review exists without one. However, the participation on the **CUSTOMER** side is **partial**, as not all customers necessarily write reviews.

This relationship follows a **one-to-many** cardinality (1:M), meaning that a customer "Writes" one or more reviews, and each review must be associated with exactly one customer.

ER to Relational Mapping Algorithms

After designing the ER schema and defining the database for Triple 2 Movie Theatre as a structured system of entity types, attributes, and relationship types, the next crucial step is transforming this conceptual model into a well-structured relational database. This transformation ensures that the database design aligns with industry best practices, optimizing data storage, enforcing integrity constraints, and minimizing redundancy.

A systematic seven-step algorithm is employed to achieve this transformation, providing a clear methodology to translate the ER model into a relational schema while preserving the essential relationships and attributes of the system. The primary objectives of this design phase include maintaining data consistency, ensuring referential integrity, and enhancing query efficiency. By following these structured mapping steps, Triple 2 Movie Theatre's database will support seamless operations, from movie scheduling and ticket reservations to customer memberships and financial transactions.

Relational Database Design – Color Scheme

STRONG_ENTITY_TYPE_MAPPED_AS_ENTITY_RELATION

<u>KeyAttribute mapped as PrimaryKey</u>	Attribute	ForeignKey

WEAK_ENTITY_TYPE_MAPPED_AS_ENTITY_RELATION

<u>PartialKeyAttribute mapped as PrimaryKey</u>	<u>ForeignKey that is part of the entity type's PrimaryKey</u>	Attribute

MULTIVALUED_ATTRIBUTE_BASED_RELATION

<u>ForeignKey that is part of the entity type's PrimaryKey</u>	<u>PrimaryKey</u>

N-ARY_RELATIONSHIP_MAPPED_AS_RELATIONSHIP_RELATION

<u>ForeignKey that is part of the entity type's PrimaryKey</u>

STEP 1: Mapping of Regular Entity Types

All regular entity types are mapped into relation schemas. By regular, we mean only non-weak entity types will be mapped in this step. Each entity type is transformed into a relation, where only the simple attributes are included. Composite attributes are broken down into their atomic components. Multivalued and derived attributes are not included in this step and will be handled separately in Step 6.

The strong entity types are: EMPLOYEE, SCHEDULE, DEPARTMENT, CUSTOMER, MEMBERSHIP, PAYMENT, BRANCH, THEATRE, SCREENING, SEAT, TICKET, MOVIE, EQUIPMENT and SUPPLIER.

1. EMPLOYEE

EmployeeID	FirstName	MiddleName	LastName	Gender	DOB	Country	Region
City	Street	Building	Floor	Email	Position	Salary	ICEContact
Age							

The **EMPLOYEE** entity type includes simple, composite, multivalued, and derived attributes. The underlined attribute *EmployeeID* serves as the primary key. The *Name* attribute is composite and consists of *FirstName*, *MiddleName*, *LastName* which are included in the relation. Similarly, the *Address* attribute is composite and consists of *Country*, *Region*, *City*, *Street*, *Building*, *Floor*. Other simple attributes in the relation include *Gender*, *DOB*, *Email*, *Position*, *Salary* and *ICEContact*. Additionally, *Phone* is a multivalued and is not included in this relation, as it will be handled separately. The *Age* attribute is derived from *DOB.*

2. DEPARTMENT

EXTNumber	Name	Email

The entity type **DEPARTMENT** contains simple attribute *Email* that is added to the relation. Both *Name* and *EXTNumber* are candidate keys; however, *EXTNumber* was chosen as the primary key because numbers are more efficient and require less storage. The primary key, *EXTNumber*, is underlined.

3. CUSTOMER

CustomerID	FirstName	MiddleName	LastName	Gender	DOB	Age	Email
Username	Password	CountryCode	LocalPhone				

The entity type **CUSTOMER** contains simple, composite, and derived attributes. This relation includes the underlined primary key *CustomerID*. The **CUSTOMER** entity type has 3 composite attributes: *Name*, *Account*, and *Phone*. The *Name* attribute consists of the simple attributes *FirstName*, *MiddleName*, and *LastName*, which are included in the relation. The *Account* attribute comprises the simple attributes *Username* and *Password*. The *Phone* attribute includes the simple attributes *CountryCode* and *LocalPhone***.** Additionally, the relation contains other simple attributes: *Gender*, *DOB*, and *Email*, which provide essential customer details; and the derived attribute *Age*.

4. MEMBERSHIP

MembershipID	Type	StartDate	ExpDate

The **MEMBERSHIP** entity type includes both simple and composite attributes. The underlined attribute *MembershipID* serves as the primary key. Additionally, the composite attribute *Period* consists of the simple attributes *StartDate* and *ExpDate*, which define the membership duration and are included in the relation. The relation also contains the *Type* simple attribute, which specifies the membership category.

5. PAYMENT

PaymentID	Status	Method	Date	Amount	Discount	FinalAmount

The **PAYMENT** entity type includes simple and derived attributes. This relation consists of all simple attributes, *Status*, *Method*, *Date*, *Amount*, and *Discount*, the derived attribute *FinalAmount*, and the underlined primary key *PaymentID*, capturing essential details about each payment.

6. BRANCH

BranchID	Name	Country	Region	City	Street	LocalPhone	CountryCode

The entity type **BRANCH** contains simple and composite attributes. This relation includes simple attributes *Name* and the primary key BranchID which is underlined. Address is a composite attribute of which only the simple attributes *Country*, *Region*, *City*, and *Street* are included in the relation. Phone is also a composite attribute which we include the simple attributes *LocalPhone* and *CountryCode* in the relation.

7. THEATRE

<u>TheatreID</u>	Theatre Type	Capacity

The entity type **THEATRE** contains simple attribute *Capacity* which is added to the relation including the primary key *TheatreID* which is underlined.

8. SCREENING

<u>ScreeningID</u>	Type	Date	StartTime	EndTime

The entity type **SCREENING** contains simple and composite attributes. The underlined attribute *ScreeningID* is considered the primary key in this relation. **SCREENING** represents individual movie showings at the **222** movie theatre, ensuring proper scheduling and organization. It consists of simple attributes *Type* and *Date*, which indicate the nature of the screening and its scheduled date. The *ShowTime* attribute is a composite attribute that includes *StartTime* and *EndTime*, detailing the screening's exact duration.

9. SEAT

<u>SeatID</u>	Availability

The entity type **SEAT** contains only simple attributes. The underlined attribute *SeatID* serves as the primary key, uniquely identifying each seat within a theatre. The *Availability* attribute is used to track whether a seat is available, reserved, or occupied, allowing for real-time seat management and efficient booking.

10. TICKET

<u>TicketID</u>	Price

The **TICKET** entity type consists of only simple attributes, all of which are included in the relation. The underlined attribute *TicketID* serves as the primary key, while *Price* is an essential attribute that defines the cost of the ticket.

11. MOVIE

<u>MovieID</u>	Description	Name	Duration	ReleaseDate	PGRating	ImdbRating

The entity type **MOVIE** contains simple and multi-valued attributes. The underlined attribute *MovieID* is the primary keys for uniquely identifying each movie in the theatre's database. The relation includes simple attributes such as *Description*, *Name*, *Duration*, *ReleaseDate*, *PGRating*, and *ImdbRating*, which provide essential details about each film.

12. EQUIPMENT

<u>EquipmentID</u>	Type	Brand	Model	LastChecked

The entity type **EQUIPMENT** contains simple attributes *Type, Brand, Model, LastChecked* which are added to the relation including the primary key *EquipmentID* which is underlined.

13. SUPPLIER

<u>SupplierID</u>	Name	CountryCode	LocalPhone	Email

The entity type **SUPPLIER** contains simple and composite attributes. The underlined attribute *SupplierID* is the primary key for this relation. The SUPPLIER entity includes details about vendors providing movies and equipment to the **222** movie theatre. It consists of simple attributes *Name* and *Email*, which store the supplier's official identification and contact information. Additionally, the *Phone* attribute is a composite attribute that includes *CountryCode* and *LocalPhone*, ensuring proper communication with suppliers operating across different regions.

STEP 2: Mapping of Weak Entity Types

All weak entity types are mapped into relation schemas. Similar to Step 1, only the simple attributes are mapped. Each weak entity type is assigned a relation, which includes the primary key of the owner entity type as a foreign key. The primary key of the weak entity type is formed by combining its partial key with the primary key of the owner entity type. Composite attributes are also decomposed into simple attributes. Multivalued and derived attributes will be handled in Step 6.

The weak entity types are: SCHEDULE and REVIEW.

1. SCHEDULE

Date	EmpID	StartTime	EndTime

The **SCHEDULE** is a weak entity type and depends on the **EMPLOYEE** entity type. The *Date* attribute is a partial key, used in combination with the primary key of the **EMPLOYEE** entity type, *EmployeeID* renamed as *EmpID*, to form a complete primary key for this relation to establish uniqueness. The relation also includes simple attributes *StartTime* and *EndTime* to represent an employee's working schedule. Additionally, since weak entity relationships are inherently managed through the inclusion of the owner's primary key, no separate handling of the **"HAS4"** relationship, which links the **SCHEDULE** relation to the **EMPLOYEE** relation, is required.

2. REVIEW

Date	CustID	Comment	FacilitiesR	CleanlinessR	ServiceR	AmbianceR	AverageR

The **REVIEW** is a weak entity type that depends on the **CUSTOMER** entity type. The *Date* attribute is a partial key, used with the primary key of the **CUSTOMER** entity type, *CustomerID* renamed as *CustID*, to form a complete primary key for this relation to establish uniqueness. The relation includes the simple attributes *Comment*, *FacilitiesR*, *CleanlinessR*, *ServiceR* and *AmbianceR*, which capture customer feedback. Additionally, the *AverageR* attribute is derived from, *FacilitiesR*, *CleanlinessR*, *ServiceR* and *AmbianceR*. Additionally, since weak entity relationships are inherently managed through the inclusion of the owner's primary key, no separate handling of the "**WRITES**" relationship, which links the **REVIEW** relation to the **CUSTOMER** relation, is required.

STEP 3: Mapping of Binary 1:1 Relationship Types

In this step, we are going to map the binary one-to-one relationships. In order to accomplish our goal, we can follow one of three approaches. The first approach, called the foreign key approach, is where we choose the entity type on the total participation side of the relation, then we add as a foreign key the primary key of the other entity type participating in this relation.

The second approach, called the merged relation approach, is where we merge the two entity types participating in the relationship into a single relation. This is only used when both participations are total and thus not useful in our case.

The third approach, called the cross-reference or relationship relation approach, is where we create a third relation that includes the primary keys of both entity types participating in the relationship.

We are going to follow the foreign key approach because it is the most useful in our case. The foreign key approach is applied, where the primary key of one participating entity type is added as a foreign key to the relation on the total participation side of the relationship. If both sides have total participation, an alternative approach may be considered, such as merging the two entity types into a single relation.

The binary one-to-one relationships that need to be mapped are: Has3, CorrespondsTo and Reserves.

1. Has3

MEMBERSHIP

<u>MembershipID</u>	Type	StartDate	ExpDate	CustID

A customer can have a membership at the movie theatre “222”. The **“HAS3”** relationship serves as a link between **MEMBERSHIP** and **CUSTOMER** entity types. Since **MEMBERSHIP** has total participation in the relationship, we add the primary key of the **CUSTOMER** entity type, CustomerID renamed as *CustID*, as a foreign key in the **MEMBERSHIP** entity type.

2. CorrespondsTo

TICKET

<u>TicketID</u>	Price	SeID

The ticket that a customer buy must relate to reserve a specific seat in the theatre. The “**CORRESPONDSTO”** relationship serves as a link between **TICKET** and **SEAT** entity types. On both sides the participation is partial; thus it doesn’t matter where we add the foreign key that relates both entity types. We chose to place the foreign key to the **TICKET** relation by including the primary key *SeatID* from the **SEAT** entity type and renaming it *SeID*.

3. Reserves

TICKET

<u>TicketID</u>	Price	SeID	CustID

Customers have the options to book tickets for a movie screening. The “**RESERVES”** relationship serves as a link between **TICKET** and **CUSTOMER** entity types. On both sides the participation is partial; thus it doesn’t matter where we add the foreign key that relates both entity types. We chose to place the foreign key to the **TICKET** relation by including the primary key *CustomerID* from the **CUSTOMER** entity type and renaming it *CustID*.

STEP 4: Mapping of Binary 1:N Relationship Types

All binary 1:N relationship types are mapped into relation schemas. The foreign key approach is applied, where the primary key of the entity type on the "1" side is added as a foreign key in the relation representing the "N" side of the relationship. If the relationship contains attributes, they are stored within the "N" side relation.

The one-to-many relationships that need to be mapped are: Contains1, Contains2, WorksIn, Processes, Makes, Has1, Has2, Hosts, IsAt, Shows, and Surpervises.

1. Contains1

EQUIPMENT

<u>EquipmentID</u>	Type	Brand	Model	LastChecked	ThID

Each theatre has different equipment. The "**CONTAINS1**" relationship links the **THEATRE** and **EQUIPMENT** entity types. With **EQUIPMENT** being on the “many” side of the relationship, the primary key *TheatreID* of **THEATRE** is added as a foreign key in **EQUIPMENT** and renamed as *ThID*.

2. Contains2

EQUIPMENT

<u>EquipmentID</u>	Type	Brand	Model	LastChecked	ThID	BrID

The branches also contain equipment. The "**CONTAINS2**" relationship links the **BRANCH** and **EQUIPMENT** entity types. With **EQUIPMENT** being on the “many” side of the relationship, the primary key *BranchID* of **BRANCH** is added as a foreign key in **EQUIPMENT** and renamed as *BrID*.

3. WorksIn

EMPLOYEE

<u>EmployeeID</u>	FirstName	MiddleName	LastName	Gender	DOB	Country	Region
City	Street	Building	Floor	Email	Position	Salary	ICEContact
Age	EXTNb						

Each employee has a specific position or role in a department at 222. The "**WORKS IN**" relationship links the **EMPLOYEE** and **DEPARTMENT** entity types. With **EMPLOYEE** being on the “many” side of the relationship, the primary key *EXTNumber* of **DEPARTMENT** is added as a foreign key in **EMPLOYEE** and renamed as *EXTNb*.

4. Processes

PAYMENT

<u>PaymentID</u>	Status	Method	Date	Amount	Discount	FinalAmount	EmpID

Employees handle and manage customer transactions and payments within the theater. The "**PROCESSES**" relationship links the **EMPLOYEE** and **PAYMENT** entity types. With **PAYMENT** being on the "many" side of the relationship, the primary key *EmployeeID* of **EMPLOYEE** is added as a foreign key in **PAYMENT** and renamed as *EmpID*.

5. Makes

PAYMENT

<u>PaymentID</u>	Status	Method	Date	Amount	Discount	FinalAmount	EmpID
CustID							

A customer is required to make a payment when purchasing tickets for movie screenings at the theater**.** The "**MAKES**" relationship links the **CUSTOMER** and **PAYMENT** entity types. With **PAYMENT** being on the "many" side of the relationship, the primary key *CustomerID* of **CUSTOMER** is added as a foreign key in **PAYMENT** and renamed as *CustID*.

6. Has1

EMPLOYEE

<u>EmployeeID</u>	FirstName	MiddleName	LastName	Gender	DOB	Country	Region
City	Street	Building	Floor	Email	Position	Salary	ICEContact
Age	EXTNb	BrID					

Each employee works at a specific branch of 222. The "**HAS1**" relationship links the **BRANCH** and **EMPLOYEE** entity types. With **EMPLOYEE** being on the "many" side of the relationship, the primary key *BranchID* of **BRANCH** is added as a foreign key in **EMPLOYEE** and renamed as *BrID*.

7. Has2

THEATRE

<u>TheatreID</u>	Theatre Type	Capacity	BrID

Each branch has many theatres. The "**HAS2**" relationship links the **BRANCH** and **THEATRE** entity types. With **THEATRE** being on the “many” side of the relationship, the primary key *BranchID* of **BRANCH** is added as a foreign key in **THEATRE** and renamed as *BrID*.

8. Hosts

SCREENING

<u>ScreeningID</u>	Type	Date	StartTime	EndTime	ThID

Each theatre can host several screenings. The "**HOSTS**" relationship links the **THEATRE** and **SCREENING** entity types. With **SCREENING** being on the “many” side of the relationship, the primary key *TheatreID* of **THEATRE** is added as a foreign key in **SCREENING** and renamed as *ThID*.

9. IsAt

SEAT

<u>SeatID</u>	Availability	ScrID

A unique seat is located at a specific screening, and a screening has many seats. The "**ISAT**" relationship links the **SEAT** and **SCREENING** entity types. With **SEAT** being on the “many” side of the relationship, the primary key *ScreeningID* of **SCREENING** is added as a foreign key in **SEAT** and renamed as *ScrID*.

10. Shows

SCREENING

<u>ScreeningID</u>	Type	Date	StartTime	EndTime	ThID	MovID

Each screening presents a specific movie. The **"SHOWS"** relationship links the **SCREENING** and **MOVIE** entity types, ensuring that each scheduled screening is associated with a particular film. With **SCREENING** being on the "many" side of the relationship, the primary key *MovieID* of **MOVIE** is added as a foreign key in **SCREENING** and renamed as *MovID*.

11. Supervises

EMPLOYEE

<u>EmployeeID</u>	FirstName	MiddleName	LastName	Gender	DOB	Country	Region
City	Street	Building	Floor	Email	Position	Salary	ICEContact
EXTNb	BrID	SupervisorID					

Each employee may have a supervisor overseeing their work. The **"SUPERVISES"** relationship links the **EMPLOYEE** entity type to itself, representing a hierarchical structure where some employees oversee others. With **EMPLOYEE** being on the "many" side of the relationship, the primary key *EmployeeID* of **EMPLOYEE** is added as a foreign key in the same entity and renamed as *SupervisorID*.

STEP 5: Mapping of Binary M:N Relationship Types

All binary M:N relationship types are mapped into relation schemas. Unlike Steps 3 and 4, these relationships require the creation of a new relation, which contains the primary keys of both participating entity types as foreign keys. The combination of these foreign keys forms the primary key of the new relation. Any attributes associated with the relationship are included in this relation.

But plot twist! Turns out, we don't have any M:N relationship types to map. Time to move on to what is next, or maybe just take a well-deserved snack break :)

STEP 6: Mapping of Multivalued Attributes

All multivalued attributes that were left out from the previous steps are mapped. A separate relation is created for each multivalued attribute. This relation consists of the primary key of the entity type that owns the multivalued attribute along with the multivalued attribute itself. The combination of these forms the primary key of the newly created relation.

The multivalued attributes are: Employee_Phone, Movie_Genre, Movie_Language, and Movie_Subtitle.

1. EMPLOYEE_PHONE

<u>**EmpID**</u>	<u>**CountryCode**</u>	<u>**LocalPhoneNumber**</u>

The multivalued and composite attribute *Phone*, which includes *CountryCode* and *LocalPhoneNumber*, belongs to the **EMPLOYEE** entity type. To represent it, we create a relation called "**EMPLOYEE_PHONE**". This relation is characterized by a *EmployeeID*, *CountryCode*, and *LocalPhoneNumber*. The foreign key *EmployeeID* is the primary key of the **EMPLOYEE** entity type and ensures that each phone number is associated with a specific employee. The combination of *EmployeeID*, *CountryCode*, and *LocalPhoneNumber* guarantees the uniqueness of each phone number within the database, recognizing that an employee can have multiple unique phone numbers, potentially across different countries.

2. MOVIE_GENRE

<u>**MovID**</u>	<u>**Genre**</u>

The multivalued attribute *Genre* is associated with the **MOVIE** entity type. To represent this in the relational model, we create a relation called "**MOVIE_GENRE**". This relation includes the attribute *Genre* and a foreign key *MovID*, originally called *MovieID*, which is the primary key of the **MOVIE** entity type. The primary key of the **MOVIE_GENRE** relation is the combination of *MovID* and *Genre*, with both attributes serving as primary keys to ensure unique identification of each genre associated with a movie.

3. MOVIE_LANGUAGE

<u>**MovID**</u>	<u>**Language**</u>

The multivalued attribute *Language* belongs to the **MOVIE** entity type. To represent this in the relational model, we create a relation called "**MOVIE_LANGUAGE**". This relation includes the attribute *Language* and a foreign key *MovID*, originally called *MovieID*, which is the primary key of the **MOVIE** entity type. The primary key of the **MOVIE_LANGUAGE** relation is formed by the combination of *MovID* and *Language*, with both attributes serving as primary keys to record multiple languages for each movie uniquely.

4. MOVIE_SUBTITLE

<u>**MovID**</u>	<u>**Subtitle**</u>

To map this attribute to the relational model, we create a relation called the "**MOVIE_SUBTITLE**" relation. This relation encompasses the attribute ***Subtitle*** and a foreign key *MovID*, originally called *MovieID*, which is the primary key of the **MOVIE** entity type. The primary key of the **MOVIE_SUBTITLE** relation is the combination of *MovID* and *Subtitle*, with both attributes serving as primary keys to ensure that each set of subtitles associated with a movie is uniquely identified.

STEP 7: Mapping of N-ary Relationship Types

All N-ary relationship types are mapped into relation schemas. A new relation is created, including the primary keys of all participating entity types as foreign keys. The primary key of this relation is the combination of all these foreign keys. If the N-ary relationship includes attributes, they are also included in this relation.

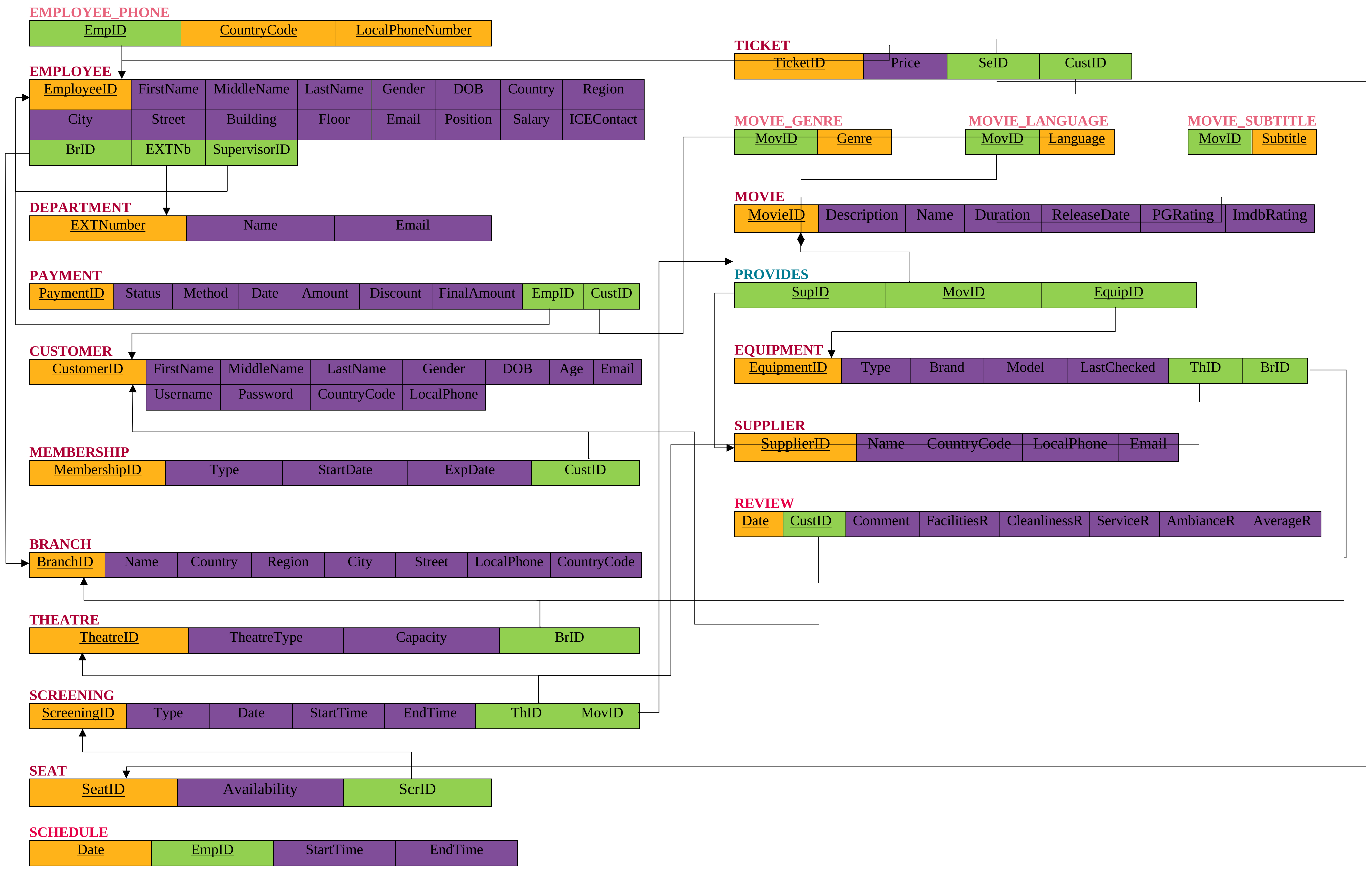
The N-ary relationship that needs to be mapped is: Provides.

1. PROVIDES

<u>**SupID**</u>	<u>**MovID**</u>	<u>**EquipID**</u>

In the ER model, the "**PROVIDES**" relationship type connects the entity types **SUPPLIER**, **MOVIE**, and **EQUIPMENT**. This relationship type is vital for tracking which supplier provides which movies or equipment. To map this relationship type to the relational model, we create a new relationship called "**PROVIDES**". This relation includes the foreign keys: *SupID* (originally called *SupID*) from the **SUPPLIER** entity type, *MovID* (originally called *MovieID*) from the **MOVIE** entity type, and *EquipID* (originally called *EquipmentID*) from the **EQUIPMENT** entity type. The primary key of **PROVIDES** is the combination of *SupID*, *MovID*, and *EquipID*. This ensures that each record in **PROVIDES** uniquely identifies one instance of a supplier providing a specific movie or piece of equipment, maintaining the integrity of the relationship across the database.

FINAL STEP: Relational Model for 222

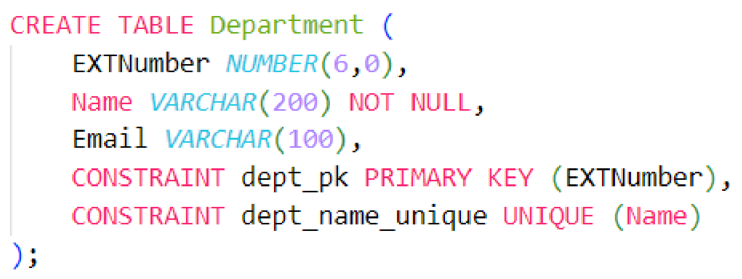


Database Creation on Oracle Server

After designing the ER diagram for 222 and mapping this diagram into relational database design, now it is time to start creating the actual tables for our database on the Oracle Database Server. We will start by creating all tables and then inserting data into these tables. Finally, we will execute some queries to display the importance of the database and especially in a hospital.

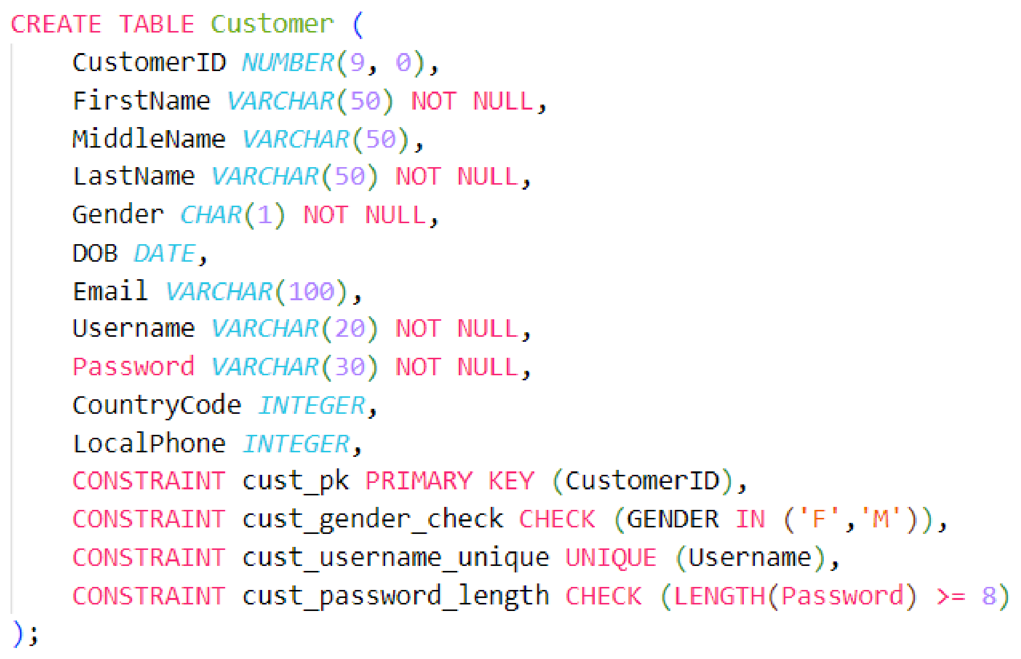
Table Structure for 222

1. DEPARTMENT



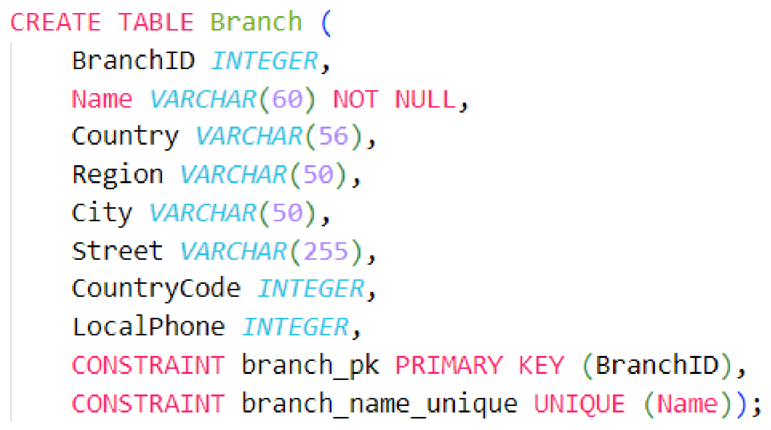
```
CREATE TABLE Department (
    EXTNumber NUMBER(6,0),
    Name VARCHAR(200) NOT NULL,
    Email VARCHAR(100),
    CONSTRAINT dept_pk PRIMARY KEY (EXTNumber),
    CONSTRAINT dept_name_unique UNIQUE (Name)
);
```

2. CUSTOMER



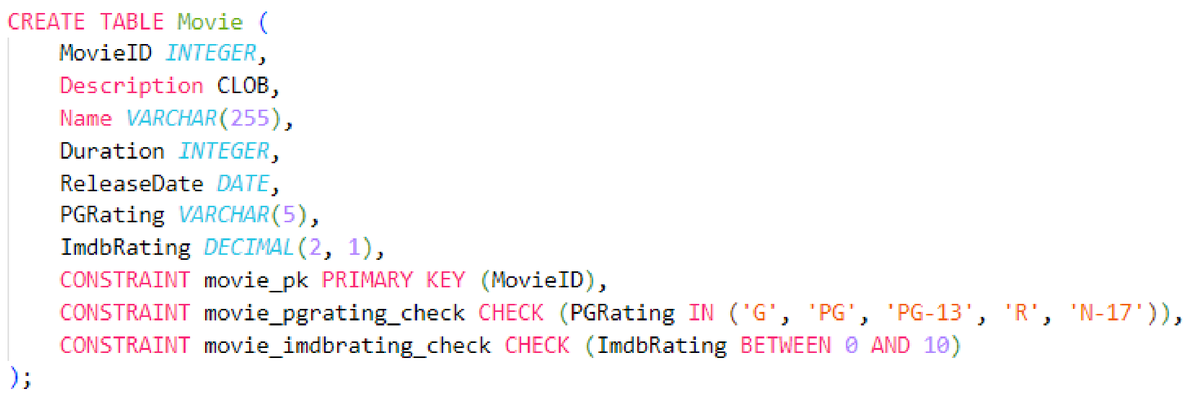
```
CREATE TABLE Customer (
    CustomerID NUMBER(9, 0),
    FirstName VARCHAR(50) NOT NULL,
    MiddleName VARCHAR(50),
    LastName VARCHAR(50) NOT NULL,
    Gender CHAR(1) NOT NULL,
    DOB DATE,
    Email VARCHAR(100),
    Username VARCHAR(20) NOT NULL,
    Password VARCHAR(30) NOT NULL,
    CountryCode INTEGER,
    LocalPhone INTEGER,
    CONSTRAINT cust_pk PRIMARY KEY (CustomerID),
    CONSTRAINT cust_gender_check CHECK (GENDER IN ('F','M')),
    CONSTRAINT cust_username_unique UNIQUE (Username),
    CONSTRAINT cust_password_length CHECK (LENGTH(Password) >= 8)
);
```

3. BRANCH



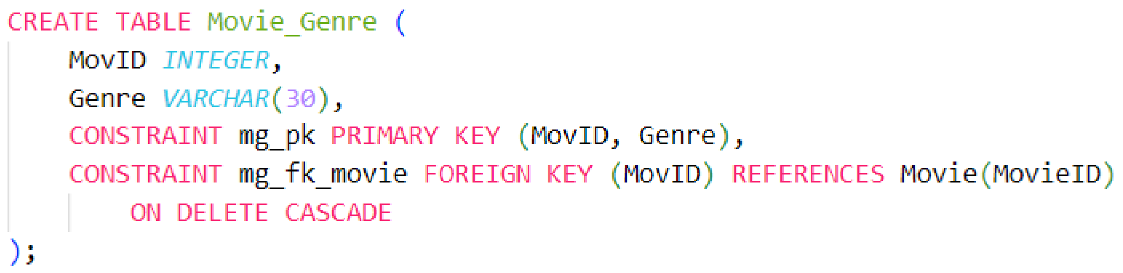
```sql
CREATE TABLE Branch (
    BranchID INTEGER,
    Name VARCHAR(60) NOT NULL,
    Country VARCHAR(56),
    Region VARCHAR(50),
    City VARCHAR(50),
    Street VARCHAR(255),
    CountryCode INTEGER,
    LocalPhone INTEGER,
    CONSTRAINT branch_pk PRIMARY KEY (BranchID),
    CONSTRAINT branch_name_unique UNIQUE (Name));
```

4. MOVIE



```sql
CREATE TABLE Movie (
    MovieID INTEGER,
    Description CLOB,
    Name VARCHAR(255),
    Duration INTEGER,
    ReleaseDate DATE,
    PGRating VARCHAR(5),
    ImdbRating DECIMAL(2, 1),
    CONSTRAINT movie_pk PRIMARY KEY (MovieID),
    CONSTRAINT movie_pgrating_check CHECK (PGRating IN ('G', 'PG', 'PG-13', 'R', 'N-17')),
    CONSTRAINT movie_imdbrating_check CHECK (ImdbRating BETWEEN 0 AND 10)
);
```

5. MOVIE_GENRE



```sql
CREATE TABLE Movie_Genre (
    MovID INTEGER,
    Genre VARCHAR(30),
    CONSTRAINT mg_pk PRIMARY KEY (MovID, Genre),
    CONSTRAINT mg_fk_movie FOREIGN KEY (MovID) REFERENCES Movie(MovieID)
        ON DELETE CASCADE
);
```

6. MOVIE_LANGUAGE

```
CREATE TABLE Movie_Language (
    MovID INTEGER,
    Language VARCHAR(30),
    CONSTRAINT ml_pk PRIMARY KEY (MovID, Language),
    CONSTRAINT ml_fk_movie FOREIGN KEY (MovID) REFERENCES Movie(MovieID)
        ON DELETE CASCADE
);
```

7. MOVIE_SUBTITLE

```
CREATE TABLE Movie_Subtitle (
    MovID INTEGER,
    Subtitle VARCHAR(30),
    CONSTRAINT ms_pk PRIMARY KEY (MovID, Subtitle),
    CONSTRAINT ms_fk_movie FOREIGN KEY (MovID) REFERENCES Movie(MovieID)
        ON DELETE CASCADE
);
```

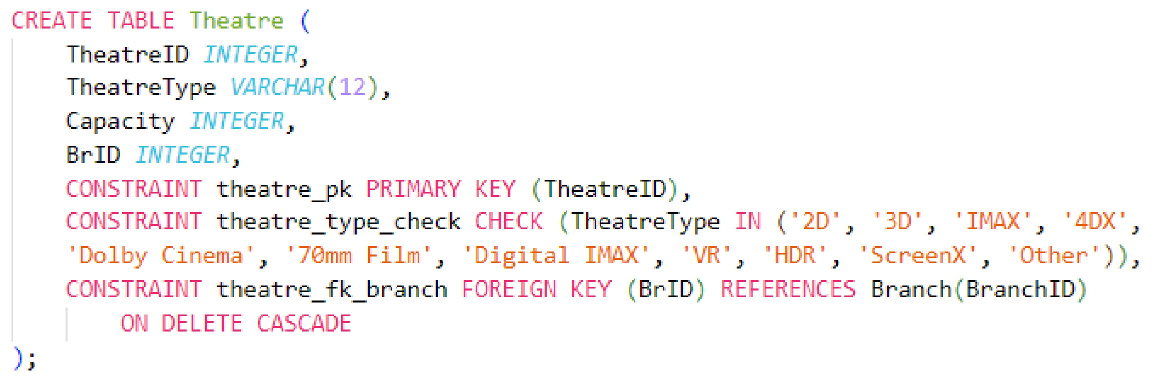
8. SUPPLIER

```
CREATE TABLE Supplier (
    SupplierID INTEGER,
    Name VARCHAR(255),
    CountryCode INTEGER,
    LocalPhone INTEGER,
    Email VARCHAR(100),
    CONSTRAINT supplier_pk PRIMARY KEY (SupplierID)
);
```

9. REVIEW

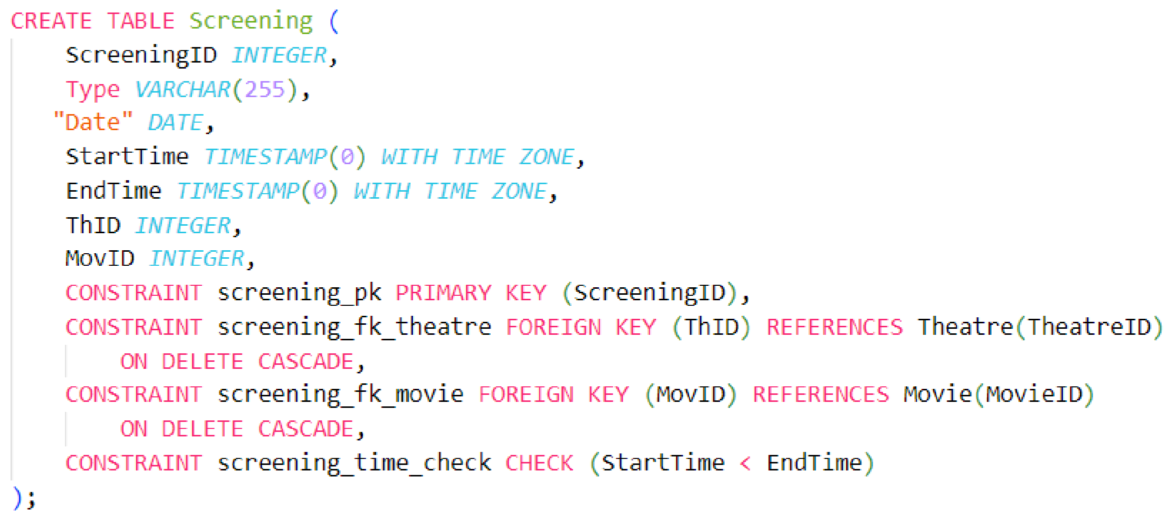
```
CREATE TABLE Review (
    ReviewID INTEGER,
  "Date" DATE,
    CustID INTEGER,
    "Comment" CLOB,
    AmbianceR DECIMAL(2,1),
    ServiceR DECIMAL(2,1),
    CleanlinessR DECIMAL(2,1),
    FacilitiesR DECIMAL(2,1),
    CONSTRAINT review_pk PRIMARY KEY (ReviewID, CustID),
    CONSTRAINT review_fk_customer FOREIGN KEY (CustID) REFERENCES Customer(CustomerID)
        ON DELETE CASCADE
);
```

10.THEATRE



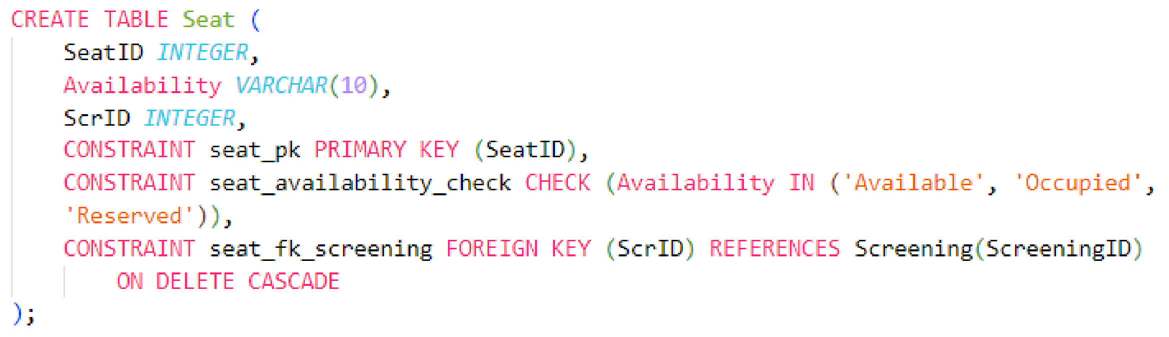
```
CREATE TABLE Theatre (
    TheatreID INTEGER,
    TheatreType VARCHAR(12),
    Capacity INTEGER,
    BrID INTEGER,
    CONSTRAINT theatre_pk PRIMARY KEY (TheatreID),
    CONSTRAINT theatre_type_check CHECK (TheatreType IN ('2D', '3D', 'IMAX', '4DX',
    'Dolby Cinema', '70mm Film', 'Digital IMAX', 'VR', 'HDR', 'ScreenX', 'Other')),
    CONSTRAINT theatre_fk_branch FOREIGN KEY (BrID) REFERENCES Branch(BranchID)
        ON DELETE CASCADE
);
```

11.SCREENING



```
CREATE TABLE Screening (
    ScreeningID INTEGER,
    Type VARCHAR(255),
   "Date" DATE,
    StartTime TIMESTAMP(0) WITH TIME ZONE,
    EndTime TIMESTAMP(0) WITH TIME ZONE,
    ThID INTEGER,
    MovID INTEGER,
    CONSTRAINT screening_pk PRIMARY KEY (ScreeningID),
    CONSTRAINT screening_fk_theatre FOREIGN KEY (ThID) REFERENCES Theatre(TheatreID)
        ON DELETE CASCADE,
    CONSTRAINT screening_fk_movie FOREIGN KEY (MovID) REFERENCES Movie(MovieID)
        ON DELETE CASCADE,
    CONSTRAINT screening_time_check CHECK (StartTime < EndTime)
);
```

12.SEAT



```
CREATE TABLE Seat (
    SeatID INTEGER,
    Availability VARCHAR(10),
    ScrID INTEGER,
    CONSTRAINT seat_pk PRIMARY KEY (SeatID),
    CONSTRAINT seat_availability_check CHECK (Availability IN ('Available', 'Occupied',
    'Reserved')),
    CONSTRAINT seat_fk_screening FOREIGN KEY (ScrID) REFERENCES Screening(ScreeningID)
        ON DELETE CASCADE
);
```

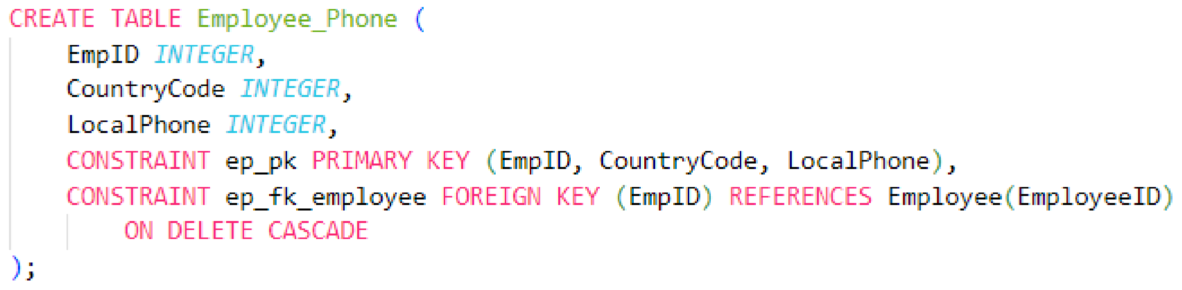
13.TICKET

```sql
CREATE TABLE Ticket (
    TicketID INTEGER,
    Price DECIMAL(5,2),
    SeID INTEGER,
    CustID INTEGER,
    CONSTRAINT ticket_pk PRIMARY KEY (TicketID),
    CONSTRAINT ticket_price_check CHECK (Price >= 0),
    CONSTRAINT ticket_fk_seat FOREIGN KEY (SeID) REFERENCES Seat(SeatID)
        ON DELETE CASCADE
    CONSTRAINT ticket_fk_customer FOREIGN KEY (CustID) REFERENCES Customer(CustomerID)
        ON DELETE CASCADE
    CONSTRAINT unique_seat_customer UNIQUE (SeID, CustID)
);
```

14.EMPLOYEE

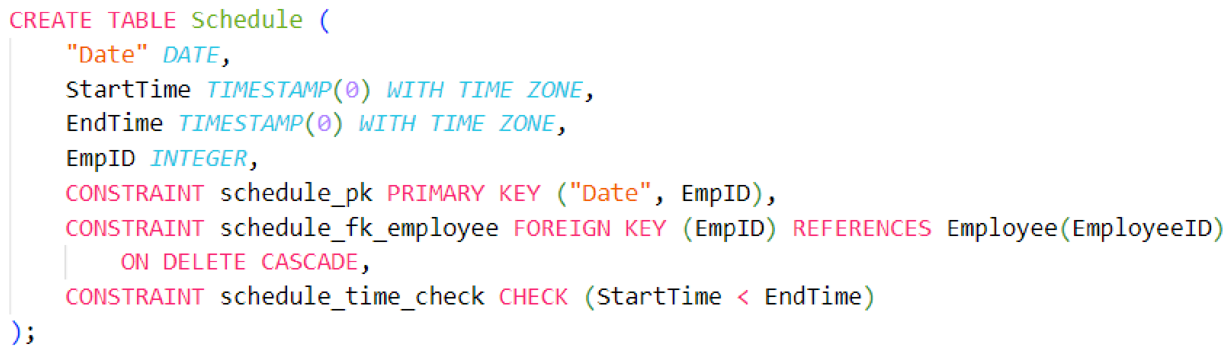
```sql
CREATE TABLE Employee (
    EmployeeID NUMBER(9, 0),
    FirstName VARCHAR(50) NOT NULL,
    MiddleName VARCHAR(50),
    LastName VARCHAR(50) NOT NULL,
    Gender CHAR(1),
    DOB DATE,
    Country VARCHAR(50),
    Region VARCHAR(50),
    City VARCHAR(50),
    Street VARCHAR(100),
    Building VARCHAR(50),
    Floor INTEGER,
    Email VARCHAR(100),
    Position VARCHAR(50),
    Salary DECIMAL(8,2),
    ICEContact VARCHAR(50),
    BrID INTEGER,
    EXTNb INTEGER,
    SupervisorID INTEGER,
    CONSTRAINT emp_pk PRIMARY KEY (EmployeeID),
    CONSTRAINT gender_check CHECK (Gender IN ('F', 'M')),
    CONSTRAINT emp_email_unique UNIQUE (Email),
    CONSTRAINT emp_super_fk FOREIGN KEY (SupervisorID) REFERENCES Employee(EmployeeID)
        ON DELETE SET NULL,
    CONSTRAINT emp_fk_branch FOREIGN KEY (BrID) REFERENCES Branch(BranchID),
    CONSTRAINT emp_fk_department FOREIGN KEY (EXTNb) REFERENCES Department(EXTNumber)
);
```

15.EMPLOYEE_PHONE



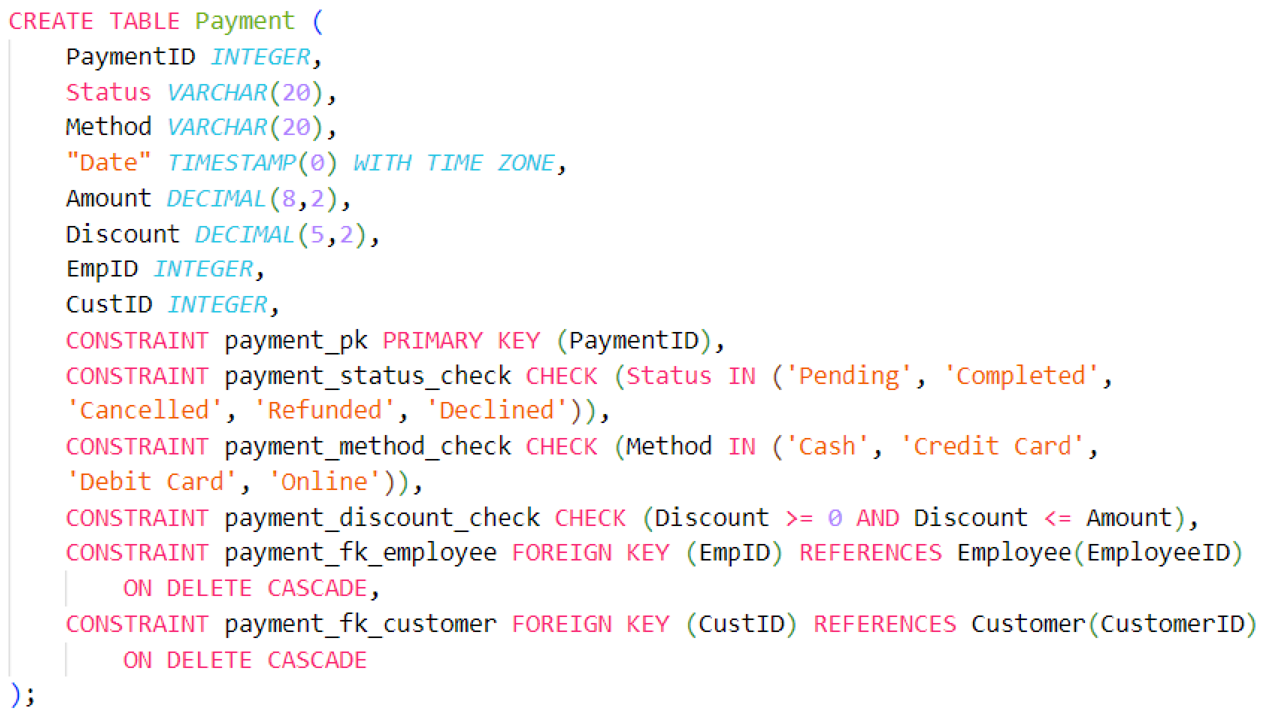
```
CREATE TABLE Employee_Phone (
    EmpID INTEGER,
    CountryCode INTEGER,
    LocalPhone INTEGER,
    CONSTRAINT ep_pk PRIMARY KEY (EmpID, CountryCode, LocalPhone),
    CONSTRAINT ep_fk_employee FOREIGN KEY (EmpID) REFERENCES Employee(EmployeeID)
        ON DELETE CASCADE
);
```

16.SCHEDULE



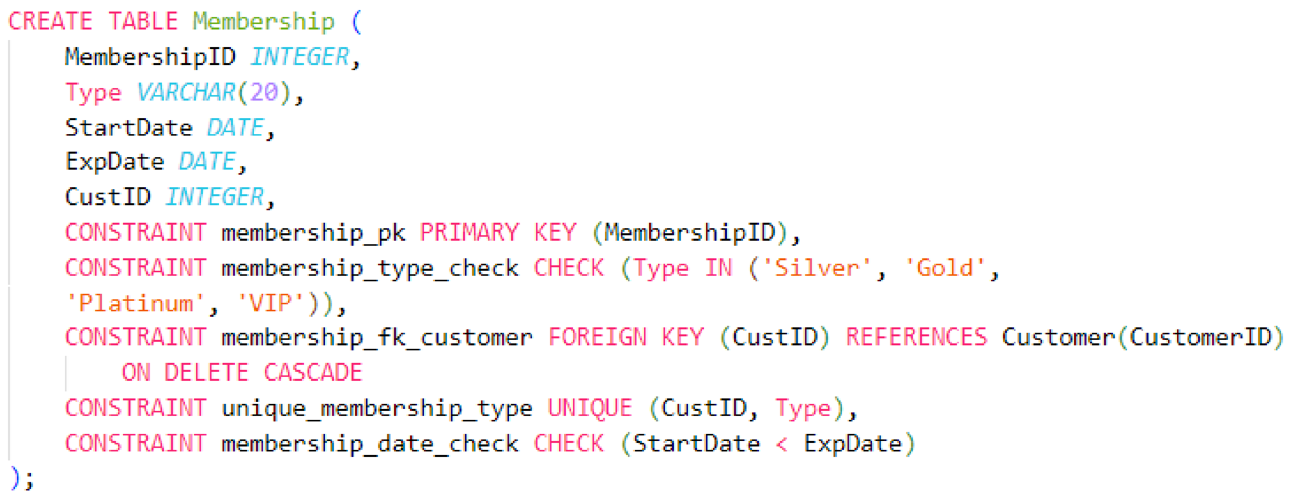
```
CREATE TABLE Schedule (
    "Date" DATE,
    StartTime TIMESTAMP(0) WITH TIME ZONE,
    EndTime TIMESTAMP(0) WITH TIME ZONE,
    EmpID INTEGER,
    CONSTRAINT schedule_pk PRIMARY KEY ("Date", EmpID),
    CONSTRAINT schedule_fk_employee FOREIGN KEY (EmpID) REFERENCES Employee(EmployeeID)
        ON DELETE CASCADE,
    CONSTRAINT schedule_time_check CHECK (StartTime < EndTime)
);
```

17.PAYMENT



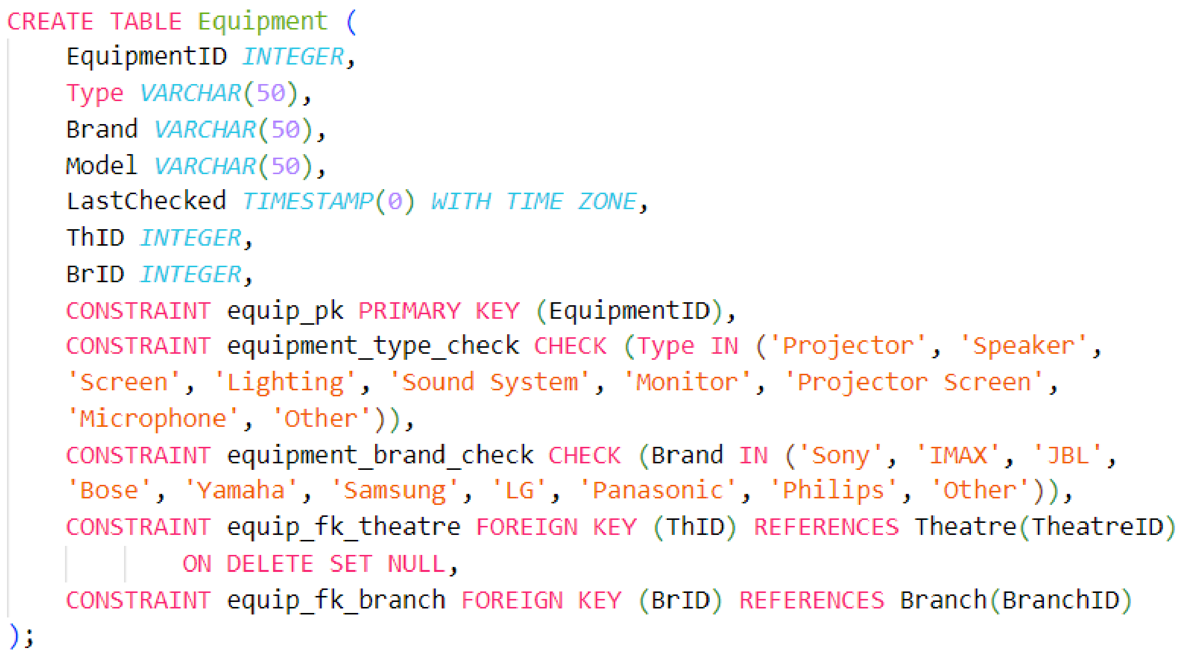
```
CREATE TABLE Payment (
    PaymentID INTEGER,
    Status VARCHAR(20),
    Method VARCHAR(20),
    "Date" TIMESTAMP(0) WITH TIME ZONE,
    Amount DECIMAL(8,2),
    Discount DECIMAL(5,2),
    EmpID INTEGER,
    CustID INTEGER,
    CONSTRAINT payment_pk PRIMARY KEY (PaymentID),
    CONSTRAINT payment_status_check CHECK (Status IN ('Pending', 'Completed',
    'Cancelled', 'Refunded', 'Declined')),
    CONSTRAINT payment_method_check CHECK (Method IN ('Cash', 'Credit Card',
    'Debit Card', 'Online')),
    CONSTRAINT payment_discount_check CHECK (Discount >= 0 AND Discount <= Amount),
    CONSTRAINT payment_fk_employee FOREIGN KEY (EmpID) REFERENCES Employee(EmployeeID)
        ON DELETE CASCADE,
    CONSTRAINT payment_fk_customer FOREIGN KEY (CustID) REFERENCES Customer(CustomerID)
        ON DELETE CASCADE
);
```

18.MEMBERSHIP



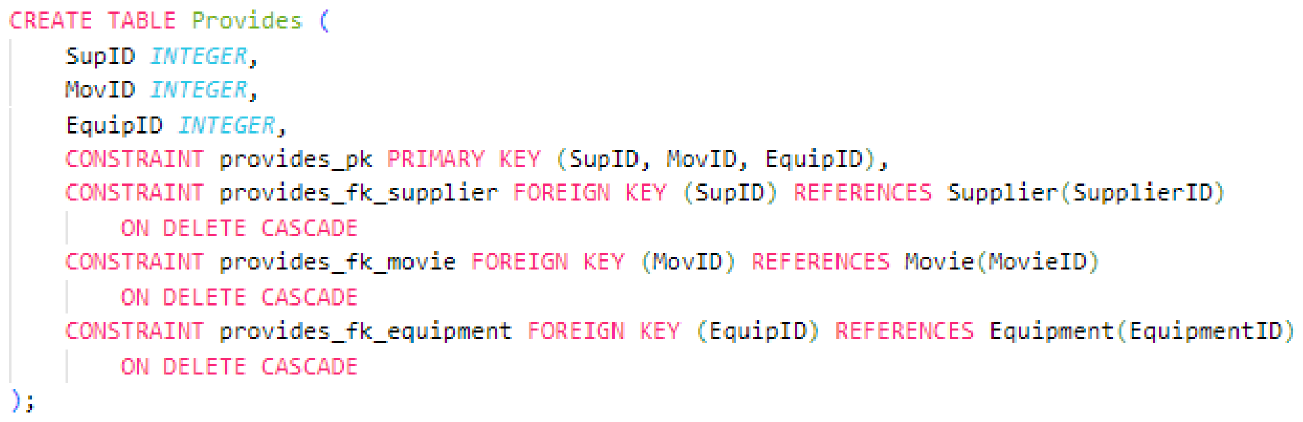
```sql
CREATE TABLE Membership (
    MembershipID INTEGER,
    Type VARCHAR(20),
    StartDate DATE,
    ExpDate DATE,
    CustID INTEGER,
    CONSTRAINT membership_pk PRIMARY KEY (MembershipID),
    CONSTRAINT membership_type_check CHECK (Type IN ('Silver', 'Gold',
    'Platinum', 'VIP')),
    CONSTRAINT membership_fk_customer FOREIGN KEY (CustID) REFERENCES Customer(CustomerID)
        ON DELETE CASCADE
    CONSTRAINT unique_membership_type UNIQUE (CustID, Type),
    CONSTRAINT membership_date_check CHECK (StartDate < ExpDate)
);
```

19.EQUIPMENT



```sql
CREATE TABLE Equipment (
    EquipmentID INTEGER,
    Type VARCHAR(50),
    Brand VARCHAR(50),
    Model VARCHAR(50),
    LastChecked TIMESTAMP(0) WITH TIME ZONE,
    ThID INTEGER,
    BrID INTEGER,
    CONSTRAINT equip_pk PRIMARY KEY (EquipmentID),
    CONSTRAINT equipment_type_check CHECK (Type IN ('Projector', 'Speaker',
    'Screen', 'Lighting', 'Sound System', 'Monitor', 'Projector Screen',
    'Microphone', 'Other')),
    CONSTRAINT equipment_brand_check CHECK (Brand IN ('Sony', 'IMAX', 'JBL',
    'Bose', 'Yamaha', 'Samsung', 'LG', 'Panasonic', 'Philips', 'Other')),
    CONSTRAINT equip_fk_theatre FOREIGN KEY (ThID) REFERENCES Theatre(TheatreID)
            ON DELETE SET NULL,
    CONSTRAINT equip_fk_branch FOREIGN KEY (BrID) REFERENCES Branch(BranchID)
);
```

20.PROVIDES



```
CREATE TABLE Provides (
    SupID INTEGER,
    MovID INTEGER,
    EquipID INTEGER,
    CONSTRAINT provides_pk PRIMARY KEY (SupID, MovID, EquipID),
    CONSTRAINT provides_fk_supplier FOREIGN KEY (SupID) REFERENCES Supplier(SupplierID)
        ON DELETE CASCADE
    CONSTRAINT provides_fk_movie FOREIGN KEY (MovID) REFERENCES Movie(MovieID)
        ON DELETE CASCADE
    CONSTRAINT provides_fk_equipment FOREIGN KEY (EquipID) REFERENCES Equipment(EquipmentID)
        ON DELETE CASCADE
);
```

N.B:

It is important to note that **Oracle does not support the `ON UPDATE` clause** for foreign key constraints. As a result, we did not include any `ON UPDATE` statements in our database schema.

However, we did consider the potential need for updating foreign key values to maintain data integrity. Although not implemented due to Oracle's limitations, we took this into account when designing the structure, ensuring that our database remains consistent and reliable.

Data Population

1. DEPARTMENT

```
INSERT ALL
    INTO Department (EXTNumber, Name, Email) VALUES (201001, 'Box Office',
'boxoffice@222.com')
    INTO Department (EXTNumber, Name, Email) VALUES (201002, 'Events & Special
Screenings' , 'events@movietheater.com')
    INTO Department (EXTNumber, Name, Email) VALUES (201003, 'Projection',
'projection@222.com')
    INTO Department (EXTNumber, Name, Email) VALUES (201004, 'Customer Service',
'customerservice@movietheater.com')
    INTO Department (EXTNumber, Name, Email) VALUES (201005, 'Marketing',
'marketing@222.com')
    INTO Department (EXTNumber, Name, Email) VALUES (201006, 'Finance',
'finance@222.com')
    INTO Department (EXTNumber, Name, Email) VALUES (201007, 'Human Resources',
'hr@222.com')
    INTO Department (EXTNumber, Name, Email) VALUES (201008, 'IT Support',
'itsupport@222.com')
    INTO Department (EXTNumber, Name, Email) VALUES (201009, 'Cleaning and
Maintenance', 'maintenance@222.com')
    INTO Department (EXTNumber, Name, Email) VALUES (201010, 'Security',
'security@222.com')
SELECT * FROM DUAL;
```

2. CUSTOMER

```
INSERT ALL
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100001, 'John', 'Michael', 'Smith', 'M', TO_DATE('1990-05-15', 'YYYY-
MM-DD'), 'john.smith@gmail.com', 'johnsmith90', 'Passw0rd123', 1, 5551234567)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100002, 'Maria', 'Luisa', 'Gonzalez', 'F', TO_DATE('1990-07-22',
'YYYY-MM-DD'), 'maria.gonzalez@hotmail.com', 'maria22', 'password456', 34,
912345679)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100003, 'Liu', 'Wei', 'Zhang', 'M', TO_DATE('1993-11-05', 'YYYY-MM-
DD'), 'liu.zhang@outlook.com','liuzhang93', 'securepass789', 86, 1387654321)
```

```
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100004, 'Olivia', NULL, 'Jones', 'F', TO_DATE('1982-02-18', 'YYYY-MM-
DD'), 'olivia.jones@outlook.com', 'olivia82', 'strongpass234', 44, 2071234567)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100005, 'Ahmed', 'Khaled', 'Al-Farsi', 'M', TO_DATE('1995-09-10',
'YYYY-MM-DD'), 'ahmed.alfarsi@gmail.com','ahmed1995', 'mypassword01', 971,
41234567)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100006, 'Sophia', 'Marie', 'Lemoine', 'F', TO_DATE('1998-06-28',
'YYYY-MM-DD'), 'sophia.lemoine@yahoo.com', 'sophie98', 'passw0rd123', 33,
767715974)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100007, 'Carlos', 'Miguel', 'Ramirez', 'M', TO_DATE('1988-01-25',
'YYYY-MM-DD'), 'carlos.ramirez@icloud.com', 'carlos88', 'mypassword98', 52,
5512345678)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100008, 'Ava', 'Marie', 'Martinez', 'F', TO_DATE('1996-11-25', 'YYYY-
MM-DD'), 'ava.martinez@gmail.com', 'avaM96', 'LoveMovies22', 52, 8529634753)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100009, 'Ava', NULL, 'Patel', 'F', TO_DATE('1992-12-11', 'YYYY-MM-
DD'), 'ava.patel@outlook.com', 'ava92', 'password7890', 91, 2212345678)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100010, 'Giovanni', 'Luciano', 'Bianchi', 'M', TO_DATE('1984-05-16',
'YYYY-MM-DD'), 'giovanni.bianchi@hotmail.com','giovanni84', 'securepassword11',
39, 0226830102)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100011, 'Jia', 'Ling', 'Wang', 'F', TO_DATE('1997-08-19', 'YYYY-MM-
DD'), 'jia.wang@outlook.com', 'jia1997', 'password1234', 86, 13912345678)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100012, 'Ethan', 'Michael', 'Brown', 'M', TO_DATE('1990-10-13',
'YYYY-MM-DD'), 'ethan.brown@gmail.com', 'ethan90', 'mypassword321', 1,
1239876543)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
```

```
    VALUES (100013, 'Isabella', 'Grace', 'Santos', 'F', TO_DATE('2000-04-09',
'YYYY-MM-DD'), 'isabella.santos@yahoo.com', 'bella2000', 'password0987', 55,
11345678)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100014, 'Tom', 'Richard', 'Williams', 'M', TO_DATE('1987-06-03',
'YYYY-MM-DD'), 'tom.williams@icloud.com', 'tom87', 'passw0rd567', 61, 212345678)
    INTO Customer (CustomerID, FirstName, MiddleName, LastName, Gender, DOB,
Email, Username, Password, CountryCode, LocalPhone)
    VALUES (100015, 'Haya', NULL , 'Mazyad', 'F', TO_DATE('2005-01-13', 'YYYY-MM-
DD'), 'haya.mazyad@outlook.com', 'hayamazyad', 'hayaM2005', 961, 81869581)
SELECT * FROM DUAL;
```

3. BRANCH

```
INSERT ALL
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (1, '222 New York', 'USA', 'New York', 'New York City', '123 Broadway
Ave', 1, 5551000001)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (2, '222 Madrid', 'Spain', 'Madrid', 'Madrid', 'Gran Via 45', 34,
910200301)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (3, '222 Shanghai', 'China', 'Shanghai', 'Shanghai', 'Huangpu District
99', 86, 2123456789)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (4, '222 London', 'UK', 'England', 'London', 'Piccadilly Street 32',
44, 2076543210)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (5, '222 Dubai', 'UAE', 'Dubai', 'Dubai', 'Sheikh Zayed Road 88', 971,
43012345)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (6, '222 Paris', 'France', 'Île-de-France', 'Paris', 'Avenue des
Champs-Élysées 78', 33, 144556677)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (7, '222 Mexico City', 'Mexico', 'Mexico City', 'Mexico City',
'Reforma Avenue 55', 52, 5512346789)
```

```
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (8, '222 Mumbai', 'India', 'Maharashtra', 'Mumbai', 'Marine Drive 21',
91, 2245678910)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (9, '222 Rome', 'Italy', 'Lazio', 'Rome', 'Via del Corso 101', 39,
0667123456)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (10, '222 Beijing', 'China', 'Beijing', 'Beijing', 'Wangfujing Street
50', 86, 1087654321)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (11, '222 São Paulo', 'Brazil', 'São Paulo', 'São Paulo', 'Avenida
Paulista 80', 55, 1132123456)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (12, '222 Sydney', 'Australia', 'New South Wales', 'Sydney', 'George
Street 67', 61, 292123456)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (13, '222 Beirut', 'Lebanon', 'Beirut', 'Beirut', 'Hamra Street 99',
961, 17456789)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (14, '222 Los Angeles', 'USA', 'California', 'Los Angeles', 'Hollywood
Blvd 88', 1, 3234567890)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (15, '222 Barcelona', 'Spain', 'Catalonia', 'Barcelona', 'La Rambla
77', 34, 932123456)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (16, '222 Berlin', 'Germany', 'Berlin', 'Berlin', 'Unter den Linden
34', 49, 3021234567)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (17, '222 Tokyo', 'Japan', 'Tokyo', 'Tokyo', 'Shinjuku Avenue 21', 81,
3523456789)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (18, '222 Buenos Aires', 'Argentina', 'Buenos Aires', 'Buenos Aires',
'Florida Street 58', 54, 1145671234)
```

```
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (19, '222 Toronto', 'Canada', 'Ontario', 'Toronto', 'Yonge Street
123', 1, 6472345678)
    INTO Branch (BranchID, Name, Country, Region, City, Street, CountryCode,
LocalPhone)
    VALUES (20, '222 Cape Town', 'South Africa', 'Western Cape', 'Cape Town',
'Long Street 31', 27, 217654321)
SELECT * FROM DUAL;
```

4. MOVIE

```
INSERT ALL
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (1, 'A computer hacker learns about the true nature of reality and his
role in the war against its controllers.', 'The Matrix', 136, TO_DATE('1999-03-
31', 'YYYY-MM-DD'), 'R', 8.7)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (2, 'The aging patriarch of an organized crime dynasty transfers
control to his reluctant son.', 'The Godfather', 175, TO_DATE('1972-03-24',
'YYYY-MM-DD'), 'R', 9.2)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (3, 'A group of explorers travel through a wormhole in space in an
attempt to ensure humanity's survival.', 'Interstellar', 169, TO_DATE('2014-11-
07', 'YYYY-MM-DD'), 'PG-13', 8.7)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (4, 'A young lion prince flees his kingdom only to learn the true
meaning of responsibility and bravery.', 'The Lion King', 88, TO_DATE('1994-06-
24', 'YYYY-MM-DD'), 'G', 8.5)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (5, 'After losing his wife, a 78-year-old balloon salesman embarks on
an adventure to South America.', 'Up', 96, TO_DATE('2009-05-29', 'YYYY-MM-DD'),
'PG', 8.3)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (6, 'In Gotham City, the Joker emerges as a criminal mastermind who
challenges Batman.', 'The Dark Knight', 152, TO_DATE('2008-07-18', 'YYYY-MM-DD'),
'PG-13', 9.0)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
```

```
    VALUES (7, 'A team of thieves enter dreams to steal secrets from their
targets.', 'Inception', 148, TO_DATE('2010-07-16', 'YYYY-MM-DD'), 'PG-13', 8.8)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (8, 'A New York cop tries to save hostages taken by terrorists in a
Los Angeles skyscraper.', 'Die Hard', 132, TO_DATE('1988-07-15', 'YYYY-MM-DD'),
'R', 8.2)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (9, 'A young boy discovers he is a wizard and attends a magical
school.', 'Harry Potter and the Sorcerer''s Stone', 152, TO_DATE('2001-11-16',
'YYYY-MM-DD'), 'PG', 7.6)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (10, 'A teenager travels back in time and inadvertently alters his
parents'' future.', 'Back to the Future', 116, TO_DATE('1985-07-03', 'YYYY-MM-
DD'), 'PG', 8.5)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (11, 'A group of friends venture into the mountains and encounter a
terrifying force.', 'The Blair Witch Project', 81, TO_DATE('1999-07-30', 'YYYY-
MM-DD'), 'R', 6.5)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (12, 'A young clownfish gets separated from his father, who sets off
to find him.', 'Finding Nemo', 100, TO_DATE('2003-05-30', 'YYYY-MM-DD'), 'G',
8.2)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (13, 'An animated adventure following a cowboy and his toy friends in
a child’s room.', 'Toy Story', 81, TO_DATE('1995-11-22', 'YYYY-MM-DD'), 'G', 8.3)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (14, 'A young woman in ancient China disguises herself as a man to
fight in a war.', 'Mulan', 88, TO_DATE('1998-06-19', 'YYYY-MM-DD'), 'G', 7.6)
    INTO Movie (MovieID, Description, Name, Duration, ReleaseDate, PGRating,
ImdbRating)
    VALUES (15, 'A scientist creates dinosaurs for a theme park, but things go
terribly wrong.', 'Jurassic Park', 127, TO_DATE('1993-06-11', 'YYYY-MM-DD'), 'PG-
13', 8.2)
SELECT 1 FROM DUAL;
```

5. MOVIE_GENRE

```
INSERT ALL
    INTO Movie_Genre (MovID, Genre) VALUES (1, 'Sci-Fi')
    INTO Movie_Genre (MovID, Genre) VALUES (1, 'Action')
    INTO Movie_Genre (MovID, Genre) VALUES (2, 'Crime')
    INTO Movie_Genre (MovID, Genre) VALUES (2, 'Drama')
    INTO Movie_Genre (MovID, Genre) VALUES (3, 'Sci-Fi')
    INTO Movie_Genre (MovID, Genre) VALUES (3, 'Drama')
    INTO Movie_Genre (MovID, Genre) VALUES (3, 'Adventure')
    INTO Movie_Genre (MovID, Genre) VALUES (4, 'Animation')
    INTO Movie_Genre (MovID, Genre) VALUES (4, 'Family')
    INTO Movie_Genre (MovID, Genre) VALUES (4, 'Musical')
    INTO Movie_Genre (MovID, Genre) VALUES (5, 'Animation')
    INTO Movie_Genre (MovID, Genre) VALUES (5, 'Adventure')
    INTO Movie_Genre (MovID, Genre) VALUES (5, 'Comedy')
    INTO Movie_Genre (MovID, Genre) VALUES (6, 'Action')
    INTO Movie_Genre (MovID, Genre) VALUES (6, 'Crime')
    INTO Movie_Genre (MovID, Genre) VALUES (6, 'Thriller')
    INTO Movie_Genre (MovID, Genre) VALUES (7, 'Sci-Fi')
    INTO Movie_Genre (MovID, Genre) VALUES (7, 'Action')
    INTO Movie_Genre (MovID, Genre) VALUES (7, 'Thriller')
    INTO Movie_Genre (MovID, Genre) VALUES (8, 'Action')
    INTO Movie_Genre (MovID, Genre) VALUES (8, 'Thriller')
    INTO Movie_Genre (MovID, Genre) VALUES (9, 'Fantasy')
    INTO Movie_Genre (MovID, Genre) VALUES (9, 'Adventure')
    INTO Movie_Genre (MovID, Genre) VALUES (9, 'Family')
    INTO Movie_Genre (MovID, Genre) VALUES (10, 'Sci-Fi')
    INTO Movie_Genre (MovID, Genre) VALUES (10, 'Adventure')
    INTO Movie_Genre (MovID, Genre) VALUES (10, 'Comedy')
    INTO Movie_Genre (MovID, Genre) VALUES (11, 'Horror')
    INTO Movie_Genre (MovID, Genre) VALUES (11, 'Mystery')
    INTO Movie_Genre (MovID, Genre) VALUES (12, 'Animation')
    INTO Movie_Genre (MovID, Genre) VALUES (12, 'Family')
    INTO Movie_Genre (MovID, Genre) VALUES (12, 'Adventure')
    INTO Movie_Genre (MovID, Genre) VALUES (13, 'Animation')
    INTO Movie_Genre (MovID, Genre) VALUES (13, 'Family')
    INTO Movie_Genre (MovID, Genre) VALUES (13, 'Comedy')
    INTO Movie_Genre (MovID, Genre) VALUES (14, 'Animation')
    INTO Movie_Genre (MovID, Genre) VALUES (14, 'Action')
    INTO Movie_Genre (MovID, Genre) VALUES (14, 'Adventure')
    INTO Movie_Genre (MovID, Genre) VALUES (15, 'Sci-Fi')
    INTO Movie_Genre (MovID, Genre) VALUES (15, 'Adventure')
    INTO Movie_Genre (MovID, Genre) VALUES (15, 'Thriller')
SELECT 1 FROM DUAL;
```

6. MOVIE_LANGUAGE

```
INSERT ALL
    INTO Movie_Language (MovID, Language) VALUES (1, 'English')
    INTO Movie_Language (MovID, Language) VALUES (1, 'French')
    INTO Movie_Language (MovID, Language) VALUES (2, 'English')
    INTO Movie_Language (MovID, Language) VALUES (2, 'Spanish')
    INTO Movie_Language (MovID, Language) VALUES (3, 'English')
    INTO Movie_Language (MovID, Language) VALUES (4, 'English')
    INTO Movie_Language (MovID, Language) VALUES (5, 'English')
    INTO Movie_Language (MovID, Language) VALUES (5, 'Japanese')
    INTO Movie_Language (MovID, Language) VALUES (6, 'English')
    INTO Movie_Language (MovID, Language) VALUES (6, 'Hindi')
    INTO Movie_Language (MovID, Language) VALUES (7, 'English')
    INTO Movie_Language (MovID, Language) VALUES (7, 'Mandarin')
    INTO Movie_Language (MovID, Language) VALUES (8, 'English')
    INTO Movie_Language (MovID, Language) VALUES (8, 'Korean')
    INTO Movie_Language (MovID, Language) VALUES (9, 'English')
    INTO Movie_Language (MovID, Language) VALUES (9, 'French')
    INTO Movie_Language (MovID, Language) VALUES (9, 'German')
    INTO Movie_Language (MovID, Language) VALUES (10, 'English')
    INTO Movie_Language (MovID, Language) VALUES (10, 'Italian')
    INTO Movie_Language (MovID, Language) VALUES (11, 'English')
    INTO Movie_Language (MovID, Language) VALUES (11, 'Spanish')
    INTO Movie_Language (MovID, Language) VALUES (12, 'English')
    INTO Movie_Language (MovID, Language) VALUES (12, 'Portuguese')
    INTO Movie_Language (MovID, Language) VALUES (13, 'English')
    INTO Movie_Language (MovID, Language) VALUES (13, 'Arabic')
    INTO Movie_Language (MovID, Language) VALUES (13, 'French')
    INTO Movie_Language (MovID, Language) VALUES (14, 'English')
    INTO Movie_Language (MovID, Language) VALUES (14, 'German')
    INTO Movie_Language (MovID, Language) VALUES (15, 'English')
SELECT 1 FROM DUAL;
```

7. MOVIE_SUBTITLE

```
INSERT ALL
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (1, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (1, 'Spanish')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (2, 'Spanish')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (2, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (3, 'Spanish')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (3, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (3, 'German')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (4, 'German')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (4, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (5, 'Japanese')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (5, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (6, 'Hindi')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (6, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (7, 'Mandarin')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (7, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (8, 'Korean')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (8, 'Japanese')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (9, 'German')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (9, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (9, 'Italian')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (10, 'Italian')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (10, 'Spanish')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (11, 'Spanish')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (11, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (12, 'Portuguese')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (12, 'Spanish')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (13, 'Arabic')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (13, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (14, 'German')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (14, 'Spanish')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (15, 'French')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (15, 'Italian')
    INTO Movie_Subtitle (MovID, Subtitle) VALUES (15, 'Spanish')
SELECT 1 FROM DUAL;
```

8. SUPPLIER

```
INSERT ALL
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (1,
'Acme Electronics', 1, 1234567890, 'contact@acmeelectronics.com')
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (2,
'BestTech Solutions', 44, 9876543210, 'info@besttechsolutions.co.uk')
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (3,
'Global Industries', 61, 2345678901, 'sales@globalindustries.com.au')
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (4,
'MegaGoods Supplier', 49, 3456789012, 'support@megagoodssupplier.de')
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (5,
'TechWave Suppliers', 33, 4567890123, 'contact@techwavesuppliers.fr')
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (6,
'FutureTech Providers', 81, 5678901234, 'info@futuretechproviders.jp')
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (7,
'EliteSupply Co.', 34, 6789012345, 'support@elitesupplyco.es')
```

```
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (8,
'TechMasters Group', 55, 7890123456, 'contact@techmastersgroup.br')
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (9,
'Innovative Supplies', 91, 8901234567, 'info@innovativesupplies.in')
    INTO Supplier (SupplierID, Name, CountryCode, LocalPhone, Email) VALUES (10,
'Prime Supplies', 52, 9012345678, 'sales@primesupplies.mx')
SELECT 1 FROM DUAL;
```

9. REVIEW

```
INSERT ALL
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (1, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100001, 'Great experience,
would love to come back!', 4.5, 4.5, 4.7, 4.8)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (2, TO_DATE('2025-03-02', 'YYYY-MM-DD'), 100002, 'Excellent service,
but the ambiance could improve.', 4.0, 4.7, 4.6, 4.5)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (3, TO_DATE('2024-11-02', 'YYYY-MM-DD'), 100003, 'Loved the
facilities, but the service was slow.', 4.3, 3.8, 4.5, 4.7)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (4, TO_DATE('2023-04-02', 'YYYY-MM-DD'), 100004, 'The place is clean,
but it was too noisy for my taste.', 3.8, 4.2, 4.7, 4.6)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (5, TO_DATE('2025-02-02', 'YYYY-MM-DD'), 100005, 'Wonderful ambiance,
perfect for a family outing.', 5.0, 4.9, 4.8, 4.9)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (6, TO_DATE('2025-01-13', 'YYYY-MM-DD'), 100006, 'Friendly staff, and
the facilities were top-notch!', 4.8, 5.0, 4.9, 5.0)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (7, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100007, 'It was an okay
experience, nothing extraordinary.', 3.9, 3.8, 4.0, 4.2)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (8, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100008, 'I had a great time
with friends, would visit again!', 4.6, 4.7, 4.5, 4.6)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
```

```
    VALUES (9, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100009, 'The food was
amazing, but it was quite crowded.', 4.2, 4.4, 4.7, 4.5)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (10, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100010, 'I loved the vibe,
but the service was a bit slow.', 4.3, 3.9, 4.8, 4.6)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (11, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100011, 'The staff was very
friendly, but the facilities could be improved.', 4.0, 4.5, 4.3, 4.4)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (12, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100012, 'Had a good time,
but I expected more variety in the menu.', 4.2, 4.1, 4.6, 4.5)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (13, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100013, 'Amazing food and
atmosphere, I will definitely be back!', 5.0, 5.0, 5.0, 5.0)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (14, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100014, 'The place was nice
but a little too pricey for my liking.', 3.7, 3.9, 4.4, 4.3)
    INTO Review (ReviewID, "Date", CustID, "Comment" , AmbianceR, ServiceR,
CleanlinessR, FacilitiesR)
    VALUES (15, TO_DATE('2025-04-02', 'YYYY-MM-DD'), 100015, 'I had a wonderful
experience, everything was perfect!', 5.0, 5.0, 5.0, 5.0)
SELECT 1 FROM DUAL;
```

10. THEATRE

```
INSERT ALL
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (1, '2D', 250, 1)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (2, '3D', 200, 2)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (3, 'IMAX', 150, 3)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (4, '4DX', 180, 4)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (5, 'Dolby Cinema', 220, 5)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (6, '70mm Film', 120, 6)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (7, 'Digital IMAX', 300, 7)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (8, 'VR', 100, 8)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (9, 'HDR', 180, 9)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (10, 'ScreenX', 160, 10)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (11, '2D', 250, 11)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (12, '3D', 200, 12)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (13, 'IMAX', 150, 13)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (14, '4DX', 180, 14)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (15, 'Dolby Cinema', 220, 15)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (16, '70mm Film', 120, 16)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (17, 'Digital IMAX', 300, 17)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (18, 'VR', 100, 18)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (19, 'HDR', 180, 19)
    INTO Theatre (TheatreID, TheatreType, Capacity, BrID)
    VALUES (20, 'ScreenX', 160, 20)
SELECT * FROM DUAL;
```

11. SCREENING

```
INSERT ALL
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (1, 'Regular', DATE '2025-04-03', TIMESTAMP '2025-04-03 18:00:00 -
05:00', TIMESTAMP '2025-04-03 20:30:00 -05:00', 1, 1)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (2, 'IMAX', DATE '2025-04-03', TIMESTAMP '2025-04-03 19:00:00 +01:00',
TIMESTAMP '2025-04-03 21:45:00 +01:00', 2, 1)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (3, '3D', DATE '2025-04-03', TIMESTAMP '2025-04-03 20:00:00 +08:00',
TIMESTAMP '2025-04-03 22:50:00 +08:00', 3, 1)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (4, '4DX', DATE '2025-04-03', TIMESTAMP '2025-04-03 17:30:00 +00:00',
TIMESTAMP '2025-04-03 19:00:00 +00:00', 4, 4)
```

```
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (5, 'Dolby Cinema', DATE '2025-04-03', TIMESTAMP '2025-04-03 21:00:00
+04:00', TIMESTAMP '2025-04-03 22:40:00 +04:00', 5, 5)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (6, 'Regular', DATE '2025-04-03', TIMESTAMP '2025-04-03 15:00:00 -
06:00', TIMESTAMP '2025-04-03 17:30:00 -06:00', 6, 6)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (7, 'IMAX', DATE '2025-04-03', TIMESTAMP '2025-04-03 14:30:00 -03:00',
TIMESTAMP '2025-04-03 17:00:00 -03:00', 7, 7)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (8, '70mm Film', DATE '2025-04-03', TIMESTAMP '2025-04-03 20:00:00
+05:30', TIMESTAMP '2025-04-03 22:30:00 +05:30', 8, 8)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (9, 'VR', DATE '2025-04-03', TIMESTAMP '2025-04-03 19:00:00 +02:00',
TIMESTAMP '2025-04-03 21:00:00 +02:00', 9, 9)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (10, 'HDR', DATE '2025-04-03', TIMESTAMP '2025-04-03 18:00:00 -08:00',
TIMESTAMP '2025-04-03 20:15:00 -08:00', 10, 1)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (11, 'Digital IMAX', DATE '2025-04-03', TIMESTAMP '2025-04-03 21:30:00
+09:00', TIMESTAMP '2025-04-03 23:50:00 +09:00', 11, 11)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (12, 'ScreenX', DATE '2025-04-03', TIMESTAMP '2025-04-03 16:45:00
+11:00', TIMESTAMP '2025-04-03 19:15:00 +11:00', 12, 12)
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (13, '4DX', DATE '2025-04-12', TIMESTAMP '2025-04-12 17:00:00 +00:00',
TIMESTAMP '2025-04-12 19:00:00 +00:00', 4, 2);
    INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID, MovID)
    VALUES (14, 'IMAX', DATE '2025-04-13', TIMESTAMP '2025-04-13 20:00:00
+01:00', TIMESTAMP '2025-04-13 22:30:00 +01:00', 2, 3);

SELECT 1 FROM DUAL;
```

12. SEAT

```
INSERT ALL
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (1, 'Available', 1)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (2, 'Occupied', 1)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (3, 'Reserved', 1)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (4, 'Available', 1)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (5, 'Available', 2)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (6, 'Occupied', 2)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (7, 'Reserved', 2)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (8, 'Available', 2)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (9, 'Available', 3)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (10, 'Occupied', 3)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (11, 'Reserved', 3)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (12, 'Available', 3)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (13, 'Available', 4)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (14, 'Occupied', 4)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (15, 'Reserved', 4)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (16, 'Available', 4)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (17, 'Available', 5)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (18, 'Occupied', 5)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (19, 'Reserved', 5)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (20, 'Available', 5)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (21, 'Reserved', 13)
    INTO Seat (SeatID, Availability, ScrID)
    VALUES (22, 'Reserved', 14)
SELECT 1 FROM DUAL;
```

13. TICKET

```
INSERT ALL
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (1, 15.00, 1, 100001)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (2, 20.00, 2, 100001)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (3, 18.00, 3, 100003)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (4, 25.00, 4, 100003)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (5, 22.00, 5, 100003)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (6, 15.00, 6, 100006)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (7, 20.00, 7, 100006)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (8, 17.00, 8, 100008)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (9, 18.50, 9, 100009)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (10, 16.00, 10, 100010)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (11, 25.00, 11, 100011)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (12, 19.00, 12, 100012)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (13, 19.00, 21, 100013)
    INTO Ticket (TicketID, Price, SeID, CustID)
    VALUES (14, 22.00, 22, 100014)
SELECT 1 FROM DUAL;
```

14. EMPLOYEE

```
-- Management & Administration
   INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
    VALUES (1, 'James', 'Andrew', 'Smith', 'M', TO_DATE('1980-03-25', 'YYYY-MM-
DD'), 'USA', 'California', 'Los Angeles', 'Hollywood Blvd', '88', 2,
'james.smith@222.com', 'Theater Manager', 6500.00, 'Emma Smith', 14, NULL, NULL)
   INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (2, 'Sophie', NULL, 'Dupont', 'F', TO_DATE('1985-07-19', 'YYYY-MM-DD'),
'France', 'Île-de-France', 'Paris', 'Avenue des Champs', '78', 3,
'sophie.dupont@222.com', 'Assistant Manager', 4800.00, 'Louis Dupont', 6, NULL,
1)
   INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
```

```
 VALUES (3, 'Liam', 'Michael', 'Johnson', 'M', TO_DATE('1990-05-10', 'YYYY-MM-
DD'), 'Canada', 'Ontario', 'Toronto', 'Yonge Street', '123', 1,
'liam.johnson@222.com', 'Box Office Manager', 4200.00, 'Sarah Johnson', 19,
201001, 1)
   INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (4, 'Aisha', NULL, 'Al-Farsi', 'F', TO_DATE('1988-11-02', 'YYYY-MM-DD'),
'UAE', 'Dubai', 'Dubai', 'Sheikh Zayed Road', '88', 4, 'aisha.alfarsi@222.com',
'Events and Special Screening Manager', 4000.00, 'Mohammed Al-Farsi', 5, 201002,
1)
   INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (5, 'Chen', 'Wei', 'Zhang', 'M', TO_DATE('1986-02-18', 'YYYY-MM-DD'),
'China', 'Beijing', 'Beijing', 'Wangfujing Street', '50', 2,
'chen.zhang@222.com', 'Customer Service Manager', 4300.00, 'Ying Zhang', 10,
201004, 1)
   INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (6, 'Daniel', 'Scott', 'Miller', 'M', TO_DATE('1982-09-14', 'YYYY-MM-DD'),
'Australia', 'New South Wales', 'Sydney', 'George Street', '67', 3,
'daniel.miller@222.com', 'Marketing Coordinator', 3900.00, 'Amy Miller', 12,
201005, 1)

-- Box Office Staff
    INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (7, 'Jack', 'Thomas', 'Brown', 'M', TO_DATE('1995-01-05', 'YYYY-MM-DD'),
'USA', 'New York', 'New York City', 'Broadway Ave', '123', 2,
'jack.brown@222.com', 'Head Cashier', 3200.00, 'Laura Brown', 1, 201001, 3)
    INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (8, 'Isabella', 'Marie', 'Rossi', 'F', TO_DATE('1997-12-14', 'YYYY-MM-
DD'), 'Italy', 'Lazio', 'Rome', 'Via del Corso', '101', 1,
'isabella.rossi@222.com', 'Ticket Seller (Cashier)', 2900.00, 'Marco Rossi', 9,
201001, 7)

-- Events and Special Screenin
    INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
```

```
VALUES (9, 'Carlos', NULL, 'Lopez', 'M', TO_DATE('1994-08-21', 'YYYY-MM-DD'),
'Mexico', 'Mexico City', 'Mexico City', 'Reforma Avenue', '55', 2,
'carlos.lopez@222.com', 'Events and Special Screening Supervisor', 3500.00, 'Ana
Lopez', 7, 201002, 4)
    INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (10, 'Maya', NULL, 'Haddad', 'F', TO_DATE('1996-04-17', 'YYYY-MM-DD'),
'Lebanon', 'Beirut', 'Beirut', 'Hamra Street', '99', 1, 'maya.haddad@222.com',
'Events and Special Screening Worker', 2700.00, 'Jad Haddad', 13, 201002, 9)

-- Customer Service
    INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (11, 'Ethan', 'David', 'Nguyen', 'M', TO_DATE('1998-10-09', 'YYYY-MM-DD'),
'South Africa', 'Western Cape', 'Cape Town', 'Long Street', '31', 1,
'ethan.nguyen@222.com', 'Usher', 2500.00, 'Linda Nguyen', 20, 201004, 5)

-- Technical & Maintenance Staff (Projection + Cleaning and Maintenance)
     INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (13, 'Tom', 'Edward', 'Wilson', 'M', TO_DATE('1983-11-11', 'YYYY-MM-DD'),
'Japan', 'Tokyo', 'Tokyo', 'Shinjuku Avenue', '21', 2, 'tom.wilson@222.com',
'Projection Manager', 5000.00, 'Emma Wilson', 17, 201003, 1)
    INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (12, 'Oliver', NULL, 'Lee', 'M', TO_DATE('1995-07-30', 'YYYY-MM-DD'),
'China', 'Shanghai', 'Shanghai', 'Huangpu District', '99', 2,
'oliver.lee@222.com', 'Projectionist', 3100.00, 'Kevin Lee', 3, 201003, 13)
    INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (14, 'Hassan', NULL, 'Khan', 'M', TO_DATE('1992-05-13', 'YYYY-MM-DD'),
'Mexico', 'Mexico City', 'Mexico City', 'Reforma Avenue', '55', 2,
'hassan.khan@222.com', 'Cleaning Supervisor', 2800.00, 'Ali Khan', 7, 201009, 1)

-- Security Staff
    INSERT INTO Employee(EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
 VALUES (16, 'Walter', NULL, 'White', 'M', TO_DATE('1980-09-12', 'YYYY-MM-DD'),
'USA', 'New York', 'New York City', 'Broadway Ave', '123', 2,
'walter.white@222.com', 'Security Supervisor', 4000.00, 'Skyler White', 1,
201010, 1)
    INSERT INTO Employee (EmployeeID, FirstName, MiddleName, LastName, Gender,
DOB, Country, Region, City, Street, Building, Floor, Email, Position, Salary,
ICEContact, BrID, EXTNb, SupervisorID)
VALUES (15, 'Lucy', 'Jane', 'Taylor', 'F', TO_DATE('1996-12-07', 'YYYY-MM-DD'),
'Spain', 'Madrid', 'Madrid', 'Gran Via', '45', 3, 'lucy.taylor@222.com',
'Security Guard', 2700.00, 'Robert Taylor', 2, 201010, 16)
```

15. EMPLOYEE_PHONE

```
INSERT ALL
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (1, 1,
2135551001)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (7, 1,
2125553003)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (7, 1,
6465554000)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (2, 33,
175432198)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (3, 1,
4165552002)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (4, 971,
502345678)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (5, 86,
1087654321)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (12, 86,
2165432109)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (12, 61,
291234567)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (8, 39,
0649876543)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (9, 52,
5512345678)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (14, 52,
5578901234)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (10, 961,
31234567)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (11, 27,
218765432)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (11, 27,
218789423)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (11, 33,
795526438)
```

```
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (13, 81, 
364512345)
    INTO Employee_Phone (EmpID, CountryCode, LocalPhone) VALUES (15, 34, 
914567890)
SELECT * FROM DUAL;
```

16. SCHEDULE

```
INSERT ALL
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03 
08:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 16:00:00', 'YYYY-
MM-DD HH24:MI:SS'), 1)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03 
09:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 17:00:00', 'YYYY-
MM-DD HH24:MI:SS'), 2)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03 
10:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 18:00:00', 'YYYY-
MM-DD HH24:MI:SS'), 3)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03 
11:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 19:00:00', 'YYYY-
MM-DD HH24:MI:SS'), 4)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03 
16:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 23:59:00', 'YYYY-
MM-DD HH24:MI:SS'), 5)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03 
17:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 23:59:00', 'YYYY-
MM-DD HH24:MI:SS'), 6)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03 
20:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 23:59:00', 'YYYY-
MM-DD HH24:MI:SS'), 7)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03 
22:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 23:59:00', 'YYYY-
MM-DD HH24:MI:SS'), 8)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03 
23:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 23:59:00', 'YYYY-
MM-DD HH24:MI:SS'), 9)
```

```
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03
08:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 16:00:00', 'YYYY-
MM-DD HH24:MI:SS'), 10)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03
16:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 23:59:00', 'YYYY-
MM-DD HH24:MI:SS'), 11)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03
23:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 23:59:00', 'YYYY-
MM-DD HH24:MI:SS'), 12)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03
10:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 18:00:00', 'YYYY-
MM-DD HH24:MI:SS'), 13)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03
19:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 23:59:00', 'YYYY-
MM-DD HH24:MI:SS'), 14)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03
08:00:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 16:00:00', 'YYYY-
MM-DD HH24:MI:SS'), 15)
    INTO Schedule ("Date", StartTime, EndTime, EmpID)
    VALUES (TO_DATE('2025-04-03', 'YYYY-MM-DD'), TO_TIMESTAMP('2025-04-03
20:30:00', 'YYYY-MM-DD HH24:MI:SS'), TO_TIMESTAMP('2025-04-03 23:59:00', 'YYYY-
MM-DD HH24:MI:SS'), 16)
SELECT * FROM DUAL;
```

17. PAYMENT

```
INSERT ALL
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (1, 'Completed', 'Credit Card', TO_TIMESTAMP('2025-04-03
14:30:00', 'YYYY-MM-DD HH24:MI:SS'), 45.50, 5.00, 7, 100001)
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (2, 'Pending', 'Debit Card', TO_TIMESTAMP('2025-04-03 15:00:00',
'YYYY-MM-DD HH24:MI:SS'), 33.00, 3.00, 8, 100002)
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (3, 'Completed', 'Cash', TO_TIMESTAMP('2025-04-03 16:00:00',
'YYYY-MM-DD HH24:MI:SS'), 72.00, 0.00, 7, 100003)
```

```
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (4, 'Cancelled', 'Online', TO_TIMESTAMP('2025-04-03 17:30:00',
'YYYY-MM-DD HH24:MI:SS'), 25.00, 2.00, 8, 100004)
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (5, 'Refunded', 'Credit Card', TO_TIMESTAMP('2025-04-03 18:00:00',
'YYYY-MM-DD HH24:MI:SS'), 50.00, 5.00, 7, 100005)
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (6, 'Completed', 'Debit Card', TO_TIMESTAMP('2025-04-03 18:30:00',
'YYYY-MM-DD HH24:MI:SS'), 60.00, 7.50, 8, 100006)
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (7, 'Completed', 'Cash', TO_TIMESTAMP('2025-04-03 19:00:00',
'YYYY-MM-DD HH24:MI:SS'), 55.00, 5.00, 7, 100007)
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (8, 'Pending', 'Credit Card', TO_TIMESTAMP('2025-04-03 19:30:00',
'YYYY-MM-DD HH24:MI:SS'), 40.00, 4.00, 8, 100008)
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (9, 'Completed', 'Online', TO_TIMESTAMP('2025-04-03 20:00:00',
'YYYY-MM-DD HH24:MI:SS'), 65.00, 6.00, 7, 100009)
    INTO Payment (PaymentID, Status, Method, "Date", Amount, Discount, EmpID,
CustID) VALUES (10, 'Refunded', 'Debit Card', TO_TIMESTAMP('2025-04-03 20:30:00',
'YYYY-MM-DD HH24:MI:SS'), 30.00, 3.00, 8, 100010)
SELECT * FROM dual;
```

18. MEMBERSHIP

```
INSERT ALL
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (1,
'Gold', TO_DATE('2025-01-01', 'YYYY-MM-DD'), TO_DATE('2026-01-01', 'YYYY-MM-DD'),
100001)
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (2,
'Platinum', TO_DATE('2025-02-01', 'YYYY-MM-DD'), TO_DATE('2026-02-01', 'YYYY-MM-
DD'), 100002)
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (3,
'Silver', TO_DATE('2025-03-01', 'YYYY-MM-DD'), TO_DATE('2026-03-01', 'YYYY-MM-
DD'), 100003)
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (4,
'VIP', TO_DATE('2025-04-01', 'YYYY-MM-DD'), TO_DATE('2026-04-01', 'YYYY-MM-DD'),
100004)
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (5,
'Gold', TO_DATE('2025-05-01', 'YYYY-MM-DD'), TO_DATE('2026-05-01', 'YYYY-MM-DD'),
100006)
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (6,
'Silver', TO_DATE('2025-06-01', 'YYYY-MM-DD'), TO_DATE('2026-06-01', 'YYYY-MM-
DD'), 100008)
```

```
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (7,
'Platinum', TO_DATE('2025-07-01', 'YYYY-MM-DD'), TO_DATE('2026-07-01', 'YYYY-MM-
DD'), 100011)
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (8,
'VIP', TO_DATE('2025-08-01', 'YYYY-MM-DD'), TO_DATE('2026-08-01', 'YYYY-MM-DD'),
100013)
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (9,
'Gold', TO_DATE('2025-09-01', 'YYYY-MM-DD'), TO_DATE('2026-09-01', 'YYYY-MM-DD'),
100014)
    INTO Membership (MembershipID, Type, StartDate, ExpDate, CustID) VALUES (10,
'Silver', TO_DATE('2025-10-01', 'YYYY-MM-DD'), TO_DATE('2026-10-01', 'YYYY-MM-
DD'), 100015)
SELECT * FROM dual;
```

19. EQUIPMENT

```
INSERT ALL
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (1, 'Projector', 'Sony', 'VPL-FHZ75', TO_TIMESTAMP('2025-03-01
12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1, 2)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (2, 'Speaker', 'JBL', 'EON615', TO_TIMESTAMP('2025-03-10 14:30:00',
'YYYY-MM-DD HH24:MI:SS'), 2, 4)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (3, 'Screen', 'Panasonic', 'TH-65EQ1W', TO_TIMESTAMP('2025-03-05
11:00:00', 'YYYY-MM-DD HH24:MI:SS'), 5, 3)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (4, 'Lighting', 'Yamaha', 'CL5', TO_TIMESTAMP('2025-03-08 16:00:00',
'YYYY-MM-DD HH24:MI:SS'), 4, 4)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (5, 'Sound System', 'Bose', 'L1 Compact', TO_TIMESTAMP('2025-03-03
09:30:00', 'YYYY-MM-DD HH24:MI:SS'), 5, 5)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (6, 'Monitor', 'Samsung', 'U32J590', TO_TIMESTAMP('2025-03-06
13:00:00', 'YYYY-MM-DD HH24:MI:SS'), 6, 9)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (7, 'Projector Screen', 'IMAX', 'L3X4', TO_TIMESTAMP('2025-03-04
10:30:00', 'YYYY-MM-DD HH24:MI:SS'), 8, 7)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (8, 'Microphone', 'Philips', 'SHM1900', TO_TIMESTAMP('2025-03-07
17:45:00', 'YYYY-MM-DD HH24:MI:SS'), 8, 12)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (9, 'Lighting', 'Sony', 'LMD-A240', TO_TIMESTAMP('2025-03-09
18:20:00', 'YYYY-MM-DD HH24:MI:SS'), 9, 9)
```

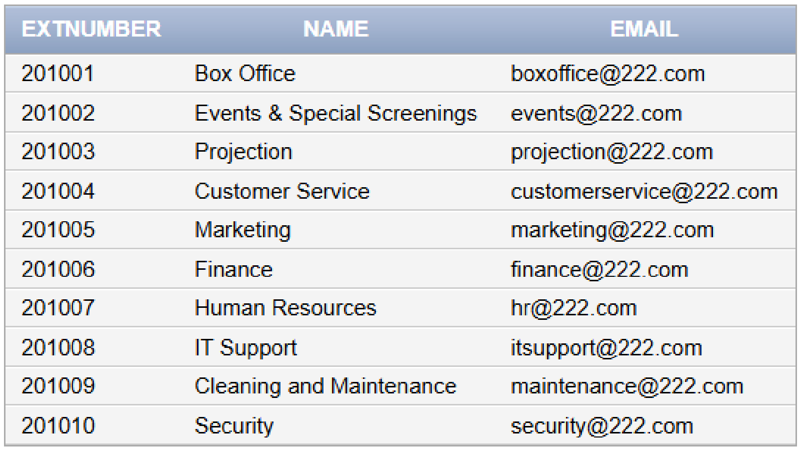
```
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (10, 'Speaker', 'JBL', 'Professional EON', TO_TIMESTAMP('2025-03-02
12:45:00', 'YYYY-MM-DD HH24:MI:SS'), 15, 18)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (11, 'Sound System', 'Yamaha', 'Stagepas 600BT', TO_TIMESTAMP('2025-
03-11 15:00:00', 'YYYY-MM-DD HH24:MI:SS'), 13, 17)
    INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
    VALUES (12, 'Monitor', 'LG', '27GL850-B', TO_TIMESTAMP('2025-03-12 08:00:00',
'YYYY-MM-DD HH24:MI:SS'), 12, 14)
SELECT * FROM DUAL;
```

20. PROVIDES

```
INSERT ALL
    INTO Provides (SupID, MovID, EquipID)
    VALUES (1, 1, 2)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (2, 3, 4)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (3, 5, 6)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (4, 7, 8)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (5, 9, 10)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (6, 10, 12)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (7, 11, 1)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (8, 12, 2)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (9, 13, 4)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (10, 14, 6)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (1, 15, 8)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (2, 6, 10)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (3, 4, 12)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (4, 8, 2)
    INTO Provides (SupID, MovID, EquipID)
    VALUES (5, 2, 4)
SELECT * FROM DUAL;
```

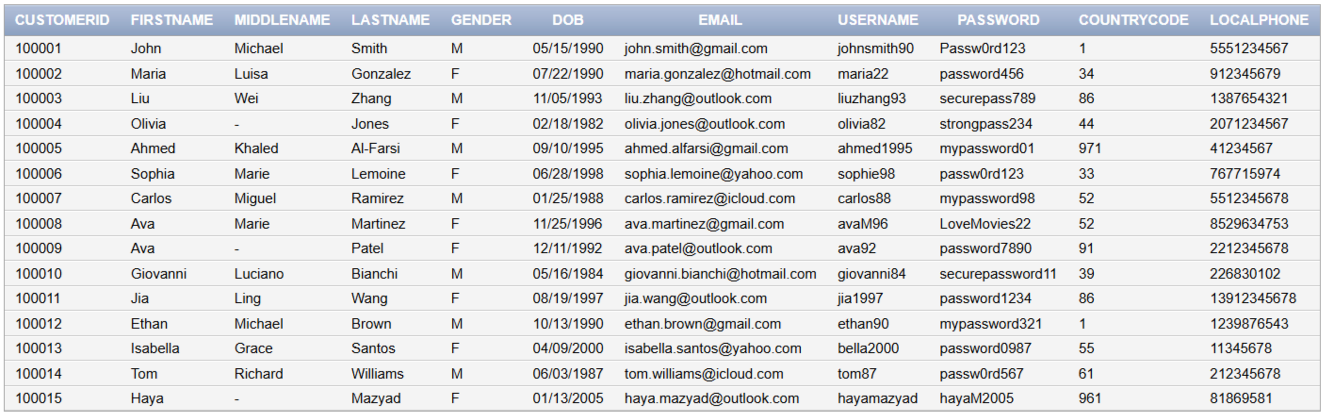
Final Table State

1. DEPARTMENT



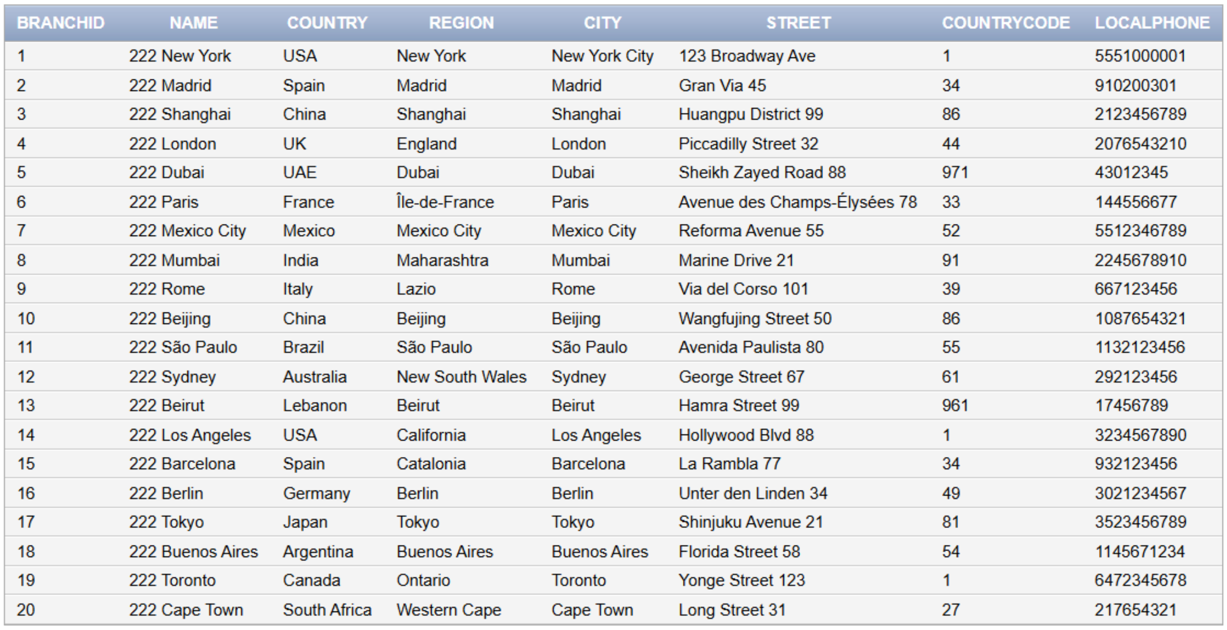
EXTNUMBER	NAME	EMAIL
201001	Box Office	boxoffice@222.com
201002	Events & Special Screenings	events@222.com
201003	Projection	projection@222.com
201004	Customer Service	customerservice@222.com
201005	Marketing	marketing@222.com
201006	Finance	finance@222.com
201007	Human Resources	hr@222.com
201008	IT Support	itsupport@222.com
201009	Cleaning and Maintenance	maintenance@222.com
201010	Security	security@222.com

2. CUSTOMER



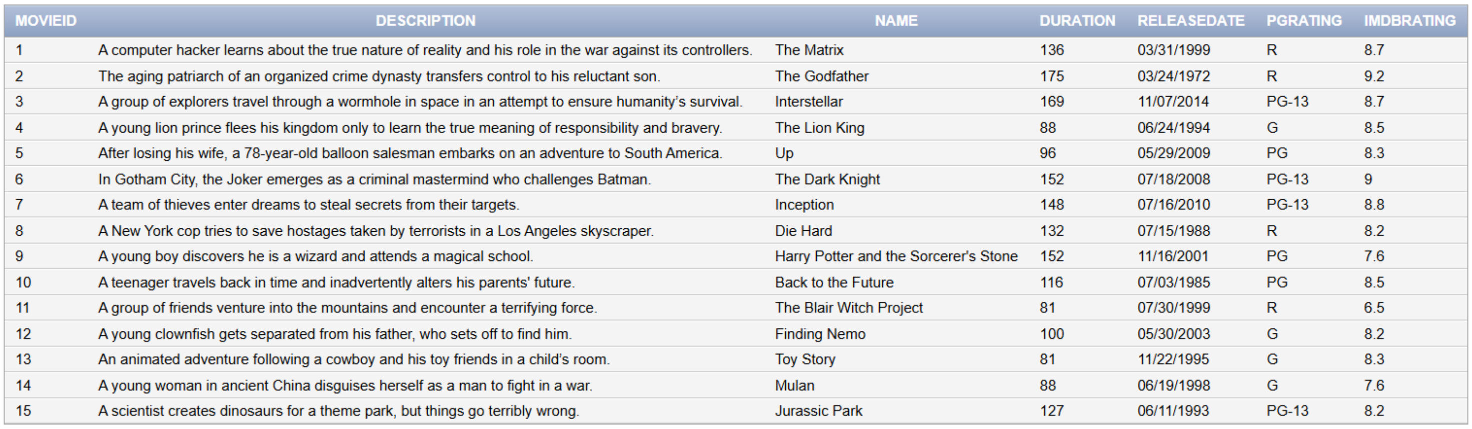
CUSTOMERID	FIRSTNAME	MIDDLENAME	LASTNAME	GENDER	DOB	EMAIL	USERNAME	PASSWORD	COUNTRYCODE	LOCALPHONE
100001	John	Michael	Smith	M	05/15/1990	john.smith@gmail.com	johnsmith90	Passw0rd123	1	5551234567
100002	Maria	Luisa	Gonzalez	F	07/22/1990	maria.gonzalez@hotmail.com	maria22	password456	34	912345679
100003	Liu	Wei	Zhang	M	11/05/1993	liu.zhang@outlook.com	liuzhang93	securepass789	86	1387654321
100004	Olivia	-	Jones	F	02/18/1982	olivia.jones@outlook.com	olivia82	strongpass234	44	2071234567
100005	Ahmed	Khaled	Al-Farsi	M	09/10/1995	ahmed.alfarsi@gmail.com	ahmed1995	mypassword01	971	41234567
100006	Sophia	Marie	Lemoine	F	06/28/1998	sophia.lemoine@yahoo.com	sophie98	passw0rd123	33	767715974
100007	Carlos	Miguel	Ramirez	M	01/25/1988	carlos.ramirez@icloud.com	carlos88	mypassword98	52	5512345678
100008	Ava	Marie	Martinez	F	11/25/1996	ava.martinez@gmail.com	avaM96	LoveMovies22	52	8529634753
100009	Ava	-	Patel	F	12/11/1992	ava.patel@outlook.com	ava92	password7890	91	2212345678
100010	Giovanni	Luciano	Bianchi	M	05/16/1984	giovanni.bianchi@hotmail.com	giovanni84	securepassword11	39	226830102
100011	Jia	Ling	Wang	F	08/19/1997	jia.wang@outlook.com	jia1997	password1234	86	13912345678
100012	Ethan	Michael	Brown	M	10/13/1990	ethan.brown@gmail.com	ethan90	mypassword321	1	1239876543
100013	Isabella	Grace	Santos	F	04/09/2000	isabella.santos@yahoo.com	bella2000	password0987	55	11345678
100014	Tom	Richard	Williams	M	06/03/1987	tom.williams@icloud.com	tom87	passw0rd567	61	212345678
100015	Haya	-	Mazyad	F	01/13/2005	haya.mazyad@outlook.com	hayamazyad	hayaM2005	961	81869581

3. BRANCH



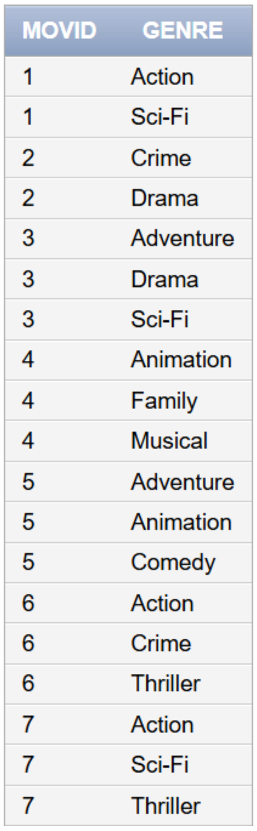
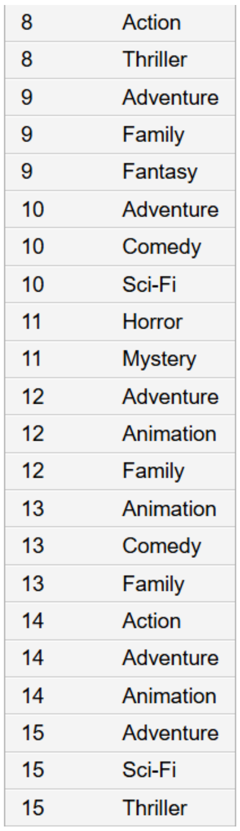
BRANCHID	NAME	COUNTRY	REGION	CITY	STREET	COUNTRYCODE	LOCALPHONE
1	222 New York	USA	New York	New York City	123 Broadway Ave	1	5551000001
2	222 Madrid	Spain	Madrid	Madrid	Gran Via 45	34	910200301
3	222 Shanghai	China	Shanghai	Shanghai	Huangpu District 99	86	2123456789
4	222 London	UK	England	London	Piccadilly Street 32	44	2076543210
5	222 Dubai	UAE	Dubai	Dubai	Sheikh Zayed Road 88	971	43012345
6	222 Paris	France	Île-de-France	Paris	Avenue des Champs-Élysées 78	33	144556677
7	222 Mexico City	Mexico	Mexico City	Mexico City	Reforma Avenue 55	52	5512346789
8	222 Mumbai	India	Maharashtra	Mumbai	Marine Drive 21	91	2245678910
9	222 Rome	Italy	Lazio	Rome	Via del Corso 101	39	667123456
10	222 Beijing	China	Beijing	Beijing	Wangfujing Street 50	86	1087654321
11	222 São Paulo	Brazil	São Paulo	São Paulo	Avenida Paulista 80	55	1132123456
12	222 Sydney	Australia	New South Wales	Sydney	George Street 67	61	292123456
13	222 Beirut	Lebanon	Beirut	Beirut	Hamra Street 99	961	17456789
14	222 Los Angeles	USA	California	Los Angeles	Hollywood Blvd 88	1	3234567890
15	222 Barcelona	Spain	Catalonia	Barcelona	La Rambla 77	34	932123456
16	222 Berlin	Germany	Berlin	Berlin	Unter den Linden 34	49	3021234567
17	222 Tokyo	Japan	Tokyo	Tokyo	Shinjuku Avenue 21	81	3523456789
18	222 Buenos Aires	Argentina	Buenos Aires	Buenos Aires	Florida Street 58	54	1145671234
19	222 Toronto	Canada	Ontario	Toronto	Yonge Street 123	1	6472345678
20	222 Cape Town	South Africa	Western Cape	Cape Town	Long Street 31	27	217654321

4. MOVIE



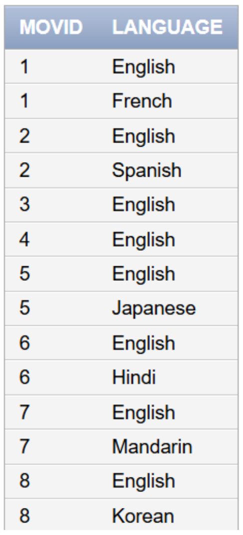
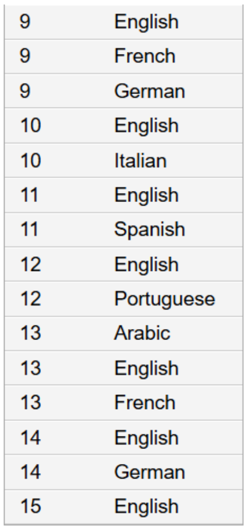
MOVIEID	DESCRIPTION	NAME	DURATION	RELEASEDATE	PGRATING	IMDBRATING
1	A computer hacker learns about the true nature of reality and his role in the war against its controllers.	The Matrix	136	03/31/1999	R	8.7
2	The aging patriarch of an organized crime dynasty transfers control to his reluctant son.	The Godfather	175	03/24/1972	R	9.2
3	A group of explorers travel through a wormhole in space in an attempt to ensure humanity's survival.	Interstellar	169	11/07/2014	PG-13	8.7
4	A young lion prince flees his kingdom only to learn the true meaning of responsibility and bravery.	The Lion King	88	06/24/1994	G	8.5
5	After losing his wife, a 78-year-old balloon salesman embarks on an adventure to South America.	Up	96	05/29/2009	PG	8.3
6	In Gotham City, the Joker emerges as a criminal mastermind who challenges Batman.	The Dark Knight	152	07/18/2008	PG-13	9
7	A team of thieves enter dreams to steal secrets from their targets.	Inception	148	07/16/2010	PG-13	8.8
8	A New York cop tries to save hostages taken by terrorists in a Los Angeles skyscraper.	Die Hard	132	07/15/1988	R	8.2
9	A young boy discovers he is a wizard and attends a magical school.	Harry Potter and the Sorcerer's Stone	152	11/16/2001	PG	7.6
10	A teenager travels back in time and inadvertently alters his parents' future.	Back to the Future	116	07/03/1985	PG	8.5
11	A group of friends venture into the mountains and encounter a terrifying force.	The Blair Witch Project	81	07/30/1999	R	6.5
12	A young clownfish gets separated from his father, who sets off to find him.	Finding Nemo	100	05/30/2003	G	8.2
13	An animated adventure following a cowboy and his toy friends in a child's room.	Toy Story	81	11/22/1995	G	8.3
14	A young woman in ancient China disguises herself as a man to fight in a war.	Mulan	88	06/19/1998	G	7.6
15	A scientist creates dinosaurs for a theme park, but things go terribly wrong.	Jurassic Park	127	06/11/1993	PG-13	8.2

5. MOVIE_GENRE

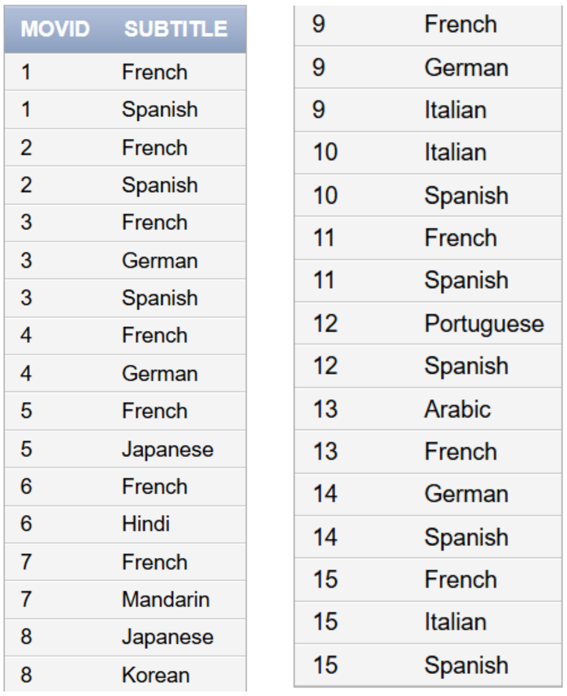
MOVID	GENRE
1	Action
1	Sci-Fi
2	Crime
2	Drama
3	Adventure
3	Drama
3	Sci-Fi
4	Animation
4	Family
4	Musical
5	Adventure
5	Animation
5	Comedy
6	Action
6	Crime
6	Thriller
7	Action
7	Sci-Fi
7	Thriller
8	Action
8	Thriller
9	Adventure
9	Family
9	Fantasy
10	Adventure
10	Comedy
10	Sci-Fi
11	Horror
11	Mystery
12	Adventure
12	Animation
12	Family
13	Animation
13	Comedy
13	Family
14	Action
14	Adventure
14	Animation
15	Adventure
15	Sci-Fi
15	Thriller

6. MOVIE_LANGUAGE

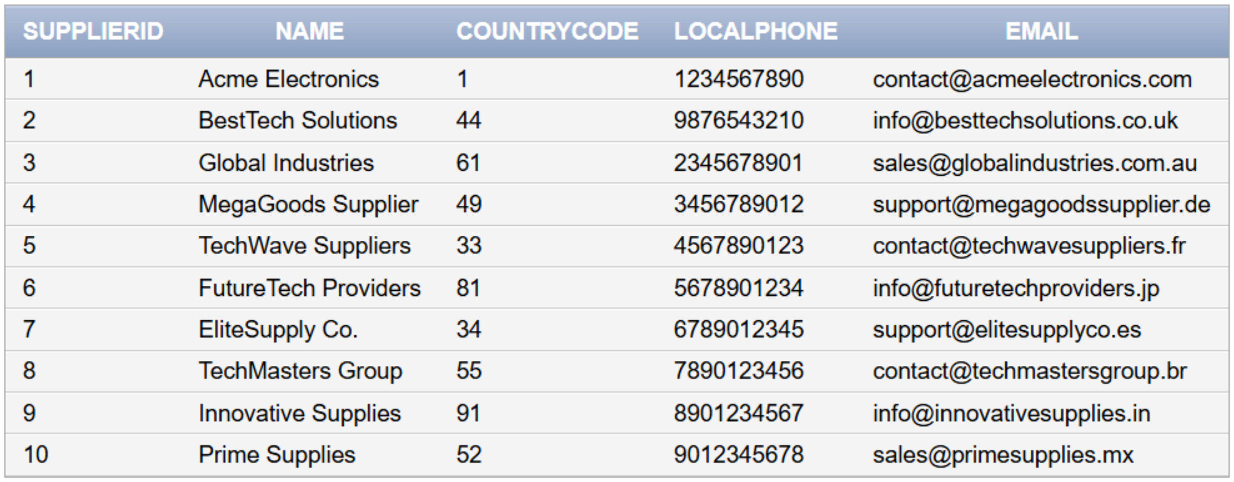
MOVID	LANGUAGE
1	English
1	French
2	English
2	Spanish
3	English
4	English
5	English
5	Japanese
6	English
6	Hindi
7	English
7	Mandarin
8	English
8	Korean
9	English
9	French
9	German
10	English
10	Italian
11	English
11	Spanish
12	English
12	Portuguese
13	Arabic
13	English
13	French
14	English
14	German
15	English

7. MOVIE_SUBTITLE



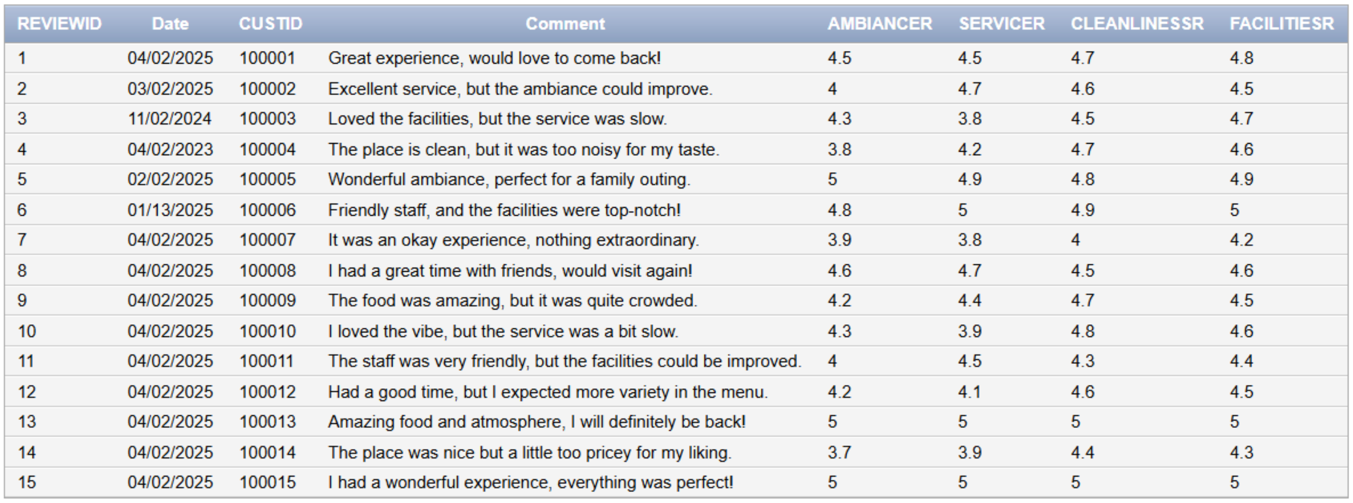
MOVID	SUBTITLE
1	French
1	Spanish
2	French
2	Spanish
3	French
3	German
3	Spanish
4	French
4	German
5	French
5	Japanese
6	French
6	Hindi
7	French
7	Mandarin
8	Japanese
8	Korean
9	French
9	German
9	Italian
10	Italian
10	Spanish
11	French
11	Spanish
12	Portuguese
12	Spanish
13	Arabic
13	French
14	German
14	Spanish
15	French
15	Italian
15	Spanish

8. SUPPLIER



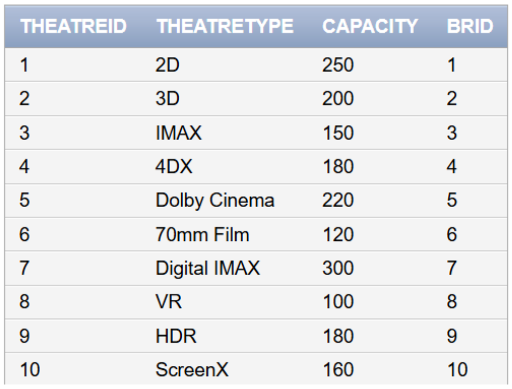
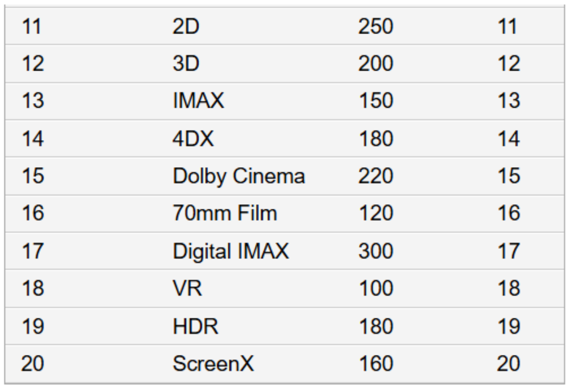
SUPPLIERID	NAME	COUNTRYCODE	LOCALPHONE	EMAIL
1	Acme Electronics	1	1234567890	contact@acmeelectronics.com
2	BestTech Solutions	44	9876543210	info@besttechsolutions.co.uk
3	Global Industries	61	2345678901	sales@globalindustries.com.au
4	MegaGoods Supplier	49	3456789012	support@megagoodssupplier.de
5	TechWave Suppliers	33	4567890123	contact@techwavesuppliers.fr
6	FutureTech Providers	81	5678901234	info@futuretechproviders.jp
7	EliteSupply Co.	34	6789012345	support@elitesupplyco.es
8	TechMasters Group	55	7890123456	contact@techmastersgroup.br
9	Innovative Supplies	91	8901234567	info@innovativesupplies.in
10	Prime Supplies	52	9012345678	sales@primesupplies.mx

9. REVIEW



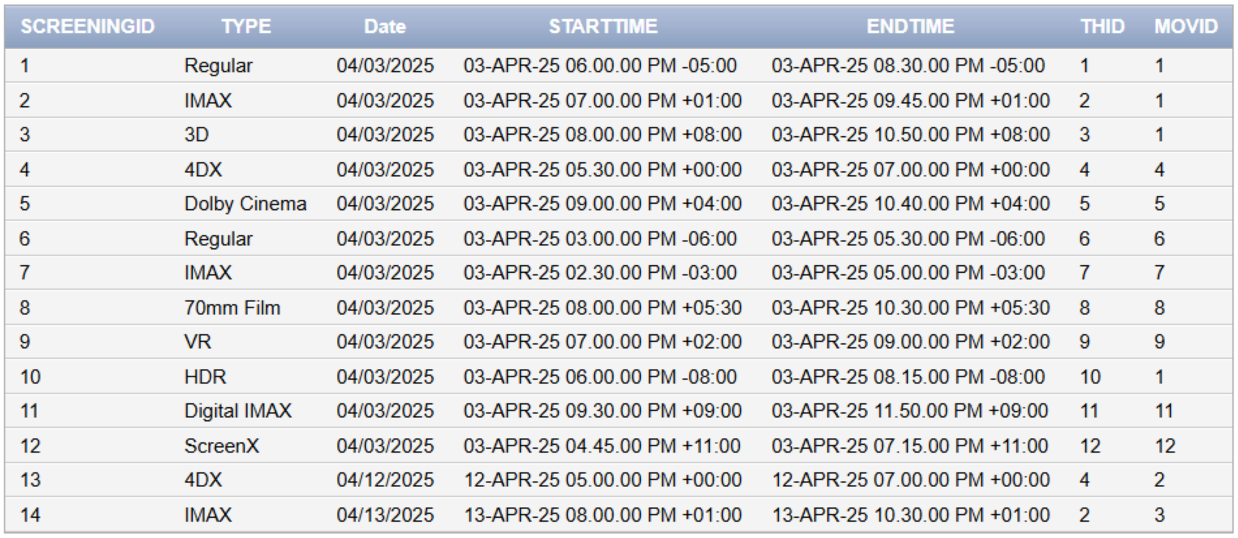
REVIEWID	Date	CUSTID	Comment	AMBIANCER	SERVICER	CLEANLINESSR	FACILITIESR
1	04/02/2025	100001	Great experience, would love to come back!	4.5	4.5	4.7	4.8
2	03/02/2025	100002	Excellent service, but the ambiance could improve.	4	4.7	4.6	4.5
3	11/02/2024	100003	Loved the facilities, but the service was slow.	4.3	3.8	4.5	4.7
4	04/02/2023	100004	The place is clean, but it was too noisy for my taste.	3.8	4.2	4.7	4.6
5	02/02/2025	100005	Wonderful ambiance, perfect for a family outing.	5	4.9	4.8	4.9
6	01/13/2025	100006	Friendly staff, and the facilities were top-notch!	4.8	5	4.9	5
7	04/02/2025	100007	It was an okay experience, nothing extraordinary.	3.9	3.8	4	4.2
8	04/02/2025	100008	I had a great time with friends, would visit again!	4.6	4.7	4.5	4.6
9	04/02/2025	100009	The food was amazing, but it was quite crowded.	4.2	4.4	4.7	4.5
10	04/02/2025	100010	I loved the vibe, but the service was a bit slow.	4.3	3.9	4.8	4.6
11	04/02/2025	100011	The staff was very friendly, but the facilities could be improved.	4	4.5	4.3	4.4
12	04/02/2025	100012	Had a good time, but I expected more variety in the menu.	4.2	4.1	4.6	4.5
13	04/02/2025	100013	Amazing food and atmosphere, I will definitely be back!	5	5	5	5
14	04/02/2025	100014	The place was nice but a little too pricey for my liking.	3.7	3.9	4.4	4.3
15	04/02/2025	100015	I had a wonderful experience, everything was perfect!	5	5	5	5

10. THEATRE

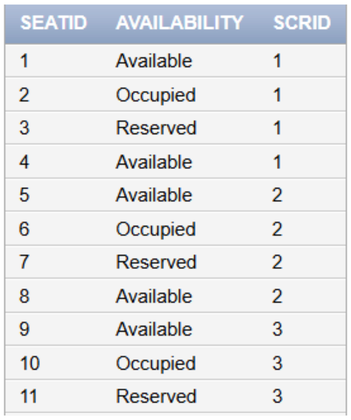
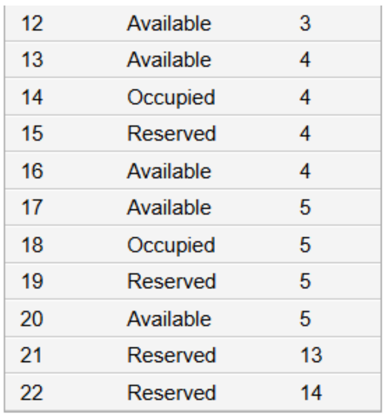
THEATREID	THEATRETYPE	CAPACITY	BRID
1	2D	250	1
2	3D	200	2
3	IMAX	150	3
4	4DX	180	4
5	Dolby Cinema	220	5
6	70mm Film	120	6
7	Digital IMAX	300	7
8	VR	100	8
9	HDR	180	9
10	ScreenX	160	10
11	2D	250	11
12	3D	200	12
13	IMAX	150	13
14	4DX	180	14
15	Dolby Cinema	220	15
16	70mm Film	120	16
17	Digital IMAX	300	17
18	VR	100	18
19	HDR	180	19
20	ScreenX	160	20

11. SCREENING



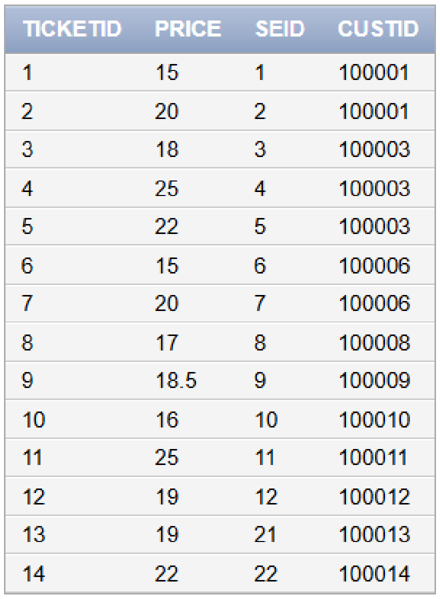
SCREENINGID	TYPE	Date	STARTTIME	ENDTIME	THID	MOVID
1	Regular	04/03/2025	03-APR-25 06.00.00 PM -05:00	03-APR-25 08.30.00 PM -05:00	1	1
2	IMAX	04/03/2025	03-APR-25 07.00.00 PM +01:00	03-APR-25 09.45.00 PM +01:00	2	1
3	3D	04/03/2025	03-APR-25 08.00.00 PM +08:00	03-APR-25 10.50.00 PM +08:00	3	1
4	4DX	04/03/2025	03-APR-25 05.30.00 PM +00:00	03-APR-25 07.00.00 PM +00:00	4	4
5	Dolby Cinema	04/03/2025	03-APR-25 09.00.00 PM +04:00	03-APR-25 10.40.00 PM +04:00	5	5
6	Regular	04/03/2025	03-APR-25 03.00.00 PM -06:00	03-APR-25 05.30.00 PM -06:00	6	6
7	IMAX	04/03/2025	03-APR-25 02.30.00 PM -03:00	03-APR-25 05.00.00 PM -03:00	7	7
8	70mm Film	04/03/2025	03-APR-25 08.00.00 PM +05:30	03-APR-25 10.30.00 PM +05:30	8	8
9	VR	04/03/2025	03-APR-25 07.00.00 PM +02:00	03-APR-25 09.00.00 PM +02:00	9	9
10	HDR	04/03/2025	03-APR-25 06.00.00 PM -08:00	03-APR-25 08.15.00 PM -08:00	10	1
11	Digital IMAX	04/03/2025	03-APR-25 09.30.00 PM +09:00	03-APR-25 11.50.00 PM +09:00	11	11
12	ScreenX	04/03/2025	03-APR-25 04.45.00 PM +11:00	03-APR-25 07.15.00 PM +11:00	12	12
13	4DX	04/12/2025	12-APR-25 05.00.00 PM +00:00	12-APR-25 07.00.00 PM +00:00	4	2
14	IMAX	04/13/2025	13-APR-25 08.00.00 PM +01:00	13-APR-25 10.30.00 PM +01:00	2	3

12. SEAT

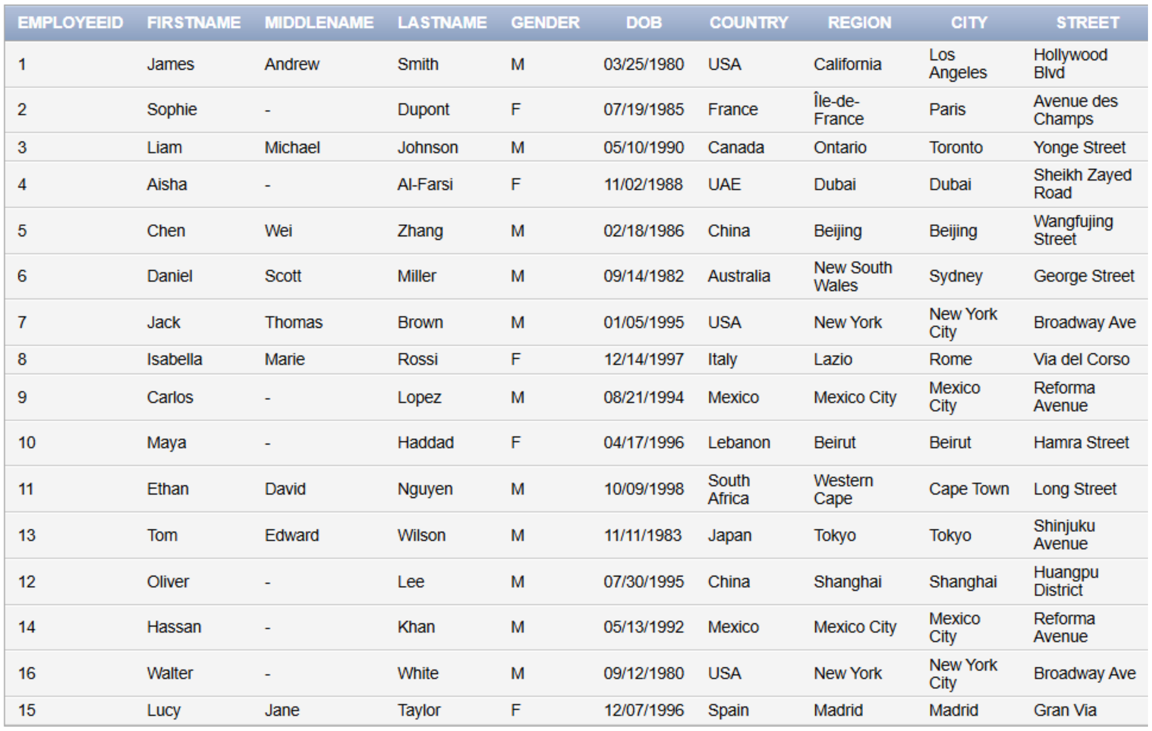
SEATID	AVAILABILITY	SCRID
1	Available	1
2	Occupied	1
3	Reserved	1
4	Available	1
5	Available	2
6	Occupied	2
7	Reserved	2
8	Available	2
9	Available	3
10	Occupied	3
11	Reserved	3
12	Available	3
13	Available	4
14	Occupied	4
15	Reserved	4
16	Available	4
17	Available	5
18	Occupied	5
19	Reserved	5
20	Available	5
21	Reserved	13
22	Reserved	14

13. TICKET

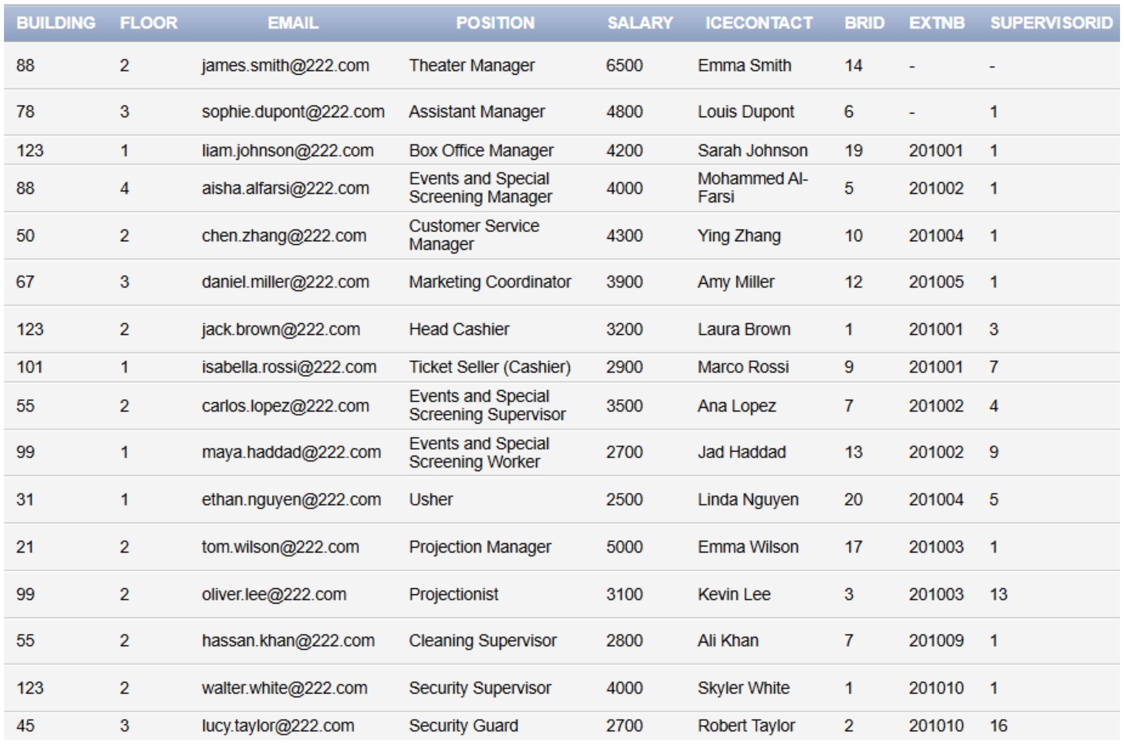


TICKETID	PRICE	SEID	CUSTID
1	15	1	100001
2	20	2	100001
3	18	3	100003
4	25	4	100003
5	22	5	100003
6	15	6	100006
7	20	7	100006
8	17	8	100008
9	18.5	9	100009
10	16	10	100010
11	25	11	100011
12	19	12	100012
13	19	21	100013
14	22	22	100014

14. EMPLOYEE

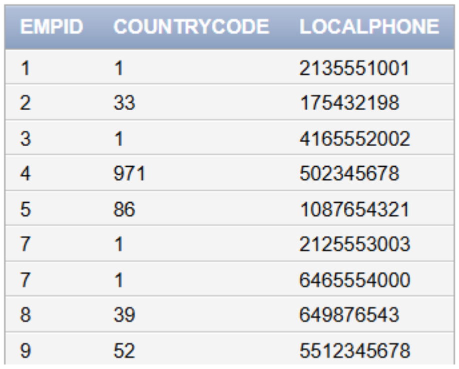
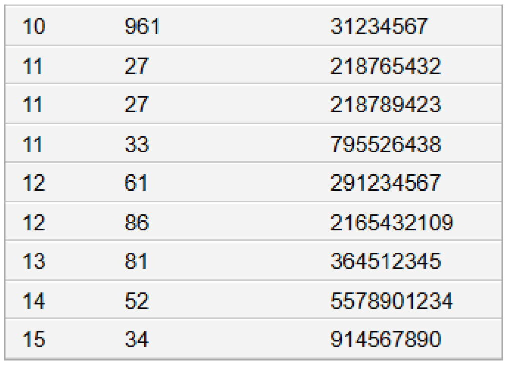


EMPLOYEEID	FIRSTNAME	MIDDLENAME	LASTNAME	GENDER	DOB	COUNTRY	REGION	CITY	STREET
1	James	Andrew	Smith	M	03/25/1980	USA	California	Los Angeles	Hollywood Blvd
2	Sophie	-	Dupont	F	07/19/1985	France	Île-de-France	Paris	Avenue des Champs
3	Liam	Michael	Johnson	M	05/10/1990	Canada	Ontario	Toronto	Yonge Street
4	Aisha	-	Al-Farsi	F	11/02/1988	UAE	Dubai	Dubai	Sheikh Zayed Road
5	Chen	Wei	Zhang	M	02/18/1986	China	Beijing	Beijing	Wangfujing Street
6	Daniel	Scott	Miller	M	09/14/1982	Australia	New South Wales	Sydney	George Street
7	Jack	Thomas	Brown	M	01/05/1995	USA	New York	New York City	Broadway Ave
8	Isabella	Marie	Rossi	F	12/14/1997	Italy	Lazio	Rome	Via del Corso
9	Carlos	-	Lopez	M	08/21/1994	Mexico	Mexico City	Mexico City	Reforma Avenue
10	Maya	-	Haddad	F	04/17/1996	Lebanon	Beirut	Beirut	Hamra Street
11	Ethan	David	Nguyen	M	10/09/1998	South Africa	Western Cape	Cape Town	Long Street
13	Tom	Edward	Wilson	M	11/11/1983	Japan	Tokyo	Tokyo	Shinjuku Avenue
12	Oliver	-	Lee	M	07/30/1995	China	Shanghai	Shanghai	Huangpu District
14	Hassan	-	Khan	M	05/13/1992	Mexico	Mexico City	Mexico City	Reforma Avenue
16	Walter	-	White	M	09/12/1980	USA	New York	New York City	Broadway Ave
15	Lucy	Jane	Taylor	F	12/07/1996	Spain	Madrid	Madrid	Gran Via



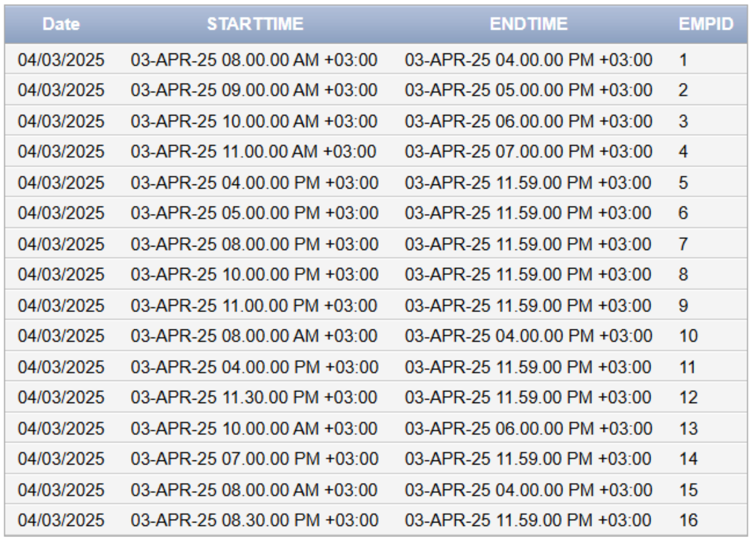
BUILDING	FLOOR	EMAIL	POSITION	SALARY	ICECONTACT	BRID	EXTNB	SUPERVISORID
88	2	james.smith@222.com	Theater Manager	6500	Emma Smith	14	-	-
78	3	sophie.dupont@222.com	Assistant Manager	4800	Louis Dupont	6	-	1
123	1	liam.johnson@222.com	Box Office Manager	4200	Sarah Johnson	19	201001	1
88	4	aisha.alfarsi@222.com	Events and Special Screening Manager	4000	Mohammed Al-Farsi	5	201002	1
50	2	chen.zhang@222.com	Customer Service Manager	4300	Ying Zhang	10	201004	1
67	3	daniel.miller@222.com	Marketing Coordinator	3900	Amy Miller	12	201005	1
123	2	jack.brown@222.com	Head Cashier	3200	Laura Brown	1	201001	3
101	1	isabella.rossi@222.com	Ticket Seller (Cashier)	2900	Marco Rossi	9	201001	7
55	2	carlos.lopez@222.com	Events and Special Screening Supervisor	3500	Ana Lopez	7	201002	4
99	1	maya.haddad@222.com	Events and Special Screening Worker	2700	Jad Haddad	13	201002	9
31	1	ethan.nguyen@222.com	Usher	2500	Linda Nguyen	20	201004	5
21	2	tom.wilson@222.com	Projection Manager	5000	Emma Wilson	17	201003	1
99	2	oliver.lee@222.com	Projectionist	3100	Kevin Lee	3	201003	13
55	2	hassan.khan@222.com	Cleaning Supervisor	2800	Ali Khan	7	201009	1
123	2	walter.white@222.com	Security Supervisor	4000	Skyler White	1	201010	1
45	3	lucy.taylor@222.com	Security Guard	2700	Robert Taylor	2	201010	16

15. EMPLOYEE_PHONE

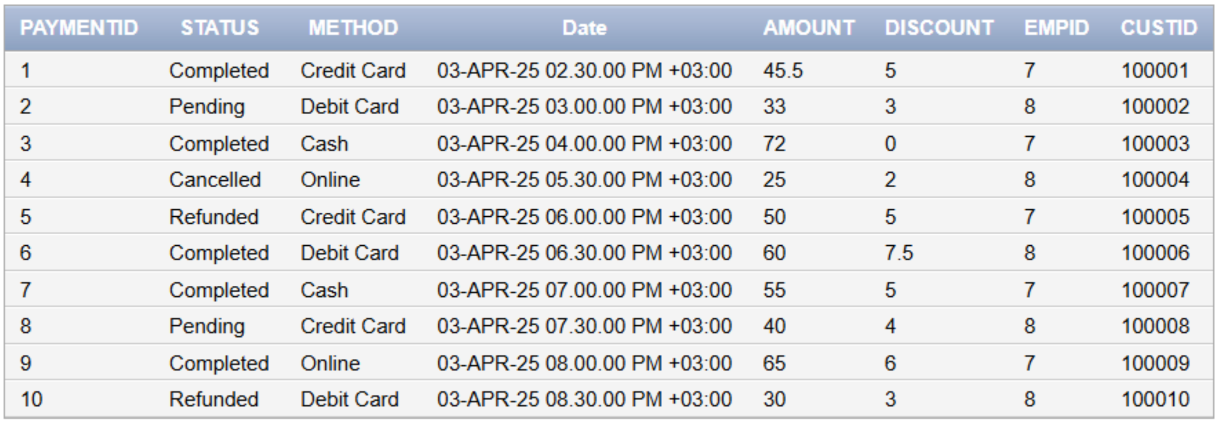
EMPID	COUNTRYCODE	LOCALPHONE
1	1	2135551001
2	33	175432198
3	1	4165552002
4	971	502345678
5	86	1087654321
7	1	2125553003
7	1	6465554000
8	39	649876543
9	52	5512345678
10	961	31234567
11	27	218765432
11	27	218789423
11	33	795526438
12	61	291234567
12	86	2165432109
13	81	364512345
14	52	5578901234
15	34	914567890

16. SCHEDULE



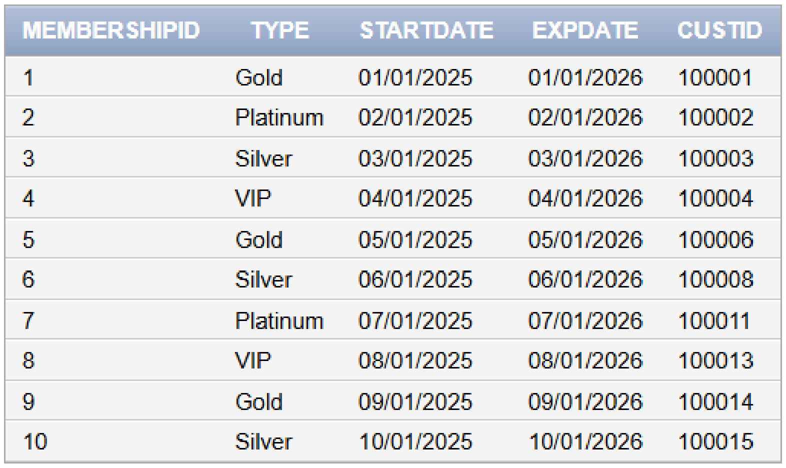
Date	STARTTIME	ENDTIME	EMPID
04/03/2025	03-APR-25 08.00.00 AM +03:00	03-APR-25 04.00.00 PM +03:00	1
04/03/2025	03-APR-25 09.00.00 AM +03:00	03-APR-25 05.00.00 PM +03:00	2
04/03/2025	03-APR-25 10.00.00 AM +03:00	03-APR-25 06.00.00 PM +03:00	3
04/03/2025	03-APR-25 11.00.00 AM +03:00	03-APR-25 07.00.00 PM +03:00	4
04/03/2025	03-APR-25 04.00.00 PM +03:00	03-APR-25 11.59.00 PM +03:00	5
04/03/2025	03-APR-25 05.00.00 PM +03:00	03-APR-25 11.59.00 PM +03:00	6
04/03/2025	03-APR-25 08.00.00 PM +03:00	03-APR-25 11.59.00 PM +03:00	7
04/03/2025	03-APR-25 10.00.00 PM +03:00	03-APR-25 11.59.00 PM +03:00	8
04/03/2025	03-APR-25 11.00.00 PM +03:00	03-APR-25 11.59.00 PM +03:00	9
04/03/2025	03-APR-25 08.00.00 AM +03:00	03-APR-25 04.00.00 PM +03:00	10
04/03/2025	03-APR-25 04.00.00 PM +03:00	03-APR-25 11.59.00 PM +03:00	11
04/03/2025	03-APR-25 11.30.00 PM +03:00	03-APR-25 11.59.00 PM +03:00	12
04/03/2025	03-APR-25 10.00.00 AM +03:00	03-APR-25 06.00.00 PM +03:00	13
04/03/2025	03-APR-25 07.00.00 PM +03:00	03-APR-25 11.59.00 PM +03:00	14
04/03/2025	03-APR-25 08.00.00 AM +03:00	03-APR-25 04.00.00 PM +03:00	15
04/03/2025	03-APR-25 08.30.00 PM +03:00	03-APR-25 11.59.00 PM +03:00	16

17. PAYMENT



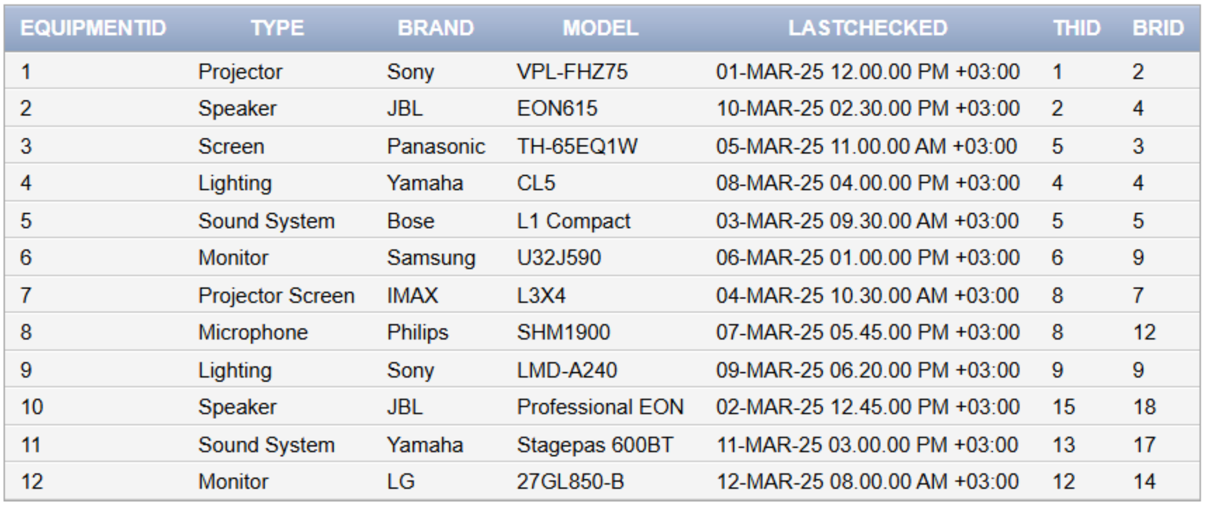
PAYMENTID	STATUS	METHOD	Date	AMOUNT	DISCOUNT	EMPID	CUSTID
1	Completed	Credit Card	03-APR-25 02.30.00 PM +03:00	45.5	5	7	100001
2	Pending	Debit Card	03-APR-25 03.00.00 PM +03:00	33	3	8	100002
3	Completed	Cash	03-APR-25 04.00.00 PM +03:00	72	0	7	100003
4	Cancelled	Online	03-APR-25 05.30.00 PM +03:00	25	2	8	100004
5	Refunded	Credit Card	03-APR-25 06.00.00 PM +03:00	50	5	7	100005
6	Completed	Debit Card	03-APR-25 06.30.00 PM +03:00	60	7.5	8	100006
7	Completed	Cash	03-APR-25 07.00.00 PM +03:00	55	5	7	100007
8	Pending	Credit Card	03-APR-25 07.30.00 PM +03:00	40	4	8	100008
9	Completed	Online	03-APR-25 08.00.00 PM +03:00	65	6	7	100009
10	Refunded	Debit Card	03-APR-25 08.30.00 PM +03:00	30	3	8	100010

18. MEMBERSHIP



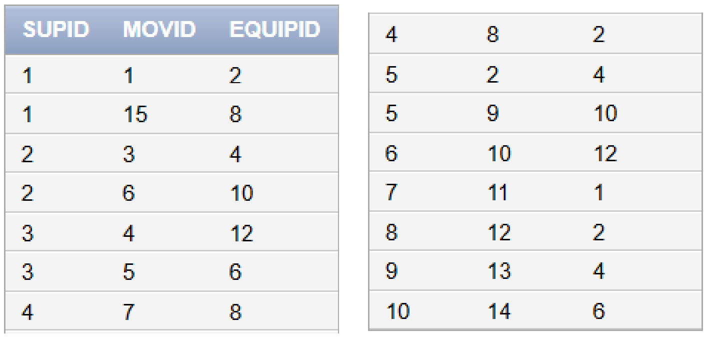
MEMBERSHIPID	TYPE	STARTDATE	EXPDATE	CUSTID
1	Gold	01/01/2025	01/01/2026	100001
2	Platinum	02/01/2025	02/01/2026	100002
3	Silver	03/01/2025	03/01/2026	100003
4	VIP	04/01/2025	04/01/2026	100004
5	Gold	05/01/2025	05/01/2026	100006
6	Silver	06/01/2025	06/01/2026	100008
7	Platinum	07/01/2025	07/01/2026	100011
8	VIP	08/01/2025	08/01/2026	100013
9	Gold	09/01/2025	09/01/2026	100014
10	Silver	10/01/2025	10/01/2026	100015

19. EQUIPMENT



EQUIPMENTID	TYPE	BRAND	MODEL	LASTCHECKED	THID	BRID
1	Projector	Sony	VPL-FHZ75	01-MAR-25 12.00.00 PM +03:00	1	2
2	Speaker	JBL	EON615	10-MAR-25 02.30.00 PM +03:00	2	4
3	Screen	Panasonic	TH-65EQ1W	05-MAR-25 11.00.00 AM +03:00	5	3
4	Lighting	Yamaha	CL5	08-MAR-25 04.00.00 PM +03:00	4	4
5	Sound System	Bose	L1 Compact	03-MAR-25 09.30.00 AM +03:00	5	5
6	Monitor	Samsung	U32J590	06-MAR-25 01.00.00 PM +03:00	6	9
7	Projector Screen	IMAX	L3X4	04-MAR-25 10.30.00 AM +03:00	8	7
8	Microphone	Philips	SHM1900	07-MAR-25 05.45.00 PM +03:00	8	12
9	Lighting	Sony	LMD-A240	09-MAR-25 06.20.00 PM +03:00	9	9
10	Speaker	JBL	Professional EON	02-MAR-25 12.45.00 PM +03:00	15	18
11	Sound System	Yamaha	Stagepas 600BT	11-MAR-25 03.00.00 PM +03:00	13	17
12	Monitor	LG	27GL850-B	12-MAR-25 08.00.00 AM +03:00	12	14

20. PROVIDES



SUPID	MOVID	EQUIPID
1	1	2
1	15	8
2	3	4
2	6	10
3	4	12
3	5	6
4	7	8
4	8	2
5	2	4
5	9	10
6	10	12
7	11	1
8	12	2
9	13	4
10	14	6

Queries and Transactions

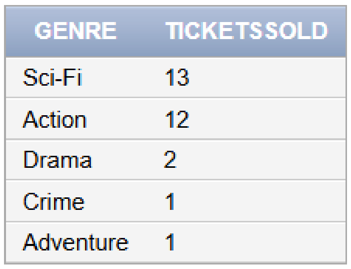
1. Identify the Most Popular Movie Genre

This query finds which movie genre has attracted the most customers, based on total tickets sold. It joins the movie genre classification with ticket sales to count how many tickets were sold for movies of each genre. The result shows genres ranked by ticket count, helping the theatre understand audience preferences by genre.

SQL QUERY:

```
SELECT mg.Genre, COUNT(t.TicketID) AS TicketsSold
FROM Ticket t
JOIN Seat s        ON t.SeID = s.SeatID
JOIN Screening sc ON s.ScrID = sc.ScreeningID
JOIN Movie_Genre mg ON sc.MovID = mg.MovID
GROUP BY mg.Genre
ORDER BY TicketsSold DESC;
```

OUTPUT:



GENRE	TICKETSSOLD
Sci-Fi	13
Action	12
Drama	2
Crime	1
Adventure	1

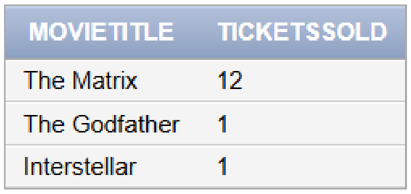
2. Top 5 Movies by Ticket Sales

This query determines the five movies with the highest number of tickets sold. It aggregates ticket records per movie and orders the results in descending order of ticket count. Managers can use this to identify blockbuster films. The query uses a multi-table join (ticket → seat → screening → movie) and the FETCH FIRST clause to limit the output to the top 5 movies.

SQL QUERY:

```
SELECT * FROM (
    SELECT m.Name AS MovieTitle, COUNT(t.TicketID) AS TicketsSold
    FROM Ticket t
    JOIN Seat s        ON t.SeID = s.SeatID
    JOIN Screening sc ON s.ScrID = sc.ScreeningID
    JOIN Movie m       ON sc.MovID = m.MovieID
    GROUP BY m.Name
    ORDER BY COUNT(t.TicketID) DESC
)
WHERE ROWNUM <= 5;
```

OUTPUT:



MOVIETITLE	TICKETSSOLD
The Matrix	12
The Godfather	1
Interstellar	1

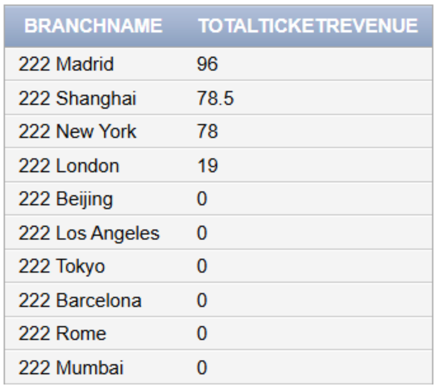
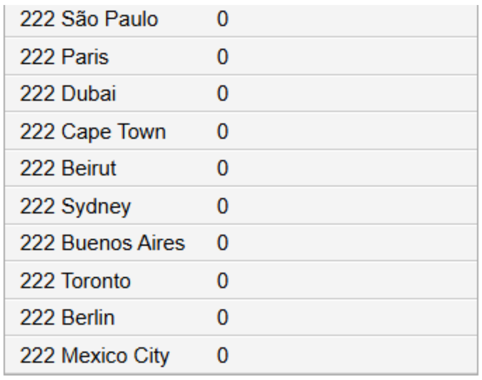
3. Calculate Total Ticket Revenue per Branch

This query computes the total revenue from ticket sales for each branch. It joins tickets through seats and screenings to the theatre and branch, then sums the ticket prices per branch. The result shows how much money each branch brought in from ticket sales. (This assumes Ticket.Price is the full price; any discounts would be applied separately via payments.)

SQL QUERY:

```
SELECT b.Name AS BranchName,
       NVL(SUM(t.Price), 0) AS TotalTicketRevenue
FROM Branch b
LEFT JOIN Theatre th     ON b.BranchID = th.BrID
LEFT JOIN Screening sc   ON th.TheatreID = sc.ThID
LEFT JOIN Seat st        ON sc.ScreeningID = st.ScrID
LEFT JOIN Ticket t       ON st.SeatID = t.SeID
GROUP BY b.Name
ORDER BY TotalTicketRevenue DESC;
```

OUTPUT:

BRANCHNAME	TOTALTICKETREVENUE
222 Madrid	96
222 Shanghai	78.5
222 New York	78
222 London	19
222 Beijing	0
222 Los Angeles	0
222 Tokyo	0
222 Barcelona	0
222 Rome	0
222 Mumbai	0
222 São Paulo	0
222 Paris	0
222 Dubai	0
222 Cape Town	0
222 Beirut	0
222 Sydney	0
222 Buenos Aires	0
222 Toronto	0
222 Berlin	0
222 Mexico City	0

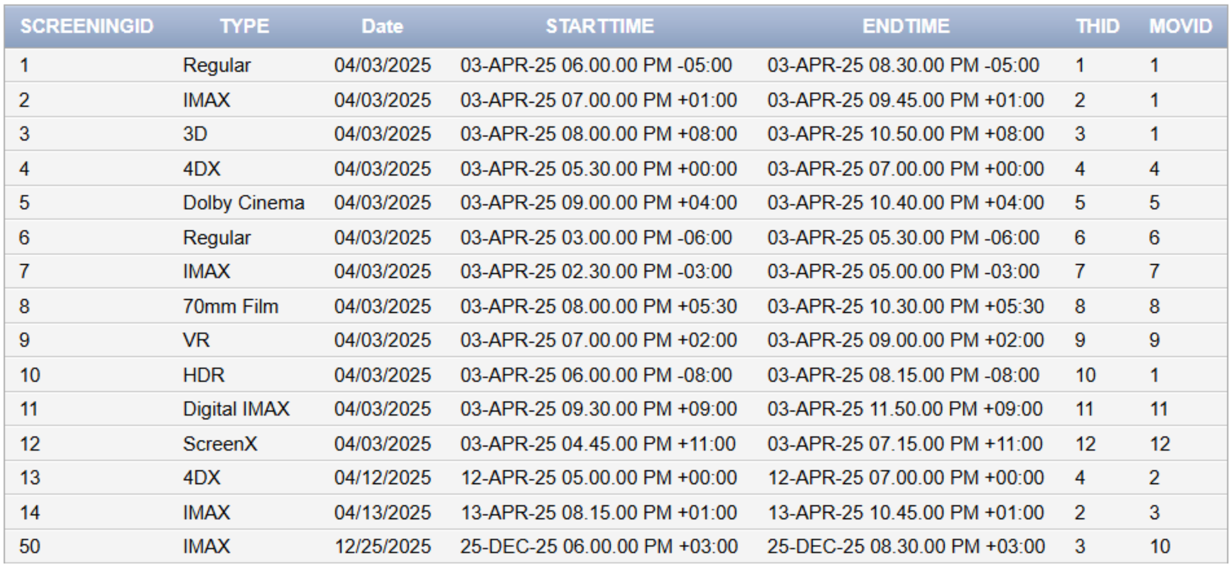
4. Insert a New Movie Screening Schedule

A new screening is added to the schedule. In this example, we insert a screening for Movie ID 10 in Theatre ID 3 on 25-Dec-2025 at 6:00 PM, marked as an IMAX showing with a 2.5-hour duration. This INSERT adds a record to the Screening table. (After this, seats for the new screening would be populated in the Seat table in practice.)

SQL QUERY

```
INSERT INTO Screening (ScreeningID, Type, "Date", StartTime, EndTime, ThID,
MovID)
VALUES (50, 'IMAX', DATE '2025-12-25',
        TIMESTAMP '2025-12-25 18:00:00',
        TIMESTAMP '2025-12-25 20:30:00',
        3, 10);
```

OUTPUT:



SCREENINGID	TYPE	Date	STARTTIME	ENDTIME	THID	MOVID
1	Regular	04/03/2025	03-APR-25 06.00.00 PM -05:00	03-APR-25 08.30.00 PM -05:00	1	1
2	IMAX	04/03/2025	03-APR-25 07.00.00 PM +01:00	03-APR-25 09.45.00 PM +01:00	2	1
3	3D	04/03/2025	03-APR-25 08.00.00 PM +08:00	03-APR-25 10.50.00 PM +08:00	3	1
4	4DX	04/03/2025	03-APR-25 05.30.00 PM +00:00	03-APR-25 07.00.00 PM +00:00	4	4
5	Dolby Cinema	04/03/2025	03-APR-25 09.00.00 PM +04:00	03-APR-25 10.40.00 PM +04:00	5	5
6	Regular	04/03/2025	03-APR-25 03.00.00 PM -06:00	03-APR-25 05.30.00 PM -06:00	6	6
7	IMAX	04/03/2025	03-APR-25 02.30.00 PM -03:00	03-APR-25 05.00.00 PM -03:00	7	7
8	70mm Film	04/03/2025	03-APR-25 08.00.00 PM +05:30	03-APR-25 10.30.00 PM +05:30	8	8
9	VR	04/03/2025	03-APR-25 07.00.00 PM +02:00	03-APR-25 09.00.00 PM +02:00	9	9
10	HDR	04/03/2025	03-APR-25 06.00.00 PM -08:00	03-APR-25 08.15.00 PM -08:00	10	1
11	Digital IMAX	04/03/2025	03-APR-25 09.30.00 PM +09:00	03-APR-25 11.50.00 PM +09:00	11	11
12	ScreenX	04/03/2025	03-APR-25 04.45.00 PM +11:00	03-APR-25 07.15.00 PM +11:00	12	12
13	4DX	04/12/2025	12-APR-25 05.00.00 PM +00:00	12-APR-25 07.00.00 PM +00:00	4	2
14	IMAX	04/13/2025	13-APR-25 08.15.00 PM +01:00	13-APR-25 10.45.00 PM +01:00	2	3
50	IMAX	12/25/2025	25-DEC-25 06.00.00 PM +03:00	25-DEC-25 08.30.00 PM +03:00	3	10

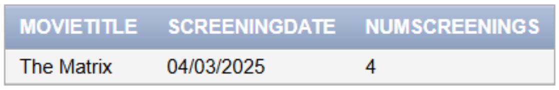
5. Detect Movies with >3 Screenings on the Same Day

This query finds movies that are being shown more than three times in a single day. It groups screenings by movie and date, counting how many showtimes each movie has per day, and uses HAVING to filter those counts greater than 3. The result highlights films with very frequent showings in one day (indicating high demand or scheduling issues).

SQL QUERY:

```
SELECT m.Name AS MovieTitle,
       sc."Date" AS ScreeningDate,
       COUNT(sc.ScreeningID) AS NumScreenings
FROM Screening sc
JOIN Movie m ON sc.MovID = m.MovieID
GROUP BY m.Name, sc."Date"
HAVING COUNT(sc.ScreeningID) > 3
ORDER BY NumScreenings DESC;
```

<u>OUTPUT:</u>



MOVIETITLE	SCREENINGDATE	NUMSCREENINGS
The Matrix	04/03/2025	4

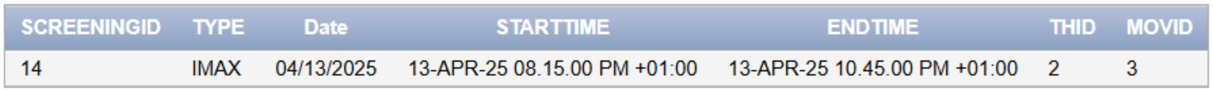
6. Reschedule Screenings of a Specific Movie

This transaction adjusts the schedule for all screenings of "Intersteller" by delaying them 15 minutes. It uses an UPDATE with a subquery to find the movie's ID by name, then adds 15 minutes to both the start and end times of every screening of that movie. This could be used to handle a last-minute delay while keeping the screening duration intact.

<u>SQL QUERY:</u>

```
UPDATE Screening
SET StartTime = StartTime + INTERVAL '15' MINUTE,
    EndTime   = EndTime + INTERVAL '15' MINUTE
WHERE MovID = (
    SELECT MovieID
    FROM Movie
    WHERE Name = 'Interstellar'
);
```

<u>OUTPUT:</u>



SCREENINGID	TYPE	Date	STARTTIME	ENDTIME	THID	MOVID
14	IMAX	04/13/2025	13-APR-25 08.15.00 PM +01:00	13-APR-25 10.45.00 PM +01:00	2	3

7. Find Departments with at Least 3 Employees

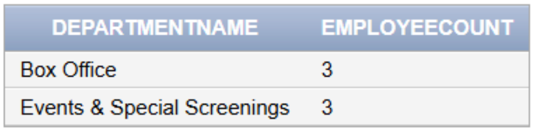
This query identifies all departments that have five or more employees. It joins employees to their departments and groups by department, using a HAVING clause to filter only those counts >= 3. This helps management see which departments are large and possibly need more resources or reorganization.

<u>SQL QUERY</u>:

```
SELECT d.Name AS DepartmentName, COUNT(e.EmployeeID) AS EmployeeCount
```

```sql
FROM Department d
JOIN Employee e ON e.EXTNb = d.EXTNumber
GROUP BY d.Name
HAVING COUNT(e.EmployeeID) >= 3;
```

OUTPUT:



DEPARTMENTNAME	EMPLOYEECOUNT
Box Office	3
Events & Special Screenings	3

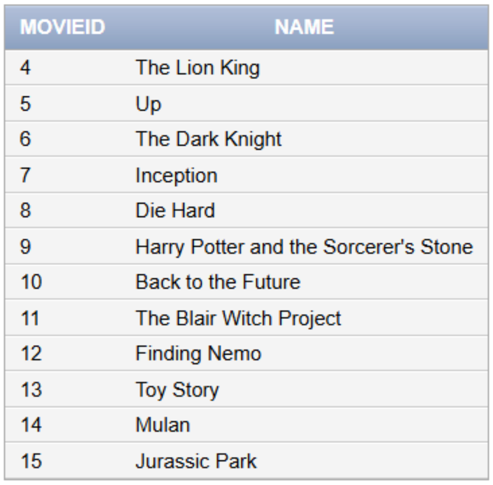
8. Find Movies That Have Never Sold a Ticket

This query lists any movies that have no tickets sold. Using a subquery with a join across screenings, seats, and tickets, it checks for movies with zero ticket records. If a movie has no associated ticket in any screening, it appears in the result. This helps identify underperforming movies (or ones never screened) which might be candidates for removal or special promotion.

SQL QUERY:

```sql
SELECT m.MovieID, m.Name
FROM Movie m
WHERE NOT EXISTS (
    SELECT 1
    FROM Screening sc
    JOIN Seat st ON sc.ScreeningID = st.ScrID
    JOIN Ticket t ON t.SeID = st.SeatID
    WHERE sc.MovID = m.MovieID
);
```

OUTPUT:



MOVIEID	NAME
4	The Lion King
5	Up
6	The Dark Knight
7	Inception
8	Die Hard
9	Harry Potter and the Sorcerer's Stone
10	Back to the Future
11	The Blair Witch Project
12	Finding Nemo
13	Toy Story
14	Mulan
15	Jurassic Park

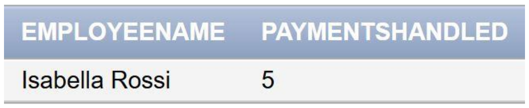
9. Identify the Top Cashier (Employee with Most Payments Processed)

This query determines which employee has processed the highest number of payment transactions (assuming cashiers or staff record payments). It joins Payment with Employee (via the EmpID foreign key) and counts payments per employee. The results are sorted by count, and the first row gives the employee who handled the most payments. Management can use this to reward the busiest cashier or examine workload distribution.

SQL QUERY:

```
SELECT *
FROM (
    SELECT e.FirstName || ' ' || e.LastName AS EmployeeName,
           COUNT(p.PaymentID) AS PaymentsHandled
    FROM Payment p
    JOIN Employee e ON p.EmpID = e.EmployeeID
    GROUP BY e.FirstName, e.LastName
    ORDER BY COUNT(p.PaymentID) DESC
)
WHERE ROWNUM = 1;
```

OUTPUT:



EMPLOYEENAME	PAYMENTSHANDLED
Isabella Rossi	5

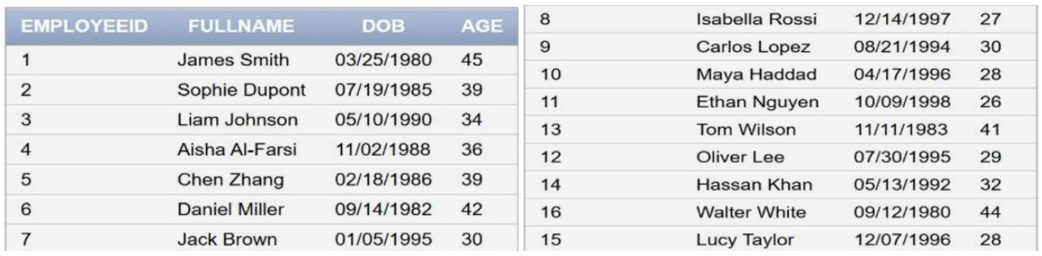
10. Derived Attribute 1 : Employee Age

This query is to calculate the derived attribute AGE from the Customer table using DOB.

SQL QUERY:

```
SELECT EmployeeID,
       FirstName || ' ' || LastName AS FullName,
       DOB,
       FLOOR(MONTHS_BETWEEN(SYSDATE, DOB) / 12) AS Age
FROM Employee;
```

OUTPUT:



EMPLOYEEID	FULLNAME	DOB	AGE
1	James Smith	03/25/1980	45
2	Sophie Dupont	07/19/1985	39
3	Liam Johnson	05/10/1990	34
4	Aisha Al-Farsi	11/02/1988	36
5	Chen Zhang	02/18/1986	39
6	Daniel Miller	09/14/1982	42
7	Jack Brown	01/05/1995	30
8	Isabella Rossi	12/14/1997	27
9	Carlos Lopez	08/21/1994	30
10	Maya Haddad	04/17/1996	28
11	Ethan Nguyen	10/09/1998	26
13	Tom Wilson	11/11/1983	41
12	Oliver Lee	07/30/1995	29
14	Hassan Khan	05/13/1992	32
16	Walter White	09/12/1980	44
15	Lucy Taylor	12/07/1996	28

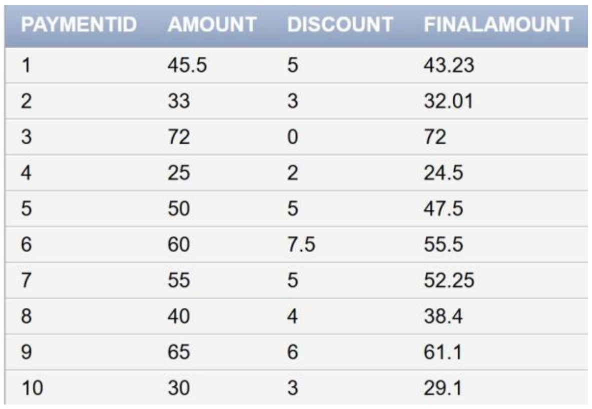
11. Final Payment Amount

This query is to calculate the final payment after the discount is added.

SQL QUERY:

```
SELECT PaymentID,
       Amount,
       Discount,
       ROUND(Amount - (Amount * Discount / 100), 2) AS FinalAmount
FROM Payment;
```

OUTPUT:



PAYMENTID	AMOUNT	DISCOUNT	FINALAMOUNT
1	45.5	5	43.23
2	33	3	32.01
3	72	0	72
4	25	2	24.5
5	50	5	47.5
6	60	7.5	55.5
7	55	5	52.25
8	40	4	38.4
9	65	6	61.1
10	30	3	29.1

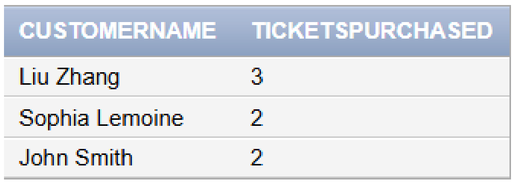
12. Identify Top 3 Loyal Customers (Most Tickets Purchased)

This query finds the three customers who have purchased the most tickets, indicating highly engaged patrons. It joins tickets with customers, counts tickets per customer, and then limits the result to the top 3. This insight helps the theatre recognize and potentially reward its most loyal moviegoers.

SQL QUERY:

```
SELECT * FROM (
    SELECT c.FirstName || ' ' || c.LastName AS CustomerName,
           COUNT(t.TicketID) AS TicketsPurchased
    FROM Customer c
    JOIN Ticket t ON c.CustomerID = t.CustID
    GROUP BY c.FirstName, c.LastName
    ORDER BY COUNT(t.TicketID) DESC
)
WHERE ROWNUM <= 3;
```

OUTPUT:



CUSTOMERNAME	TICKETSPURCHASED
Liu Zhang	3
Sophia Lemoine	2
John Smith	2

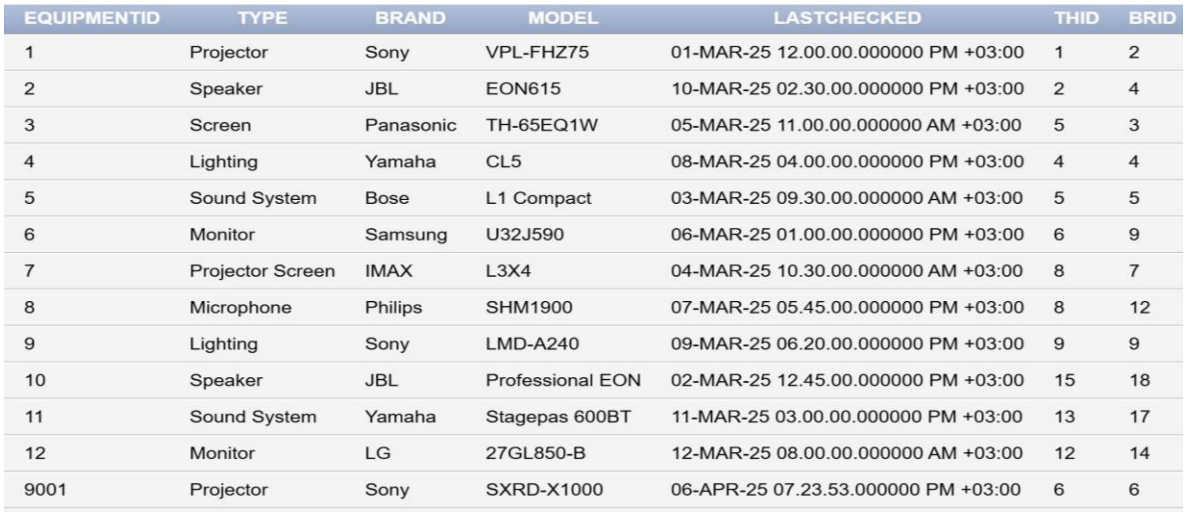
13. Insert New Equipment

Insert a new projector for each theatre in the 222 PARIS branch.

SQL QUERY:

```
INSERT INTO Equipment (EquipmentID, Type, Brand, Model, LastChecked, ThID, BrID)
SELECT 9000 + ROW_NUMBER() OVER (ORDER BY th.TheatreID),
       'Projector', 'Sony', 'SXRD-X1000',
       SYSDATE, th.TheatreID, b.BranchID
FROM Theatre th
JOIN Branch b ON th.BrID = b.BranchID
WHERE b.Name = '222 Paris';
```

OUTPUT:



EQUIPMENTID	TYPE	BRAND	MODEL	LASTCHECKED	THID	BRID
1	Projector	Sony	VPL-FHZ75	01-MAR-25 12.00.00.000000 PM +03:00	1	2
2	Speaker	JBL	EON615	10-MAR-25 02.30.00.000000 PM +03:00	2	4
3	Screen	Panasonic	TH-65EQ1W	05-MAR-25 11.00.00.000000 AM +03:00	5	3
4	Lighting	Yamaha	CL5	08-MAR-25 04.00.00.000000 PM +03:00	4	4
5	Sound System	Bose	L1 Compact	03-MAR-25 09.30.00.000000 AM +03:00	5	5
6	Monitor	Samsung	U32J590	06-MAR-25 01.00.00.000000 PM +03:00	6	9
7	Projector Screen	IMAX	L3X4	04-MAR-25 10.30.00.000000 AM +03:00	8	7
8	Microphone	Philips	SHM1900	07-MAR-25 05.45.00.000000 PM +03:00	8	12
9	Lighting	Sony	LMD-A240	09-MAR-25 06.20.00.000000 PM +03:00	9	9
10	Speaker	JBL	Professional EON	02-MAR-25 12.45.00.000000 PM +03:00	15	18
11	Sound System	Yamaha	Stagepas 600BT	11-MAR-25 03.00.00.000000 PM +03:00	13	17
12	Monitor	LG	27GL850-B	12-MAR-25 08.00.00.000000 AM +03:00	12	14
9001	Projector	Sony	SXRD-X1000	06-APR-25 07.23.53.000000 PM +03:00	6	6

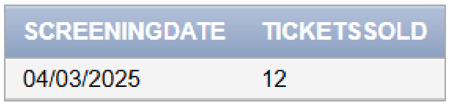
14. Find the Busiest Screening Day

Determine which date had the most ticket sales, helps identify peak business days (public holidays or release days)

SQL QUERY:

```
SELECT *
FROM (
  SELECT sc."Date" AS ScreeningDate,
         COUNT(t.TicketID) AS TicketsSold
  FROM Ticket t
  JOIN Seat st      ON t.SeID = st.SeatID
  JOIN Screening sc ON st.ScrID = sc.ScreeningID
  GROUP BY sc."Date"
  ORDER BY COUNT(t.TicketID) DESC
)
WHERE ROWNUM = 1;
```

OUTPUT:



SCREENINGDATE	TICKETSSOLD
04/03/2025	12

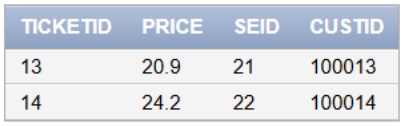
15. Apply Weekend Pricing Policy

This query calculates the average overall review rating for each branch by combining ambiance, service, cleanliness, and facilities scores. It aggregates ratings from the Review table and traces them through customers, tickets, seats, screenings, and theatres back to their corresponding branches. By grouping the results by city and averaging the total of all rating categories, the theater gains insights into customer satisfaction levels per location, helping management identify top-performing branches and those needing improvement.

SQL QUERY:

```
UPDATE Ticket
SET Price = Price * 1.10
WHERE SeID IN (
    SELECT s.SeatID
    FROM Seat s
    JOIN Screening sc ON s.ScrID = sc.ScreeningID
    WHERE TO_CHAR(sc."Date", 'DY', 'NLS_DATE_LANGUAGE = AMERICAN') IN ('SAT',
'SUN')
);
```

OUTPUT:



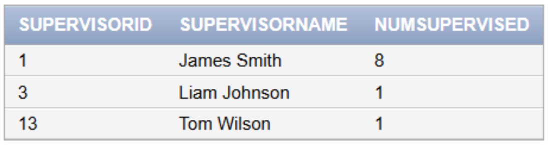
TICKETID	PRICE	SEID	CUSTID
13	20.9	21	100013
14	24.2	22	100014

16. Find employees supervising the most people

Identify the top 3 employees who supervise the largest number of employees, This helps the HR analyze management load.

SQL QUERY:

```
SELECT *
FROM (
  SELECT s.EmployeeID AS SupervisorID,
         s.FirstName || ' ' || s.LastName AS SupervisorName,
         COUNT(e.EmployeeID) AS NumSupervised
  FROM Employee s
  JOIN Employee e ON s.EmployeeID = e.SupervisorID
  GROUP BY s.EmployeeID, s.FirstName, s.LastName
  ORDER BY COUNT(e.EmployeeID) DESC
)
WHERE ROWNUM <= 3;
```
OUTPUT:



SUPERVISORID	SUPERVISORNAME	NUMSUPERVISED
1	James Smith	8
3	Liam Johnson	1
13	Tom Wilson	1

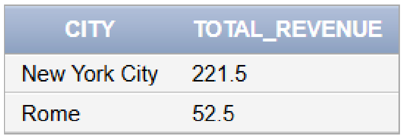
17. Calculate the Revenue Per City

This query determines the total revenue generated in each city by summing completed payments, factoring in applied discounts. It joins the Payment, Employee, and Branch tables to trace each transaction back to its corresponding city. By grouping the results by city and ordering them in descending order of revenue, the theater can identify which locations are performing best financially and make informed business decisions based on geographic performance.

SQL QUERY:

```
SELECT b.City, SUM(p.Amount - p.Discount) AS Total_Revenue
FROM Payment p
JOIN Employee e ON p.EmpID = e.EmployeeID
JOIN Branch b ON e.BrID = b.BranchID
WHERE p.Status = 'Completed'
GROUP BY b.City
ORDER BY Total_Revenue DESC;
```

OUTPUT:



CITY	TOTAL_REVENUE
New York City	221.5
Rome	52.5

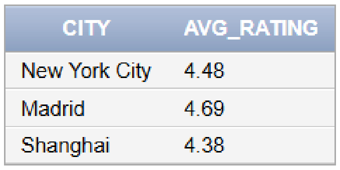
18. Get Average Review Ratings Per Branch

This query calculates the average overall review rating for each branch by combining ambiance, service, cleanliness, and facilities scores. It aggregates ratings from the Review table and traces them through customers, tickets, seats, screenings, and theatres back to their corresponding branches. By grouping the results by city and averaging the total of all rating categories, the theater gains insights into customer satisfaction levels per location, helping management identify top-performing branches and those needing improvement.

SQL QUERY:

```
SELECT b.City, ROUND(AVG(r.AmbianceR + r.ServiceR + r.CleanlinessR +
r.FacilitiesR)/4, 2) AS Avg_Rating
FROM Review r
JOIN Customer c ON r.CustID = c.CustomerID
JOIN Ticket t ON c.CustomerID = t.CustID
JOIN Seat s ON t.SeID = s.SeatID
JOIN Screening sc ON s.ScrID = sc.ScreeningID
JOIN Theatre th ON sc.ThID = th.TheatreID
JOIN Branch b ON th.BrID = b.BranchID
GROUP BY b.City;
```

OUTPUT:



CITY	AVG_RATING
New York City	4.48
Madrid	4.69
Shanghai	4.38

Normalization Up to The BCNF Normal Form

Here we are going to normalize our database up to the fourth normal form which is the Boyce-Codd Normal Form. On each relation we are going to apply the four normal forms. We start with the first then second then third and at last the BCNF normal form. Let us first start by a general description to each normal form.

First Normal Form:

This form does not allow multivalued attributes, composite attributes, and their combinations to exist in a relation.

1. Only attribute values permitted are single atomic values.
2. Domain of an attribute must only include atomic values and the value of an attribute in a tuple must be a single value from the domain of that attribute.
3. Disallows having a set of values as an attribute value for a single tuple.

Second Normal Form:

The Second normal form is based on the concept of full functional dependency. Before explaining the second form let us define some concepts used in this form and other forms also.

Functional Dependencies: A constraint between two sets of attributes from the database. The values of the Y component of a tuple in relation R depend on, or are determined by the values of an X component. We say that Y is functionally dependent on X.

Prime attribute: An attribute that is a member of a candidate key in a relation R. An attribute is called nonprime if it is not a prime attribute that is, if it is not a member of any candidate key.

Full functional dependency: A functional dependency $X \rightarrow Y$ is a full functional dependency if removal of any attribute A from X means that the dependency does not hold anymore.

Partial Dependency: A functional dependency $X \rightarrow Y$ is a partial functional dependency if removal of any attribute A from X means that the dependency still holds.

A relation schema R is in the second normal form if every nonprime attribute in R is fully functionally dependent on the every key of R and every nonprime attribute A in R is not partially dependent on any key in R.

Third Normal Form:

The third normal form is based on the concept of transitive dependency. So let us first define a transitive dependency.

Transitive Dependency: A functional dependency X →Y in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X →Z and Z →Y hold.

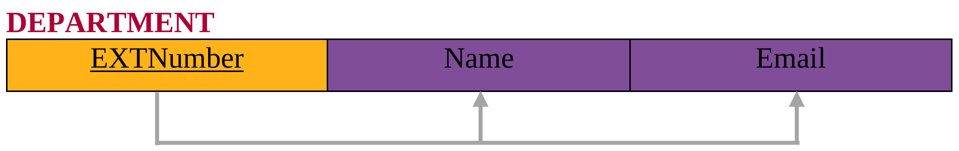
A relation schema R is in the third normal form if it satisfies the second normal form and no nonprime attribute of R is transitively dependent on the primary key. For every nontrivial functional dependency X →Y either X should be a super key or Y is a prime attribute.

Boycee-Codd Normal Form:

The Boycee-Codd normal form is a stricter form than the third normal form. The BCNF differs from the definition of the third normal form in only one condition. The third normal form allows the right-hand side of the functional dependency to be a prime attribute while BCNF does not allow that.

Normalization of Individual Relations

1. DEPARTMENT



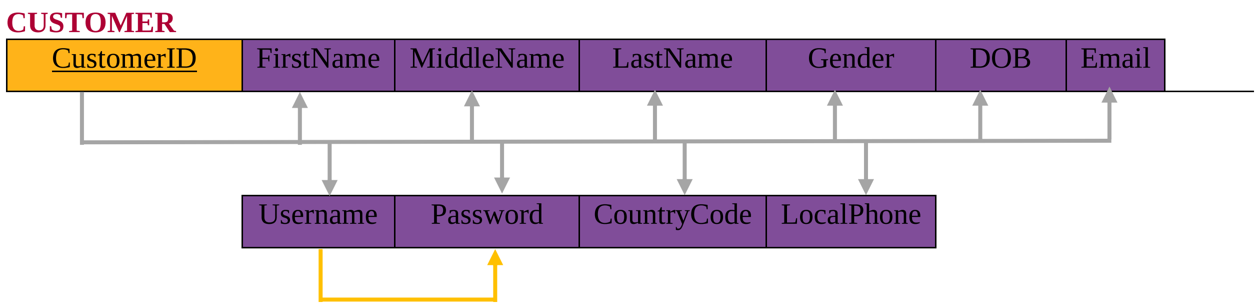
First Normal Form (1NF): The DEPARTMENT relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

Second Normal Form (2NF): The DEPARTMENT relation schema satisfies all conditions of the 2NF because every nonprime attribute (Name, Email) is fully functionally dependent on the primary key "EXTNumber".

Third Normal Form (3NF): The DEPARTMENT relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no transitive dependencies between nonprime attributes and the primary key.

Boyce-Codd Normal Form (BCNF): The DEPARTMENT relation satisfies all conditions of the BCNF because every determinant (here, EXTNumber) is a superkey.

2. CUSTOMER

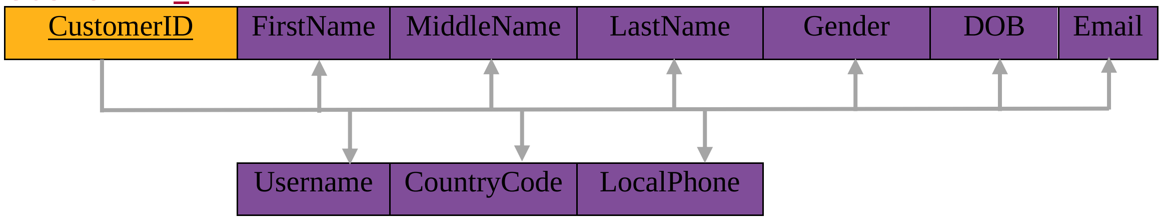


First Normal Form (1NF): The CUSTOMER relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

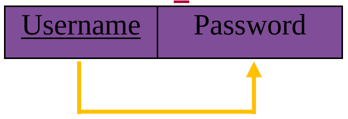
Second Normal Form (2NF): The CUSTOMER relation schema satisfies all conditions of the 2NF because every nonprime attribute (FirstName, MiddleName, LastName, Gender, DOB, Email, Username, Password, CountryCode, LocalPhone) is fully functionally dependent on the primary key "CustomerID".

Third Normal Form (3NF): The CUSTOMER relation schema does not satisfy all conditions of the 3NF because it includes a transitive dependency: **CustomerID→Username→Password** and. Thus, further decomposition is needed.

CUSTOMER_A



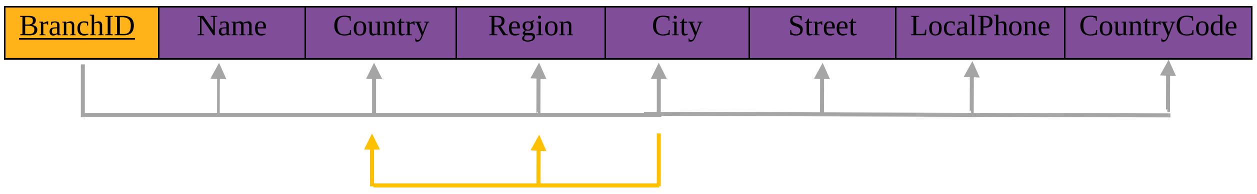
CUSTOMER_B



Boyce-Codd Normal Form (BCNF): The CUSTOMER relation satisfies all conditions of the BCNF because every determinant is a superkey.

3. BRANCH

Branch

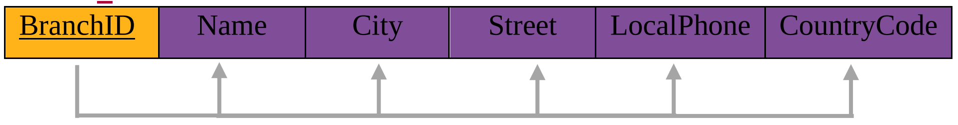


First Normal Form (1NF): The BRANCH relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

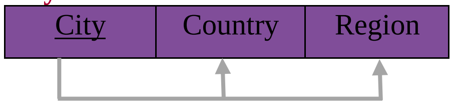
Second Normal Form (2NF): The schema satisfies all conditions of the 2NF because every nonprime attribute (Name, Country, Region, City, Street, CountryCode, LocalPhone) is fully functionally dependent on the primary key. There are no partial dependencies.

Third Normal Form (3NF): The BRANCH relation schema does not satisfy all conditions of the 3NF because it includes one transitive dependency: **BranchID → City → (Region, Country)**.
Thus, further decomposition is needed.

Branch_A



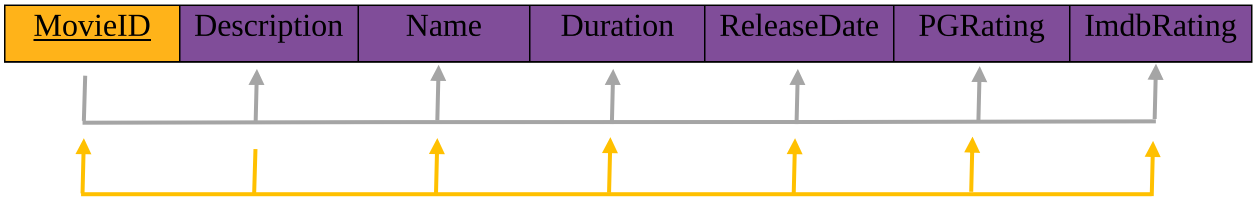
City



Boyce-Codd Normal Form (BCNF): After the decomposition, both BRANCH_A and CITY relations satisfy the conditions of BCNF because every determinant is a superkey and all functional dependencies are preserved.

4. MOVIE

Movie

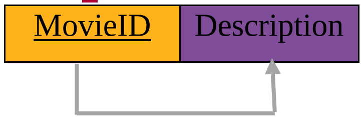


First Normal Form (1NF): The MOVIE relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

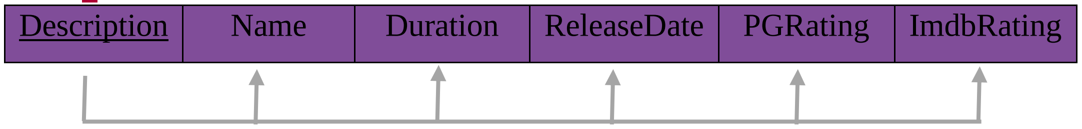
Second Normal Form (2NF): The MOVIE relation schema satisfies all conditions of the 2NF because every nonprime attribute (Description, Name, Duration, ReleaseDate, PGRating, ImdbRating) is fully functionally dependent on the primary key MovieID. There are no partial dependencies.

Third Normal Form (3NF): The MOVIE relation schema does not satisfy all conditions of the 3NF because it includes a transitive dependency:
MovieID → Description → (Name, Duration, ReleaseDate, PGRating, ImdbRating).
Thus, further decomposition is needed.

Movie_A



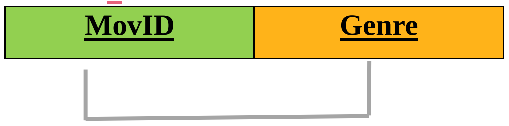
Movie_B



Boyce-Codd Normal Form (BCNF): After decomposing the MOVIE relation into two relations — one with MovieID and Description, and another with Description determining the remaining attributes — both resulting relations satisfy the conditions of the BCNF because every determinant is a superkey.

5. MOVIE_GENRE

MOVIE_GENRE



The MOVIE_GENRE relation does not contain any non-prime attributes, so there are **no functional dependencies** beyond the composite key (**MovID, Genre**), which ensures uniqueness of each genre associated with a movie (Multivalued Dependencies – MVDs – 4NF).

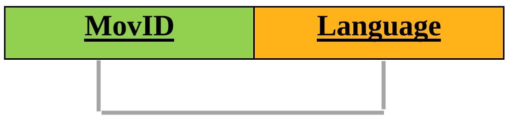
First Normal Form (1NF): The MOVIE_GENRE relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

Second Normal Form (2NF): The MOVIE_GENRE relation schema satisfies all conditions of the 2NF because it has a composite primary key (MovID, Genre) and no non-prime attributes exist. Therefore, there are no partial dependencies.

Third Normal Form (3NF): The MOVIE_GENRE relation schema satisfies all conditions of the 3NF because there are no transitive dependencies between nonprime attributes. Every attribute is part of the primary key.

Boyce-Codd Normal Form (BCNF): The MOVIE_GENRE relation schema satisfies all conditions of the BCNF because every determinant (MovID and Genre) is a superkey.

6. MOVIE_LANGUAGE



The MOVIE_LANGUAGE relation does not contain any non-prime attributes, so there are **no functional dependencies** beyond the composite key (**MovID, Language**), which ensures uniqueness of each genre associated with a movie (Multivalued Dependencies – MVDs – 4NF).

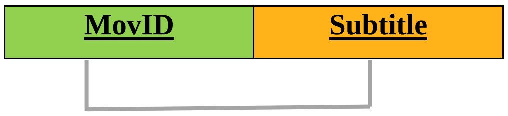
First Normal Form (1NF): The MOVIE_LANGUAGE relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

Second Normal Form (2NF): The MOVIE_LANGUAGE relation schema satisfies all conditions of the 2NF because it has a composite primary key (MovID, Language) and no non-prime attributes exist. Therefore, there are no partial dependencies.

Third Normal Form (3NF): The MOVIE_LANGUAGE relation schema satisfies all conditions of the 3NF because there are no transitive dependencies between nonprime attributes. Every attribute is part of the primary key.

Boyce-Codd Normal Form (BCNF): The MOVIE_LANGUAGE relation schema satisfies all conditions of the BCNF because every determinant (MovID and Language) is a superkey.

7. MOVIE_SUBTITLE



The MOVIE_SUBTITLE relation does not contain any non-prime attributes, so there are no functional dependencies beyond the composite key (MovID, Subtitle), which ensures uniqueness of each set of subtitles associated with a movie (Multivalued Dependencies – MVDs – 4NF).

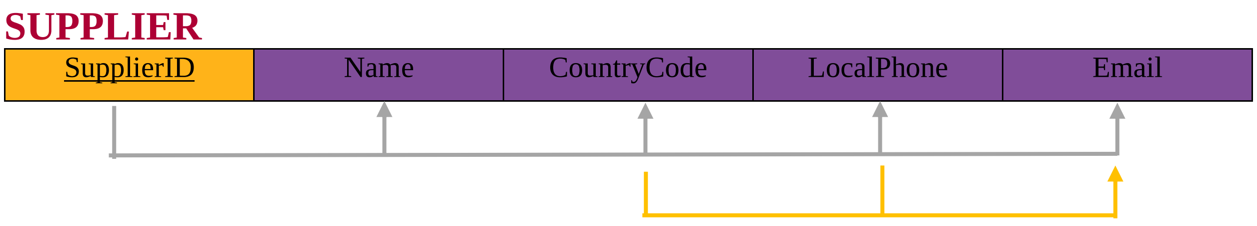
First Normal Form (1NF): The MOVIE_SUBTITLE relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

Second Normal Form (2NF): The MOVIE_SUBTITLE relation schema satisfies all conditions of the 2NF because it has a composite primary key (MovID, Subtitle) and no non-prime attributes exist. Therefore, there are no partial dependencies.

Third Normal Form (3NF): The MOVIE_SUBTITLE relation schema satisfies all conditions of the 3NF because there are no transitive dependencies between nonprime attributes. Every attribute is part of the primary key.

Boyce-Codd Normal Form (BCNF): The MOVIE_SUBTITLE relation schema satisfies all conditions of the BCNF because every determinant (MovID and Subtitle) is a superkey.

8. SUPPLIER



First Normal Form (1NF):

The SUPPLIER relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

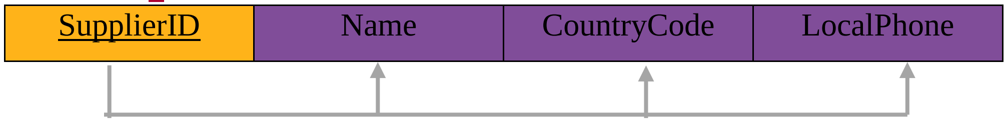
Second Normal Form (2NF):

The SUPPLIER relation schema satisfies all conditions of the 2NF because every nonprime attribute (SupplierName, CountryCode, LocalPhone, Email) is fully functionally dependent on the primary key "SupplierID".

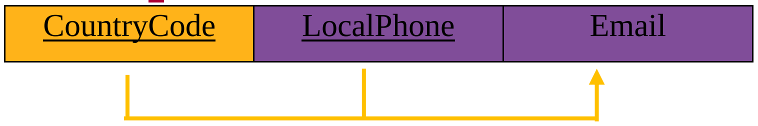
Third Normal Form (3NF):

The SUPPLIER relation schema does not satisfy all conditions of the 3NF because it includes a transitive dependency: SupplierID → CountryCode, LocalPhone → Email. Thus, further decomposition is needed.

SUPPLIER_A



SUPPLIER_B

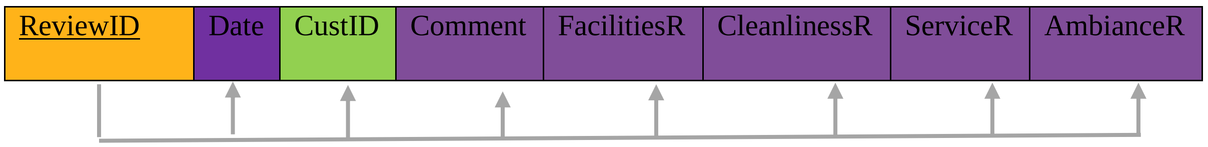


Boyce-Codd Normal Form (BCNF):

The SUPPLIER relation satisfies all conditions of the BCNF because every determinant is a superkey after decomposition.

9. REVIEW

REVIEW



First Normal Form (1NF):

The REVIEW relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

Second Normal Form (2NF):

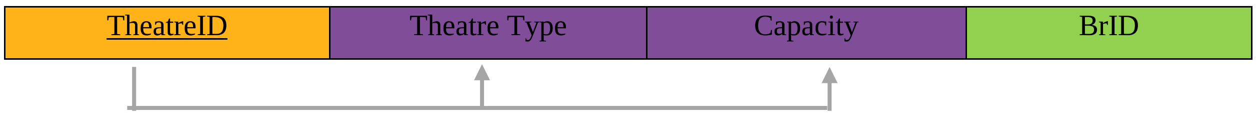
The REVIEW relation schema satisfies all conditions of the 2NF because every nonprime attribute (Date, CustID, Comment, FacilitiesR, CleanlinessR, ServiceR, AmbianceR) is fully functionally dependent on the primary key “ReviewID”.

Third Normal Form (3NF):
The REVIEW relation schema satisfies all conditions of the 3NF because there are no transitive dependencies among the nonprime attributes.

Boyce-Codd Normal Form (BCNF):
The REVIEW relation satisfies all conditions of the BCNF because every determinant is a superkey.

10.THEATRE

THEATRE



First Normal Form (1NF):
The THEATRE relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

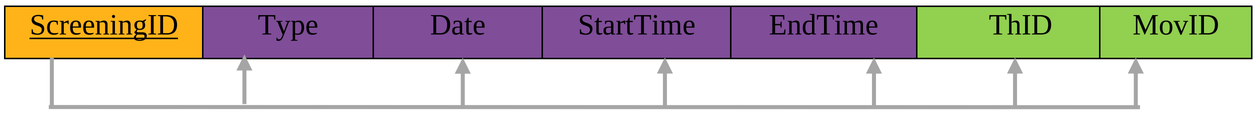
Second Normal Form (2NF):
The THEATRE relation schema satisfies all conditions of the 2NF because every nonprime attribute (TheatreType, Capacity) is fully functionally dependent on the primary key “TheatreID”.

Third Normal Form (3NF):
The THEATRE relation schema satisfies all conditions of the 3NF because there are no transitive dependencies among the nonprime attributes.

Boyce-Codd Normal Form (BCNF):
The THEATRE relation satisfies all conditions of the BCNF because every determinant is a superkey.

11.SCREENING

SCREENING



First Normal Form (1NF):

The SCREENING relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. The previously composite attribute ShowTime was properly separated into atomic attributes StartTime and EndTime.

Second Normal Form (2NF):

The SCREENING relation schema satisfies all conditions of the 2NF because every nonprime attribute (Type, Date, StartTime, EndTime, ThID, MovID) is fully functionally dependent on the primary key "ScreeningID".

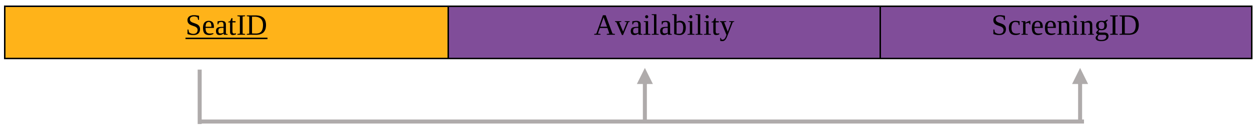
Third Normal Form (3NF):

The SCREENING relation schema satisfies all conditions of the 3NF because there are no transitive dependencies among the nonprime attributes.

Boyce-Codd Normal Form (BCNF):

The SCREENING relation satisfies all conditions of the BCNF because every determinant is a superkey.

12.SEAT



The SEAT relation consists of the attributes SeatID, Availability, and ScrID. The primary key is SeatID, which uniquely identifies each seat. ScrID is a foreign key referencing the screening in which the seat is located. Availability indicates the current status of the seat (e.g., Available, Occupied, Reserved).

First Normal Form (1NF):

The SEAT relation satisfies 1NF because all attributes contain atomic values and there are no repeating groups or multivalued attributes.

Second Normal Form (2NF):

The SEAT relation satisfies 2NF because it has a single-attribute primary key (SeatID), and all other attributes (Availability, ScrID) are fully functionally dependent on it. There are no partial dependencies.

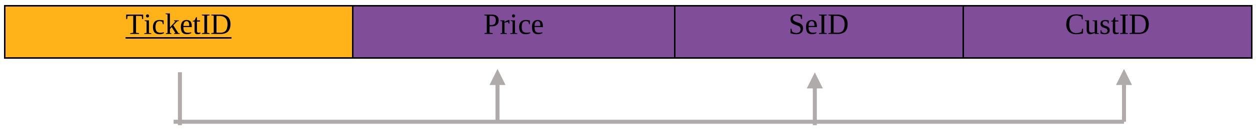
Third Normal Form (3NF):

The SEAT relation satisfies 3NF because there are no transitive dependencies. All non-prime attributes (Availability, ScrID) depend directly on the primary key and not on one another.

Boyce-Codd Normal Form (BCNF):

The SEAT relation satisfies BCNF because all functional dependencies have a determinant that is a superkey. Specifically, SeatID → Availability, ScrID, and SeatID is a superkey.

13.TICKET



The TICKET relation consists of four attributes: TicketID, Price, SeID, and CustID. The primary key is TicketID, which uniquely identifies each ticket issued. The relation includes a foreign key to the SEAT table (SeID) and a foreign key to the CUSTOMER table (CustID). A unique constraint is also imposed on the combination (SeID, CustID) to prevent multiple bookings of the same seat by the same customer.

First Normal Form (1NF):

The TICKET relation schema satisfies all conditions of 1NF because it has no multivalued or composite attributes. All attributes are atomic and contain indivisible values.

Second Normal Form (2NF):

The TICKET relation schema satisfies all conditions of 2NF because it has a single-attribute primary key (TicketID), and all non-key attributes (Price, SeID, CustID) are fully functionally dependent on that primary key. There are no partial dependencies.

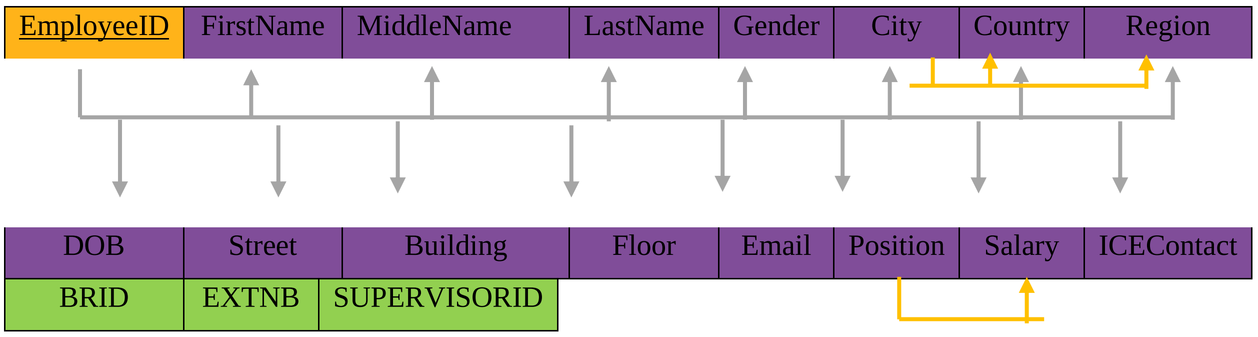
Third Normal Form (3NF):

The TICKET relation satisfies 3NF because there are no transitive dependencies among non-prime attributes. Price, SeID, and CustID depend only on TicketID, and no non-key attribute determines another non-key attribute.

Boyce-Codd Normal Form (BCNF):

The TICKET relation satisfies BCNF because the only non-trivial functional dependency is TicketID → Price, SeID, CustID, and TicketID is a superkey. All determinants in the relation are superkeys.

14. EMPLOYEE



First Normal Form (1NF):

The EMPLOYEE relation schema satisfies all conditions of 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic, as composite attributes like Name and Address were already broken down into their atomic parts (FirstName, MiddleName, LastName, Country, Region, City, etc.).

Second Normal Form (2NF):

The schema satisfies all conditions of 2NF because the primary key is a single attribute (EmployeeID), and every non-prime attribute is fully functionally dependent on the entire primary key. There are no partial dependencies.

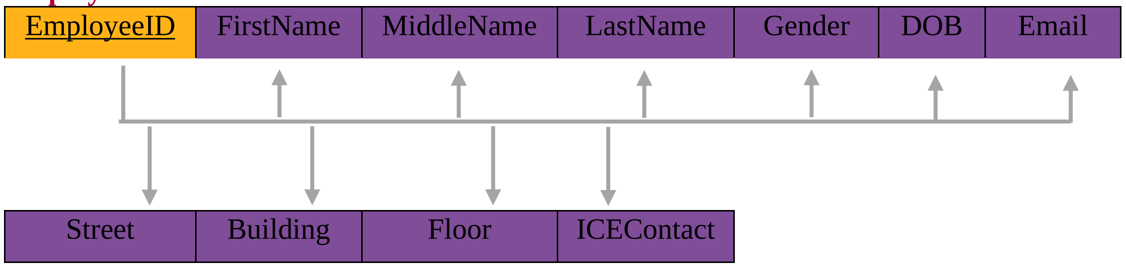
Third Normal Form (3NF):

The EMPLOYEE relation schema does not satisfy all conditions of 3NF because it contains two transitive dependencies:

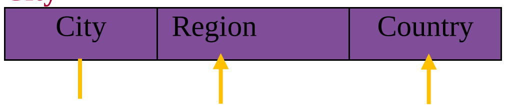
- EmployeeID → City → (Region, Country)
- EmployeeID → Position → Salary

Thus, further decomposition is needed.

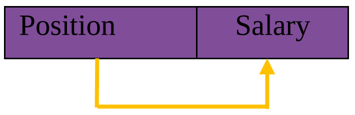
Employee



City



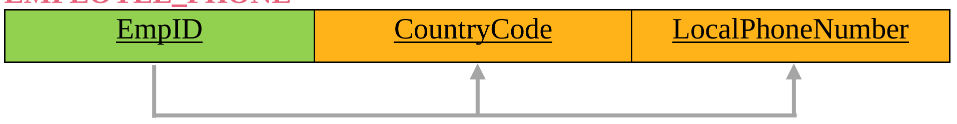
Position



After decomposition, the EMPLOYEE, CITY, and POSITION_SALARY relations all satisfy the conditions of BCNF. Every determinant in each relation is a candidate key (or superkey), and all functional dependencies are preserved. This ensures the schema is free from redundancy and update anomalies, while respecting the rule of not introducing new attributes.

15.EMPLOYEE_PHONE

EMPLOYEE_PHONE



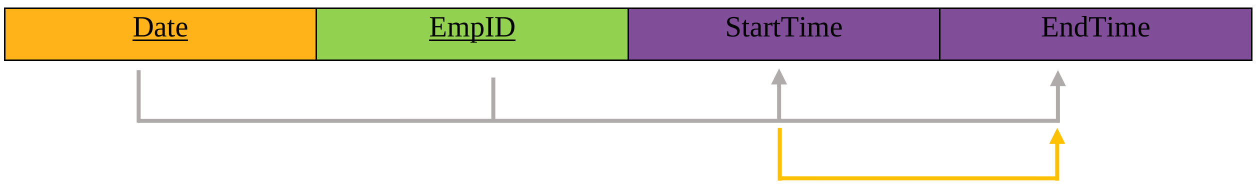
First Normal Form (1NF): The EMPLOYEE_PHONE relation schema satisfies all conditions of the 1NF because it has neither multivalued attributes nor composite attributes. All attributes are single and atomic.

Second Normal Form (2NF): The EMPLOYEE_PHONE relation schema satisfies all conditions of the 2NF because every nonprime attribute (CountryCode, LocalPhoneNumber) is fully functionally dependent on the primary key "EmpID".

Third Normal Form (3NF): The EMPLOYEE_PHONE relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no transitive dependencies between nonprime attributes and the primary key.

Boyce-Codd Normal Form (BCNF): The EMPLOYEE_PHONE relation satisfies all conditions of the BCNF because every determinant (here, EmpID) is a superkey.

16.SCHEDULE



The SCHEDULE relation initially includes attributes Date, StartTime, EndTime, and EmpID, with the primary key being (EmpID, Date).

First Normal Form (1NF):

The SCHEDULE relation satisfies all conditions of the 1NF because all attributes are atomic and there are no repeating groups or multivalued attributes.

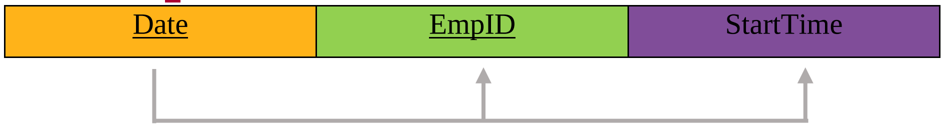
Second Normal Form (2NF):

The relation satisfies 2NF because all non-prime attributes are fully functionally dependent on the entire composite primary key (EmpID, Date). There are no partial dependencies.

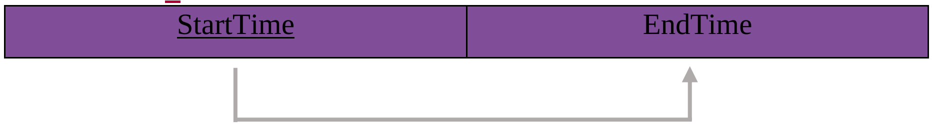
Third Normal Form (3NF):

The relation does not satisfy 3NF due to the presence of a transitive dependency: (EmpID, Date) → StartTime → EndTime. Here, EndTime is transitively dependent on the primary key through StartTime, which is not a candidate key. Thus, further decomposition is needed.

SCHEDULE_A

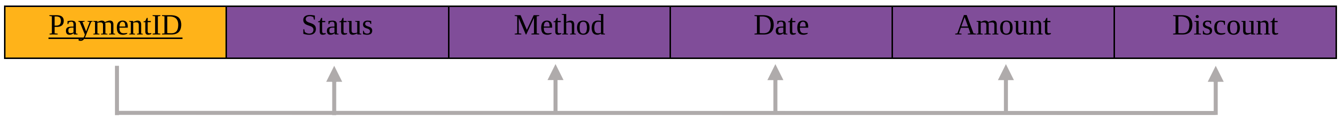


SCHEDULE_B



Boyce-Codd Normal Form (BCNF):
The REVIEW relation satisfies all conditions of the BCNF because every determinant is a superkey after decomposition.

17.PAYMENT



The **PAYMENT** relation includes transaction records with attributes such as status, method of payment, date, amount, discount, and the employee and customer involved. The underlined attribute PaymentID serves as the primary key and uniquely identifies each payment. All other attributes in the relation depend on this identifier.

First Normal Form (1NF):

The PAYMENT relation satisfies 1NF because all attributes are atomic. There are no repeating groups or multivalued attributes—each cell holds a single value.

Second Normal Form (2NF):

Since PaymentID is the only primary key (a single attribute), there are no partial dependencies. Each non-key attribute depends entirely on the whole key. Thus, the relation satisfies 2NF.

Third Normal Form (3NF):

There are no transitive dependencies among non-prime attributes. Each non-key attribute is directly and fully dependent on the primary key PaymentID, and not on any other non-key attribute. Hence, the relation satisfies 3NF.

Boyce-Codd Normal Form (BCNF):

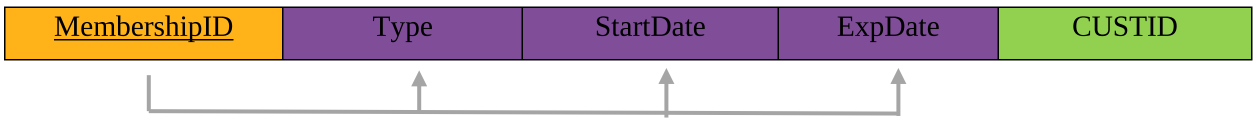
Every functional dependency in the PAYMENT table has a determinant that is a superkey. Specifically, the only meaningful functional dependency is:

PaymentID → Status, Method, Date, Amount, Discount, EmpID, CustID

No other combinations of attributes uniquely determine another attribute. Therefore, the PAYMENT table satisfies the conditions of BCNF.

Although FinalAmount can be functionally determined by (Amount, Discount), we chose not to decompose the table based on this derived attribute. Instead, FinalAmount is calculated dynamically through queries to avoid redundancy and ensure consistency with any future changes to Amount or Discount.

18.MEMBERSHIP

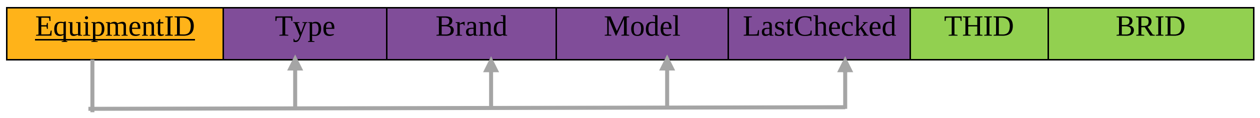


First Normal Form (1NF): The MEMBERSHIP relation schema satisfies all conditions of 1NF because it has neither multivalued attributes nor composite attributes in its final representation. The Period composite attribute was decomposed into StartDate and ExpDate, making all attributes single and atomic.

Second Normal Forms (2NF): The schema satisfies all conditions of 2NF because the primary key is a single attribute (MembershipID), and every non-prime attribute (Type, StartDate, ExpDate) is fully functionally dependent on the entire primary key. There are no partial dependencies.

Third Normal Form (3NF): The MEMBERSHIP relation schema also satisfies all conditions of 3NF because there are no transitive dependencies. None of the non-prime attributes functionally determine other non-prime attributes

19.EQUIPMENT



First Normal Form(1NF): The EQUIPMENT relation schema satisfies all conditions of 1NF because all attributes are simple and atomic. There are no multivalued or composite attributes.

Second Normal Form (2NF): The schema satisfies all conditions of 2NF because the primary key is a single attribute (EquipmentID), and every non-prime attribute (Type, Brand, Model, LastChecked) is fully functionally dependent on the entire primary key. No partial dependencies exist.

Third Normal Form (3NF): The EQUIPMENT relation also satisfies all conditions of 3NF. There are **no transitive dependencies** — no non-prime attribute determines another non-prime attribute. Type, Brand, and Model are directly dependent on EquipmentID.

Boyce-Codd Normal Form (BCNF): The EQUIPMENT relation satisfies the conditions of BCNF because the only functional dependency is:

- EquipmentID → Type, Brand, Model, LastChecked

Since the determinant is a candidate key (EquipmentID), and no violations exist, the schema is already in BCNF. No decomposition is needed.

20.PROVIDES

SupID	**MovID**	**EquipID**

First Normal Form(1NF): The PROVIDES relation satisfies all conditions of 1NF because all attributes are atomic and there are no multivalued or composite attributes.

Second Normal Form(2NF): The schema satisfies 2NF because its primary key is a composite of three attributes (SupID, MovID, EquipID), and there are no partial dependencies. Each non-prime attribute (in this case, none) would be fully dependent on the entire composite key.

Third Normal Form (3NF): The PROVIDES relation satisfies 3NF because there are no transitive dependencies. All attributes are part of the primary key, and there are no non-prime attributes in the relation that depend on other non-prime attributes.

Boyce-Codd Normal Form (BCNF): The PROVIDES relation satisfies BCNF because every determinant is a candidate key. The only dependencies present are:

- (SupID, MovID, EquipID) → SupID, MovID, EquipID

  Since the only determinants are candidate keys, the relation is in BCNF and no decomposition is needed.

Final Table State

DEPARTMENT

EXTNumber	Name	Email

CUSTOMER_A

CustomerID	FirstName	MiddleName	LastName	Gender	DOB	Email
	Username	CountryCode	LocalPhone			

CUSTOMER_B

Username	Password

Branch_A

BranchID	Name	City	Street	LocalPhone	CountryCode

City

City	Country	Region

Movie_A

MovieID	Description

Movie_B

Description	Name	Duration	ReleaseDate	PGRating	ImdbRating

MOVIE_GENRE

MovID	**Genre**

MOVIE_LANGUAGE

MovID	**Language**

MOVIE_SUBTITLE

MovID	**Subtitle**

SUPPLIER_A

SupplierID	Name	CountryCode	LocalPhone

SUPPLIER_B

CountryCode	LocalPhone	Email

REVIEW

ReviewID	Date	CustID	Comment	FacilitiesR	CleanlinessR	ServiceR	AmbianceR

THEATRE

TheatreID	Theatre Type	Capacity	BrID

SCREENING

ScreeningID	Type	Date	StartTime	EndTime	ThID	MovID

SEAT

SeatID	Availability	ScreeningID

TICKET

TicketID	Price	SeID	CustID

EMPLOYEE

EmployeeID	FirstName	MiddleName	LastName	Gender	DOB	Email
Street	Building	Floor	ICEContact	BRID	EXTNB	SUPERVISORID

CITY

City	Region	Country

POSITION

Position	Salary

EMPLOYEE_PHONE

EmpID	CountryCode	LocalPhoneNumber

SCHEDULE

Date	EmpID	StartTime	EndTime

PAYMENT

PaymentID	Status	Method	Date	Amount	Discount

MEMBERSHIP

MembershipID	Type	StartDate	ExpDate
ExpDate	CUSTID		

EQUIPMENT

EquipmentID	Type	Brand	Model	LastChecked
THID	BRID			

PROVIDES

SupID	**MovID**	**EquipID**