**BUAN 6320 - Group 7**

**Technical Report**

INTRODUCTION

This Database Design Document describes the design and implementation of a database that will store data for a movie theater management system. This document was developed and written concurrently with the development of the proposed database and is intended strictly for internal use of the management and its business partners.

The purpose of this report is to create a database for a movie theater company, in turn enabling them to make data-driven decisions using a relational database management system (RDBMS). In the database, the following attributes have been defined:

* Theater
* Movie
* Ticket
* Ticket Type
* Booking Info
* Customer

With the above attributes, the company will streamline its operations and switch to a database solution rather than worry about manual data entry. Doing so dramatically reduces human error and boosts operational efficiency.

# Project Objectives

Using the data highlighted in this report, the cinema will have more detailed insights into its customers. Information about repeated visits that maximize a customer’s lifetime value can be obtained through the booking and customer information tables. Additionally, data about where customers live can provide information about potential regions where more cinemas can be established. Since demographic data like customers’ home addresses are also collected, buying habits can be predicted too. For example, people living in affluent neighborhoods may be more likely to purchase expensive tickets or visit the cinema more often, making it easier for the cinema to curate region-specific offers to increase revenue. Similarly, each customer’s purchasing habits can be tracked using the database, the movies they like and whether they are more likely to buy one or multiple tickets make it easier for the company to create campaigns that give it an edge over the competition.

# Scope of the Project

Several scripts have been created using the Structure Query Language (SQL) to create and alter the database. These scripts make it easy for the cinema to find and update the data as needed when new customers are added, more cinema locations open, new movie timings are added, or when new ticket tiers are introduced.

## Topics Covered

In this report, the following topics have been covered:

* Details about the database with information about all business rules, attributes, entities, and relationships between each entity within a table and between various attributes (foreign keys).
* Explanation of the database including a detailed entity relationship diagram.
* Data Definition Language (DDL) and Data Manipulation Language (DML) scripts.

Business Rules

1. A THEATER may show zero or many MOVIES.

2. Each MOVIE is shown at one THEATER.

3. A MOVIE may sell zero or many TICKETS.

4. Each TICKET is associated with one MOVIE.

5. A TICKET has one TICKET TYPE.

6. Each TICKET TYPE has many TICKETS.

7. A BOOKING INFO may have one or many TICKETS associated with it.

8. Each TICKET is associated with one BOOKING INFO.

9. A CUSTOMER generates one or many BOOKINGS.

10. Each BOOKING INFO is generated by one CUSTOMER.

Entity and Attribute Description

**Entity Name: THEATER**  
  
Entity Description: The THEATER table describes movie theaters with regards to location, type, and rating. It is related to the MOVIE entity.

Main Attributes of THEATER:   
THEATER\_ID: (Primary Key) A unique identifier for each theater where the movie will be played.

THEATER\_NAME: Name of the theater.

LOCATION: Geographical location of the theater.

THEATER\_TYPE: Denotes the type of theater. A theater could be a single screen or a multiplex.

THEATER\_RATING: Customer rating of the theater is used as a numerical representation of the popularity of said theater at the time. The rating is given from 1-5, 1 being the least quality of experience and 5 being the best.

**Entity Name: MOVIE**  
  
Entity Description: Next entity of our database is MOVIE, and it stores details about all the movies that are shown in theaters. It is related to both THEATER and TICKET tables.

Main Attributes of MOVIE:   
MOVIE\_ID: (Primary Key) A unique identifier for each movie shown at the theater.  
FK\_THEATER\_ID: (Foreign key) The ID of the theater where movie is shown.  
MOVIE\_NAME: Name of the movie.

DURATION: Duration/playtime of the movie.

MOVIE\_DATETIME: The date and time of the movie screening.

MOVIE\_RATING: Customer rating of the movie is used as a numerical representation of the popularity of said movie at the time. The rating is given from 1-5, 1 being the least entertaining and 5 being the most.

**Entity Name: TICKET**  
  
Entity Description: TICKET stores information about the movie and screen details and is a critical table that connects to 3 other entities; MOVIE, TICKET\_TYPE and BOOKING\_INFO.

Main Attributes of TICKET:   
TICKET\_ID: (Primary Key) A unique identifier for each movie ticket.

FK\_MOVIE\_ID: (Foreign key) The ID of the movie shown.

SEAT NO: Seat number where the customer will sit during the movie.  
FK\_BOOKING\_ID: (Foreign key) The ID of ticket booking information.

MOVIE\_TIME: The time at which the movie screening starts.

MOVIE\_DATE: The date of movie screening.

SCREEN\_NO: A theater has multiple screens. The screen number identifies at which screen is said movie shown.

FK\_TICKET\_TYPE\_ID: The ID of the ticket type.  
  
**Entity Name: TICKET\_TYPE**  
  
Entity Description: This entity in our database provides information about the various categories of tickets available for the movie. The type includes categories such as Gold, Silver or Platinum, and these types come with different benefits and perks. It connects to only the TICKET table.

Main Attributes of TICKET\_TYPE:

TICKET\_TYPE\_ID: (Primary Key) A unique identifier for each category of the ticket.

TIER: Ticket type category such as Gold, Silver, or Platinum. The Platinum tier has the most perks, followed by Gold and then Silver. Platinum ticket holders can choose seats from the top 5 rows and avail up to 15% discounts on theater snacks and drinks. They will also get discounts on movie tickets and will be able to access ticket bookings earlier than others. Similarly, Gold and Silver ticket holders will have benefits and perks lesser than those of the Platinum category.

PRICE: The price of a ticket based on ticket type.

PERKS: Description of bonuses based on ticket type, such as complimentary snacks or a 10% discount on the next ticket.

SCREEN\_TYPE: The screen type for movies is either 2D or 3D at this cinema.

**Entity Name: CUSTOMER**  
  
Entity Description: This table contains all the relevant information about all the customers attending the Movie. It relates to only one table: BOOKING\_INFO.

Main Attributes of CUSTOMER:

CUSTOMER\_ID: (Primary Key) A unique identifier for each customer who purchased a ticket.

F\_NAME: The customer’s first name.

L\_NAME: The customer's last name.

PHONE: Customer’s phone number.

EMAIL: The customer’s email address.

ADDRESS: The customer’s location.

**Entity Name: BOOKING\_INFO**  
  
Entity Description: Table that stores relevant information about the booking details of the customer. It relates to 2 tables: TICKET and CUSTOMER.

Main Attributes of BOOKING\_INFO:

BOOKING\_ID: (Primary Key) A unique identifier for each booking done by the customer.

BOOKING\_DATE: The date when the booking is made by the customer.

TIMINGS: The time at which the customer makes the booking.

PRICE: The total price of tickets and add on purchases made by the customer in one booking.

NO\_OF\_TICKETS: Number of movie tickets purchased in one booking.

FK\_CUSTOMER\_ID: (Foreign key) The ID of the customer who made the booking.

Relationship and Cardinality Description

Relationship: (shows) between THEATER and MOVIE.

Cardinality: 1:M between THEATER and MOVIE.

Business rule: A THEATER may show zero or many MOVIES, and each MOVIE is shown at one THEATER.

Relationship: (sells) between MOVIE and TICKETS.

Cardinality: 1:M relationship between MOVIE and TICKETS.

Business rule: A MOVIE may sell zero or many TICKETS, and each TICKET is associated with one MOVIE.

Relationship: (has) between TICKET and TICKET\_TYPE.

Cardinality: M:1 between TICKET and TICKET\_TYPE.

Business rule: A TICKET has one TICKET TYPE, and each TICKET TYPE has many TICKETS.

Relationship: (has) between BOOKING\_INFO and TICKET.

Cardinality: 1:M between BOOKING\_INFO and TICKET.

Business rule: A BOOKING INFO may have one or many TICKETS associated with it and each TICKET is associated with one BOOKING INFO.

Relationship: (generates) between CUSTOMER and BOOKING\_INFO.

Cardinality: 1:M between CUSTOMER and BOOKING\_INFO.

Business rule: A CUSTOMER generates one or many BOOKINGS and each BOOKING INFO is generated by one CUSTOMER.

# Key Assumptions and Special Considerations

**Assumptions**

A key business assumption being implemented in the design of the database, is that there can be zero tickets being purchased per Movie. Therefore, our database design and ERD diagrams will reflect this assumption, rather than having a one-to-many relationship. The reason for this, is that if the movie comes across a controversy and the audience decides to boycott the movie, it will not sell any tickets.

Another assumption is that a theater will not show any movies due to unforeseen circumstances such as bad weather or lockdowns.

**Design Decisions**

The key factors influencing design decisions was that it was decided to keep the number of entities (tables) to a manageable number of 6. This was a decision made in part to ensure overcomplexity would not be a problem for this relatively small movie theater business, but also to help with the overall efficiency of the database. While some entities, such as Reviews and Location were ruled out, other potential entities were combined. For example, the table named “Booking Info” was created to take the place of both the “Ticket Booking Info” and “Customer Order Details” entities. Therefore, an imperative in our design process was to create something simple, manageable, effective, but overwhelmingly practical for this business to use for years to come.

Entity Relationship Diagram

An entity relationship diagram (ERD) has been constructed below.

Diagram, schematic

Description automatically generated

# Database Construction

The next step is creating the databases, which was done with the code below:

# Drop tables to clear previous run

-- Triggers

DROP TRIGGER TRG\_BOOKING\_INFO;

DROP TRIGGER TRG\_CUSTOMER;

DROP TRIGGER TRG\_MOVIE;

DROP TRIGGER TRG\_THEATER;

DROP TRIGGER TRG\_TICKET;

DROP TRIGGER TRG\_TICKET\_TYPE;

-- Sequences

DROP SEQUENCE SEQ\_BOOKING\_INFO\_BOOKING\_ID;

DROP SEQUENCE SEQ\_CUSTOMER\_CUSTOMER\_ID;

DROP SEQUENCE SEQ\_MOVIE\_MOVIE\_ID;

DROP SEQUENCE SEQ\_THEATER\_THEATER\_ID;

DROP SEQUENCE SEQ\_TICKET\_TICKET\_ID;

-- Views

DROP VIEW CUSTOMER\_INFO;

DROP VIEW MOVIE\_INFO;

DROP VIEW THEATER\_INFO;

DROP VIEW TICKET\_TYPE\_INFO;

-- Indices

DROP INDEX IDX\_THEATER\_theater\_id;

DROP INDEX IDX\_MOVIE\_movie\_id;

DROP INDEX IDX\_MOVIE\_theater\_id\_FK;

DROP INDEX IDX\_CUSTOMER\_customer\_id;

DROP INDEX IDX\_BOOKING\_INFO\_booking\_id;

DROP INDEX IDX\_BOOKING\_INFO\_customer\_id\_FK;

DROP INDEX IDX\_BOOKING\_INFO\_price;

DROP INDEX IDX\_TICKET\_ticket\_id;

DROP INDEX IDX\_TICKET\_movie\_id\_FK;

DROP INDEX IDX\_TICKET\_booking\_id\_FK;

DROP INDEX IDX\_TICKET\_ticket\_type\_id\_FK;

DROP INDEX IDX\_TICKET\_TYPE\_ticket\_type\_id;

DROP INDEX IDX\_TICKET\_TYPE\_tier;

-- Tables

DROP TABLE TICKET\_TYPE;

DROP TABLE TICKET;

DROP TABLE BOOKING\_INFO;

DROP TABLE CUSTOMER;

DROP TABLE MOVIE;

DROP TABLE THEATER;

## Creating tables

CREATE TABLE THEATER

(

theater\_id INTEGER NOT NULL,

theater\_name VARCHAR(50) NOT NULL,

location VARCHAR(50) NOT NULL,

theater\_type VARCHAR(50) NOT NULL,

theater\_rating FLOAT,

CONSTRAINT PK\_theater\_id PRIMARY KEY (theater\_id)

);

CREATE TABLE MOVIE

(

movie\_id INTEGER NOT NULL,

theater\_id INTEGER NOT NULL,

movie\_name VARCHAR(100) NOT NULL,

duration INTEGER NOT NULL,

movie\_datetime TIMESTAMP NOT NULL,

movie\_rating FLOAT,

CONSTRAINT PK\_movie\_id PRIMARY KEY (movie\_id),

CONSTRAINT FK\_theater\_id FOREIGN KEY (theater\_id) REFERENCES THEATER

);

CREATE TABLE CUSTOMER

(

customer\_id INTEGER NOT NULL,

f\_name VARCHAR(100) NOT NULL,

l\_name VARCHAR(100) NOT NULL,

phone VARCHAR(30) NOT NULL,

email VARCHAR(30) NOT NULL,

address VARCHAR(100) NOT NULL,

CONSTRAINT PK\_customer\_id PRIMARY KEY (customer\_id)

);

CREATE TABLE BOOKING\_INFO

(

booking\_id INTEGER NOT NULL,

booking\_date DATE NOT NULL,

timings TIMESTAMP NOT NULL,

price INTEGER NOT NULL,

no\_of\_tickets INTEGER NOT NULL,

customer\_id INTEGER NOT NULL,

CONSTRAINT PK\_booking\_id PRIMARY KEY (booking\_id),

CONSTRAINT FK\_customer\_id FOREIGN KEY (customer\_id) REFERENCES CUSTOMER

);

CREATE TABLE TICKET

(

ticket\_id INTEGER NOT NULL,

movie\_id INTEGER NOT NULL,

seat\_no VARCHAR(30) NOT NULL,

booking\_id INTEGER NOT NULL,

movie\_time TIMESTAMP NOT NULL,

movie\_date DATE NOT NULL,

screen\_no INTEGER NOT NULL,

ticket\_type\_id INTEGER NOT NULL,

CONSTRAINT PK\_ticket\_id PRIMARY KEY (ticket\_id),

CONSTRAINT FK\_movie\_id FOREIGN KEY (movie\_id) REFERENCES MOVIE,

CONSTRAINT FK\_booking\_id FOREIGN KEY (booking\_id) REFERENCES BOOKING\_INFO

);

CREATE TABLE TICKET\_TYPE

(

ticket\_type\_id INTEGER NOT NULL,

tier VARCHAR(30) NOT NULL,

price FLOAT NOT NULL,

perks VARCHAR(100),

screen\_type VARCHAR(30) NOT NULL,

CONSTRAINT PK\_ticket\_type\_id PRIMARY KEY (ticket\_type\_id)

);

## Creating Index

CREATE INDEX IDX\_THEATER\_theater\_id ON THEATER (theater\_id);

CREATE INDEX IDX\_MOVIE\_movie\_id ON MOVIE (movie\_id);

CREATE INDEX IDX\_MOVIE\_theater\_id\_FK ON MOVIE (theater\_id);

CREATE INDEX IDX\_CUSTOMER\_customer\_id ON CUSTOMER (customer\_id);

CREATE INDEX IDX\_BOOKING\_INFO\_booking\_id ON BOOKING\_INFO (booking\_id);

CREATE INDEX IDX\_BOOKING\_INFO\_customer\_id\_FK ON BOOKING\_INFO (customer\_id);

CREATE INDEX IDX\_BOOKING\_INFO\_price ON BOOKING\_INFO (price);

CREATE INDEX IDX\_TICKET\_ticket\_id ON TICKET (ticket\_id);

CREATE INDEX IDX\_TICKET\_movie\_id\_FK ON TICKET (movie\_id);

CREATE INDEX IDX\_TICKET\_booking\_id\_FK ON TICKET (booking\_id);

CREATE INDEX IDX\_TICKET\_ticket\_type\_id\_FK ON TICKET (ticket\_type\_id);

CREATE INDEX IDX\_TICKET\_TYPE\_ticket\_type\_id ON TICKET\_TYPE (ticket\_type\_id);

CREATE INDEX IDX\_TICKET\_TYPE\_tier ON TICKET\_TYPE (tier);

## Updating Tables

ALTER TABLE THEATER ADD (

created\_by VARCHAR2(30),

date\_created DATE,

modified\_by VARCHAR2(30),

date\_modified DATE

);

ALTER TABLE MOVIE ADD (

created\_by VARCHAR2(30),

date\_created DATE,

modified\_by VARCHAR2(30),

date\_modified DATE

);

ALTER TABLE CUSTOMER ADD (

created\_by VARCHAR2(30),

date\_created DATE,

modified\_by VARCHAR2(30),

date\_modified DATE

);

ALTER TABLE BOOKING\_INFO ADD (

created\_by VARCHAR2(30),

date\_created DATE,

modified\_by VARCHAR2(30),

date\_modified DATE

);

ALTER TABLE TICKET ADD (

created\_by VARCHAR2(30),

date\_created DATE,

modified\_by VARCHAR2(30),

date\_modified DATE

);

ALTER TABLE TICKET\_TYPE ADD (

created\_by VARCHAR2(30),

date\_created DATE,

modified\_by VARCHAR2(30),

date\_modified DATE

);

## Creating Views

CREATE OR REPLACE VIEW THEATER\_INFO AS

SELECT THEATER\_ID,THEATER\_NAME,LOCATION,THEATER\_RATING

FROM THEATER;

CREATE OR REPLACE VIEW MOVIE\_INFO AS

SELECT MOVIE\_ID,MOVIE\_NAME,DURATION,MOVIE\_RATING

FROM MOVIE;

CREATE OR REPLACE VIEW TICKET\_TYPE\_INFO AS

SELECT TICKET\_TYPE\_ID,TIER,PRICE,PERKS

FROM TICKET\_TYPE;

CREATE OR REPLACE VIEW CUSTOMER\_INFO AS

SELECT CUSTOMER\_ID,F\_NAME || ' ' || L\_NAME as CUSTOMER\_NAME,PHONE,EMAIL,ADDRESS

FROM CUSTOMER;

## Creating Sequences

CREATE SEQUENCE SEQ\_THEATER\_THEATER\_ID

INCREMENT BY 1

START WITH 0

NOMAXVALUE

MINVALUE 0

NOCACHE;

CREATE SEQUENCE SEQ\_MOVIE\_MOVIE\_ID

INCREMENT BY 1

START WITH 0

NOMAXVALUE

MINVALUE 0

NOCACHE;

CREATE SEQUENCE SEQ\_TICKET\_TICKET\_ID

INCREMENT BY 1

START WITH 0

NOMAXVALUE

MINVALUE 0

NOCACHE;

CREATE SEQUENCE SEQ\_BOOKING\_INFO\_BOOKING\_ID

INCREMENT BY 1

START WITH 0

NOMAXVALUE

MINVALUE 0

NOCACHE;

CREATE SEQUENCE SEQ\_CUSTOMER\_CUSTOMER\_ID

INCREMENT BY 1

START WITH 0

NOMAXVALUE

MINVALUE 0

NOCACHE;

## Creating Triggers

CREATE OR REPLACE TRIGGER TRG\_CUSTOMER

BEFORE INSERT OR UPDATE ON CUSTOMER

FOR EACH ROW

BEGIN

IF INSERTING THEN

IF :NEW.CUSTOMER\_ID IS NULL THEN

:NEW.CUSTOMER\_ID := SEQ\_CUSTOMER\_CUSTOMER\_ID.NEXTVAL;

END IF;

IF :NEW.CREATED\_BY IS NULL THEN

:NEW.CREATED\_BY := USER;

END IF;

IF :NEW.DATE\_CREATED IS NULL THEN

:NEW.DATE\_CREATED := SYSDATE;

END IF;

END IF;

IF INSERTING OR UPDATING THEN

:NEW.MODIFIED\_BY := USER;

:NEW.DATE\_MODIFIED := SYSDATE;

END IF;

END;

/

CREATE OR REPLACE TRIGGER TRG\_BOOKING\_INFO

BEFORE INSERT OR UPDATE ON BOOKING\_INFO

FOR EACH ROW

BEGIN

IF INSERTING THEN

IF :NEW.BOOKING\_ID IS NULL THEN

:NEW.BOOKING\_ID := SEQ\_BOOKING\_INFO\_BOOKING\_ID.NEXTVAL;

END IF;

IF :NEW.CREATED\_BY IS NULL THEN

:NEW.CREATED\_BY := USER;

END IF;

IF :NEW.DATE\_CREATED IS NULL THEN

:NEW.DATE\_CREATED := SYSDATE;

END IF;

END IF;

IF INSERTING OR UPDATING THEN

:NEW.MODIFIED\_BY := USER;

:NEW.DATE\_MODIFIED := SYSDATE;

END IF;

END;

/

CREATE OR REPLACE TRIGGER TRG\_MOVIE

BEFORE INSERT OR UPDATE ON MOVIE

FOR EACH ROW

BEGIN

IF INSERTING THEN

IF :NEW.MOVIE\_ID IS NULL THEN

:NEW.MOVIE\_ID := SEQ\_MOVIE\_MOVIE\_ID.NEXTVAL;

END IF;

IF :NEW.CREATED\_BY IS NULL THEN

:NEW.CREATED\_BY := USER;

END IF;

IF :NEW.DATE\_CREATED IS NULL THEN

:NEW.DATE\_CREATED := SYSDATE;

END IF;

END IF;

IF INSERTING OR UPDATING THEN

:NEW.MODIFIED\_BY := USER;

:NEW.DATE\_MODIFIED := SYSDATE;

END IF;

END;

/

CREATE OR REPLACE TRIGGER TRG\_THEATER

BEFORE INSERT OR UPDATE ON THEATER

FOR EACH ROW

BEGIN

IF INSERTING THEN

IF :NEW.THEATER\_ID IS NULL THEN

:NEW.THEATER\_ID := SEQ\_THEATER\_THEATER\_ID.NEXTVAL;

END IF;

IF :NEW.CREATED\_BY IS NULL THEN

:NEW.CREATED\_BY := USER;

END IF;

IF :NEW.DATE\_CREATED IS NULL THEN

:NEW.DATE\_CREATED := SYSDATE;

END IF;

END IF;

IF INSERTING OR UPDATING THEN

:NEW.MODIFIED\_BY := USER;

:NEW.DATE\_MODIFIED := SYSDATE;

END IF;

END;

/

CREATE OR REPLACE TRIGGER TRG\_TICKET

BEFORE INSERT OR UPDATE ON TICKET

FOR EACH ROW

BEGIN

IF INSERTING THEN

IF :NEW.TICKET\_ID IS NULL THEN

:NEW.TICKET\_ID := SEQ\_TICKET\_TICKET\_ID.NEXTVAL;

END IF;

IF :NEW.CREATED\_BY IS NULL THEN

:NEW.CREATED\_BY := USER;

END IF;

IF :NEW.DATE\_CREATED IS NULL THEN

:NEW.DATE\_CREATED := SYSDATE;

END IF;

END IF;

IF INSERTING OR UPDATING THEN

:NEW.MODIFIED\_BY := USER;

:NEW.DATE\_MODIFIED := SYSDATE;

END IF;

END;

/

CREATE OR REPLACE TRIGGER TRG\_TICKET\_TYPE

BEFORE INSERT OR UPDATE ON TICKET\_TYPE

FOR EACH ROW

BEGIN

IF INSERTING THEN

IF :NEW.CREATED\_BY IS NULL THEN

:NEW.CREATED\_BY := USER;

END IF;

IF :NEW.DATE\_CREATED IS NULL THEN

:NEW.DATE\_CREATED := SYSDATE;

END IF;

END IF;

IF INSERTING OR UPDATING THEN

:NEW.MODIFIED\_BY := USER;

:NEW.DATE\_MODIFIED := SYSDATE;

END IF;

END;

/

-- Check the DBMS data dictionary to make sure that all objects have been created successfully

SELECT TABLE\_NAME FROM USER\_TABLES;

SELECT OBJECT\_NAME, STATUS, CREATED, LAST\_DDL\_TIME FROM USER\_OBJECTS;

## Insert Data

--THEATER

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (1,'Cinemax', 'Dallas', 'Single Screen', 4.2);

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (2, 'Carnival', 'Richardson', 'Single Screen', 3.7);

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (3, 'Movietime', 'Fort Worth', 'Multiplex', 3.7);

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (4, 'INOX', 'Houston', 'Multiplex', 4.9);

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (5, 'MovieMax', 'San Antonio', 'Multiplex', 4.2);

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (6, 'Regal', 'Plano', 'Multiplex',null);

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (7, 'Cinepolis', 'Austin', 'Multiplex', 4.8);

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (8, 'Metro', 'Austin', 'Multiplex', 4.7);

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (9, 'PVR', 'Dallas', 'Multiplex', 4.2);

INSERT INTO THEATER (THEATER\_ID, THEATER\_NAME, LOCATION, THEATER\_TYPE, THEATER\_RATING)

VALUES (10, 'CineFun', 'El Paso', 'Single Screen', 3.8);

--MOVIE

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (1, 2, 'Spiderman: No Way Home', 120, TO\_TIMESTAMP('2022-07-02 06:15:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 4.9);

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (2, 3, 'Batman', 120, TO\_TIMESTAMP('2022-08-14 08:50:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 4.9);

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (3, 10, 'Halloween Ends', 110,TO\_TIMESTAMP('2022-08-22 05:45:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 5.0);

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (4, 7, 'Top Gun: Maverick', 130,TO\_TIMESTAMP('2022-10-02 12:15:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 4.7);

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (5, 6, 'Wild Is the Wind', 125, TO\_TIMESTAMP('2022-04-27 03:50:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 2.6);

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (6, 9, 'Spirited', 120, TO\_TIMESTAMP('2022-05-20 19:15:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), null);

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (7, 5, 'Elvis', 160, TO\_TIMESTAMP('2022-07-03 07:25:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 4.2);

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (8, 1, 'Glass Onion: A Knives Out Mystery', 130, TO\_TIMESTAMP('2022-07-14 17:20:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 4.5);

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (9, 1, 'One Piece Film: Red', 115, TO\_TIMESTAMP('2022-07-02 06:15:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 4.2);

INSERT INTO movie (MOVIE\_ID, THEATER\_ID, MOVIE\_NAME, DURATION, MOVIE\_DATETIME, MOVIE\_RATING)

VALUES (10, 4, 'Black Panther: Wakanda Forever', 120, TO\_TIMESTAMP('2022-01-01 12:00:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 5);

--CUSTOMER

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (1, 'Jim', 'Allen', 'jallen@email.com', '(218)867-4128', '870 Waldeck Street');

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (2, 'Kate', 'Kerry', 'kk@gmail.com', '(405)349-7761', '4593 Fancher Drive');

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (3, 'Ben', 'Jackson', 'bjack@emsil.com', '(302)219-1780', '750 Synergy Park Blvd');

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (4, 'Jane', 'Nick', 'jane@email.com', '(214)799-8502', '1620 Star Route');

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (5, 'Alex', 'Hale', 'alhale@email.com', '(830)697-6409', '1284 Carolyns Circle');

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (6, 'Bob', 'Jade', 'jade.bob@email.com', '(224)442-7254','1721 Wines Lane');

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (7, 'Matthew', 'Moran', 'mj@email.com', '(334)252-8867', '425 Romines Mill Road');

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (8, 'Jack', 'Black', 'jack@email.com', '(580)837-6871', '4990 Sardis Sta');

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (9, 'Preston', 'Billingsley', 'prestonbil@email.com', '(270)920-7678', '2179 Bell Street');

INSERT INTO CUSTOMER (CUSTOMER\_ID, F\_NAME, L\_NAME, EMAIL, PHONE, ADDRESS)

VALUES (10, 'Nick', 'Campise', 'nickcampise@email.com', '(270)833-9967', '3090 Moore Avenue');

--BOOKING\_INFO

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (1, TO\_DATE('2022-07-01','YYYY-MM-DD') ,TO\_TIMESTAMP('2022-07-01 06:15:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 10, 2, 1);

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (2, TO\_DATE('2022-08-20','YYYY-MM-DD'),TO\_TIMESTAMP('2022-08-20 05:45:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 10, 2, 10);

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (3, TO\_DATE('2022-04-27','YYYY-MM-DD'),TO\_TIMESTAMP('2022-04-27 01:50:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 15, 3, 2);

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (4, TO\_DATE('2022-06-03','YYYY-MM-DD'), TO\_TIMESTAMP('2022-06-03 07:25:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'),20, 4, 3);

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (5, TO\_DATE('2022-07-02','YYYY-MM-DD'), TO\_TIMESTAMP('2022-07-02 07:25:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 35, 7, 4);

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (6, TO\_DATE('2022-08-10','YYYY-MM-DD'), TO\_TIMESTAMP('2022-08-10 07:25:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 10, 2, 5);

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (7, TO\_DATE('2021-12-23','YYYY-MM-DD'), TO\_TIMESTAMP('2021-12-23 07:25:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 25, 5, 6);

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (8, TO\_DATE('2022-10-01','YYYY-MM-DD'), TO\_TIMESTAMP('2022-10-01 07:25:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 15, 3, 7);

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (9, TO\_DATE('2022-04-26','YYYY-MM-DD'), TO\_TIMESTAMP('2022-04-26 07:25:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 35, 7, 8);

INSERT INTO BOOKING\_INFO (BOOKING\_ID, BOOKING\_DATE, TIMINGS, PRICE, NO\_OF\_TICKETS, CUSTOMER\_ID)

VALUES (10, TO\_DATE('2022-07-09','YYYY-MM-DD'), TO\_TIMESTAMP('2022-07-09 07:25:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), 10, 5, 9);

--TICKET\_TYPE

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (1, 'Platinum I', 20,'Best Seats(A-D), Early Access, 15% discount on tickets, 10% discount on food and beverages', '3D/2D Recliner');

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (2, 'Platinum II',18,'Best Seats(A-F), Early Access, 10% discount on tickets, 7% discount on food and beverages', '2D Recliner');

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (3, 'Platinum III',16,'Best Seats(A-I), Early Access, 5% discount on tickets, 3% discount on food and beverages', '2D Recliner');

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (4, 'Gold I', 15,'Middle Row, 5% discount on tickets, Medium Soda free with Large popcorn','3D/2D Premium Glider');

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (5, 'Gold II', 13.75,'Middle Row, 3% discount on tickets, Small Soda free with Large popcorn','2D Premium Glider');

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (6, 'Gold III', 12,'Middle Row, 3% discount on tickets','2D Premium Glider');

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (7, 'Silver I', 10,'Lower Middle Row, 5% discount on orders above 20$','3D/2D Rocker');

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (8, 'Silver II', 8.5,'Lower Middle Row, 5% discount on orders above 30$','2D Rocker');

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (9, 'Silver III', 7,'Lower Middle Row, 5% discount on orders above 40$','2D Rocker');

INSERT INTO TICKET\_TYPE (TICKET\_TYPE\_ID, TIER, PRICE, PERKS, SCREEN\_TYPE)

VALUES (10,'Basic', 5,'No Priority','3D/2D');

--TICKET

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (1, 1, '20F', 1,TO\_TIMESTAMP('2022-07-02 06:15:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-07-02','YYYY-MM-DD'), 3,1);

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (2, 3, '20G', 2,TO\_TIMESTAMP('2022-08-22 05:45:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-08-22','YYYY-MM-DD'), 3,6);

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (3, 5, '11S', 3,TO\_TIMESTAMP('2022-04-27 03:50:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-04-27','YYYY-MM-DD'), 5,9);

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (4, 7, '24A', 4, TO\_TIMESTAMP('2022-07-03 07:25:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-07-03','YYYY-MM-DD'), 1,1);

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (5, 9, '32E', 5,TO\_TIMESTAMP('2022-07-02 06:15:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-07-02','YYYY-MM-DD'), 2,6);

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (6, 2, '18F', 6, TO\_TIMESTAMP('2022-08-14 08:50:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-08-14','YYYY-MM-DD'), 1,5);

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (7, 10, '14A',7, TO\_TIMESTAMP('2022-01-01 12:00:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-01-01','YYYY-MM-DD'), 3,2);

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (8, 4, '14B', 8,TO\_TIMESTAMP('2022-10-02 12:15:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-10-02','YYYY-MM-DD'), 3,1);

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (9, 6, '8C', 9, TO\_TIMESTAMP('2022-04-27 03:50:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-04-27','YYYY-MM-DD'), 2,3);

INSERT INTO TICKET (TICKET\_ID, MOVIE\_ID, SEAT\_NO, BOOKING\_ID, MOVIE\_TIME, MOVIE\_DATE, SCREEN\_NO,TICKET\_TYPE\_ID)

VALUES (10, 8, '16E', 10, TO\_TIMESTAMP('2022-07-14 17:20:00.742000000', 'YYYY-MM-DD HH24:MI:SS.FF'), TO\_DATE('2022-07-14','YYYY-MM-DD'), 1,3);

## Basic Queries

**Query 1: Select all columns and all rows from one table**

select \*

from THEATER;

Graphical user interface, application, table

Description automatically generated

**Query 2: Select five columns and all rows from one table**

SELECT movie\_id, movie\_name, duration, movie\_datetime, movie\_rating

from MOVIE;

Graphical user interface, text

Description automatically generated with medium confidence

**Query 3: Select all columns from all rows from one view**

select \*

from CUSTOMER\_INFO;

Graphical user interface, text, application

Description automatically generated

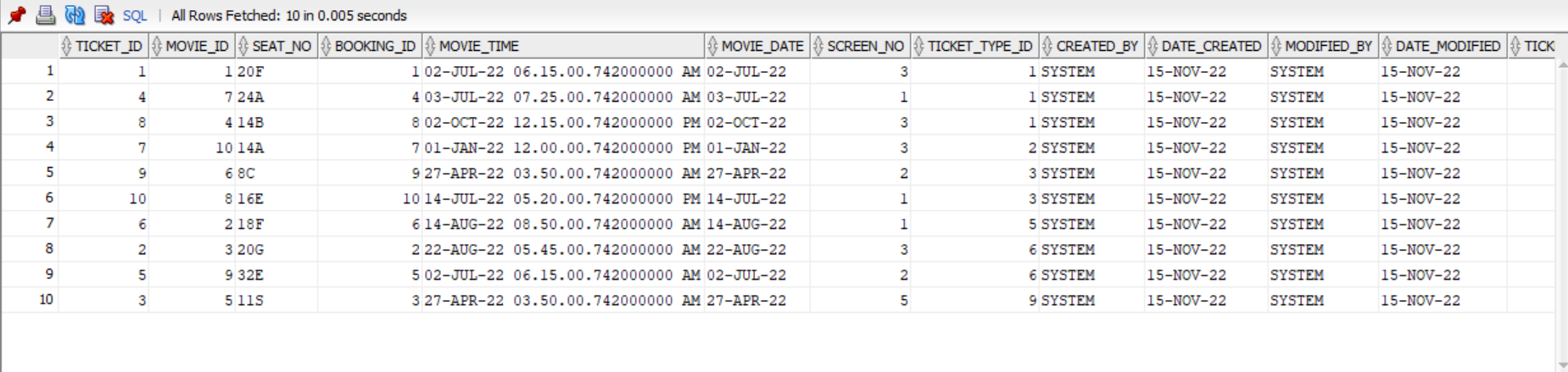
**Query 4: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product**

select \*

from TICKET

left outer join TICKET\_TYPE

on TICKET.ticket\_type\_id = TICKET\_TYPE.ticket\_type\_id;



**Query 5: Select and order data retrieved from one table**

Select \*

from MOVIE

order by MOVIE\_RATING desc nulls last;

Text

Description automatically generated

**Query 6: Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows**

select F\_name || ' ' || L\_name as Customer\_name, T.Seat\_No, T.Movie\_Time,

T.Screen\_No, B.Price

from Customer C

left join BOOKING\_INFO B

on C.Customer\_id = B.Customer\_id

left join TICKET T

on T.Booking\_id = B.Booking\_id

fetch first 10 rows only;

Text, table

Description automatically generated with medium confidence

**Query 7: Select distinct rows using joins on 3 tables**

select distinct THEATER\_NAME, LOCATION

from THEATER T

left join MOVIE M

on T.THEATER\_ID = M.THEATER\_ID

left join TICKET TC

on TC.MOVIE\_ID = M.MOVIE\_ID

where DURATION>120 and movie\_date > '27-APR-22';

Graphical user interface, table

Description automatically generated

**Query 8: Use GROUP BY and HAVING in a select statement using one or more tables**

select T.THEATER\_NAME, M.MOVIE\_NAME, AVG(M.MOVIE\_RATING) AVG\_RATING

from THEATER T

left join MOVIE M

on T.THEATER\_ID=M.THEATER\_ID

group by T.THEATER\_NAME, M.MOVIE\_NAME

having AVG(M.MOVIE\_RATING)>3;

Graphical user interface, text, application, email

Description automatically generated

**Query 9: Use IN clause to select data from one or more tables**

select \*

from THEATER

where LOCATION in ('Dallas','Austin');

Graphical user interface, table

Description automatically generated with medium confidence

**Query 10: Select length of one column from one table (use LENGTH function)**

select MOVIE\_NAME, LENGTH(Movie\_Name) LENGTH

from MOVIE

group by MOVIE\_NAME;

Graphical user interface, text, application

Description automatically generated

**Query 11: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed**

alter table MOVIE

DISABLE constraint FK\_THEATER\_ID; --disable foreign key constraint to be able to delete data

select \* from theater;

delete from theater

where theater\_rating is null;

select \* from theater;

ROLLBACK;

**Graphical user interface, text, application, email

Description automatically generated**

**Table

Description automatically generated with medium confidence**

**Graphical user interface, table

Description automatically generated**

**Query 12: Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed**

select \* from MOVIE;

update MOVIE

set MOVIE\_NAME = 'Batman Returns'

where movie\_id = 2;

select \* from MOVIE;

rollback;

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

**Perform 8 Additional Advanced Queries**

**Query 13: List name, email, phone of customers along with the tier and perks of the tickets they purchased**

select f\_name || ' ' || l\_name customer\_name, phone, email, tt.tier, perks

from customer c

left join booking\_info b

on c.customer\_id=b.customer\_id

left join ticket t

on b.booking\_id = t.booking\_id

left join ticket\_type tt

on tt.ticket\_type\_id = t.ticket\_type\_id;

Graphical user interface, text

Description automatically generated

**Query 14: List all theater names, their location and their average rating of theater types 'Multiplex' with average rating greater than 4. Order these by average theater rating from highest to last**

select THEATER\_NAME, LOCATION, AVG(THEATER\_RATING)

from THEATER

where THEATER\_TYPE = 'Multiplex'

group by THEATER\_NAME, LOCATION

having avg(THEATER\_RATING) > 4

order by avg(THEATER\_RATING) desc;

Graphical user interface, table

Description automatically generated

**Query 15: List movie names, duration, movie\_rating, price, perks and screen type of movie played in 3D**

select M.MOVIE\_NAME, DURATION, MOVIE\_RATING, PRICE, PERKS, SCREEN\_TYPE

from MOVIE M

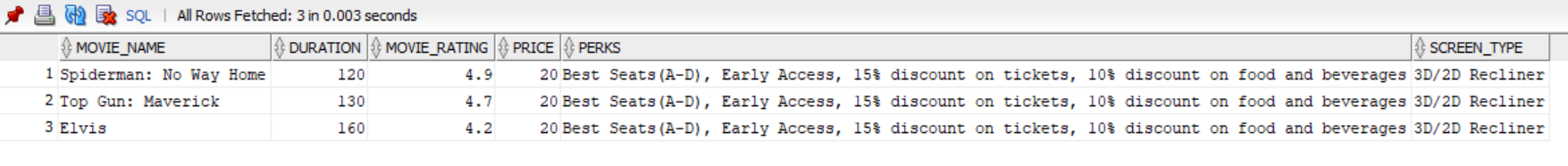
left join TICKET T

on M.movie\_id = T.movie\_id

left join TICKET\_TYPE TT

on T.ticket\_type\_id = TT.ticket\_type\_id

where SCREEN\_TYPE like '%3D%';



**Query 16: List top 3 theaters by rating and the number of movies played at these theaters. Display the name, rating and count of movies**

select t.THEATER\_NAME,THEATER\_RATING,count(m.MOVIE\_ID) as MOVIE\_COUNT

from THEATER t

left join MOVIE m

on t.THEATER\_ID = m.THEATER\_ID

group by t.THEATER\_NAME, THEATER\_RATING

order by THEATER\_RATING desc nulls last

fetch first 3 rows only;

Table

Description automatically generated

**Query 17: List all theater information for theaters that sell type Platinum I tickets**

select \* from THEATER

where theater\_id in ( select theater\_id from MOVIE

where movie\_id in (select movie\_id from ticket

where ticket\_type\_id in (select ticket\_type\_id from ticket\_type where tier='Platinum I')

)

);

Application, table

Description automatically generated

**Query 18: List movie name, duration, movie rating and movie date of all movies played after 2nd July 2022. Order by rating from highest to least**

select MOVIE\_NAME, DURATION, MOVIE\_RATING, MOVIE\_DATETIME

from movie

where MOVIE\_DATETIME > '2 jul 2022'

order by MOVIE\_RATING desc;

Graphical user interface, table

Description automatically generated with medium confidence

**Query 19: List customer names who have not spent less than 15$ while booking**

select F\_NAME || ' ' || L\_NAME CUSTOMER\_NAME

from CUSTOMER

where CUSTOMER\_ID not in ( select CUSTOMER\_ID from BOOKING\_INFO where PRICE <15)

order by F\_NAME,L\_NAME;

Graphical user interface, application, table

Description automatically generated

**Query 20: List all tiers and the count of tickets bought. Count should be greater than 0**

select tt.tier, count(ticket\_id) ticket\_count

from TICKET\_TYPE tt

left join TICKET t

on tt.ticket\_type\_id=t.ticket\_type\_id

group by tt.tier

having count(ticket\_id)<>0

order by ticket\_count desc;

Graphical user interface, application

Description automatically generated

# DDL & DML Query Outputs

Trigger TRG\_BOOKING\_INFO dropped.

Trigger TRG\_CUSTOMER dropped.

Trigger TRG\_MOVIE dropped.

Trigger TRG\_THEATER dropped.

Trigger TRG\_TICKET dropped.

Trigger TRG\_TICKET\_TYPE dropped.

Sequence SEQ\_BOOKING\_INFO\_BOOKING\_ID dropped.

Sequence SEQ\_CUSTOMER\_CUSTOMER\_ID dropped.

Sequence SEQ\_MOVIE\_MOVIE\_ID dropped.

Sequence SEQ\_THEATER\_THEATER\_ID dropped.

Sequence SEQ\_TICKET\_TICKET\_ID dropped.

View CUSTOMER\_INFO dropped.

View MOVIE\_INFO dropped.

View THEATER\_INFO dropped.

View TICKET\_TYPE\_INFO dropped.

Error starting at line : 29 in command -

DROP INDEX IDX\_THEATER\_theater\_id

Error report -

ORA-01418: specified index does not exist

01418. 00000 - "specified index does not exist"

\*Cause:

\*Action:

Error starting at line : 31 in command -

DROP INDEX IDX\_MOVIE\_movie\_id

Error report -

ORA-01418: specified index does not exist

01418. 00000 - "specified index does not exist"

\*Cause:

\*Action:

Index IDX\_MOVIE\_THEATER\_ID\_FK dropped.

Error starting at line : 34 in command -

DROP INDEX IDX\_CUSTOMER\_customer\_id

Error report -

ORA-01418: specified index does not exist

01418. 00000 - "specified index does not exist"

\*Cause:

\*Action:

Error starting at line : 36 in command -

DROP INDEX IDX\_BOOKING\_INFO\_booking\_id

Error report -

ORA-01418: specified index does not exist

01418. 00000 - "specified index does not exist"

\*Cause:

\*Action:

Index IDX\_BOOKING\_INFO\_CUSTOMER\_ID\_FK dropped.

Index IDX\_BOOKING\_INFO\_PRICE dropped.

Error starting at line : 40 in command -

DROP INDEX IDX\_TICKET\_ticket\_id

Error report -

ORA-01418: specified index does not exist

01418. 00000 - "specified index does not exist"

\*Cause:

\*Action:

Index IDX\_TICKET\_MOVIE\_ID\_FK dropped.

Index IDX\_TICKET\_BOOKING\_ID\_FK dropped.

Index IDX\_TICKET\_TICKET\_TYPE\_ID\_FK dropped.

Error starting at line : 45 in command -

DROP INDEX IDX\_TICKET\_TYPE\_ticket\_type\_id

Error report -

ORA-01418: specified index does not exist

01418. 00000 - "specified index does not exist"

\*Cause:

\*Action:

Index IDX\_TICKET\_TYPE\_TIER dropped.

Table TICKET\_TYPE dropped.

Table TICKET dropped.

Table BOOKING\_INFO dropped.

Table CUSTOMER dropped.

Table MOVIE dropped.

Table THEATER dropped.

Table THEATER created.

Table MOVIE created.

Table CUSTOMER created.

Table BOOKING\_INFO created.

Table TICKET created.

Table TICKET\_TYPE created.

Error starting at line : 130 in command -

CREATE INDEX IDX\_THEATER\_theater\_id ON THEATER (theater\_id)

Error report -

ORA-01408: such column list already indexed

01408. 00000 - "such column list already indexed"

\*Cause:

\*Action:

Error starting at line : 134 in command -

CREATE INDEX IDX\_MOVIE\_movie\_id ON MOVIE (movie\_id)

Error report -

ORA-01408: such column list already indexed

01408. 00000 - "such column list already indexed"

\*Cause:

\*Action:

Index IDX\_MOVIE\_THEATER\_ID\_FK created.

Error starting at line : 140 in command -

CREATE INDEX IDX\_CUSTOMER\_customer\_id ON CUSTOMER (customer\_id)

Error report -

ORA-01408: such column list already indexed

01408. 00000 - "such column list already indexed"

\*Cause:

\*Action:

Error starting at line : 144 in command -

CREATE INDEX IDX\_BOOKING\_INFO\_booking\_id ON BOOKING\_INFO (booking\_id)

Error report -

ORA-01408: such column list already indexed

01408. 00000 - "such column list already indexed"

\*Cause:

\*Action:

Index IDX\_BOOKING\_INFO\_CUSTOMER\_ID\_FK created.

Index IDX\_BOOKING\_INFO\_PRICE created.

Error starting at line : 152 in command -

CREATE INDEX IDX\_TICKET\_ticket\_id ON TICKET (ticket\_id)

Error report -

ORA-01408: such column list already indexed

01408. 00000 - "such column list already indexed"

\*Cause:

\*Action:

Index IDX\_TICKET\_MOVIE\_ID\_FK created.

Index IDX\_TICKET\_BOOKING\_ID\_FK created.

Index IDX\_TICKET\_TICKET\_TYPE\_ID\_FK created.

Error starting at line : 160 in command -

CREATE INDEX IDX\_TICKET\_TYPE\_ticket\_type\_id ON TICKET\_TYPE (ticket\_type\_id)

Error report -

ORA-01408: such column list already indexed

01408. 00000 - "such column list already indexed"

\*Cause:

\*Action:

Index IDX\_TICKET\_TYPE\_TIER created.

Table THEATER altered.

Table MOVIE altered.

Table CUSTOMER altered.

Table BOOKING\_INFO altered.

Table TICKET altered.

Table TICKET\_TYPE altered.

View THEATER\_INFO created.

View MOVIE\_INFO created.

View TICKET\_TYPE\_INFO created.

View CUSTOMER\_INFO created.

Sequence SEQ\_THEATER\_THEATER\_ID created.

Sequence SEQ\_MOVIE\_MOVIE\_ID created.

Sequence SEQ\_TICKET\_TICKET\_ID created.

Sequence SEQ\_BOOKING\_INFO\_BOOKING\_ID created.

Sequence SEQ\_CUSTOMER\_CUSTOMER\_ID created.

Trigger TRG\_CUSTOMER compiled

Trigger TRG\_BOOKING\_INFO compiled

Trigger TRG\_MOVIE compiled

Trigger TRG\_THEATER compiled

Trigger TRG\_TICKET compiled

Trigger TRG\_TICKET\_TYPE compiled

>>Query Run In:Query Result

>>Query Run In:Query Result 1

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.