Movie Ticket Booking System

Directed by

ROHIT GHARAL

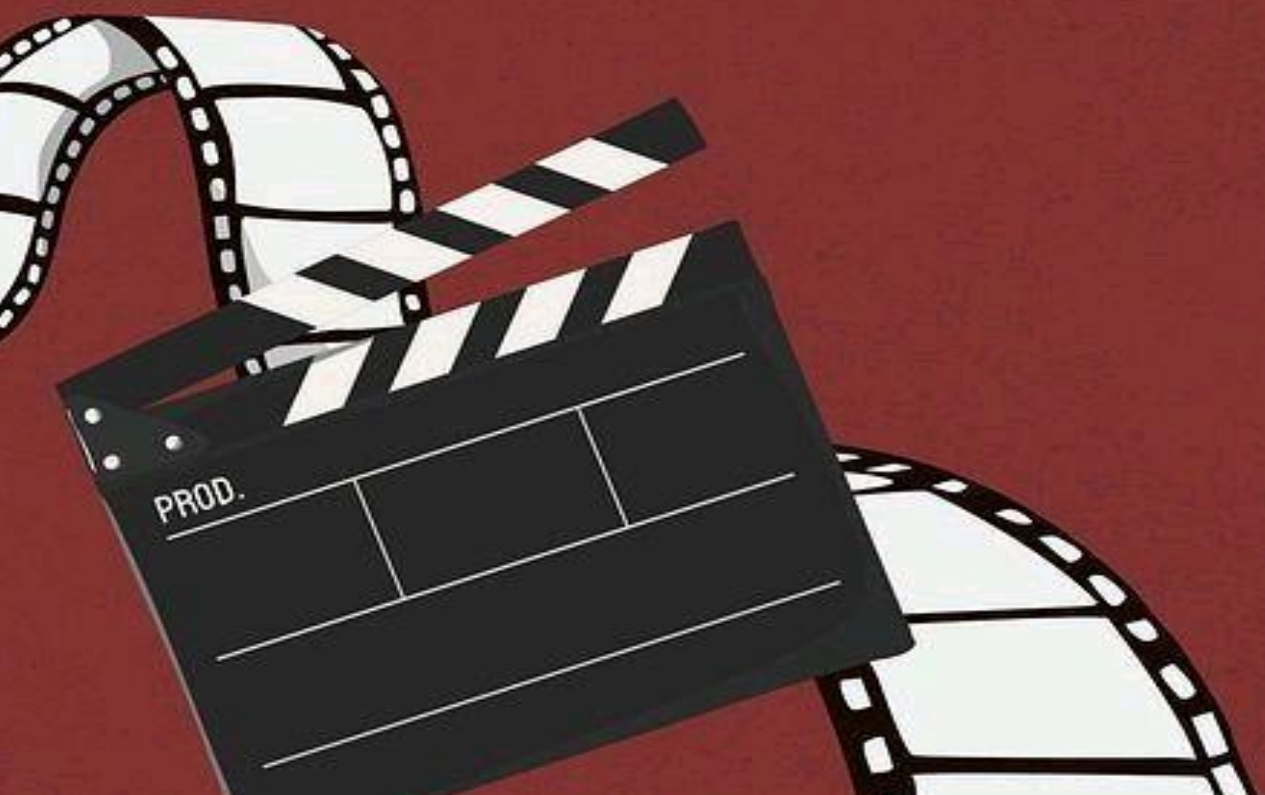
INTRODUCATION

The **Movie Ticket Booking System** is a database project designed to manage movie shows, theaters, customers, bookings, and payments. It helps to keep track of which movies are playing, where they are playing, customer details, ticket bookings, and the payments made.

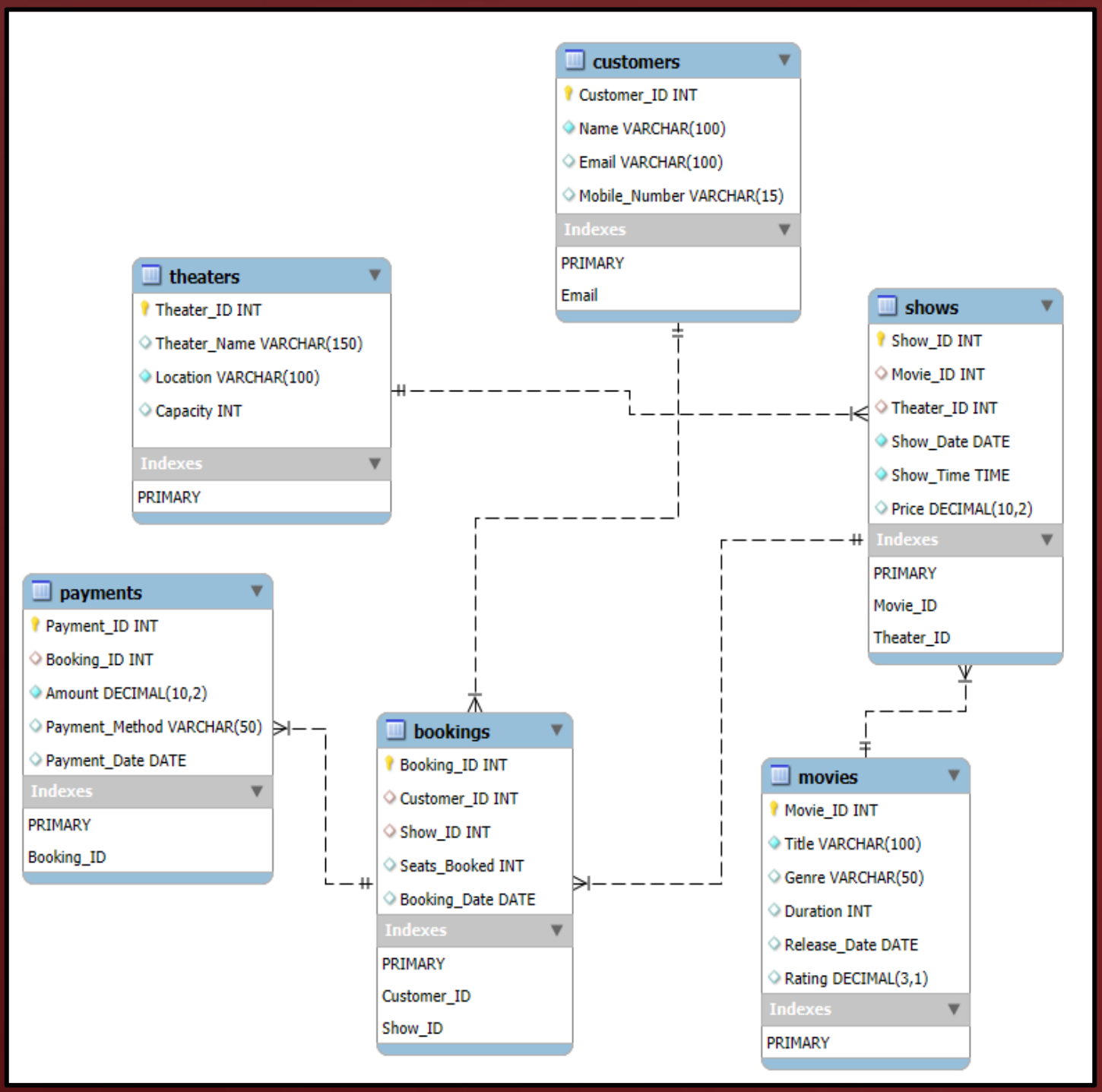
This project uses **MySQL** to create tables, insert data, and run queries to get useful information. With the help of SQL commands like SELECT, JOIN, GROUP BY, HAVING, and subqueries, we can find insights such as:

- Which movies are most popular
- How much revenue each theatre earns
- Which customers book the most tickets

The project shows how SQL can be used in real-life situations like an online movie booking platform. It covers **DDL (create/alter/drop)**, **DML (insert/update/delete)**, and **DQL (queries for analysis)**, making it a complete end-to-end database project.



ER DIAGRAM

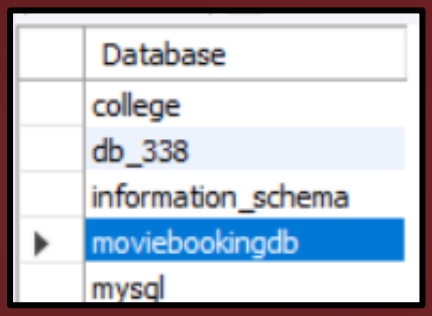

customers
Customer_ID INT
Name VARCHAR(100)
Email VARCHAR(100)
Mobile_Number VARCHAR(15)
Indexes
PRIMARY
Email
theaters
Theater_ID INT
Theater_Name VARCHAR(150)
Location VARCHAR(100)
Capacity INT
Indexes
PRIMARY
shows
Show_ID INT
Movie_ID INT
Theater_ID INT
Show_Date DATE
Show_Time TIME
Price DECIMAL(10,2)
Indexes
PRIMARY
Movie_ID
Theater_ID
payments
Payment_ID INT
Booking_ID INT
Amount DECIMAL(10,2)
Payment_Method VARCHAR(50)
Payment_Date DATE
Indexes
PRIMARY
Booking_ID
bookings
Booking_ID INT
Customer_ID INT
Show_ID INT
Seats_Booked INT
Booking_Date DATE
Indexes
PRIMARY
Customer_ID
Show_ID
movies
Movie_ID INT
Title VARCHAR(100)
Genre VARCHAR(50)
Duration INT
Release_Date DATE
Rating DECIMAL(3,1)
Indexes
PRIMARY

DATABASE:

CREATE DATABASE MovieBookingDB;

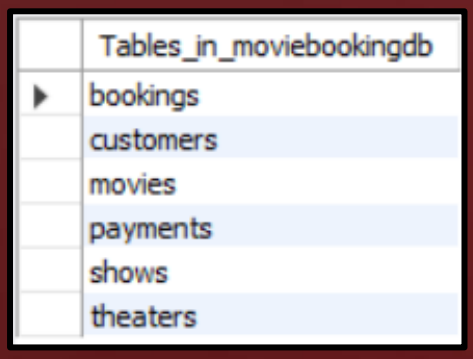
USE MovieBookingDB;

Show databases;



Database
college
db_338
information_schema
▶ moviebookingdb
mysql

Tables in Movie Booking Database:



Tables_in_moviebookingdb
▶ bookings
customers
movies
payments
shows
theaters

- **DATA DEFINITION LANGUAGE (DDL)**

1. CREATE

1. Movies:

```
CREATE TABLE Movies
 (Movie_ID INT PRIMARY KEY, Title VARCHAR(100) NOT NULL,
  Genre VARCHAR(50),Duration INT CHECK (Duration > 0), -- in minutes
  Release_Date DATE
   );
```

Field	Type	Null	Key	Default	Extra
Movie_ID	int	NO	PRI	NULL	
Title	varchar(100)	NO		NULL	
Genre	varchar(50)	YES		NULL	
Duration	int	YES		NULL	
Release_Date	date	YES		NULL	

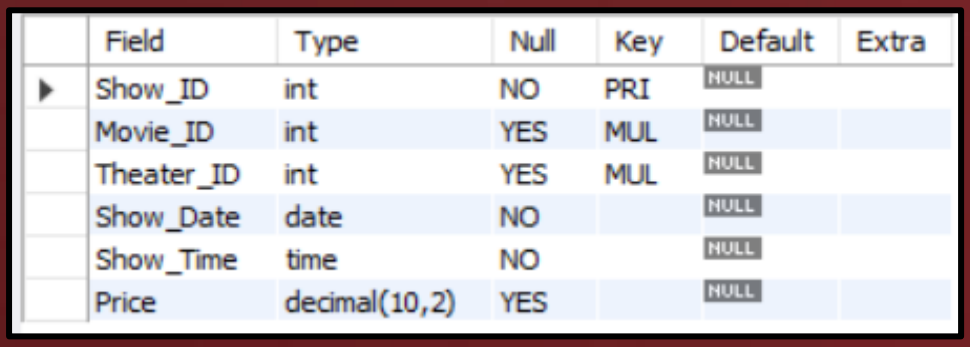
2. Theaters:

```
CREATE TABLE Theaters (
   Theater_ID INT PRIMARY KEY,
   Theater_Name VARCHAR(100) NOT NULL,
   Location VARCHAR(100) NOT NULL,
   Capacity INT CHECK (Capacity > 0)
);
```

Field	Type	Null	Key	Default	Extra
Theater_ID	int	NO	PRI	NULL	
Theater_Name	varchar(100)	NO		NULL	
Location	varchar(100)	NO		NULL	
Capacity	int	YES		NULL	

3. Shows:

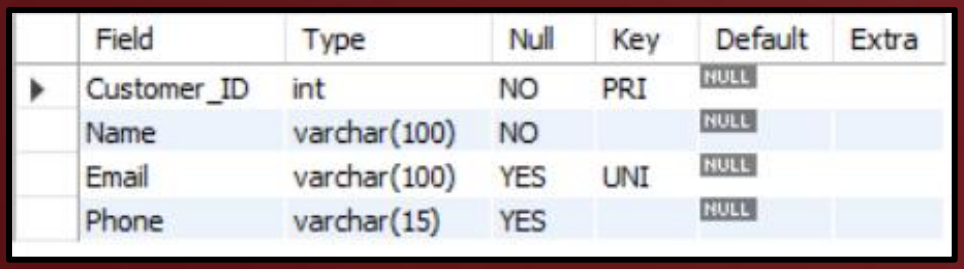
```
CREATE TABLE Shows (
    Show_ID INT PRIMARY KEY,
    Movie_ID INT,
    Theater_ID INT,
    Show_Date DATE NOT NULL,
    Show_Time TIME NOT NULL,
    Price DECIMAL(10,2) CHECK (Price > 0),
    FOREIGN KEY (Movie_ID) REFERENCES Movies(Movie_ID),
    FOREIGN KEY (Theater_ID) REFERENCES Theaters(Theater_ID)
);
```



Field	Type	Null	Key	Default	Extra
Show_ID	int	NO	PRI	NULL	
Movie_ID	int	YES	MUL	NULL	
Theater_ID	int	YES	MUL	NULL	
Show_Date	date	NO		NULL	
Show_Time	time	NO		NULL	
Price	decimal(10,2)	YES		NULL	

4. Customers:

```
CREATE TABLE Customers (
    Customer_ID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Phone VARCHAR(15)
);
```



Field	Type	Null	Key	Default	Extra
Customer_ID	int	NO	PRI	NULL	
Name	varchar(100)	NO		NULL	
Email	varchar(100)	YES	UNI	NULL	
Phone	varchar(15)	YES		NULL	

5. Bookings:

```
CREATE TABLE Bookings (
    Booking_ID INT PRIMARY KEY,
    Customer_ID INT,
    Show_ID INT,
```

```
    Seats_Booked INT CHECK (Seats_Booked > 0),
    Booking_Date DATE,
    FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID),
    FOREIGN KEY (Show_ID) REFERENCES Shows(Show_ID)
);
```

Field	Type	Null	Key	Default	Extra
Booking_ID	int	NO	PRI	NULL	
Customer_ID	int	YES	MUL	NULL	
Show_ID	int	YES	MUL	NULL	
Seats_Booked	int	YES		NULL	
Booking_Date	date	YES		NULL	

6. Payments:

```
CREATE TABLE Payments (
    Payment_ID INT PRIMARY KEY,
    Booking_ID INT,
    Amount DECIMAL(10,2) NOT NULL,
    Payment_Method VARCHAR(50),
    Payment_Date DATE,
    FOREIGN KEY (Booking_ID) REFERENCES Bookings(Booking_ID)
);
```

Field	Type	Null	Key	Default	Extra
Payment_ID	int	NO	PRI	NULL	
Booking_ID	int	YES	MUL	NULL	
Amount	decimal(10,2)	NO		NULL	
Payment_Method	varchar(50)	YES		NULL	
Payment_Date	date	YES		NULL	

2. ALTER

Q1: Add a new column Rating (decimal) to the Movies table.

```
ALTER TABLE Movies
ADD COLUMN Rating DECIMAL(3,1);
DESC MOVIES;
```

Field	Type	Null	Key	Default	Extra
Movie_ID	int	NO	PRI	NULL	
Title	varchar(100)	NO		NULL	
Genre	varchar(50)	YES		NULL	
Duration	int	YES		NULL	
Release_Date	date	YES		NULL	
Rating	decimal(3,1)	YES		NULL	

Q2: Modify the column Theater_Name in Theaters table to increase its length.

```
ALTER TABLE Theaters
MODIFY Theater_Name VARCHAR(150);
DESC THEATERS;
```

Field	Type	Null	Key	Default	Extra
Theater_ID	int	NO	PRI	NULL	
Theater_Name	varchar(150)	YES		NULL	
Location	varchar(100)	NO		NULL	
Capacity	int	YES		NULL	

Q3: Rename the column Phone in Customers table to Mobile_Number.

```
ALTER TABLE Customers
RENAME COLUMN Phone TO Mobile_Number;
DESC CUSTOMERS;
```

Field	Type	Null	Key	Default	Extra
Customer_ID	int	NO	PRI	NULL	
Name	varchar(100)	NO		NULL	
Email	varchar(100)	YES	UNI	NULL	
Mobile_Number	varchar(15)	YES		NULL	

3. TRUNCATE

Q1: Remove all records from the Payments table but keep the structure (empty the table).

TRUNCATE TABLE Payments;

Q2: Clear all data from the Bookings table (but keep the table ready for new bookings).

TRUNCATE TABLE Bookings;

4. DROP

Q1: Drop the Upcoming_Shows view (if it exists as a table by mistake).

DROP TABLE IF EXISTS Upcoming_Shows;

Q2: Completely remove the Movies table (data + structure).

DROP TABLE Movies;

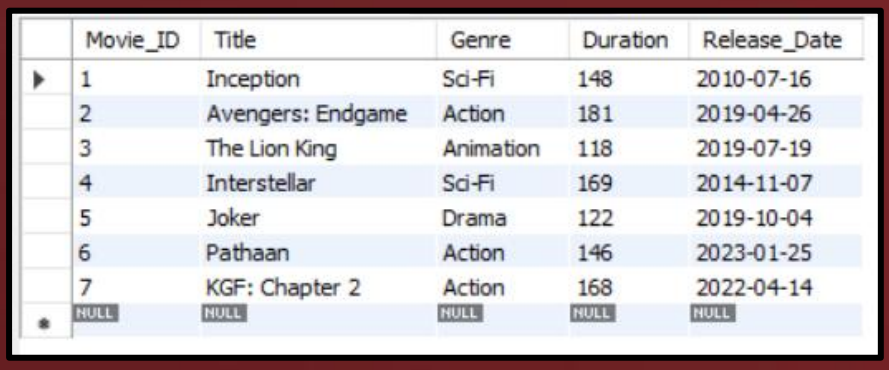
Q3: Remove the Customer_Booking_Info table if created temporarily.

DROP TABLE IF EXISTS Customer_Booking_Info;

- **DATA MANIPULATION LANGUAGE (DML):**

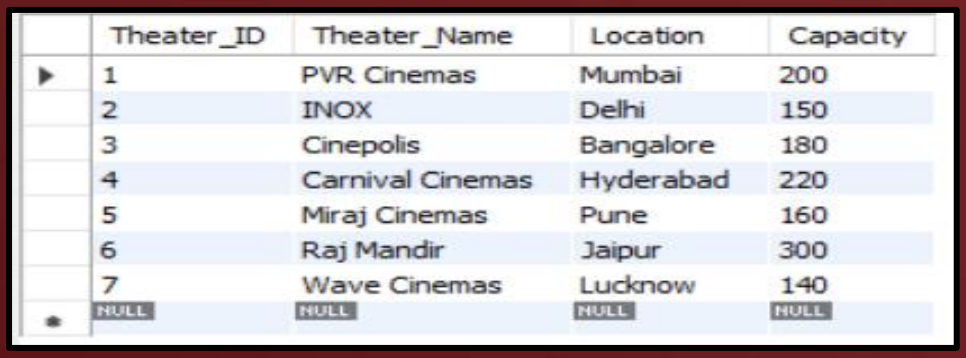
1. INSERT

INSERT INTO Movies VALUES (1, 'Inception', 'Sci-Fi', 148, '2010-07-16');
Select * from Movies;



Movie_ID	Title	Genre	Duration	Release_Date
1	Inception	Sci-Fi	148	2010-07-16
2	Avengers: Endgame	Action	181	2019-04-26
3	The Lion King	Animation	118	2019-07-19
4	Interstellar	Sci-Fi	169	2014-11-07
5	Joker	Drama	122	2019-10-04
6	Pathaan	Action	146	2023-01-25
7	KGF: Chapter 2	Action	168	2022-04-14
NULL	NULL	NULL	NULL	NULL

INSERT INTO Theaters VALUES (1, 'PVR Cinemas', 'Mumbai', 200);
INSERT INTO Theaters VALUES (2, 'INOX', 'Delhi', 150);
Select * from Theaters;



Theater_ID	Theater_Name	Location	Capacity
1	PVR Cinemas	Mumbai	200
2	INOX	Delhi	150
3	Cinepolis	Bangalore	180
4	Carnival Cinemas	Hyderabad	220
5	Miraj Cinemas	Pune	160
6	Raj Mandir	Jaipur	300
7	Wave Cinemas	Lucknow	140
NULL	NULL	NULL	NULL

2. UPDATE

Q1: Update ticket price of Show_ID = 101 to 300.

UPDATE Shows

SET Price = 300

WHERE Show_ID = 101;

Select * from shows;

Show_ID	Movie_ID	Theater_ID	Show_Date	Show_Time	Price
101	1	1	2025-09-25	18:00:00	300.00
102	2	2	2025-09-25	21:00:00	300.00
103	3	3	2025-09-26	15:00:00	200.00
104	4	4	2025-09-26	19:30:00	280.00
105	5	5	2025-09-27	20:00:00	220.00
106	6	6	2025-09-27	17:00:00	350.00
107	7	7	2025-09-28	21:30:00	300.00
NULL	NULL	NULL	NULL	NULL	NULL

Q2: Change the location of theater INOX to Gurgaon.

UPDATE Theaters

SET Location = 'Gurgaon'

WHERE Theater_Name = 'INOX';

Select * from theaters;

Theater_ID	Theater_Name	Location	Capacity
1	PVR Cinemas	Mumbai	200
2	INOX	Gurgaon	150
3	Cinepolis	Bangalore	180
4	Carnival Cinemas	Hyderabad	220
5	Miraj Cinemas	Pune	160
6	Raj Mandir	Jaipur	300
7	Wave Cinemas	Lucknow	140
NULL	NULL	NULL	NULL

Q3: Update customer name from Ravi Kumar to Ravi K.

UPDATE Customers

SET Name = 'Ravi K.'

WHERE Name = 'Ravi Kumar';

Select * from customers;

Customer_ID	Name	Email	Mobile_Number
1001	Ravi K.	ravi@example.com	9876543210
1002	Priya Sharma	priya@example.com	9123456780
1003	Amit Verma	amit@example.com	9812345678
1004	Sneha Gupta	sneha@example.com	9765432101
1005	Arjun Reddy	arjun@example.com	9900123456
1006	Neha Singh	neha@example.com	9988776655
1007	Rohit Mehta	rohit@example.com	9090909090
NULL	NULL	NULL	NULL

3. DELETE

Q1: Delete a booking with Booking_ID = 5007.

```
DELETE FROM Bookings
WHERE Booking_ID = 5007;
```

Q2: Remove a movie titled ‘Pathaan’ from the database.

```
DELETE FROM Movies
WHERE Title = 'Pathaan';
```

Q3: Delete all shows scheduled before 2025-09-25.

```
DELETE FROM Shows
WHERE Show_Date < '2025-09-25';
```

- **DATA QUERY LANGUAGE (DQL):**

1. SELECT

a) Select query for entire data.

SELECT * FROM Movies;

Movie_ID	Title	Genre	Duration	Release_Date
1	Inception	Sci-Fi	148	2010-07-16
2	Avengers: Endgame	Action	181	2019-04-26
3	The Lion King	Animation	118	2019-07-19
4	Interstellar	Sci-Fi	169	2014-11-07
5	Joker	Drama	122	2019-10-04
6	Pathaan	Action	146	2023-01-25
7	KGF: Chapter 2	Action	168	2022-04-14
NULL	NULL	NULL	NULL	NULL

b) Write a query to show movie names and genres.

SELECT Title, Genre FROM Movies;

Title	Genre
Inception	Sci-Fi
Avengers: Endgame	Action
The Lion King	Animation
Interstellar	Sci-Fi
Joker	Drama
Pathaan	Action
KGF: Chapter 2	Action

c) Write a query to show all customers sorted by name.

SELECT * FROM Customers ORDER BY Name ASC;

Customer_ID	Name	Email	Phone
1003	Amit Verma	amit@example.com	9812345678
1005	Arjun Reddy	arjun@example.com	9900123456
1006	Neha Singh	neha@example.com	9988776655
1002	Priya Sharma	priya@example.com	9123456780
1001	Ravi Kumar	ravi@example.com	9876543210
1007	Rohit Mehta	rohit@example.com	9090909090
1004	Sneha Gupta	sneha@example.com	9765432101
NULL	NULL	NULL	NULL

- **ORDER BY**

a) Write a query to show movies by Release Date (Newest First).
SELECT Title, Release_Date FROM Movies ORDER BY Release_Date DESC;

Title	Release_Date
Pathaan	2023-01-25
KGF: Chapter 2	2022-04-14
Joker	2019-10-04
The Lion King	2019-07-19
Avengers: Endgame	2019-04-26
Interstellar	2014-11-07
Inception	2010-07-16

b) Write a query to show Customers by Name (Alphabetical Order).
SELECT Customer_ID, Name, Email FROM Customers ORDER BY Name ASC;

Customer_ID	Name	Email
1003	Amit Verma	amit@example.com
1005	Arjun Reddy	arjun@example.com
1006	Neha Singh	neha@example.com
1002	Priya Sharma	priya@example.com
1001	Ravi Kumar	ravi@example.com
1007	Rohit Mehta	rohit@example.com
1004	Sneha Gupta	sneha@example.com
NULL	NULL	NULL

c) Write a query to Shows by Ticket Price (Lowest to Highest)
SELECT Show_ID, Movie_ID, Price FROM Shows ORDER BY Price ASC;

Show_ID	Movie_ID	Price
103	3	200.00
105	5	220.00
101	1	250.00
104	4	280.00
102	2	300.00
107	7	300.00
106	6	350.00
NULL	NULL	NULL

- **LIMIT**

a) Display Top 3 Most Expensive Shows

```
SELECT Show_ID, Price, Show_Date, Show_Time
FROM Shows
ORDER BY Price DESC
LIMIT 3;
```

Show_ID	Price	Show_Date	Show_Time
106	350.00	2025-09-27	17:00:00
102	300.00	2025-09-25	21:00:00
107	300.00	2025-09-28	21:30:00
NULL	NULL	NULL	NULL

b) Display Top 5 Largest Theaters by Capacity

```
SELECT Theater_Name, Location, Capacity
FROM Theaters
ORDER BY Capacity DESC
LIMIT 5;
```

Theater_Name	Location	Capacity
Raj Mandir	Jaipur	300
Carnival Cinemas	Hyderabad	220
PVR Cinemas	Mumbai	200
Cinepolis	Bangalore	180
Miraj Cinemas	Pune	160

- **DISTINCT**

Q. Write a query to Show all unique movie genres.

```
SELECT DISTINCT Genre FROM Movies;
```

Genre
Sci-Fi
Action
Animation
Drama

- **WHERE CLAUSE**

1. With Comparison Operator

Q. Write a query to show Movies longer than 150 minutes

SELECT Title, Duration FROM Movies WHERE Duration > 150;

Title	Duration
Avengers: Endgame	181
Interstellar	169
KGF: Chapter 2	168

2. Logical Operator

➢ Using AND operator

Q. Write a query to show Customers who booked more than 2 seats and booked after 2025-09-21.

SELECT * FROM Bookings WHERE Seats_Booked > 2 AND Booking_Date > '2025-09-21';

Booking_ID	Customer_ID	Show_ID	Seats_Booked	Booking_Date
5003	1003	103	4	2025-09-22
5005	1005	105	5	2025-09-23
5007	1007	107	3	2025-09-24
NULL	NULL	NULL	NULL	NULL

➢ Using OR operator

Q. Write a query to show Movies that are either “Action” or “Drama”.

SELECT Title, Genre FROM Movies WHERE Genre = 'Action' OR Genre = 'Drama';

Title	Genre
Avengers: Endgame	Action
Joker	Drama
Pathaan	Action
KGF: Chapter 2	Action

- Using NOT operator

Q. Write a query to show Shows that are not priced at 300.

SELECT Show_ID, Price FROM Shows WHERE NOT Price = 300;

Show_ID	Price
101	250.00
103	200.00
104	280.00
105	220.00
106	350.00
NULL	NULL

- Using NOT NULL

Q. Write a query to show Customers with phone numbers available

SELECT Name, Phone FROM Customers WHERE Phone IS NOT NULL;

Name	Phone
Ravi Kumar	9876543210
Priya Sharma	9123456780
Amit Verma	9812345678
Sneha Gupta	9765432101
Arjun Reddy	9900123456
Neha Singh	9988776655
Rohit Mehta	9090909090

- Using BETWEEN operator

Q. Write a query to show Shows priced between 200 and 300

SELECT Show_ID, Price FROM Shows WHERE Price BETWEEN 200 AND 300;

Show_ID	Price
101	250.00
102	300.00
103	200.00
104	280.00
105	220.00
107	300.00
NULL	NULL

- Using IN operator

Q. Write a query to show Movies from specific genres (Action, Sci-Fi, Animation)
SELECT Title, Genre FROM Movies WHERE Genre IN ('Action', 'Sci-Fi', 'Animation');

Title	Genre
Inception	Sci-Fi
Avengers: Endgame	Action
The Lion King	Animation
Interstellar	Sci-Fi
Pathaan	Action
KGF: Chapter 2	Action

- Using ANY operator

Q. Write a query to show Shows priced higher than any show in Theater_ID = 2

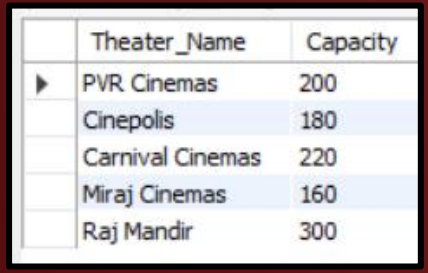
SELECT Show_ID, Price FROM Shows WHERE Price > ANY (SELECT Price FROM Shows WHERE Theater_ID = 2);

Show_ID	Price
106	350.00
NULL	NULL

- Using ALL operator

Q. Write a query to show Theaters with capacity greater than all theaters in Delhi

SELECT Theater_Name, Capacity FROM Theaters WHERE Capacity > ALL (SELECT Capacity FROM Theaters WHERE Location = 'Delhi');



Theater_Name	Capacity
PVR Cinemas	200
Cinepolis	180
Carnival Cinemas	220
Miraj Cinemas	160
Raj Mandir	300

- **AGGREGATE FUNCTIONS**

1. COUNT() Function

Q: Find the total number of customers registered in the system.

```
SELECT COUNT(*) AS Total_Customers FROM Customers;
```

Total_Customers
7

2. SUM() Function

Q: Display the total revenue collected from all payments.

```
SELECT SUM(Amount) AS Total_Revenue FROM Payments;
```

Total_Revenue
5110.00

3. AVG() Function

Q: Find the average ticket price across all shows.

```
SELECT AVG(Price) AS Average_Ticket_Price FROM Shows;
```

Average_Ticket_Price
271.428571

4. MAX() Function

Q: Find the highest ticket price among all shows.

```
SELECT MAX(Price) AS Highest_Ticket_Price FROM Shows;
```

Highest_Ticket_Price
350.00

5. MIN() Function

Q: Find the lowest ticket price among all shows.

```
SELECT MIN(Price) AS Lowest_Ticket_Price FROM Shows;
```

Lowest_Ticket_Price
200.00

6. Multiple Aggregates Together

Q: Find the highest, lowest, and average ticket price in the system.

```
SELECT MAX(Price) AS Highest_Price,
       MIN(Price) AS Lowest_Price,
       ROUND(AVG(Price),2) AS Avg_Price
FROM Shows;
```

Highest_Price	Lowest_Price	Avg_Price
350.00	200.00	271.43

- **GROUP BY CLAUSE**

Q: Show the total seats booked for each movie.

```
SELECT m.Title, SUM(b.Seats_Booked) AS Total_Seats  FROM Bookings b
  JOIN Shows s ON b.Show_ID = s.Show_ID
  JOIN Movies m ON s.Movie_ID = m.Movie_ID
  GROUP BY m.Title;
```

Title	Total_Seats
Inception	2
Avengers: Endgame	3
The Lion King	4
Interstellar	2
Joker	5
Pathaan	1
KGF: Chapter 2	3

- **HAVING CLAUSE**

Q: Find theaters where the total revenue is greater than 1000.

```
SELECT t.Theater_Name, SUM(p.Amount) AS Total_Revenue
FROM Payments p
JOIN Bookings b ON p.Booking_ID = b.Booking_ID
JOIN Shows s ON b.Show_ID = s.Show_ID
JOIN Theaters t ON s.Theater_ID = t.Theater_ID
GROUP BY t.Theater_Name
HAVING SUM(p.Amount) > 1000;
```

Theater_Name	Total_Revenue
Miraj Cinemas	1100.00

- **LIKE Operator**

Q: Find movies that have the word “King” in their title.

```
SELECT Title
FROM Movies
WHERE Title LIKE '%King%';
```

Title
The Lion King

- **UNION**

Q: Display a combined list of all movie titles and theater names.

```
SELECT Title AS Name, 'Movie' AS Type FROM Movies
UNION SELECT Theater_Name AS Name, 'Theater' AS Type
FROM Theaters;
```

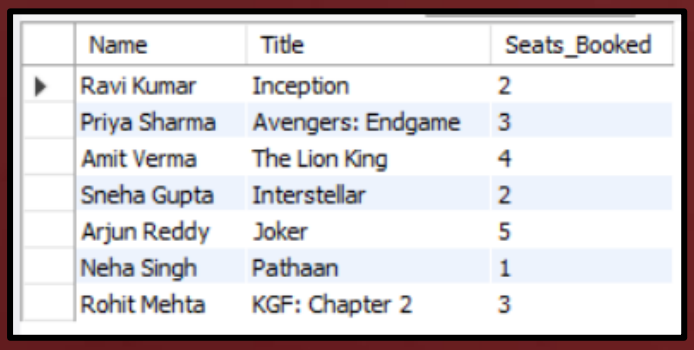
Name	Type
Inception	Movie
Avengers: Endgame	Movie
The Lion King	Movie
Interstellar	Movie
Joker	Movie
Pathaan	Movie
KGF: Chapter 2	Movie
PVR Cinemas	Theater
INOX	Theater
Cinepolis	Theater
Carnival Cinemas	Theater

- **JOINS**

1. INNER JOIN

Q. Write a query to show all customers with the movies they booked.

```
SELECT c.Name, m.Title, b.Seats_Booked
FROM Customers c
INNER JOIN Bookings b ON c.Customer_ID = b.Customer_ID
INNER JOIN Shows s ON b.Show_ID = s.Show_ID
INNER JOIN Movies m ON s.Movie_ID = m.Movie_ID;
```

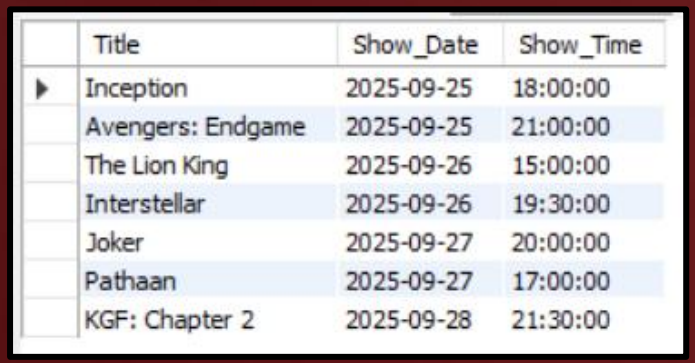


Name	Title	Seats_Booked
Ravi Kumar	Inception	2
Priya Sharma	Avengers: Endgame	3
Amit Verma	The Lion King	4
Sneha Gupta	Interstellar	2
Arjun Reddy	Joker	5
Neha Singh	Pathaan	1
Rohit Mehta	KGF: Chapter 2	3

2. LEFT JOIN

Q. List all movies and their shows (even if no shows are scheduled yet).

```
SELECT m.Title, s.Show_Date, s.Show_Time
FROM Movies m
LEFT JOIN Shows s ON m.Movie_ID = s.Movie_ID;
```



Title	Show_Date	Show_Time
Inception	2025-09-25	18:00:00
Avengers: Endgame	2025-09-25	21:00:00
The Lion King	2025-09-26	15:00:00
Interstellar	2025-09-26	19:30:00
Joker	2025-09-27	20:00:00
Pathaan	2025-09-27	17:00:00
KGF: Chapter 2	2025-09-28	21:30:00

3. RIGHT JOIN

Q. List all theaters and their scheduled movies (even if no movie is assigned).

```
SELECT t.Theater_Name, m.Title, s.Show_Date
FROM Movies m
RIGHT JOIN Shows s ON m.Movie_ID = s.Movie_ID
RIGHT JOIN Theaters t ON s.Theater_ID = t.Theater_ID;
```

Theater_Name	Title	Show_Date
PVR Cinemas	Inception	2025-09-25
INOX	Avengers: Endgame	2025-09-25
Cinepolis	The Lion King	2025-09-26
Carnival Cinemas	Interstellar	2025-09-26
Miraj Cinemas	Joker	2025-09-27
Raj Mandir	Pathaan	2025-09-27
Wave Cinemas	KGF: Chapter 2	2025-09-28

4. SELF JOIN

Q. Find customers who used the same email domain (like @example.com).

```
SELECT c1.Name AS Customer1, c2.Name AS Customer2, c1.Email
FROM Customers c1
JOIN Customers c2 ON c1.Customer_ID <> c2.Customer_ID
AND SUBSTRING_INDEX(c1.Email, '@', -1) = SUBSTRING_INDEX(c2.Email, '@', -1);
```

Customer1	Customer2	Email
Rohit Mehta	Ravi Kumar	rohit@example.com
Neha Singh	Ravi Kumar	neha@example.com
Arjun Reddy	Ravi Kumar	arjun@example.com
Sneha Gupta	Ravi Kumar	sneha@example.com
Amit Verma	Ravi Kumar	amit@example.com
Priya Sharma	Ravi Kumar	priya@example.com
Rohit Mehta	Priya Sharma	rohit@example.com
Neha Singh	Priya Sharma	neha@example.com
Arjun Reddy	Priya Sharma	arjun@example.com
Sneha Gupta	Priya Sharma	sneha@example.com
Amit Verma	Priya Sharma	amit@example.com
Ravi Kumar	Priya Sharma	ravi@example.com

Rohit Mehta	Amit Verma	rohit@example.com
Neha Singh	Amit Verma	neha@example.com
Arjun Reddy	Amit Verma	arjun@example.com
Sneha Gupta	Amit Verma	sneha@example.com
Priya Sharma	Amit Verma	priya@example.com
Ravi Kumar	Amit Verma	ravi@example.com
Rohit Mehta	Sneha Gupta	rohit@example.com
Neha Singh	Sneha Gupta	neha@example.com
Arjun Reddy	Sneha Gupta	arjun@example.com
Amit Verma	Sneha Gupta	amit@example.com
Priya Sharma	Sneha Gupta	priya@example.com
Ravi Kumar	Sneha Gupta	ravi@example.com
Rohit Mehta	Arjun Reddy	rohit@example.com

5. CROSS JOIN

Q. Generate all possible combinations of customers and movies (useful for testing promotions).

```
SELECT c.Name, m.Title
FROM Customers c
CROSS JOIN Movies m;
```

Name	Title
Rohit Mehta	Inception
Neha Singh	Inception
Arjun Reddy	Inception
Sneha Gupta	Inception
Amit Verma	Inception
Priya Sharma	Inception
Ravi Kumar	Inception
Rohit Mehta	Avengers: Endgame
Neha Singh	Avengers: Endgame
Arjun Reddy	Avengers: Endgame
Sneha Gupta	Avengers: Endgame
Amit Verma	Avengers: Endgame
Priya Sharma	Avengers: Endgame
Ravi Kumar	Avengers: Endgame
Rohit Mehta	The Lion King
Neha Singh	The Lion King
Arjun Reddy	The Lion King
Sneha Gupta	The Lion King
Amit Verma	The Lion King
Priya Sharma	The Lion King
Ravi Kumar	The Lion King
Rohit Mehta	Interstellar
Neha Singh	Interstellar
Arjun Reddy	Interstellar

Sneha Gupta	Joker
Amit Verma	Joker
Priya Sharma	Joker
Ravi Kumar	Joker
Rohit Mehta	Pathaan
Neha Singh	Pathaan
Arjun Reddy	Pathaan
Sneha Gupta	Pathaan
Amit Verma	Pathaan
Priya Sharma	Pathaan
Ravi Kumar	Pathaan
Rohit Mehta	KGF: Chapter 2
Neha Singh	KGF: Chapter 2
Arjun Reddy	KGF: Chapter 2
Sneha Gupta	KGF: Chapter 2
Amit Verma	KGF: Chapter 2
Priya Sharma	KGF: Chapter 2
Ravi Kumar	KGF: Chapter 2

- **SUB-QUERIES**

1. Single Row Subquery

Q1: Find the movie(s) that has a ticket price higher than the average ticket price.

```
SELECT Title FROM Movies
WHERE Movie_ID IN ( SELECT Movie_ID
   FROM Shows WHERE Price > (SELECT AVG(Price) FROM Shows)
);
```

Title
Avengers: Endgame
Interstellar
Pathaan
KGF: Chapter 2

Q2: Find the customer(s) who booked the maximum number of seats in a single booking.

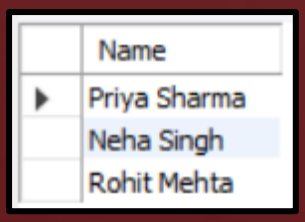
```
SELECT Name, Customer_ID
FROM Customers
WHERE Customer_ID = (
   SELECT Customer_ID
   FROM Bookings
   WHERE Seats_Booked = (SELECT MAX(Seats_Booked) FROM Bookings)
);
```

Name	Customer_ID
Arjun Reddy	1005
NULL	NULL

2. Multiple Row Subquery

Q1: Find customers who booked shows of movies belonging to Action genre.

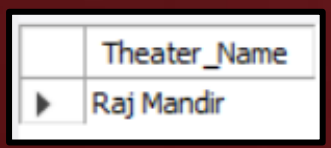
```
SELECT Name FROM Customers WHERE Customer_ID IN (
SELECT Customer_ID  FROM Bookings WHERE Show_ID IN (
   SELECT Show_ID   FROM Shows   WHERE Movie_ID IN (
    SELECT Movie_ID  FROM Movies  WHERE Genre = 'Action'   )   )
     );
```



Name
Priya Sharma
Neha Singh
Rohit Mehta

Q2: Find theaters that hosted at least one show priced above 300.

```
SELECT Theater_Name
FROM Theaters
WHERE Theater_ID IN (
   SELECT Theater_ID
   FROM Shows
   WHERE Price > 300
);
```



Theater_Name
Raj Mandir

3. Multiple Column Subquery

Q: Find the show ID and price of the highest-priced show in each theater.

```
SELECT Show_ID, Theater_ID, Price
FROM Shows WHERE (Theater_ID, Price) IN (
   SELECT Theater_ID, MAX(Price)
   FROM Shows
   GROUP BY Theater_ID
);
```

Show_ID	Theater_ID	Price
101	1	250.00
102	2	300.00
103	3	200.00
104	4	280.00
105	5	220.00
106	6	350.00
107	7	300.00
NULL	NULL	NULL

- **VIEW**

1. Customer Booking Information View

Q: Create a view that shows customers along with the movies they booked and number of seats.

```
CREATE VIEW Customer_Booking_Info AS
SELECT c.Name AS Customer_Name, m.Title AS Movie_Title, b.Seats_Booked,
b.Booking_Date
FROM Customers c
JOIN Bookings b ON c.Customer_ID = b.Customer_ID
JOIN Shows s ON b.Show_ID = s.Show_ID
JOIN Movies m ON s.Movie_ID = m.Movie_ID;
-- To see data
SELECT * FROM Customer_Booking_Info;
```

Customer_Name	Movie_Title	Seats_Booked	Booking_Date
Ravi Kumar	Inception	2	2025-09-20
Priya Sharma	Avengers: Endgame	3	2025-09-21
Amit Verma	The Lion King	4	2025-09-22
Sneha Gupta	Interstellar	2	2025-09-22
Arjun Reddy	Joker	5	2025-09-23
Neha Singh	Pathaan	1	2025-09-24
Rohit Mehta	KGF: Chapter 2	3	2025-09-24

2. Theater Revenue View

Q: Create a view to display total revenue generated by each theater.

```
CREATE VIEW Theater_Revenue AS
SELECT t.Theater_Name, SUM(p.Amount) AS Total_Revenue
FROM Payments p
JOIN Bookings b ON p.Booking_ID = b.Booking_ID
JOIN Shows s ON b.Show_ID = s.Show_ID
JOIN Theaters t ON s.Theater_ID = t.Theater_ID
GROUP BY t.Theater_Name;

-- To see data
SELECT * FROM Theater_Revenue;
```

Theater_Name	Total_Revenue
PVR Cinemas	500.00
INOX	900.00
Cinepolis	800.00
Carnival Cinemas	560.00
Miraj Cinemas	1100.00
Raj Mandir	350.00
Wave Cinemas	900.00

3. Movie Popularity View

Q: Create a view that shows movies and the total seats booked for each movie.

```
CREATE VIEW Movie_Popularity AS

SELECT m.Title, SUM(b.Seats_Booked) AS Total_Seats_Booked

FROM Movies m

JOIN Shows s ON m.Movie_ID = s.Movie_ID

JOIN Bookings b ON s.Show_ID = b.Show_ID

GROUP BY m.Title;


-- To see data

SELECT * FROM Movie_Popularity;
```

Title	Total_Seats_Booked
Inception	2
Avengers: Endgame	3
The Lion King	4
Interstellar	2
Joker	5
Pathaan	1
KGF: Chapter 2	3

4. High Value Customers View

Q: Create a view to show customers whose total spending is more than 1000.

```
CREATE VIEW High_Value_Customers AS

SELECT c.Name, SUM(p.Amount) AS Total_Spending

FROM Customers c

JOIN Bookings b ON c.Customer_ID = b.Customer_ID
```

```
JOIN Payments p ON b.Booking_ID = p.Booking_ID
GROUP BY c.Name
HAVING SUM(p.Amount) > 1000;

-- To see data
SELECT * FROM High_Value_Customers;
```

Name	Total_Spending
Arjun Reddy	1100.00

5. Upcoming Shows View

Q: Create a view that shows all shows scheduled after today's date.

```
CREATE VIEW Upcoming_Shows AS
SELECT s.Show_ID, m.Title AS Movie, t.Theater_Name, s.Show_Date, s.Show_Time, s.Price
FROM Shows s
JOIN Movies m ON s.Movie_ID = m.Movie_ID
JOIN Theaters t ON s.Theater_ID = t.Theater_ID
WHERE s.Show_Date > CURDATE();

-- To see data
SELECT * FROM Upcoming_Shows;
```

Show_ID	Movie	Theater_Name	Show_Date	Show_Time	Price
103	The Lion King	Cinepolis	2025-09-26	15:00:00	200.00
104	Interstellar	Carnival Cinemas	2025-09-26	19:30:00	280.00
105	Joker	Miraj Cinemas	2025-09-27	20:00:00	220.00
106	Pathaan	Raj Mandir	2025-09-27	17:00:00	350.00
107	KGF: Chapter 2	Wave Cinemas	2025-09-28	21:30:00	300.00