Sri Lanka Institute of Information Technology

Assignment 01- Part 02

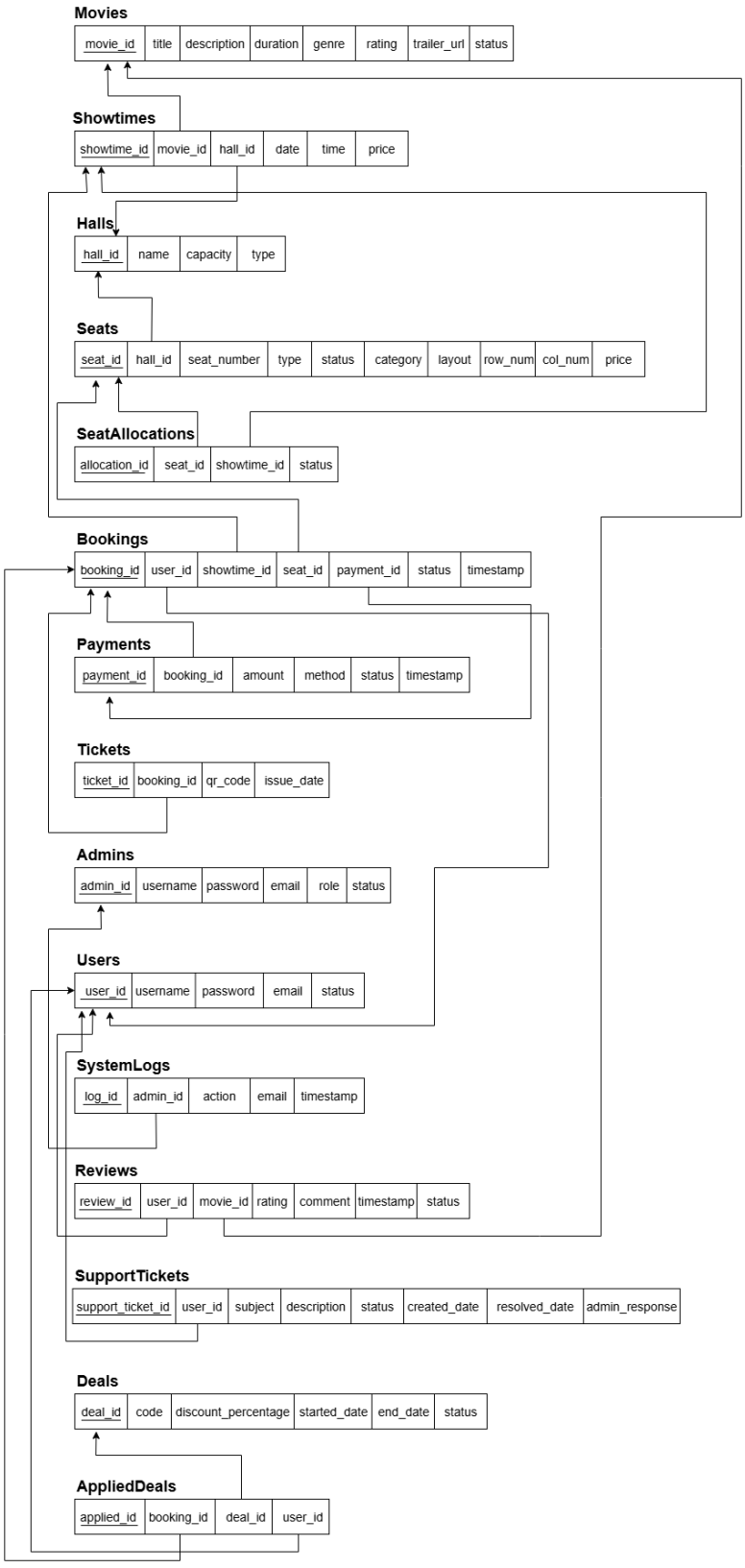
IT Number	Name
IT24100697	Kamsha S
IT24100565	De Silva S M K C P
IT24100590	Pesara J K L
IT24100673	Weeravaradana T
IT24100580	Hettigedara N.M.T.U

2025-Y2-S1-MLB-B4G1-08

Database Design and Development | IT2140

B.Sc. (Hons) in Information Technology

Part A - Mapping EER to Relational Schema



Constraints

1.Movies
Constraints:
title – NOT NULL
duration – CHECK (duration > 0)
rating – CHECK (rating BETWEEN 0 AND 10)
status – DEFAULT 'active’

2.Showtimes
Constraints:
date – NOT NULL
time – NOT NULL
price – CHECK (price >= 0)

3.Halls
Constraints:
name – NOT NULL, UNIQUE
capacity – CHECK (capacity > 0)
type – CHECK (type IN ('normal', 'balcony'))

4.Seats
Constraints:
seat_number → NOT NULL
type → CHECK (type IN ('Normal', 'Couple'))
category → CHECK (category IN ('standard', 'box'))
layout → CHECK (layout IN ('Front', 'Middle', 'Back', 'Balcony'))
status → CHECK (status IN ('available', 'booked', 'blocked'))
price → CHECK (price >= 0)
row_num and col_num → CHECK (row_num > 0 AND col_num > 0)

5.SeatAllocations
Constraints:
status → CHECK (status IN ('booked', 'available', 'blocked'))

6.Bookings
Constraints:
status – DEFAULT 'pending'
timestamp – DEFAULT CURRENT_TIMESTAMP

7.Payments
Constraints:
amount – CHECK (amount >= 0)
method – CHECK (method IN ('CreditCard', 'DebitCard', 'Cash', 'Online'))
status – DEFAULT 'success'
timestamp – DEFAULT CURRENT_TIMESTAMP

8.Tickets
Constraints:
qr_code – UNIQUE, NOT NULL
issue_date – DEFAULT CURRENT_TIMESTAMP

9.Admins
Constraints:
username – NOT NULL, UNIQUE
email – UNIQUE
password – NOT NULL
role – CHECK (role IN ('Manager', 'Staff'))
status – DEFAULT 'active'

10.Users
Constraints:
username – NOT NULL, UNIQUE
email – UNIQUE
password – NOT NULL
status – DEFAULT 'active'

11.SystemLogs
Constraints:
action – NOT NULL

timestamp – DEFAULT CURRENT_TIMESTAMP

12.Reviews
Constraints:
rating – CHECK (rating BETWEEN 1 AND 5)
timestamp – DEFAULT CURRENT_TIMESTAMP
status – DEFAULT 'visible'

13.SupportTickets
Constraints:
subject – NOT NULL
status – DEFAULT 'open'
created_date – DEFAULT CURRENT_TIMESTAMP
resolved_date – NULL allowed
admin_response – NULL allowed

14.Deals
Constraints:
code – UNIQUE, NOT NULL
discount_percentage – CHECK (discount_percentage BETWEEN 0 AND 100)
status – DEFAULT 'active'

Explain mapping choices for ISA

In the EER diagram, an ISA (is-a) relationship exists between Account and its subclasses User and Admin. This represents a disjoint and total specialization, where every account is either a user or an admin. The subclasses derive all properties from Account (account_id, username, password, email, status), and Admin has one more attribute: role.

For this ISA relationship, Mapping Option 2 was used — separate tables were created for each subclass, containing both the inherited and subclass-specific attributes. The superclass Account was not implemented as a separate table. This approach simplifies data access and clearly distinguishes between users and admins, making the design straightforward and efficient for this system.

Refine the schema if applicable

1.SeatAllocations
seat_id, showtime_id – UNIQUE

2. Bookings
status – CHECK (status IN ('pending','confirmed','cancelled'))

3. Payments
status – CHECK (status IN ('success','failed','pending'))

Part B – SQL DDL Implementation

```sql
------------
---Movies---
------------
CREATE TABLE Movies (
    movie_id INTEGER,
    title VARCHAR(255) NOT NULL,
    description VARCHAR(500),
    duration INTEGER,
    genre VARCHAR(100),
    rating REAL,
    trailer_url VARCHAR(255),
    status VARCHAR(20) CONSTRAINT df_Movies_status DEFAULT 'active',
    CONSTRAINT pk_Movies PRIMARY KEY (movie_id),
    CONSTRAINT chk_Movies_duration CHECK (duration > 0),
    CONSTRAINT chk_Movies_rating CHECK (rating BETWEEN 0 AND 10)
);
------------
---Halls---
------------
CREATE TABLE Halls (
    hall_id INTEGER,
    name VARCHAR(100) NOT NULL,
    capacity INTEGER,
    type VARCHAR(20),
    CONSTRAINT pk_Halls PRIMARY KEY (hall_id),
    CONSTRAINT uq_Halls_name UNIQUE (name),
    CONSTRAINT chk_Halls_capacity CHECK (capacity > 0),
    CONSTRAINT chk_Halls_type CHECK (type IN ('normal','balcony'))
);
--------------
---Showtimes---
--------------
CREATE TABLE Showtimes (
    showtime_id INTEGER,
    movie_id INTEGER NOT NULL,
    hall_id INTEGER NOT NULL,
    show_date DATE NOT NULL,
    show_time TIME NOT NULL,
    price MONEY,
    CONSTRAINT pk_Showtimes PRIMARY KEY (showtime_id),
    CONSTRAINT fk_Showtimes_Movie FOREIGN KEY (movie_id) REFERENCES Movies(movie_id),
    CONSTRAINT fk_Showtimes_Hall FOREIGN KEY (hall_id) REFERENCES Halls(hall_id),
    CONSTRAINT chk_Showtimes_price CHECK (price >= 0)
);
```

```sql
---------------
-----Seats-----
---------------
CREATE TABLE Seats (
    seat_id INTEGER,
    hall_id INTEGER NOT NULL,
    seat_number VARCHAR(10) NOT NULL,
    type VARCHAR(20),
    category VARCHAR(20),
    layout VARCHAR(20),
    status VARCHAR(20),
    row_num INTEGER,
    col_num INTEGER,
    price MONEY,
    CONSTRAINT pk_Seats PRIMARY KEY (seat_id),
    CONSTRAINT fk_Seats_Hall FOREIGN KEY (hall_id) REFERENCES Halls(hall_id),
    CONSTRAINT chk_Seats_type CHECK (type IN ('Normal','Couple')),
    CONSTRAINT chk_Seats_category CHECK (category IN ('standard','box')),
    CONSTRAINT chk_Seats_layout CHECK (layout IN ('Front','Middle','Back','Balcony')),
    CONSTRAINT chk_Seats_status CHECK (status IN ('available','booked','blocked')),
    CONSTRAINT chk_Seats_rowcol CHECK (row_num > 0 AND col_num > 0),
    CONSTRAINT chk_Seats_price CHECK (price >= 0)
);
```

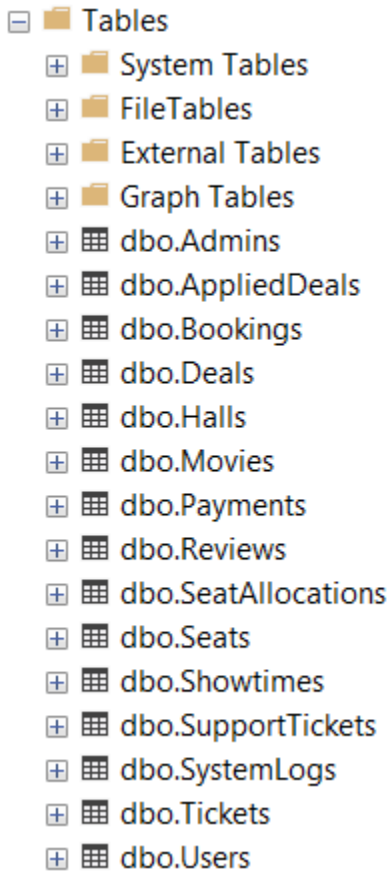
```sql
------------------------
-----SeatAllocations-----
------------------------
CREATE TABLE SeatAllocations (
    allocation_id INTEGER,
    seat_id INTEGER NOT NULL,
    showtime_id INTEGER NOT NULL,
    status VARCHAR(20),
    CONSTRAINT pk_SeatAllocations PRIMARY KEY (allocation_id),
    CONSTRAINT uq_SeatAllocations UNIQUE (seat_id, showtime_id),
    CONSTRAINT fk_SeatAllocations_Seat FOREIGN KEY (seat_id) REFERENCES Seats(seat_id),
    CONSTRAINT fk_SeatAllocations_Showtime FOREIGN KEY (showtime_id) REFERENCES Showtimes(showtime_id),
    CONSTRAINT chk_SeatAllocations_status CHECK (status IN ('booked','available','blocked'))
);
```

```sql
---------------
-----Users-----
---------------
CREATE TABLE Users (
    user_id INTEGER,
    username VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(255),
    status VARCHAR(20) CONSTRAINT df_Users_status DEFAULT 'active',
    CONSTRAINT pk_Users PRIMARY KEY (user_id),
    CONSTRAINT uq_Users_username UNIQUE (username),
    CONSTRAINT uq_Users_email UNIQUE (email)
);
```

```sql
-----------------
-----Admins-----
-----------------
CREATE TABLE Admins (
    admin_id INTEGER,
    username VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(255),
    role VARCHAR(20),
    status VARCHAR(20) CONSTRAINT df_Admins_status DEFAULT 'active',
    CONSTRAINT pk_Admins PRIMARY KEY (admin_id),
    CONSTRAINT uq_Admins_username UNIQUE (username),
    CONSTRAINT uq_Admins_email UNIQUE (email),
    CONSTRAINT chk_Admins_role CHECK (role IN ('Manager','Staff'))
);
-----------------
-----Bookings-----
-----------------
CREATE TABLE Bookings (
    booking_id INTEGER,
    user_id INTEGER NOT NULL,
    showtime_id INTEGER NOT NULL,
    seat_id INTEGER NOT NULL,
    status VARCHAR(20) CONSTRAINT df_Bookings_status DEFAULT 'pending',
    timestamp DATETIME CONSTRAINT df_Bookings_timestamp DEFAULT GETDATE(),
    CONSTRAINT pk_Bookings PRIMARY KEY (booking_id),
    CONSTRAINT fk_Bookings_User FOREIGN KEY (user_id) REFERENCES Users(user_id),
    CONSTRAINT fk_Bookings_Showtime FOREIGN KEY (showtime_id) REFERENCES Showtimes(showtime_id),
    CONSTRAINT fk_Bookings_Seat FOREIGN KEY (seat_id) REFERENCES Seats(seat_id),
    CONSTRAINT chk_Bookings_status CHECK (status IN ('pending','confirmed','cancelled'))
);
-----------------
-----Payments-----
-----------------
CREATE TABLE Payments (
    payment_id INTEGER,
    booking_id INTEGER NOT NULL,
    amount MONEY,
    method VARCHAR(20),
    status VARCHAR(20) CONSTRAINT df_Payments_status DEFAULT 'success',
    timestamp DATETIME CONSTRAINT df_Payments_timestamp DEFAULT GETDATE(),
    CONSTRAINT pk_Payments PRIMARY KEY (payment_id),
    CONSTRAINT fk_Payments_Booking FOREIGN KEY (booking_id) REFERENCES Bookings(booking_id),
    CONSTRAINT chk_Payments_amount CHECK (amount >= 0),
    CONSTRAINT chk_Payments_method CHECK (method IN ('CreditCard','DebitCard','Cash','Online')),
    CONSTRAINT chk_Payments_status CHECK (status IN ('success','failed','pending'))
);
-----------------
-----Tickets-----
-----------------
CREATE TABLE Tickets (
    ticket_id INTEGER,
    booking_id INTEGER NOT NULL,
    qr_code VARCHAR(255) NOT NULL,
    issue_date DATETIME CONSTRAINT df_Tickets_issue DEFAULT GETDATE(),
    CONSTRAINT pk_Tickets PRIMARY KEY (ticket_id),
    CONSTRAINT uq_Tickets_qr UNIQUE (qr_code),
    CONSTRAINT fk_Tickets_Booking FOREIGN KEY (booking_id) REFERENCES Bookings(booking_id)
);
```

```sql
--------------------
-----SystemLogs-----
--------------------
CREATE TABLE SystemLogs (
    log_id INTEGER,
    admin_id INTEGER NOT NULL,
    action VARCHAR(255) NOT NULL,
    timestamp DATETIME CONSTRAINT df_SystemLogs_time DEFAULT GETDATE(),
    CONSTRAINT pk_SystemLogs PRIMARY KEY (log_id),
    CONSTRAINT fk_SystemLogs_Admin FOREIGN KEY (admin_id) REFERENCES Admins(admin_id)
);
------------------
-----Reviews-----
------------------
CREATE TABLE Reviews (
    review_id INTEGER,
    user_id INTEGER NOT NULL,
    movie_id INTEGER NOT NULL,
    rating INTEGER,
    comment VARCHAR(500),
    timestamp DATETIME CONSTRAINT df_Reviews_time DEFAULT GETDATE(),
    status VARCHAR(20) CONSTRAINT df_Reviews_status DEFAULT 'visible',
    CONSTRAINT pk_Reviews PRIMARY KEY (review_id),
    CONSTRAINT fk_Reviews_User FOREIGN KEY (user_id) REFERENCES Users(user_id),
    CONSTRAINT fk_Reviews_Movie FOREIGN KEY (movie_id) REFERENCES Movies(movie_id),
    CONSTRAINT chk_Reviews_rating CHECK (rating BETWEEN 1 AND 5)
);
------------------------
-----SupportTickets-----
------------------------
CREATE TABLE SupportTickets (
    support_ticket_id INTEGER,
    user_id INTEGER NOT NULL,
    subject VARCHAR(255) NOT NULL,
    description VARCHAR(500),
    status VARCHAR(20) CONSTRAINT df_SupportTickets_status DEFAULT 'open',
    created_date DATETIME CONSTRAINT df_SupportTickets_created DEFAULT GETDATE(),
    resolved_date DATETIME NULL,
    admin_response VARCHAR(500) NULL,
    CONSTRAINT pk_SupportTickets PRIMARY KEY (support_ticket_id),
    CONSTRAINT fk_SupportTickets_User FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
------------------
-----Deals-----
------------------
CREATE TABLE Deals (
    deal_id INTEGER,
    code VARCHAR(50) NOT NULL,
    discount_percentage REAL,
    status VARCHAR(20) CONSTRAINT df_Deals_status DEFAULT 'active',
    CONSTRAINT pk_Deals PRIMARY KEY (deal_id),
    CONSTRAINT uq_Deals_code UNIQUE (code),
    CONSTRAINT chk_Deals_discount CHECK (discount_percentage BETWEEN 0 AND 100)
);
```

```sql
----------------------
-----AppliedDeals-----
----------------------
CREATE TABLE AppliedDeals (
    applied_id INTEGER,
    booking_id INTEGER NOT NULL,
    deal_id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    CONSTRAINT pk_AppliedDeals PRIMARY KEY (applied_id),
    CONSTRAINT fk_AppliedDeals_Booking FOREIGN KEY (booking_id) REFERENCES Bookings(booking_id),
    CONSTRAINT fk_AppliedDeals_Deal FOREIGN KEY (deal_id) REFERENCES Deals(deal_id),
    CONSTRAINT fk_AppliedDeals_User FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```



Part C – Insert Sample Data

```sql
----------------
-- Movies
----------------
INSERT INTO Movies VALUES
(1, 'Inception', 'A mind-bending thriller', 148, 'Sci-Fi', 8.8, 'trailer1.mp4', 'active'),
(2, 'Titanic', 'Romantic tragedy at sea', 195, 'Romance', 9.2, 'trailer2.mp4', 'active'),
(3, 'Avengers', 'Superheroes unite', 180, 'Action', 9.0, 'trailer3.mp4', 'active'),
(4, 'Joker', 'Dark psychological drama', 122, 'Drama', 8.5, 'trailer4.mp4', 'active'),
(5, 'Frozen', 'Animated musical adventure', 102, 'Animation', 7.9, 'trailer5.mp4', 'active');
```

```sql
-----------------
-- Halls
-----------------
INSERT INTO Halls VALUES
(1, 'Hall A', 100, 'normal'),
(2, 'Hall B', 120, 'balcony'),
(3, 'Hall C', 80, 'normal'),
(4, 'Hall D', 150, 'balcony'),
(5, 'Hall E', 200, 'normal');
```

```sql
-----------------
-- Showtimes
-----------------
INSERT INTO Showtimes VALUES
(1, 1, 1, '2025-10-10', '14:00', 800.00),
(2, 2, 2, '2025-10-11', '18:00', 900.00),
(3, 3, 3, '2025-10-12', '16:00', 750.00),
(4, 4, 4, '2025-10-13', '20:00', 950.00),
(5, 5, 5, '2025-10-14', '10:00', 650.00);
```

```sql
-----------------
-- Seats
-----------------
INSERT INTO Seats VALUES
(1, 1, 'A1', 'Normal', 'standard', 'Front', 'available', 1, 1, 800.00),
(2, 1, 'A2', 'Normal', 'standard', 'Front', 'available', 1, 2, 800.00),
(3, 2, 'B1', 'Couple', 'box', 'Middle', 'available', 2, 1, 1200.00),
(4, 3, 'C1', 'Normal', 'standard', 'Back', 'booked', 3, 1, 700.00),
(5, 4, 'D1', 'Couple', 'box', 'Balcony', 'available', 4, 1, 1500.00);
```

```sql
-----------------
-- SeatAllocations
-----------------
INSERT INTO SeatAllocations VALUES
(1, 1, 1, 'available'),
(2, 2, 1, 'booked'),
(3, 3, 2, 'available'),
(4, 4, 3, 'booked'),
(5, 5, 4, 'available');
-----------------
-- Users
-----------------
INSERT INTO Users VALUES
(1, 'john', 'pass123', 'john@gmail.com', 'active'),
(2, 'anna', 'anna123', 'anna@gmail.com', 'active'),
(3, 'peter', 'peter123', 'peter@gmail.com', 'active'),
(4, 'sara', 'sara123', 'sara@gmail.com', 'active'),
(5, 'mike', 'mike123', 'mike@gmail.com', 'active');
-----------------
-- Admins
-----------------
INSERT INTO Admins VALUES
(1, 'admin1', 'adminpass', 'admin1@cinema.com', 'Manager', 'active'),
(2, 'staff1', 'staffpass', 'staff1@cinema.com', 'Staff', 'active'),
(3, 'staff2', 'staffpass', 'staff2@cinema.com', 'Staff', 'active'),
(4, 'admin2', 'adminpass', 'admin2@cinema.com', 'Manager', 'active'),
(5, 'staff3', 'staffpass', 'staff3@cinema.com', 'Staff', 'active');
-----------------
-- Bookings
-----------------
INSERT INTO Bookings VALUES
(1, 1, 1, 1, 'confirmed', GETDATE()),
(2, 2, 2, 2, 'pending', GETDATE()),
(3, 3, 3, 3, 'cancelled', GETDATE()),
(4, 4, 4, 4, 'confirmed', GETDATE()),
(5, 5, 5, 5, 'pending', GETDATE());
```

```sql
-----------------
-- Payments
-----------------
INSERT INTO Payments VALUES
(1, 1, 800.00, 'CreditCard', 'success', GETDATE()),
(2, 2, 900.00, 'Cash', 'pending', GETDATE()),
(3, 3, 750.00, 'Online', 'failed', GETDATE()),
(4, 4, 950.00, 'DebitCard', 'success', GETDATE()),
(5, 5, 650.00, 'Cash', 'success', GETDATE());

-----------------
-- Tickets
-----------------
INSERT INTO Tickets VALUES
(1, 1, 'QR001', GETDATE()),
(2, 2, 'QR002', GETDATE()),
(3, 3, 'QR003', GETDATE()),
(4, 4, 'QR004', GETDATE()),
(5, 5, 'QR005', GETDATE());

-----------------
-- SystemLogs
-----------------
INSERT INTO SystemLogs VALUES
(1, 1, 'Added new movie record', GETDATE()),
(2, 2, 'Updated seat availability', GETDATE()),
(3, 3, 'Deleted old booking', GETDATE()),
(4, 4, 'Generated monthly report', GETDATE()),
(5, 5, 'Reset user password', GETDATE());

-----------------
-- Reviews
-----------------
INSERT INTO Reviews VALUES
(1, 1, 1, 5, 'Amazing movie!', GETDATE(), 'visible'),
(2, 2, 2, 4, 'Good but long', GETDATE(), 'visible'),
(3, 3, 3, 5, 'Loved the action scenes', GETDATE(), 'visible'),
(4, 4, 4, 3, 'Dark but powerful', GETDATE(), 'visible'),
(5, 5, 5, 4, 'Kids loved it', GETDATE(), 'visible');
```

```
----------------
-- SupportTickets
----------------
INSERT INTO SupportTickets VALUES
(1, 1, 'Payment issue', 'Card charged twice', 'open', GETDATE(), NULL, NULL),
(2, 2, 'Seat not showing', 'Could not select seat', 'open', GETDATE(), NULL, NULL),
(3, 3, 'Refund query', 'Booking cancelled refund not received', 'open', GETDATE(), NULL, NULL),
(4, 4, 'App issue', 'Page loading slowly', 'open', GETDATE(), NULL, NULL),
(5, 5, 'Suggestion', 'Add more showtimes', 'open', GETDATE(), NULL, NULL);

-----------------
-- Deals
-----------------
INSERT INTO Deals VALUES
(1, 'DEAL10', 10.0, 'active'),
(2, 'DEAL20', 20.0, 'active'),
(3, 'DEAL15', 15.0, 'active'),
(4, 'DEAL25', 25.0, 'inactive'),
(5, 'DEAL05', 5.0, 'active');

-----------------
-- AppliedDeals
-----------------
INSERT INTO AppliedDeals VALUES
(1, 1, 1, 1),
(2, 2, 2, 2),
(3, 3, 3, 3),
(4, 4, 4, 4),
(5, 5, 5, 5);
```

*Screenshots of inserted data (SELECT *).*

	movie_id	title	description	duration	genre	rating	trailer_url	status
1	1	Inception	A mind-bending thriller	148	Sci-Fi	8.8	trailer1.mp4	active
2	2	Titanic	Romantic tragedy at sea	195	Romance	9.2	trailer2.mp4	active
3	3	Avengers	Superheroes unite	180	Action	9	trailer3.mp4	active
4	4	Joker	Dark psychological drama	122	Drama	8.5	trailer4.mp4	active
5	5	Frozen	Animated musical adventure	102	Animation	7.9	trailer5.mp4	active

	hall_id	name	capacity	type
1	1	Hall A	100	normal
2	2	Hall B	120	balcony
3	3	Hall C	80	normal
4	4	Hall D	150	balcony
5	5	Hall E	200	normal

	showtime_id	movie_id	hall_id	show_date	show_time	price
1	1	1	1	2025-10-10	14:00:00.0000000	800.00
2	2	2	2	2025-10-13	20:00:00.0000000	900.00
3	3	3	3	2025-10-14	10:00:00.0000000	750.00
4	4	4	4	2025-10-13	20:00:00.0000000	950.00
5	5	5	5	2025-10-14	10:00:00.0000000	650.00

	seat_id	hall_id	seat_number	type	category	layout	status	row_num	col_num	price
1	1	1	A1	Normal	standard	Front	available	1	1	800.00
2	2	1	A2	Normal	standard	Front	available	1	2	800.00
3	3	2	B1	Couple	box	Middle	available	2	1	1200.00
4	4	3	C1	Normal	standard	Back	booked	3	1	700.00
5	5	4	D1	Couple	box	Balcony	available	4	1	1500.00

	allocation_id	seat_id	showtime_id	status
1	1	1	1	available
2	2	2	1	booked
3	3	3	2	available
4	4	4	3	booked
5	5	5	4	available

	user_id	username	password	email	status
1	1	john	pass123	john@gmail.com	active
2	2	anna	anna123	anna@gmail.com	active
3	3	peter	peter123	peter@gmail.com	active
4	4	sara	sara123	sara@gmail.com	active
5	5	mike	mike123	mike@gmail.com	active

	admin_id	username	password	email	role	status
1	1	admin1	adminpass	admin1@cinema.com	Manager	active
2	2	staff1	staffpass	staff1@cinema.com	Staff	active
3	3	staff2	staffpass	staff2@cinema.com	Staff	active
4	4	admin2	adminpass	admin2@cinema.com	Manager	active
5	5	staff3	staffpass	staff3@cinema.com	Staff	active

	booking_id	user_id	showtime_id	seat_id	status	timestamp
1	1	1	1	1	confirmed	2025-10-06 21:43:41.493
2	2	2	2	2	pending	2025-10-06 21:43:41.493
3	3	3	3	3	cancelled	2025-10-06 21:43:41.493
4	4	4	4	4	confirmed	2025-10-06 21:43:41.493
5	5	5	5	5	pending	2025-10-06 21:43:41.493

	payment_id	booking_id	amount	method	status	timestamp
1	1	1	800.00	CreditCard	success	2025-10-06 21:44:04.387
2	2	2	900.00	Cash	pending	2025-10-06 21:44:04.387
3	3	3	750.00	Online	failed	2025-10-06 21:44:04.387
4	4	4	950.00	DebitCard	success	2025-10-06 21:44:04.387
5	5	5	650.00	Cash	success	2025-10-06 21:44:04.387

	ticket_id	booking_id	qr_code	issue_date
1	1	1	QR001	2025-10-06 21:44:25.260
2	2	2	QR002	2025-10-06 21:44:25.260
3	3	3	QR003	2025-10-06 21:44:25.260
4	4	4	QR004	2025-10-06 21:44:25.260
5	5	5	QR005	2025-10-06 21:44:25.260

	log_id	admin_id	action	timestamp
1	1	1	Added new movie record	2025-10-06 21:44:47.053
2	2	2	Updated seat availability	2025-10-06 21:44:47.053
3	3	3	Deleted old booking	2025-10-06 21:44:47.053
4	4	4	Generated monthly report	2025-10-06 21:44:47.053
5	5	5	Reset user password	2025-10-06 21:44:47.053

	review_id	user_id	movie_id	rating	comment	timestamp	status
1	1	1	1	5	Amazing movie!	2025-10-06 21:45:21.197	visible
2	2	2	2	4	Good but long	2025-10-06 21:45:21.197	visible
3	3	3	3	5	Loved the action scenes	2025-10-06 21:45:21.197	visible
4	4	4	4	3	Dark but powerful	2025-10-06 21:45:21.197	visible
5	5	5	5	4	Kids loved it	2025-10-06 21:45:21.197	visible

	support_ticket_id	user_id	subject	description	status	created_date	resolved_date	admin_response
1	1	1	Payment issue	Card charged twice	open	2025-10-06 22:02:56.300	NULL	NULL
2	2	2	Seat not showing	Could not select seat	open	2025-10-06 22:02:56.300	NULL	NULL
3	3	3	Refund query	Booking cancelled refund not received	open	2025-10-06 22:02:56.300	NULL	NULL
4	4	4	App issue	Page loading slowly	open	2025-10-06 22:02:56.300	NULL	NULL
5	5	5	Suggestion	Add more showtimes	open	2025-10-06 22:02:56.300	NULL	NULL

	deal_id	code	discount_percentage	status
1	1	DEAL10	10	active
2	2	DEAL20	20	active
3	3	DEAL15	15	active
4	4	DEAL25	25	inactive
5	5	DEAL05	5	active

	applied_id	booking_id	deal_id	user_id
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5

Part D – SQL Queries & Outputs

```sql
--1). Listing all active movies with their title, genre, and rating
SELECT title, genre, rating
FROM Movies
WHERE status = 'active';
```

	title	genre	rating
1	Inception	Sci-Fi	8.8
2	Titanic	Romance	9.2
3	Avengers	Action	9
4	Joker	Drama	8.5
5	Frozen	Animation	7.9

Retrieve information about all the active movies.

```sql
--2).Listing all bookings with user name, movie title, and seat number.
SELECT u.username, m.title AS movie_title, s.seat_number
FROM Bookings b
JOIN Users u ON b.user_id = u.user_id
JOIN Showtimes st ON b.showtime_id = st.showtime_id
JOIN Movies m ON st.movie_id = m.movie_id
JOIN Seats s ON b.seat_id = s.seat_id;
```

	username	movie_title	seat_number
1	john	Inception	A1
2	anna	Titanic	A2
3	peter	Avengers	B1
4	sara	Joker	C1
5	mike	Frozen	D1

Use multiple joins to combine Users, Bookings, Movies, Showtimes, and Seats tables and retrieve a meaningful booking overview with all details in one row.

```sql
--3).Find the total amount of payments collected
SELECT SUM(amount) AS total_payments
FROM Payments
WHERE status = 'success';
```

	total_payments
1	2400.00

Using the aggregate function- SUM() to calculate total payments with status success.

```sql
--4).Count how many bookings each user has, showing only users with more than 0 bookings.
SELECT u.username, COUNT(b.booking_id) AS booking_count
FROM Users u
JOIN Bookings b ON u.user_id = b.user_id
GROUP BY u.username
HAVING COUNT(b.booking_id) > 0;
```

	username	booking_count
1	anna	1
2	john	1
3	mike	1
4	peter	1
5	sara	1

Retrieve the username and no. of booking each user has, from group by the username and Filters users with at least 1 booking using HAVING.

```sql
--5).Listing all users who have applied a deal with more than 10% discount
SELECT username
FROM Users
WHERE user_id IN (
    SELECT user_id
    FROM AppliedDeals ad
    JOIN Deals d ON ad.deal_id = d.deal_id
    WHERE d.discount_percentage > 10
);
```

	username
1	anna
2	peter
3	sara

The subquery finds all user IDs with applied deals over 10% and the main query retrieves their usernames.

Part E – Stored Function/Procedure

```
-- Procedure to update the booking status
CREATE PROCEDURE UpdateBookingStatus (@booking_id INT, @new_status VARCHAR(20))
as
begin
    SET NOCOUNT ON;
    -- Update booking status
    UPDATE Bookings
    SET status = @new_status
    WHERE booking_id = @booking_id;
end;
```

Before

	booking_id	user_id	seat_id	showtime_id	status
1	4	4	4	4	confirmed

```
----Execute with sample input
--Change booking 2 status to 'confirmed'
EXEC UpdateBookingStatus @booking_id = 4, @new_status = 'pending';
```

After

	booking_id	user_id	seat_id	showtime_id	status
1	4	4	4	4	pending

Part F – Trigger

```sql
-- Trigger to mark a seat as booked when a new booking is inserted
CREATE TRIGGER trg_AfterBookingInsert
ON Bookings
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;

    -- Update SeatAllocations status to 'booked' for the newly inserted booking
    UPDATE sa
    SET sa.status = 'booked'
    FROM SeatAllocations sa
    JOIN inserted i ON sa.seat_id = i.seat_id AND sa.showtime_id = i.showtime_id;

    PRINT 'Seat allocation status updated to booked for new booking.';
END;
```

```sql
-- Insert a new booking for user 4, showtime 2, seat 3
INSERT INTO Bookings (booking_id, user_id, showtime_id, seat_id, status, timestamp)
VALUES (8, 4, 2, 3, 'pending', GETDATE());
```

100 %

Messages

```
Seat allocation status updated to booked for new booking.

(1 row affected)
```

```sql
select * from Bookings
```

100 %

Results | Messages

	booking_id	user_id	showtime_id	seat_id	status	timestamp
1	1	1	1	1	confirmed	2025-10-06 21:43:41.493
2	2	2	2	2	confirmed	2025-10-06 21:43:41.493
3	3	3	3	3	cancelled	2025-10-06 21:43:41.493
4	4	4	4	4	pending	2025-10-06 21:43:41.493
5	5	5	5	5	pending	2025-10-06 21:43:41.493
6	6	3	1	1	pending	2025-10-06 22:53:32.390
7	7	3	2	2	pending	2025-10-06 22:58:10.413
8	8	4	2	3	pending	2025-10-06 23:01:54.700

```sql
--Verify the trigger worked
SELECT seat_id, showtime_id, status
FROM SeatAllocations
WHERE seat_id = 3 AND showtime_id = 2;
```

100 %

Results | Messages

	seat_id	showtime_id	status
1	3	2	booked