Cinema Hall Ticket Booking System

Ashish Dabhi – 202303039

Meet Patel – 202303012

Bhavin Dabhi – 202303041

Requirements and Queries:

1 . Get all users.

SQL: SELECT * FROM Users;

Relational Algebra: Users

	user_email [PK] character varying (250)	name character varying (100)	phone_number character varying (10)	password character varying (255)
1	john.doe@example.com	John Doe	1234567890	Password@123
2	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1
3	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23
4	serena.williams@example.com	Serena Williams	4445566778	TennisPro@1
5	lebron.james@example.com	LeBron James	2223344556	KingJames@23

2 . Find movies with a price greater than $10.

SQL: SELECT * FROM Movie WHERE price > 10;

Relational Algebra: σ(price > 10)(Movie)

	movie_title [PK] character varying (255)	genre character varying (50)	duration integer	release_date date	price numeric (10,2)
1	Avengers: Endgame	Action	180	2019-04-26	15.99
2	The Lion King	Drama	120	2019-07-19	12.99
3	Joker	Drama	122	2019-10-04	10.99
4	Interstellar	Sci-Fi	169	2014-11-07	14.99

3 . Get all bookings for a specific user (e.g., John Doe).

SQL: SELECT * FROM Booking WHERE user_email = 'john.doe@example.com';

Relational Algebra: σ(user_email = 'john.doe@example.com')(Booking)

	booking_id [PK] integer	user_email character varying (250)	showtime_id integer	booking_date date	seat_number integer
1	100000	john.doe@example.com	1	2024-10-20	1
2	100005	john.doe@example.com	2	2024-10-22	6
3	100010	john.doe@example.com	3	2024-10-23	11
4	100015	john.doe@example.com	4	2024-10-25	16

4 . Join users with their bookings.

SQL: SELECT * FROM Users u JOIN Booking b ON u.user_email = b.user_email;

Relational Algebra: Users ⋈ Booking

	user_email character varying (250)	name character varying (100)	phone_number character varying (10)	password character varying (255)	booking_id integer	user_email character varying (250)	showtime_id integer	booking_date date	seat_number integer
1	john.doe@example.com	John Doe	1234567890	Password@123	100000	john.doe@example.com	1	2024-10-20	1
2	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1	100001	jane.smith@example.com	2	2024-10-20	2
3	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23	100002	michael.jordan@example.com	1	2024-10-21	3
4	serena.williams@example.com	Serena Williams	4445566778	TennisPro@1	100003	serena.williams@example.com	3	2024-10-21	4
5	lebron.james@example.com	LeBron James	2223344556	KingJames@23	100004	lebron.james@example.com	4	2024-10-21	5
6	john.doe@example.com	John Doe	1234567890	Password@123	100005	john.doe@example.com	2	2024-10-22	6
7	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1	100006	jane.smith@example.com	3	2024-10-22	7
8	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23	100007	michael.jordan@example.com	4	2024-10-22	8
9	serena.williams@example.com	Serena Williams	4445566778	TennisPro@1	100008	serena.williams@example.com	1	2024-10-23	9
10	lebron.james@example.com	LeBron James	2223344556	KingJames@23	100009	lebron.james@example.com	2	2024-10-23	10
11	john.doe@example.com	John Doe	1234567890	Password@123	100010	john.doe@example.com	3	2024-10-23	11
12	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1	100011	jane.smith@example.com	4	2024-10-24	12
13	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23	100012	michael.jordan@example.com	1	2024-10-24	13
14	serena.williams@example.com	Serena Williams	4445566778	TennisPro@1	100013	serena.williams@example.com	2	2024-10-24	14
15	lebron.james@example.com	LeBron James	2223344556	KingJames@23	100014	lebron.james@example.com	3	2024-10-24	15
16	john.doe@example.com	John Doe	1234567890	Password@123	100015	john.doe@example.com	4	2024-10-25	16
17	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1	100016	jane.smith@example.com	1	2024-10-25	17
18	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23	100017	michael.jordan@example.com	2	2024-10-25	18
19	serena.williams@example.com	Serena Williams	4445566778	TennisPro@1	100018	serena.williams@example.com	3	2024-10-25	19
20	lebron.james@example.com	LeBron James	2223344556	KingJames@23	100019	lebron.james@example.com	4	2024-10-25	20

5 .Count the number of movies in each genre.

SQL: SELECT genre, COUNT(*) FROM Movie GROUP BY genre;

Relational Algebra: γ(genre, COUNT(*))(Movie)

	genre character varying (50)	count bigint
1	Drama	3
2	Sci-Fi	1
3	Action	1

6 . Get movies that are in the 'Drama' genre with a price less than $15.

SQL: SELECT * FROM Movie WHERE genre = 'Drama' AND price < 15;

Relational Algebra: σ(genre = 'Drama' AND price < 15)(Movie)

	movie_title [PK] character varying (255)	genre character varying (50)	duration integer	release_date date	price numeric (10,2)
1	The Lion King	Drama	120	2019-07-19	12.99
2	Parasite	Drama	132	2019-05-30	9.99
3	Joker	Drama	122	2019-10-04	10.99

7 . Find the average ticket price of all movies.

SQL: SELECT AVG(price) FROM Movie;

Relational Algebra: AVG(price)(Movie)

	avg numeric
1	12.9900000000000000

8 . Get all cinemas in New York.

SQL: SELECT * FROM Cinema WHERE cinema_city = 'New York';

Relational Algebra: σ(cinema_city = 'New York')(Cinema)

	cinema_name [PK] character varying (255)	cinema_area character varying (255)	cinema_city character varying (255)	cinema_pincode [PK] character varying (6)	total_screens integer
1	Cineplex 1	Downtown	New York	100001	5
2	Cineplex 2	Uptown	New York	100002	3

9 . List the total revenue from each movie.

SQL: SELECT movie_title, SUM(total_revenue) FROM Revenue GROUP BY movie_title;

Relational Algebra: γ(movie_title, SUM(total_revenue))(Revenue)

	movie_title character varying (255)	sum numeric
1	The Lion King	51.96
2	Joker	43.96
3	Avengers: Endgame	95.94
4	Parasite	29.97
5	Interstellar	59.96

10 . Get all showtimes for a specific movie.

SQL: SELECT * FROM Showtime WHERE movie_title = 'Avengers: Endgame';

Relational Algebra: σ(movie_title = 'Avengers: Endgame')(Showtime)

	showtime_id [PK] integer	cinema_name character varying (255)	cinema_pincode character varying (6)	movie_title character varying (255)	screen_id integer	show_date date	start_time time without time zone	end_time time without time zone
1	1	Cineplex 1	100001	Avengers: Endgame	1	2024-10-23	18:00:00	21:00:00

11 . Get all users who have made a booking and their corresponding booking details.

SQL: SELECT * FROM Users u JOIN Booking b ON u.user_email = b.user_email;

Relational Algebra: Users ⋈ Booking

	user_email character varying (250)	name character varying (100)	phone_number character varying (10)	password character varying (255)	booking_id integer	user_email character varying (250)	showtime_id integer	booking_date date	seat_number integer
1	john.doe@example.com	John Doe	1234567890	Password@123	100000	john.doe@example.com	1	2024-10-20	1
2	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1	100001	jane.smith@example.com	2	2024-10-20	2
3	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23	100002	michael.jordan@example.com	1	2024-10-21	3
4	serena.williams@example.com	Serena Williams	4445566778	TennisPro@1	100003	serena.williams@example.com	3	2024-10-21	4
5	lebron.james@example.com	LeBron James	2223344556	KingJames@23	100004	lebron.james@example.com	4	2024-10-21	5
6	john.doe@example.com	John Doe	1234567890	Password@123	100005	john.doe@example.com	2	2024-10-22	6
7	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1	100006	jane.smith@example.com	3	2024-10-22	7
8	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23	100007	michael.jordan@example.com	4	2024-10-22	8
9	serena.williams@example.com	Serena Williams	4445566778	TennisPro@1	100008	serena.williams@example.com	1	2024-10-23	9
10	lebron.james@example.com	LeBron James	2223344556	KingJames@23	100009	lebron.james@example.com	2	2024-10-23	10
11	john.doe@example.com	John Doe	1234567890	Password@123	100010	john.doe@example.com	3	2024-10-23	11
12	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1	100011	jane.smith@example.com	4	2024-10-24	12
13	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23	100012	michael.jordan@example.com	1	2024-10-24	13
14	serena.williams@example.com	Serena Williams	4445566778	TennisPro@1	100013	serena.williams@example.com	2	2024-10-24	14
15	lebron.james@example.com	LeBron James	2223344556	KingJames@23	100014	lebron.james@example.com	3	2024-10-24	15
16	john.doe@example.com	John Doe	1234567890	Password@123	100015	john.doe@example.com	4	2024-10-25	16
17	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1	100016	jane.smith@example.com	1	2024-10-25	17
18	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23	100017	michael.jordan@example.com	2	2024-10-25	18
19	serena.williams@example.com	Serena Williams	4445566778	TennisPro@1	100018	serena.williams@example.com	3	2024-10-25	19
20	lebron.james@example.com	LeBron James	2223344556	KingJames@23	100019	lebron.james@example.com	4	2024-10-25	20

12 . Count the number of bookings made by each user.

SQL: SELECT user_email, COUNT(*) FROM Booking GROUP BY user_email;

Relational Algebra: γ(user_email, COUNT(*))(Booking)

	user_email character varying (250)	count bigint
1	jane.smith@example.com	4
2	michael.jordan@example.com	4
3	lebron.james@example.com	4
4	serena.williams@example.com	4
5	john.doe@example.com	4

13 . Get all users who have booked a specific showtime.

SQL: SELECT * FROM Booking WHERE showtime_id = 1;

Relational Algebra: σ(showtime_id = 1)(Booking)

	booking_id [PK] integer	user_email character varying (250)	showtime_id integer	booking_date date	seat_number integer
1	100000	john.doe@example.com	1	2024-10-20	1
2	100002	michael.jordan@example.com	1	2024-10-21	3
3	100008	serena.williams@example.com	1	2024-10-23	9
4	100012	michael.jordan@example.com	1	2024-10-24	13
5	100016	jane.smith@example.com	1	2024-10-25	17

14 . Get showtimes and their associated cinema details.

SQL: SELECT s.*, c.cinema_name FROM Showtime s JOIN Cinema c ON s.cinema_name = c.cinema_name AND s.cinema_pincode = c.cinema_pincode;

Relational Algebra: Showtime ⋈ Cinema

	showtime_id integer	cinema_name character varying (255)	cinema_pincode character varying (6)	movie_title character varying (255)	screen_id integer	show_date date	start_time time without time zone	end_time time without time zone	cinema_name character varying (255)
1	1	Cineplex 1	100001	Avengers: Endgame	1	2024-10-23	18:00:00	21:00:00	Cineplex 1
2	2	Cineplex 1	100001	Parasite	2	2024-10-23	20:00:00	22:12:00	Cineplex 1
3	3	Galaxy Cinema	900001	Interstellar	3	2024-10-23	15:00:00	18:00:00	Galaxy Cinema
4	4	Star Cinema	600001	Joker	4	2024-10-24	19:00:00	21:02:00	Star Cinema

15 . List all movies that have not been booked yet.

SQL: SELECT * FROM Movie WHERE movie_title NOT IN (SELECT movie_title FROM Booking);

Relational Algebra: Movie - π(movie_title)(Booking)

	movie_title [PK] character varying (255)	genre character varying (50)	duration integer	release_date date	price numeric (10,2)

16 . Find all bookings made after a certain date.

SQL: SELECT * FROM Booking WHERE booking_date > '2024-10-21';

Relational Algebra: σ(booking_date > '2024-10-21')(Booking)

	booking_id [PK] integer	user_email character varying (250)	showtime_id integer	booking_date date	seat_number integer
1	100005	john.doe@example.com	2	2024-10-22	6
2	100006	jane.smith@example.com	3	2024-10-22	7
3	100007	michael.jordan@example.com	4	2024-10-22	8
4	100008	serena.williams@example.com	1	2024-10-23	9
5	100009	lebron.james@example.com	2	2024-10-23	10
6	100010	john.doe@example.com	3	2024-10-23	11
7	100011	jane.smith@example.com	4	2024-10-24	12
8	100012	michael.jordan@example.com	1	2024-10-24	13
9	100013	serena.williams@example.com	2	2024-10-24	14
10	100014	lebron.james@example.com	3	2024-10-24	15
11	100015	john.doe@example.com	4	2024-10-25	16
12	100016	jane.smith@example.com	1	2024-10-25	17
13	100017	michael.jordan@example.com	2	2024-10-25	18
14	100018	serena.williams@example.com	3	2024-10-25	19
15	100019	lebron.james@example.com	4	2024-10-25	20

17 . Get total revenue generated from each cinema.

SQL: SELECT cinema_name, SUM(total_revenue) FROM Revenue GROUP BY cinema_name;

Relational Algebra: γ(cinema_name, SUM(total_revenue))(Revenue)

	cinema_name character varying (255)	sum numeric
1	Cineplex 2	43.96
2	Cineplex 1	125.91
3	Galaxy Cinema	59.96
4	Star Cinema	51.96

18 . Find users who booked a showtime on a specific date.

SQL: SELECT DISTINCT u.* FROM Users u JOIN Booking b ON u.user_email = b.user_email WHERE b.booking_date = '2024-10-22';

Relational Algebra: π(Users)(σ(booking_date = '2024-10-22')(Users ⋈ Booking))

	user_email [PK] character varying (250)	name character varying (100)	phone_number character varying (10)	password character varying (255)
1	jane.smith@example.com	Jane Smith	9876543210	SecurePass@1
2	john.doe@example.com	John Doe	1234567890	Password@123
3	michael.jordan@example.com	Michael Jordan	5556677889	AirJordan@23

19 . Get the cinema with the maximum number of screens.

SQL: SELECT * FROM Cinema ORDER BY total_screens DESC LIMIT 1;

Relational Algebra: π(max(total_screens))(Cinema)

	cinema_name [PK] character varying (255)	cinema_area character varying (255)	cinema_city character varying (255)	cinema_pincode [PK] character varying (6)	total_screens integer
1	Galaxy Cinema	Central	Los Angeles	900001	8

20 . Find movies with a higher ticket price than the average ticket price.

SQL: SELECT * FROM Movie WHERE price > (SELECT AVG(price) FROM Movie);

Relational Algebra: σ(price > AVG(price))(Movie)

	movie_title [PK] character varying (255)	genre character varying (50)	duration integer	release_date date	price numeric (10,2)
1	Avengers: Endgame	Action	180	2019-04-26	15.99
2	Interstellar	Sci-Fi	169	2014-11-07	14.99