Cinema Database System

Muhammad Uzair Khalid 0258-BSCS-21

Wassam Rafiq 0288-BSCS-21

SUBMITTED TO: SIR HAFEEZ

May 12, 2023

Cinema Database System

➢ Introduction:

A cinema wants to build a database system for its management which would definitely enhance its business and will lure the people towards the cinema. A cinema would have multiple screens which will boost the numbers of audience resulting in the success of business.

➢ Main Points:

- A person (having atleast an id) can have a ticket(s) of one or many show(s) however a show can have many people with tickets or no one.
- A single ticket must allot a single seat.
- A screen with unique types can have multiple seats starting from one atleast.
- A movie can be displayed on multiple screens having multiple show times, also one screen can have shows of multiple movies or no any.
- A movie can belong to many genres starting from one and a single genre can have many or no any movie belonging to it.
- A person can order no-any or maybe many food items and a food item can be ordered by many as well as no-any person.

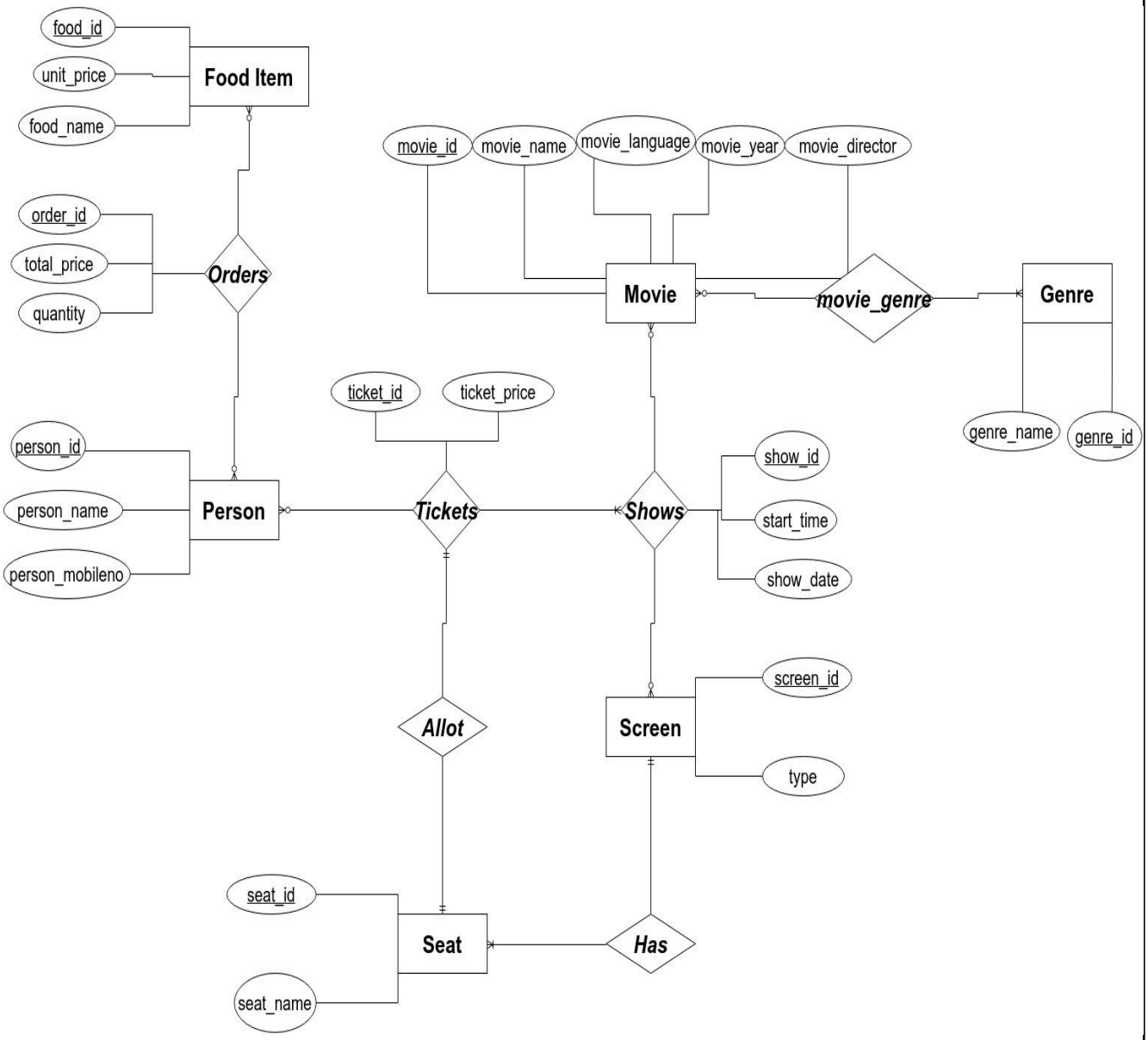
➢ Verbs:

- A person has a **TICKET** for a show.
- A ticket **ALLOT** a seat.
- A screen **HAS** many seats
- A screen has a **SHOW** for a movie.
- A movie belongs to any specific **MOVIE_GENRE** from many genres.
- A person can **ORDER** a food item.

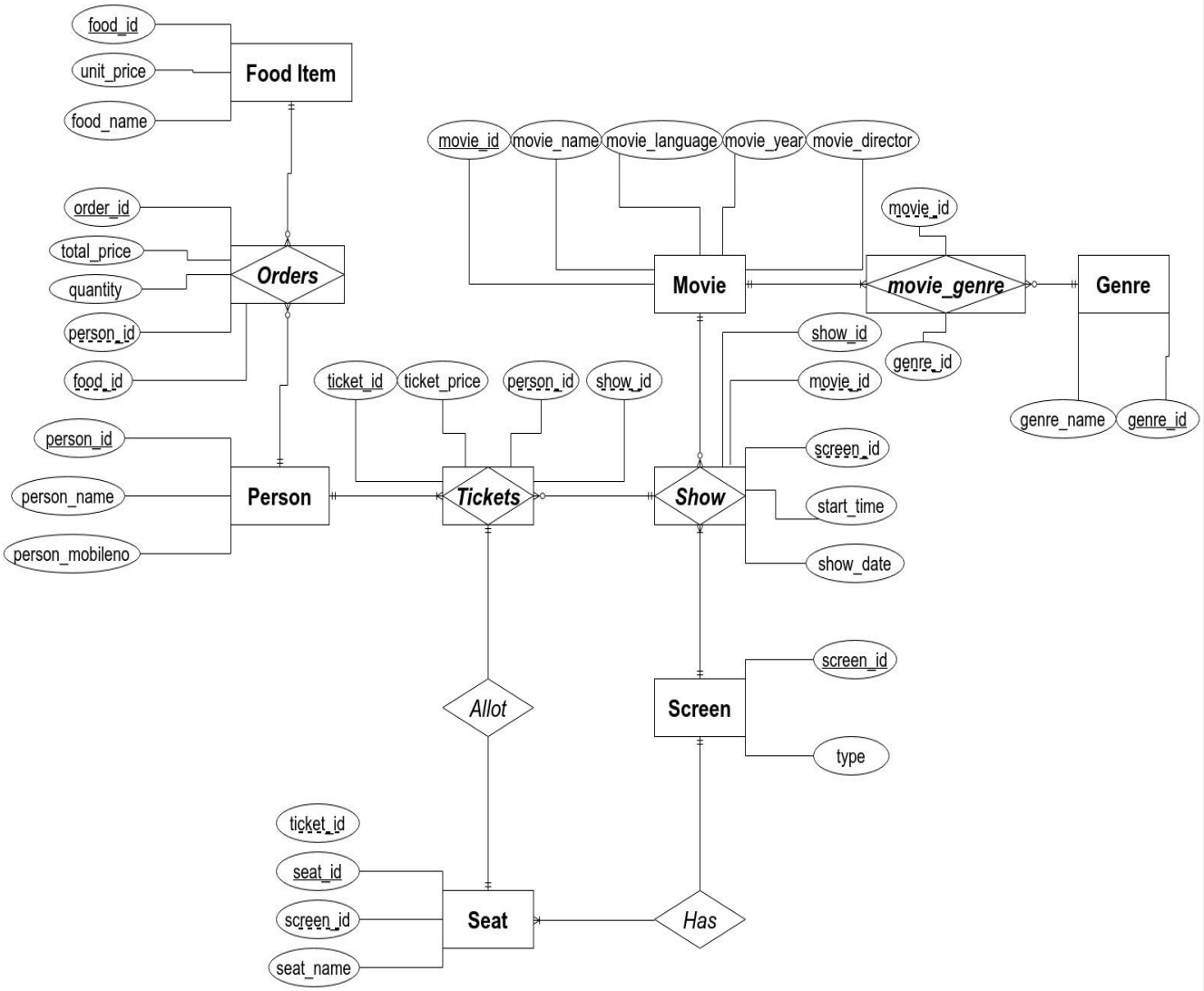
➢ Nouns & Adjectives:

NOUNS	ADJECTIVES
Person	person_name, person_ id, person_ mobileno
Ticket (Junction Entity)	ticket_price, ticket_id
Seat	seat_id, seat_name
Screen	screen_name, screen_ id
Movie	movie_name, movie_ id, movie_language, movie_director, movie_year
Show (Junction Entity)	show_id, start_time, show_date
Genre	genre_id, genre_name
Food Item	food_id, food_name, unit_price
Order (Junction Entity)	order_id, order_quantity, total_price
movie_genre (Junction Entity)	(It will only have Primary key of Movie Entity and GenreEntity as Foreign keys)

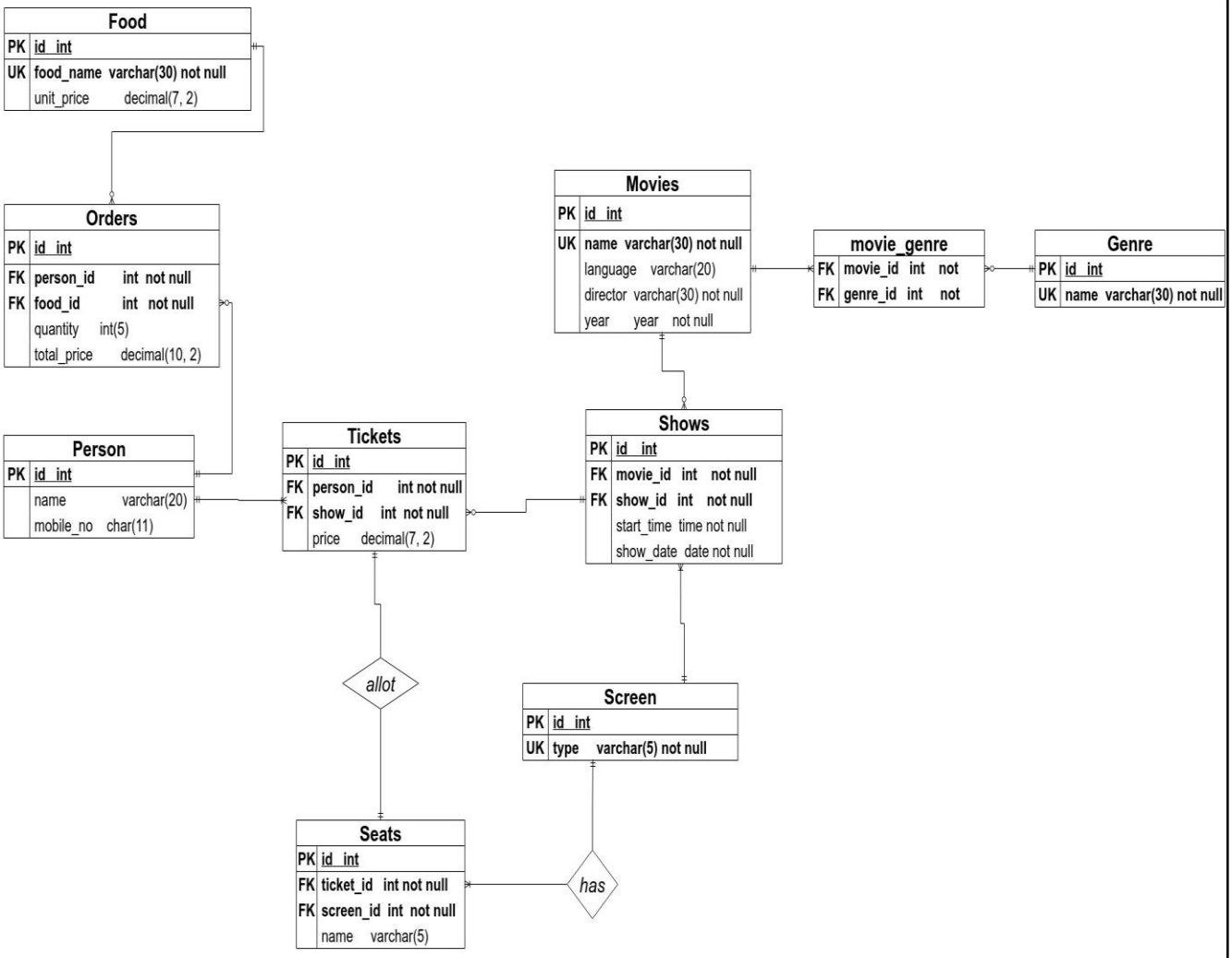
➢ ERD:


food_id
unit_price
food_name
Food Item
order_id
total_price
quantity
Orders
movie_id
movie_name
movie_language
movie_year
movie_director
Movie
movie_genre
Genre
genre_name
genre_id
ticket_id
ticket_price
person_id
person_name
person_mobileno
Person
Tickets
Shows
show_id
start_time
show_date
screen_id
Allot
Screen
type
seat_id
Seat
Has
seat_name

➢ Normalized ERD:



➢ Relational Model:



➢ Tables:

Person:

```
CREATE TABLE person(
    id int AUTO_INCREMENT,
    name VARCHAR(20),
    mobile_number char(11),
    CONSTRAINT person_id_pk PRIMARY KEY (id)
);
```

Food:

```
CREATE TABLE food(
    id int AUTO_INCREMENT,
    name VARCHAR(30) NOT NULL,
    unit_price decimal(7, 2),
    CONSTRAINT food_name_uk UNIQUE(name),
    CONSTRAINT food_id_pk PRIMARY KEY (id)
);
```

Orders:

```
CREATE TABLE orders(
    id int AUTO_INCREMENT,
    total_price decimal(10, 2),
    quantity int(5),
    person_id int NOT NULL,
    food_id int NOT NULL,
    CONSTRAINT order_id_pk PRIMARY KEY (id),
    CONSTRAINT person_id_fk FOREIGN KEY (person_id) REFERENCES person(id),
    CONSTRAINT food_id_fk FOREIGN KEY (food_id) REFERENCES food(id)
);
```

Screen:

```
CREATE TABLE screen(
    id int AUTO_INCREMENT,
    type VARCHAR(5) NOT NULL,
    CONSTRAINT screen_type_uk UNIQUE(type),
    CONSTRAINT screen_id_pk PRIMARY KEY (id)
);
```

Movies:

```
CREATE TABLE movies(
    id int AUTO_INCREMENT,
    name VARCHAR(30) NOT NULL,
    language VARCHAR(20),
    director VARCHAR(30) NOT NULL,
    year YEAR NOT NULL,
    CONSTRAINT movie_name_uk UNIQUE(name),
    CONSTRAINT movie_id_pk PRIMARY KEY (id)
);
```

Genre:

```
CREATE TABLE genre(
    id int AUTO_INCREMENT,
    name varchar(20) NOT NULL,
    CONSTRAINT genre_name_uk UNIQUE(name),
    CONSTRAINT genre_id_pk PRIMARY KEY (id)
);
```

Shows:

```
CREATE TABLE shows(
    id int AUTO_INCREMENT,
    start_time time NOT NULL,
    show_date date NOT NULL,
    screen_id int NOT NULL,
    movie_id int NOT NULL,
    CONSTRAINT show_id_pk PRIMARY KEY (id),
    CONSTRAINT screen_id_fk2 FOREIGN KEY (screen_id) REFERENCES screen(id),
    CONSTRAINT movie_id_fk2 FOREIGN KEY (movie_id) REFERENCES movies(id)
);
```

Tickets:

```
CREATE TABLE tickets(
    id int AUTO_INCREMENT,
    price DECIMAL(7, 2),
    person_id int NOT NULL,
    show_id int NOT NULL,
    CONSTRAINT tickets_id_pk PRIMARY KEY (id),
    CONSTRAINT person_id_fk2 FOREIGN KEY (person_id) REFERENCES person(id),
    CONSTRAINT show_id_fk FOREIGN KEY (show_id) REFERENCES shows(id)
);
```

Seats:

```
CREATE TABLE seats(
    id int AUTO_INCREMENT,
    name VARCHAR(5),
    ticket_id int NOT NULL,
    screen_id int NOT NULL,
    CONSTRAINT seat_id_pk PRIMARY KEY (id),
    CONSTRAINT screen_id_fk FOREIGN KEY(screen_id) REFERENCES screen(id),
    CONSTRAINT ticket_id_fk FOREIGN KEY(ticket_id) REFERENCES tickets(id)
);
```

Movie_Genre:

```
create table movie_genre(
    movie_id int NOT NULL,
    genre_id int NOT NULL,
    CONSTRAINT movie_id_fk FOREIGN KEY (movie_id) REFERENCES movies(id),
    CONSTRAINT genre_id_fk FOREIGN KEY (genre_id) REFERENCES genre(id)
);
```

➢ Data:

Person:

```
INSERT INTO person(id, name, mobile_number)
VALUES (1, 'John Kennedy', '34597522119'), (2, 'Chris Mark', '34611246785'), (3, 'John Sam', '34541262119'), (4, 'Cyrus Sen', '34617877785'),
(5, 'Sooraj Singhania', '34617877785'), (6, 'Arjun Kumar', '34211123457'), (7, 'Divya Prakash', '43326777890'), (8, 'Riya Anand', '43543437890'),
(9, 'Robert Brown', '33356789890'), (10, 'Sarah Mitchel', '34626778190'), (11, 'Omer Zain', '23456877890'), (12, 'Arif Usman', '28126517890'),
(13, 'Sophia Albert', '34326769893'), (14, 'Tim Johnson', '41326977123'), (15, 'David Kite', '42325677980'), (16, 'Elle Susane', '43321358890'),
(17, 'Kiara Kapoor', '29876991247'), (18, 'Katherine Ashton', '21046200304'), (19, 'Emma Rose', '41233321277'), (20, 'Samiya Khan', '33572289653');
```

Food:

```
INSERT INTO food(id, name, unit_price)
VALUES (1, 'Popcorn', 40.00), (2, 'Fries', 25.50), (3, 'Sprite', 9.99), (4, 'Coke', 9.99), (5, 'Fanta', 9.99), (6, 'Nuggets', 20.00),
(7, 'Pizza Slice', 9.50), (8, 'Sandwich', 6.99), (9, 'Chicken Patty', 5.00), (10, 'Vegetable Patty', 4.00), (11, 'Brownie', 14.99),
(12, 'Donut', 12.50), (13, 'Chicken Wings', 9.99), (14, 'Ice cream', 14.99), (15, 'Lays', 2.99), (16, 'Coffee', 20.00), (17, 'Tea', 18.50),
(18, 'Chocolatechip Cookies', 9.99), (19, 'Cream Roll', 9.99), (20, 'Cup cakes', 9.99);
```

Orders:

```
INSERT INTO orders(id, total_price, quantity, person_id, food_id)
VALUES (1, 120.00, 3, 2, 1), (2, 9.99, 1, 1, 5), (3, 6.99, 1, 4, 8), (4, 9.99, 1, 5, 5), (5, 51.00, 2, 3, 2), (6, 9.99, 2, 14, 5),
(7, 9.99, 1, 11, 5), (8, 9.99, 1, 15, 3), (9, 40.00, 2, 15, 16), (10, 51.00, 2, 20, 2), (11, 9.99, 1, 16, 5), (12, 29.98, 2, 20, 11),
(13, 50.00, 4, 12, 12), (14, 99.99, 10, 16, 18), (15, 37.00, 2, 16, 17), (16, 80.00, 2, 15, 1), (17, 76.50, 3, 13, 2), (18, 160.00, 4, 11, 1),
(19, 9.99, 1, 13, 13), (20, 120.00, 3, 12, 1), (21, 9.99, 1, 1, 4), (22, 9.99, 1, 4, 3), (23, 18.50, 1, 1, 17), (24, 19.98, 2, 16, 19),
(25, 19.00, 2, 3, 7), (26, 14.99, 1, 3, 11), (27, 29.97, 3, 3, 20), (28, 9.99, 1, 2, 5), (29, 9.99, 1, 12, 13), (30, 29.99, 10, 5, 15),
(31, 95.00, 10, 20, 7), (32, 50.00, 10, 5, 9), (33, 19.98, 2, 2, 19), (34, 60.00, 3, 11, 16), (35, 104.93, 7, 15, 14), (36, 12.25, 1, 1, 12),
(37, 34.95, 5, 5, 13), (38, 14.99, 1, 12, 11), (39, 8.00, 2, 8, 10), (40, 40.00, 2, 4, 16), (41, 32.00, 8, 14, 10), (42, 9.99, 1, 4, 18),
(43, 40.00, 10, 4, 10), (44, 9.99, 1, 14, 20), (45, 9.99, 1, 20, 20), (46, 9.99, 1, 8, 20), (47, 50.00, 4, 4, 12), (48, 18.50, 1, 8, 17),
(49, 4.00, 1, 15, 10), (50, 4.00, 1, 11, 10);
```

Movies:

```
INSERT INTO movies(id, name, language, director, year)
VALUES (1, 'The Hollow Man', 'English', 'Joseph Tim', 2007), (2, 'Mission Abolished', 'English', 'Samuel Santner', 1981),
(3, 'Soldier 88', 'English', 'Mark Johnson', 2009), (4, 'Lovers Trap', 'Chinese', 'Xin Jen', 2018),
(5, 'The Biggest Illusion', 'Turkish', 'Burak Ozberg', 1999), (6, 'Imaginations', 'English', 'Jonathan Perk', 2017),
(7, 'Mystery Room', 'Hindi', 'Renuka Anand', 2021), (8, 'Back Story', 'English', 'Mansi Udit', 2009),
(9, 'Last Day on Earth', 'English', 'Rachel Samuel', 2023), (10, 'Beautiful Sunshine', 'Turkish', 'Karim Basit', 1979) ,
(11, 'Paranormal Activities', 'English', 'Jacob Preterious', 2004), (12, 'The Seven Kingdoms', 'English', 'Katrina Ross', 1963),
(13, 'Double Trouble', 'Hindi', 'Gaurav Kumar', 2019), (14, 'A Trip to Paris', 'French', 'Anthony Junior', 2008),
(15, 'Hitlist 2', 'English', 'Sussane', 2020), (16, 'Black Money', 'English', 'Jurie Oscar', 1987),
(17, 'No Time to Cry', 'English', 'Andrew Blacksmith', 1995), (18, 'The Real Independence', 'Urdu', 'Maria Sameer', 2003),
(19, 'Masterpiece', 'Spanish', 'Cyrus Alberto', 2016), (20, 'Priceless Tag', 'Chinese', 'Wang Si', 1999);
```

Genre:

```
INSERT INTO genre(id, name)
VALUES (1, 'Drama'), (2, 'Science Fiction'), (3, 'Romance'), (4, 'Comedy'), (5, 'Adventure'), (6, 'Thriller'), (7, 'Crime'), (8, 'Action'),
(9, 'Historical'), (10, 'Historical Fiction'), (11, 'Mystery'), (12, 'Fantasy'), (13, 'Horror'), (14, 'Musical'), (15, 'Animation'),
(16, 'Literature'), (17, 'Documentary'), (18, 'Martial Arts'), (19, 'Magical Fiction'), (20, 'Sports');
```

Movie_Genre:

```
INSERT INTO movie_genre(movie_id, genre_id)
VALUES (1, 6), (1, 8), (1, 11), (2, 4), (2, 8), (2, 10), (3, 7), (3, 8), (3, 10), (4, 3), (4, 6), (5, 2), (5, 12), (5, 8), (6, 12), (6, 6), (7, 11),
(7, 6), (7, 7), (8, 9), (8, 10), (9, 8), (9, 3), (9, 4), (10, 3), (10, 16), (11, 13), (11, 7), (11, 11), (11, 6), (12, 10), (12, 12), (13, 3), (13, 4),
(13, 14), (14, 14), (14, 3), (15, 7), (15, 8), (16, 5), (16, 2), (17, 1), (17, 4), (18, 9), (18, 17), (19, 3), (19, 4), (19, 14), (20, 8), (20, 18);
```

Screen:

```
INSERT INTO screen(id, type)
VALUES (1, '2D'), (2, '3D'), (3, 'LD'), (4, 'SD'),(5, 'HD'), (6, '4D'), (7, '4K'), (8, '8K');
```

Shows:

```
INSERT INTO shows(id, start_time, show_date, screen_id, movie_id)
VALUES (1, '12:30:00', '2023-04-06', 1, 4), (2, '21:30:00', '2023-04-06', 2, 2), (3,'16:45:00', '2023-04-13', 1, 1),
(4, '18:00:00', '2023-04-14', 1, 2), (5,'23:15:00','2023-04-26', 2, 1), (6, '5:30:00', '2023-03-05', 1, 4), (7, '8:30:00', '2023-02-27', 4, 20),
(8,'11:45:00', '2023-02-3', 5, 11), (9, '12:00:00', '2022-08-14', 6, 12), (10,'23:15:00','2022-11-09', 4, 1), (11, '19:30:00', '2022-11-08', 1, 4),
(12, '20:30:00', '2022-06-06', 1, 6), (13,'19:45:00', '2023-04-10', 1, 11),(14, '17:00:00', '2023-02-04', 5, 2), (15,'23:15:00','2023-02-22', 6, 1),
(16, '12:40:00', '2022-08-02', 7, 16), (17, '20:45:00', '2023-03-06', 2, 12), (18,'17:45:00', '2023-03-08', 5, 1), (19, '18:00:00', '2023-02-10', 7, 2),
(20,'23:15:00','2023-03-18', 2, 16), (21, '1:00:00', '2023-05-08', 8, 4), (22, '21:35:00', '2023-04-25', 6, 2), (23,'16:55:00', '2023-03-23', 1, 19),
(24, '18:20:00', '2022-07-14', 3, 19), (25,'23:30:00','2022-12-12', 3, 5), (26, '11:40:00', '2022-02-04', 6, 14), (27, '15:30:00', '2022-12-01', 2, 20),
(28,'14:45:00', '2022-06-19', 1, 11), (29, '13:00:00', '2022-04-10', 3, 6), (30,'12:15:00','2022-07-31', 5, 11), (31,'11:15:00','2023-03-19', 2, 19),
(32, '14:30:00', '2023-05-08', 1, 14), (33, '20:10:00', '2023-02-03', 2, 6), (34,'16:15:00', '2023-01-23', 4, 11), (35, '18:30:00', '2022-07-20', 1, 13)
(36,'22:15:00','2022-12-25', 2, 1), (37, '19:30:00', '2023-02-04', 4, 15), (38, '21:35:00', '2022-12-11', 2, 12), (39,'17:45:00', '2022-06-06', 3, 11),
(40, '19:00:00', '2022-08-17', 3, 12), (41,'13:15:00','2022-07-30', 2, 7), (42,'23:00:00','2023-03-28', 2, 14), (43, '19:35:00', '2023-05-06', 3, 4),
(44, '21:40:00', '2023-05-08', 8, 12), (45,'2:10:00', '2023-05-05', 3, 11), (46, '1:45:00', '2022-06-16', 1, 7), (47, '1:15:00','2022-11-11', 2, 19),
(48, '12:00:00', '2022-10-02', 8, 8), (49, '1:50:00', '2022-07-22', 1, 4), (50,'20:45:00', '2022-10-09', 1, 3);
```

Tickets:

-
```
INSERT INTO tickets(id, price, person_id, show_id)
VALUES (1, 600.00, 1, 4), (2, 800.00, 5, 3), (3, 1000.00, 4, 4), (4, 1200.00, 2, 1), (5, 600.00, 3, 2), (6, 1000.00, 11, 34), (7, 800.00, 15, 3),
(8, 1000.00, 4, 14), (9, 1200.00, 12, 10), (10, 1000.00, 5, 2), (11, 600.00, 1, 31), (12, 800.00, 5, 9), (13, 1200.00, 4, 44), (14, 1200.00, 2, 11
(15, 600.00, 15, 12), (16, 600.00, 11, 14), (17, 800.00, 5, 33), (18, 1000.00, 1, 3), (19, 1200.00, 8, 1), (20, 600.00, 9, 4), (21, 1200.00, 7, 4
(22, 800.00, 15, 43), (23, 1200.00, 13, 14), (24, 600.00, 20, 11), (25, 600.00, 10, 12), (26, 600.00, 10, 50), (27, 800.00, 14, 14),
(28, 1000.00, 16, 8), (29, 1200.00, 20, 10), (30, 800.00, 17, 2), (31, 600.00, 18, 2), (32, 800.00, 5, 1), (33, 1000.00, 4, 47),
(34, 1200.00, 12, 48), (35, 1200.00, 17, 42), (36, 600.00, 19, 33), (37, 800.00, 8, 3), (38, 1200.00, 8, 29), (39, 1200.00, 2, 19),
(40, 600.00, 5, 50), (41, 1200.00, 20, 32), (42, 600.00, 10, 27), (43, 800.00, 6, 36), (44, 600.00, 2, 32), (45, 800.00, 15, 25),
(46, 800.00, 11, 44), (47, 600.00, 1, 12), (48, 1200.00, 5, 40), (49, 1000.00, 20, 20), (50, 1000.00, 8, 38);
```

Seats:

```
INSERT INTO seats(id, name,  ticket_id, screen_id)
VALUES (1, 'A2', 10, 2) ,(2, 'A6', 17, 6) , (3, 'A2', 5, 2), (4, 'B1', 14, 1) , (5, 'A2', 50, 2), (6, 'A2', 11, 2) ,(7, 'B6', 12, 6) ,
(8, 'B8', 34, 8), (9, 'B1', 24, 1) , (10, 'A2', 35, 2), (11, 'A1', 1, 1) ,(12, 'A1', 26, 1) , (13, 'A2', 30, 2), (14, 'B1', 20, 1) ,
(15, 'B1', 21, 1), (16, 'B5', 27, 5), (17, 'A5', 28, 5) , (18, 'B8', 13, 8), (19, 'B1', 25, 1), (20, 'B3', 29, 3), (21, 'A8', 46, 8),
(22, 'B4', 9, 4), (23, 'B4', 6, 4), (24, 'A7', 39, 7) , (25, 'B5', 16, 5), (26, 'B1', 7, 1), (27, 'B5', 8, 5), (28, 'B3', 48, 3),
(29, 'B1', 44, 1), (30, 'B1', 19, 1), (31, 'A1', 4, 1) , (32, 'B1', 40, 1), (33, 'B1', 41, 1), (34, 'B1', 43, 1), (35, 'B2', 49, 2),
(36, 'B3', 22, 3), (37, 'B5', 23, 5), (38, 'A2', 42, 2), (39, 'B2', 36, 2), (40, 'B2', 33, 2), (41, 'A3', 38, 3), (42, 'A1', 15, 1),
(43, 'B1', 18, 1), (44, 'B2', 31, 2), (45, 'A1', 3, 1) , (46, 'B1', 32, 1), (47, 'B1', 37, 1), (48, 'A1', 45, 1), (49, 'B1', 47, 1),
```

➢ Standard Queries:

1)

```
-- 1) Name of the people starting with A along with their orders id, food id and food name, also include the people having no any order.

select p.id AS person_id,p.name AS person_name, o.id AS order_id, f.id AS food_id, f.name AS food_name from person p
LEFT JOIN orders o ON p.id = o.person_id
LEFT JOIN food f ON f.id = o.food_id
where p.name LIKE 'a%'
order by p.id, o.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

person_id	person_name	order_id	food_id	food_name
6	Arjun Kumar	NULL	NULL	NULL
12	Arif Usman	13	12	Donut
12	Arif Usman	20	1	Popcorn
12	Arif Usman	29	13	Chicken Wings
12	Arif Usman	38	11	Brownie

2)

```
116    -- 2) Name of all the movies between year 1985-1999, along with the person ids, person name who have watched it in cinema
117
118 •  select p.id AS person_id, p.name AS person_name, m.name AS movie_name, m.year from movies m
119    JOIN shows s ON m.id = s.movie_id
120    JOIN tickets t ON s.id = t.show_id
121    JOIN person p ON t.person_id = p.id
122    where year BETWEEN '1985' AND '1999'
123    order by p.id, m.year;
124
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

person_id	person_name	movie_name	year
10	Sarah Mitchel	Priceless Tag	1999
15	David Kite	The Biggest Illusion	1999
20	Samiya Khan	Black Money	1987

3)

```
24
25     -- 3) List all the shows between 2am to 12pm , from the start of 2023 until now
26
27 •   select *from shows
28     where start_time between '02-00-00' AND '12-00-00'    AND  (show_date > '2023-01-01')
29     order by show_date, start_time;
30
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	start_time	show_date	screen_id	movie_id
8	11:45:00	2023-02-03	5	11
7	08:30:00	2023-02-27	4	20
6	05:30:00	2023-03-05	1	4
31	11:15:00	2023-03-19	2	19
45	02:10:00	2023-05-05	3	11
NULL	NULL	NULL	NULL	NULL

ows 14 x

4)

```
2    -- 4) List all the person id, person name with their show date whose name of the movie contains RIP in it
3
4    select p.id AS person_id, p.name AS person_name, s.show_date, m.name AS movie_name from person p
5    JOIN tickets t ON p.id = t.person_id
6    JOIN shows s ON t.show_id = s.id
7    JOIN movies m ON s.movie_id = m.id
8    where m.name LIKE '%rip%'
9    order by p.id, s.show_date;
0    select *from movies;
1
```

sult Grid | Filter Rows: | Export: | Wrap Cell Content:

person_id	person_name	show_date	movie_name
2	Chris Mark	2023-05-08	A Trip to Paris
17	Kiara Kapoor	2023-03-28	A Trip to Paris
20	Samiya Khan	2023-05-08	A Trip to Paris

5)

```
2    -- 5) List all the people along with their id, name, who have watched the movies in screen id 7,8, their seat name as well
3
4    select p.id AS person_id, p.name AS person_name, s.screen_id, s.name AS seat_name from person p
5    JOIN tickets t ON p.id = t.person_id
6    JOIN seats s ON t.id = s.ticket_id
7    where s.screen_id IN (7, 8)
8    order by p.id;
9
```

sult Grid | Filter Rows: | Export: | Wrap Cell Content:

person_id	person_name	screen_id	seat_name
2	Chris Mark	7	A7
4	Cyrus Sen	8	B8
11	Omer Zain	8	A8
12	Arif Usman	8	B8

6)

```
150    -- 6) List All movies having genres which has substring of 'RA' in it
151
152 •  select m.id AS movie_id, m.name AS movie_name, g.name from movies m
153    JOIN movie_genre mg ON m.id = mg.movie_id
154    JOIN genre g ON g.id = mg.genre_id
155    where g.name LIKE '%ra%'
156    order by m.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

movie_id	movie_name	name
10	Beautiful Sunshine	Literature
17	No Time to Cry	Drama

7)

```
158    -- 7) List all the genres which do not falls in any movie's genre type
159 •  select g.id AS genre_id, g.name AS genre_name, m.id AS movie_id, m.name AS movie_name from genre g
160    LEFT JOIN movie_genre mg ON g.id = mg.genre_id
161    LEFT JOIN movies m ON m.id = mg.movie_id
162    where mg.genre_id IS NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

genre_id	genre_name	movie_id	movie_name
15	Animation	NULL	NULL
19	Magical Fiction	NULL	NULL
20	Sports	NULL	NULL

8)

```
164    -- 8) Get all the people having order id (3,16) and have watched movie id (1-10), also list their names and show dates
165
166 •  select p.id AS person_id, p.name AS person_name, o.id AS order_id, m.id AS movie_id, s.show_date from food f
167    JOIN orders o ON f.id = o.food_id
168    JOIN person p ON p.id = o.person_id
169    JOIN tickets t ON p.id = t.person_id
170    JOIN shows s ON s.id = t.show_id
171    JOIN movies m ON m.id = s.movie_id
172    where (o.id IN (3, 16)) AND (m.id BETWEEN 1 AND 10)
173    order by p.id, o.id, m.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

person_id	person_name	order_id	movie_id	show_date
4	Cyrus Sen	3	2	2023-04-14
4	Cyrus Sen	3	2	2023-02-04
15	David Kite	16	1	2023-04-13
15	David Kite	16	4	2023-05-06
15	David Kite	16	5	2022-12-12
15	David Kite	16	6	2022-06-06

9)

```
175    -- 9) Get all the people having seat name (B2, B6, A8)
176
177 •  select p.id AS person_id, p.name AS person_name, s.name AS seat_name from person p
178    JOIN tickets t ON p.id = t.person_id
179    JOIN seats s ON t.id = s.ticket_id
180    where s.name IN ('B2', 'B6', 'A8')
181    order by p.id;
182
```

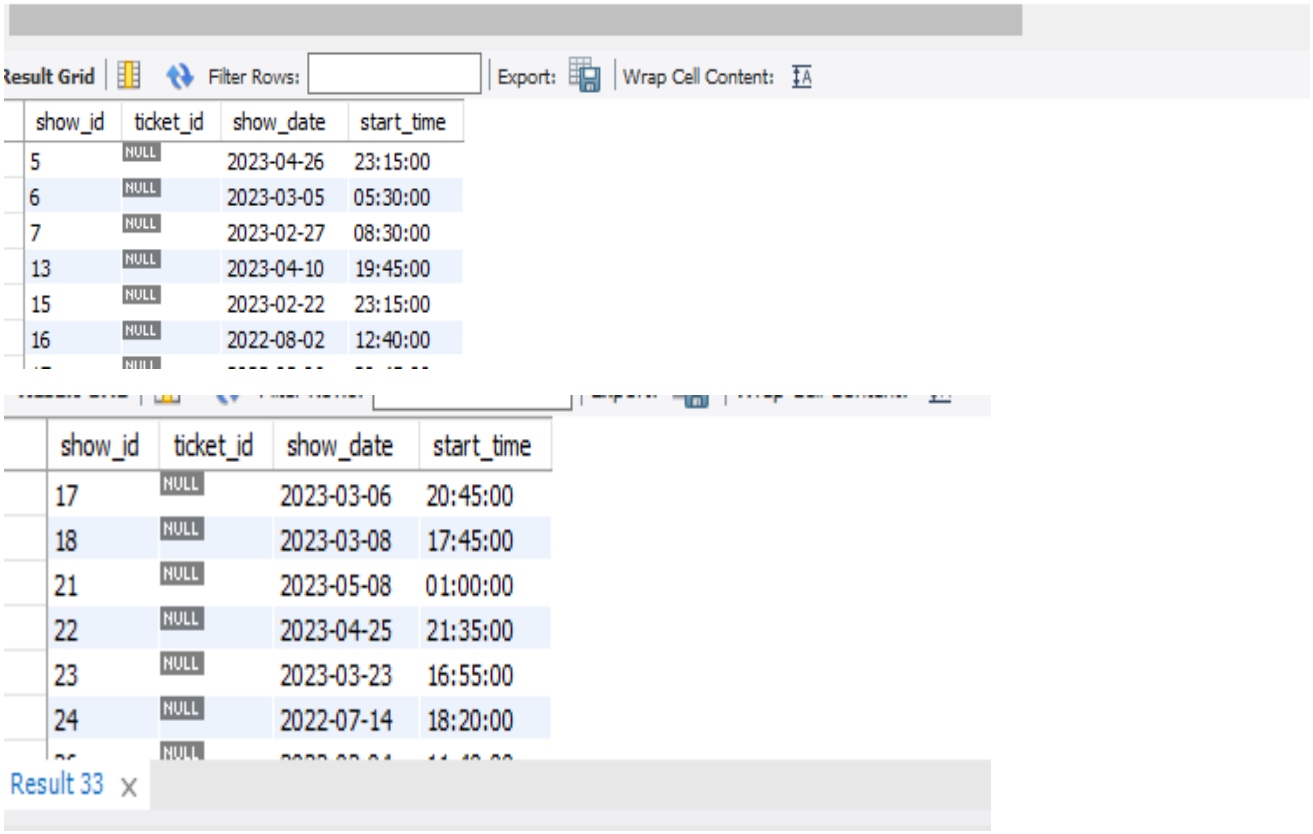
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

person_id	person_name	seat_name
4	Cyrus Sen	B2
5	Sooraj Singhania	B6
11	Omer Zain	A8
18	Katherine Ashton	B2
19	Emma Rose	B2
20	Samiya Khan	B2

10)

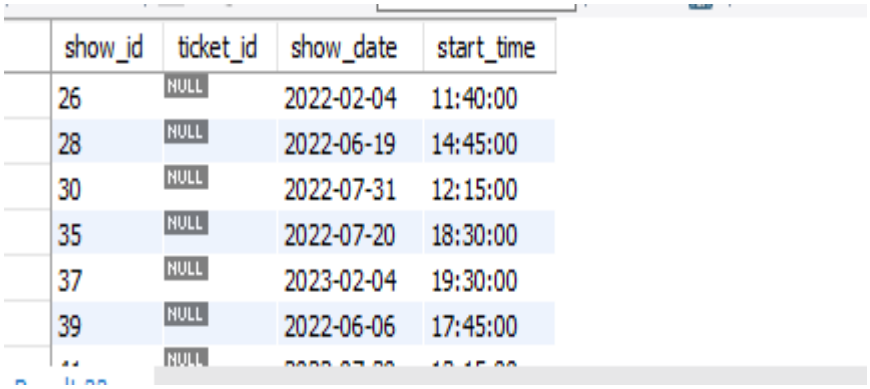
```
-- 10) Get all the shows whose tickets have not been sold

select s.id AS show_id, t.id AS ticket_id, s.show_date, s.start_time from shows s
LEFT JOIN tickets t ON s.id = t.show_id
where t.id IS NULL
order by s.id, t.id;
```



show_id	ticket_id	show_date	start_time
5	NULL	2023-04-26	23:15:00
6	NULL	2023-03-05	05:30:00
7	NULL	2023-02-27	08:30:00
13	NULL	2023-04-10	19:45:00
15	NULL	2023-02-22	23:15:00
16	NULL	2022-08-02	12:40:00
17	NULL	2023-03-06	20:45:00
18	NULL	2023-03-08	17:45:00
21	NULL	2023-05-08	01:00:00
22	NULL	2023-04-25	21:35:00
23	NULL	2023-03-23	16:55:00
24	NULL	2022-07-14	18:20:00

Result 33



show_id	ticket_id	show_date	start_time
26	NULL	2022-02-04	11:40:00
28	NULL	2022-06-19	14:45:00
30	NULL	2022-07-31	12:15:00
35	NULL	2022-07-20	18:30:00
37	NULL	2023-02-04	19:30:00
39	NULL	2022-06-06	17:45:00

show_id	ticket_id	show_date	start_time
37	NULL	2023-02-04	19:30:00
39	NULL	2022-06-06	17:45:00
41	NULL	2022-07-30	13:15:00
45	NULL	2023-05-05	02:10:00
46	NULL	2022-06-16	01:45:00
49	NULL	2022-07-22	01:50:00

11)

```
190   -- 11) Get all the people who have watched shows in screen id 8 along with their show date, time and movie name
191
192 • select p.id AS person_id, p.name AS person_name, s.id AS show_id,s.screen_id, s.show_date, s.start_time, m.name AS movie_name from person p
193   JOIN tickets t ON p.id = t.person_id
194   JOIN shows s ON s.id = t.show_id
195   JOIN movies m ON m.id = s.movie_id
196   where s.screen_id = 8
197   order by p.id, s.show_date, s.start_time;
198
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

person_id	person_name	show_id	screen_id	show_date	start_time	movie_name
4	Cyrus Sen	44	8	2023-05-08	21:40:00	The Seven Kingdoms
11	Omer Zain	44	8	2023-05-08	21:40:00	The Seven Kingdoms
12	Arif Usman	48	8	2022-10-02	12:00:00	Back Story

12)

```
9   -- 12) Get all the food items along with their ids who have not been placed in any order
0
1 • select f.id AS food_id, f.name AS food_name, o.id AS order_id from food f
2   LEFT JOIN orders o ON o.food_id = f.id
3   where o.food_id is NULL
4   order by f.id;
5
```

sult Grid | Filter Rows: | Export: | Wrap Cell Content:

food_id	food_name	order_id
6	Nuggets	NULL