

**Department of Computer Science**  
**BSCS(Hons)**  
**4<sup>th</sup> Semester**

**Database Systems**

**Project**  
**Cinema Information Management System**  
**Submitted to:**

**Sir Nadeem Zafar**  
Department of Computer Science  
GC University, Lahore

**Submitted by:**  
Jaweria Fayyaz 0237-BSCS-21  
Maira Tanveer 0265-BSCS-21

**Section: E1**

**Table Of Contents:**

<b>History Of Organization.....</b>	<b>2</b>
<b>Define Problems and Constraints.....</b>	<b>2</b>
<b>Define Objectives.....</b>	<b>2</b>
<b>Define Scope and Boundaries.....</b>	<b>3</b>
<b>Scenario.....</b>	<b>3</b>
<b>Noun Verb Analysis.....</b>	<b>4</b>
<b>Entity Relationship Diagram.....</b>	<b>5</b>
<b>Abnormal Form.....</b>	<b>5</b>
<b>Normalized Form.....</b>	<b>6</b>
<b>Relational Model.....</b>	<b>7</b>
<b>Software Requirements.....</b>	<b>7</b>
<b>Implementation.....</b>	<b>8</b>
<b>Testing.....</b>	<b>13</b>

# **CINEMA INFORMATION MANAGEMENT SYSTEM**

## **History of Organization**

Cinepax is the first cinema in Pakistan that is providing a world-class movie experience to people of Pakistan by building a state of the art movie theatre in urban cities. Cinepax is the first dedicated Cineplex company in Pakistan that is building the country's first nationally branded Cineplex chain. It is targeting larger cities of Pakistan; Karachi, Lahore, Islamabad, Faisalabad, Gujranwala, Multan and Hyderabad. The company plans to develop 120 screens over 5 years. The cinema complex would screen premium contents in a family friendly environment having world class seating arrangements and air conditioned halls that would be open from 12 noon to 12 midnight. The Cinepax Cinema Packages Mall Lahore shows English, Chinese, Spanish, Turkish, Urdu, French movies. It is spread over an area of approximately 30 acres. Total cinema screens are 7 and halls are equipped with total 1400 seats.

## **Define Problems and constraints**

Since the existing system is completely manual so it is really very difficult to maintain and to keep a full record about the daily purchase of tickets. The information is not up to date and manual system requires staff to maintain their records on registers. It creates many problems such as Black ticketing, duplication of data or information is present which creates an error. The need is to computerized the system of cinepax cinema.

General Constraints: No constraints have been noticed for the system.

## **Define Objectives**

Our objective is to create the database for their business to keep detailed record of all transactions which would definitely enhance it's business and will lure the people towards the cinema. Cinepax is also planning to increase the number of screens which will boast the number of audience resulting in the success of

business. The database that is projected is able to overcome all flaws that are mentioned before and will be helpful for further expansion.

## **Define Scope and Boundaries**

Development of Cinema Information Management System for Cinepax helps in managing the database for storing and retrieving required information about person, food\_item, screen, ticket, seat, movie and genre. Our scope is around these entities. The software will provide ability to staff to enter the new records of the movies, update and also delete them. It will also allow keeping the record of the total tickets sold. The machine which runs this system is not too heavy. This system will run on home computer also.

## **Scenario**

Cinepax Cinema Packages Mall Lahore is dedicated towards providing a best international quality cinema experience to the citizens of Pakistan. The Cinema complex would screen premium contents in a family friendly environment, now wants to build a database system to manage records for food item, ticket, screen, movie, genre etc. You were hired as an analyst. Every person has a name and mobile numbers. The system will assign id to every person. A person can order no any or maybe many food items and a food item can be ordered by many as well as no person. For food item, organization keeps record for id, name and its price. A person can book ticket(s) of one or many show(s) however a show can have many people with tickets or no one. For ticket, organization wants to store its id and price. A single ticket must allot a single seat. The cinema keeps record for id and name for every seat. A movie can be displayed on multiple screens having multiple show times, also one screens can have shows of multiple movies or no any. Every screen has a type. For movie, name, language, year and name of director will be recorded. The system assign id to every screen and movie. A screen can have multiple seats starting from one at least. A movie belongs to any specific movie-genre from many genres. Every genre has a specific id and name. A movie belongs to many genres starting from one and a single genre can have many or no any movie belonging to it.

## Noun Verb Analysis

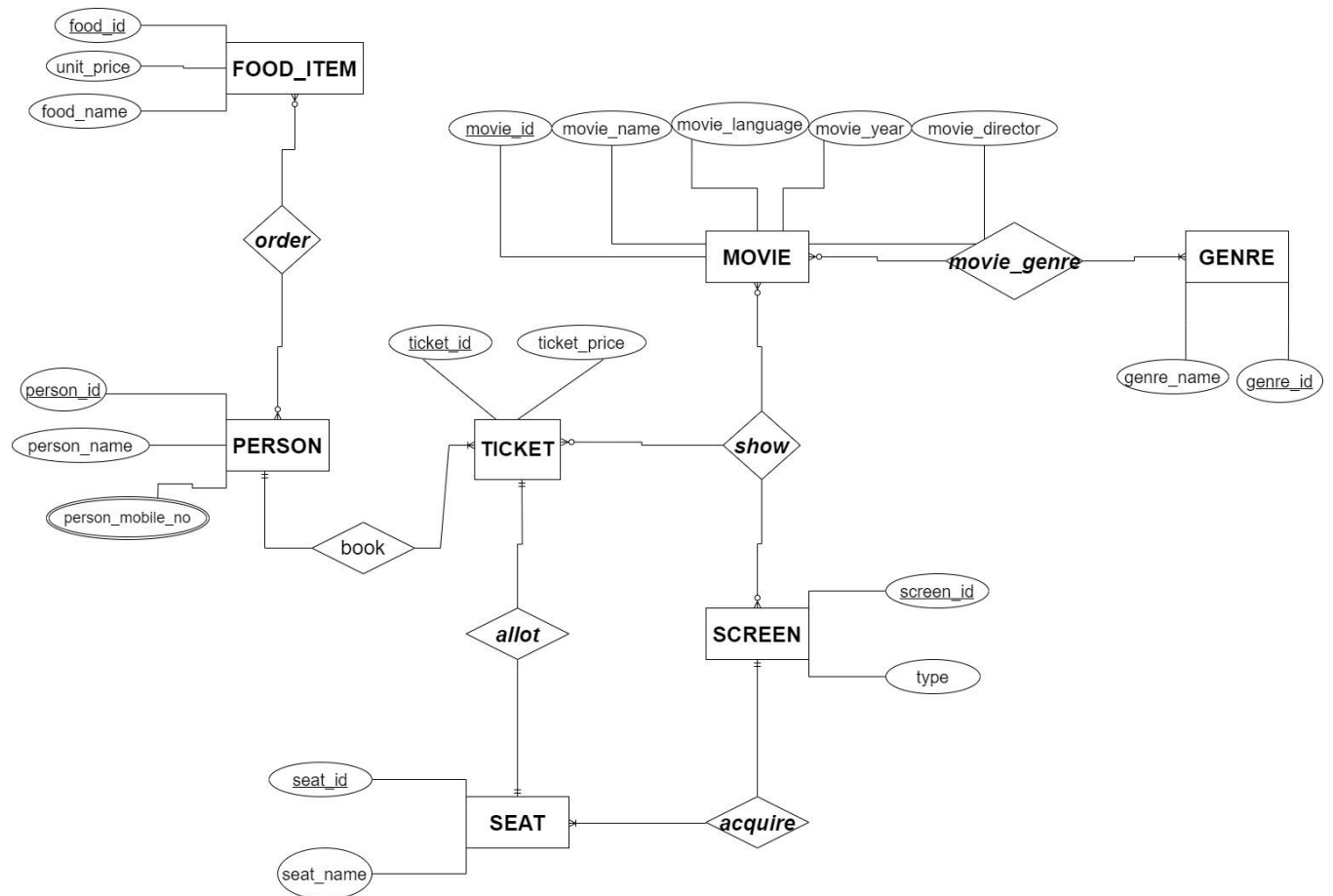
Nouns	Adjectives
Person	person_id, person_name, person_mobilenno
Order (junction entity)	order_id, order_quantity, total_price
Food_item	food_id, food_name, unit_price
Ticket	ticket_id, ticket_price
Seat	seat_id, seat_name
Screen	screen_id, screen_name
Show(associative entity)	show_id, show_date, start_time
Movie	movie_id, movie_name, movie_language, movie_director, movie_year
Genre	genre_id, genre_name
Movie_genre (associative entity)	movie_id, genre_id (it will only have primary keys of movie entity and genre entity as foreign keys)

## Verbs:

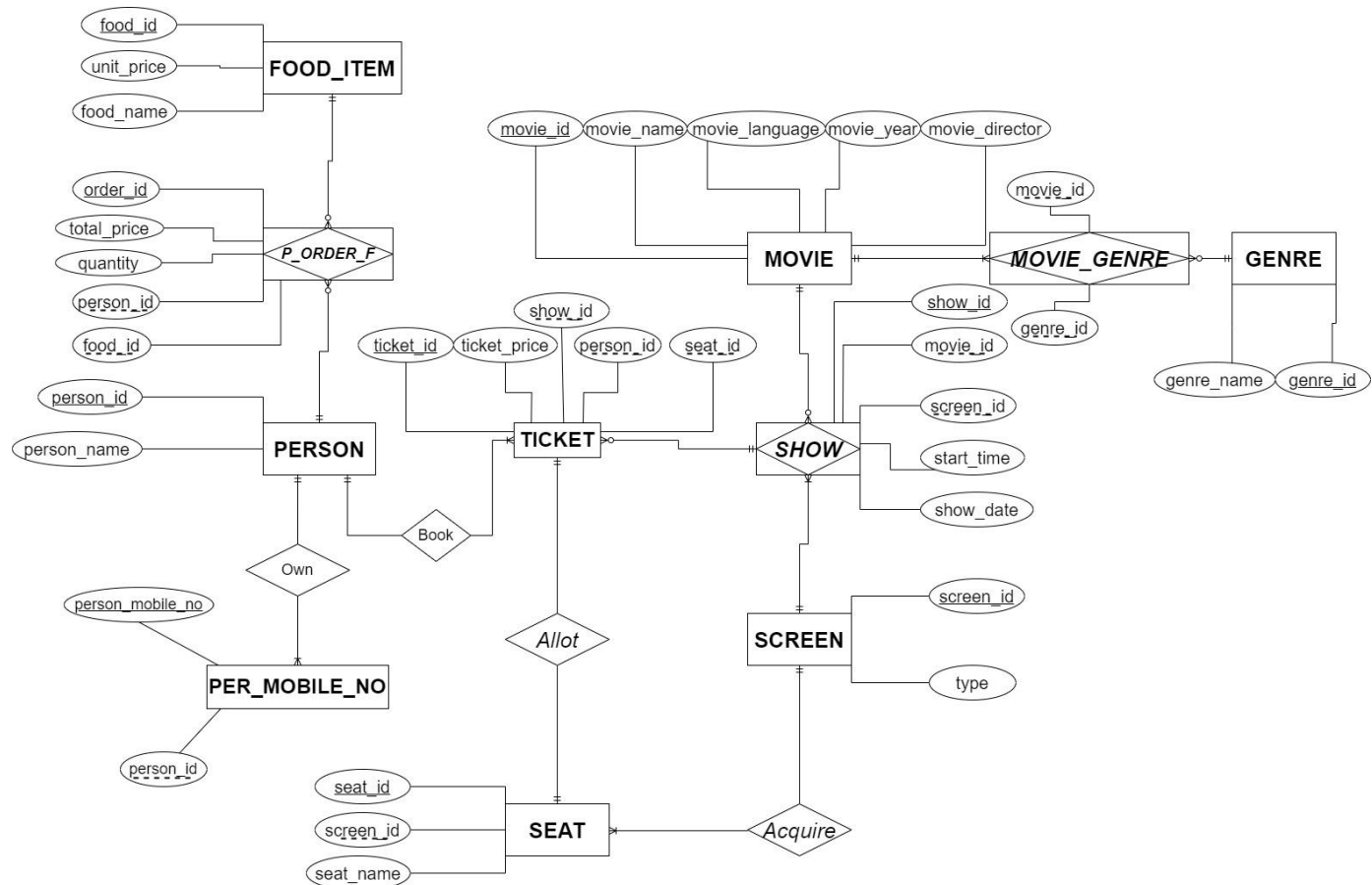
- A person can **order** a food\_item.
- A person can **buy** ticket(s) for show(s).
- A ticket **allot** a seat.
- A screen can **acquire** many seats.
- A screen has a **show** for a movie.
- A movie belongs to any specific **movie\_genre** from many genres.

# ERD(Entity Relationship Diagram)

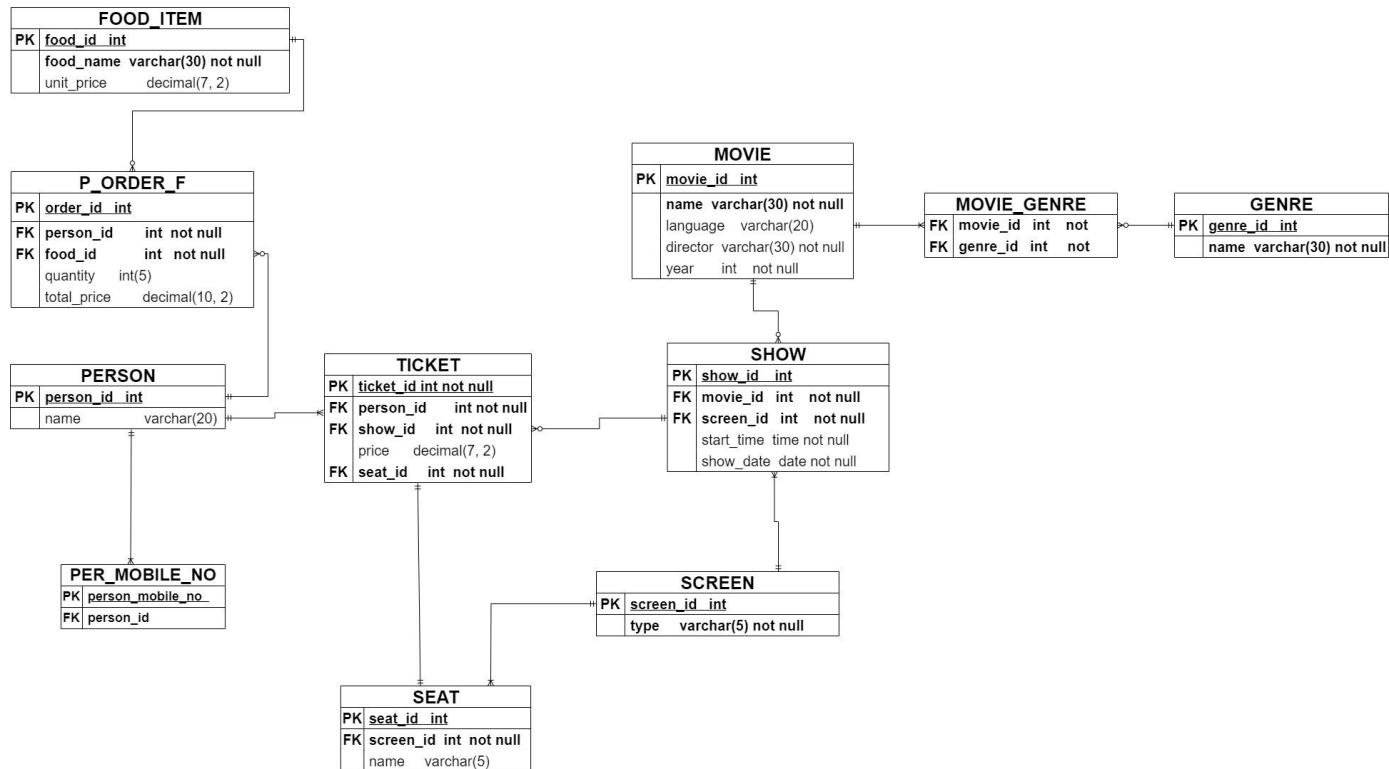
## Abnormal Form:



## Normalized Erd:



# Relational Model



## Software Requirements

- Microsoft Sql Server Management System
- Draw.io for ERD creation

# **Implementation**

## **Tables:**

### **Person:**

```
create table person(  
person_id int not null,  
person_name varchar(20),  
constraint person_pk primary key(person_id)  
)
```

### **Per\_Mobile\_No:**

```
create table per_mobile_no(  
person_mobile_no char(11) not null,  
person_id int not null,  
constraint mobno_pk primary key(person_mobile_no),  
constraint per_mobno_fk foreign key(person_id)  
references person(person_id)  
)
```

### **Food\_Item:**

```
create table food_item  
(  
food_id int not null,  
food_unit_price decimal(7,2) not null,  
food_name varchar(30) not null,  
constraint f_pk primary key(food_id)  
)
```

### **P\_Order\_F:**

```
create table p_order_f(  
order_id int not null,  
order_total_price decimal(10,2) null,  
order_quantity int,  
person_id int not null,  
food_id int not null,  
constraint order_pk primary key(order_id),  
constraint order_per_fk foreign key(person_id)  
references person(person_id),  
constraint order_food_fk foreign key (food_id)
```



```
references food_item(food_id)
)
```

### **Screen:**

```
create table screen
(
screen_id int not null,
screen_type varchar(7) not null,
constraint screen_pk primary key(screen_id)
)
```

### **Movie:**

```
create table movie(
movie_id int not null,
movie_name varchar(40) not null,
movie_language varchar(20) not null,
movie_year int,
movie_director varchar(30) not null,
constraint movie_pk primary key(movie_id)
)
```

### **Genre:**

```
create table genre(
genre_id int not null,
genre_name varchar(20) not null,
constraint genre_pk primary key (genre_id)
)
```

### **Movie\_Genre:**

```
create table movie_genre
(
movie_id int not null,
genre_id int not null,
constraint m_g_m_fk foreign key(movie_id)
references movie(movie_id),
constraint m_g_g_fk foreign key(genre_id)
references genre(genre_id)
)
```

### **Seat:**

```
create table seat(
seat_id int not null,
```

```
seat_name varchar(10) null,  
screen_id int not null,  
constraint seat_pk primary key(seat_id),  
constraint seat_screen_fk foreign key(screen_id)  
references screen(screen_id)  
)
```

### Show:

```
create table show(  
show_id int not null,  
show_start_time time not null,  
show_date date not null,  
screen_id int not null,  
movie_id int not null,  
constraint show_pk primary key(show_id),  
constraint screen_show_fk foreign key(screen_id)  
references screen(screen_id),  
constraint movie_show_fk foreign key(movie_id)  
references movie(movie_id)  
)
```

### Ticket:

```
create table ticket(  
ticket_id int not null,  
ticket_price decimal(7,2),  
person_id int not null,  
seat_id int not null,  
show_id int not null,  
constraint ticket_pk primary key(ticket_id),  
constraint person_ticket_fk foreign key(person_id)  
references person(person_id),  
constraint seat_ticket_fk foreign key(seat_id)  
references seat(seat_id),  
constraint show_ticket_fk foreign key(show_id)  
references show(show_id)  
)
```

## Insert Data:

### Person:

```
insert into person(person_id,person_name)
values('1','Maira'), ('2','Jiya'), ('3','Saira'), ('4','Hamza'), ('5','Hamid'), ('6','Haziq'),
('7','shoqia'), ('8','Maria'), ('9','Jasia'), ('10','Saira'), ('11','Hamza'), ('12','David'),
('13','Hadiq'), ('14','Shizuka'), ('15','Hadia'), ('16','Hina')
```

### Per\_Mobile\_No:

```
insert into per_mobile_no(person_id, person_mobile_no)
values('1','03004475160'), ('2','03014475160'), ('3','03114475160'),
('4','03004475750'), ('5','03007775160'), ('6','03004478160'),
('7','03224475160'), ('8','03008875160'), ('9','03004495160'),
('10','03004475188'), ('11','03004443160'), ('12','03334475160'),
('13','03924475160'), ('14','03009975160'), ('15','03008075160'),
('16','03004473760')
```

### Food\_Item:

```
insert into food_item(food_id,food_unit_price,food_name)
values('1','50.0','Lays'), ('2','40.0','Fries'), ('3','150.0','Coffee'), ('4','130.0','Tea'),
('5','350.0','Pizza'), ('6','50.0','Ice cream'), ('7','40','Popcorn'),
('8','130.0','Sandwitch'), ('9','150','Donut'), ('10','350.0','Club Sandwiches'),
('11','250.0','Loaded Fries'), ('12','340','Chicken Patty'), ('13','130.0','Zinger
burger'), ('14','150','Nuttillicious'), ('15','330.0','Brownie'), ('16','250','Nuttela
Shake')
```

### P\_Order\_F:

```
insert into
p_order_f(order_id,order_total_price,order_quantity,person_id,food_id)
values('1','100.0','2','2','1'), ('2','80.0','2','16','2'), ('3','300.0','2','15','3'),
('4','120.0','3','5','7'), ('5','700.0','2','13','5'), ('6','150.0','1','10','9'),
('7','660','4','8','15'), ('8','500.0','2','13','16'), ('9','300.0','2','10','9'),
('10','680','4','8','12')
```

### Screen:

```
insert into screen(screen_id,screen_type)
values('1','2d'), ('2','3d'), ('3','Ld'), ('4','sd'), ('5','hd'), ('6','4d'), ('7','4k'), ('8','8k'),
('9','8k'), ('10','8k'), ('11','4d'), ('12','ld'), ('13','sd'), ('14','3d'), ('15','hd')
```

### Seat:

```
insert into seat(seat_id,seat_name,screen_id)
```

```
values('1','A2','2'), ('2','A6','5'), ('3','B1','15'), ('4','B1','2'), ('5','B8','9'),
('6','A6','13'), ('7','B4','12'), ('9','A6','4'), ('10','A3','11'), ('11','B5','5'),
('12','B8','7'), ('8','B2','7')
```

## Movie:

**insert into**

```
movie(movie_id,movie_name,movie_language,movie_year,movie_director)
```

```
values('1','Hollow Man','English','2007','Joseph'), ('2','Mission
Abolished','English','1981','Samuel Santer'), ('3','Solider
88','English','2009','Mark Johnson'), ('4','Lovers Trap','Chinese','2018','Xin
Gen'), ('5','Masterpiece','Spanish','2016','Cyrus Alberto'), ('6','The Biggest
Illusion','Turkish','1999','Burak Ozberg'), ('7','Priceless
Tag','Chinese','1999','Wang Si'), ('8','Imaginations','English','2017','Jonathan
Perk'), ('9','The Real Independence','Urdu','2003','Maria Sameer'), ('10','Black
Honey','English','1987','Jurie Oscar'), ('11','Last Day On
Earth','English','2023','Rachel'), ('12','A Trip to Paris ','French','2008','Anthony
Junior '), ('13','Hitlist 2','English','2020','Sussane'), ('14','The Seven Kingdoms
','English','1963','Katrina Ross'), ('15','Beautiful
Sunshine','Turkish','1979','Karim Basit'), ('16','No Time To
Cry','English','1995','Andrew Blacksmith')
```

## Genre:

**insert into** genre(genre\_name,genre\_id)

```
values('comedy','1'), ('fiction','2'), ('action','3'), ('drama','4'), ('science fiction','5'),
('romance','6'), ('adventure','7'), ('thriller','8'), ('crime','9'), ('fantasy','10'),
('literature','11'), ('Animation','12'), ('martial arts','13'), ('horror','14'),
('mystery','15')
```

## Movie\_Genre:

**insert into** movie\_genre(movie\_id,genre\_id)

```
values('4','6'), ('1','15'), ('2','8'), ('8','7'), ('12','7'), ('5','15'), ('7','9'), ('6','5'), ('9','3'),
('10','7')
```

## Show:

**insert into** show(show\_id,show\_start\_time,show\_date,screen\_id,movie\_id)

```
values('1','10:30:00','2023-04-06','1','2'), ('2','11:30:00','2023-05-07','1','2'),
('3','12:00:00','2022-07-06','1','2'), ('4','1:50:00','2021-04-06','1','2'),
('5','2:45:00','2022-01-06','1','2'), ('6','4:17:00','2020-03-08','1','2'),
('7','8:30:00','2019-04-06','1','2'), ('8','9:50:00','2016-12-06','1','2'),
('9','7:30:00','2018-09-07','1','2'), ('10','10:10:00','2015-02-08','1','2'),
('11','8:20:00','2014-05-06','1','2')
```

## Ticket:

```
insert into ticket(ticket_id,ticket_price,person_id,seat_id,show_id)
values('1','700.00','1','1','1'), ('2','800.00','2','2','1'), ('3','900.00','3','4','1'),
('4','1000.00','4','3','2'), ('5','1200.00','5','6','3'), ('6','1300.00','6','5','3'),
('7','1800.00','7','8','2'), ('8','2000.00','8','7','4'), ('9','1900.00','9','10','5'),
('10','2300.00','10','11','6'), ('11','2500.00','11','9','7'), ('12','2600.00','12','11','8'),
('13','2700.00','13','12','7')
```

## Testing:

### Group by:

The screenshot displays the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

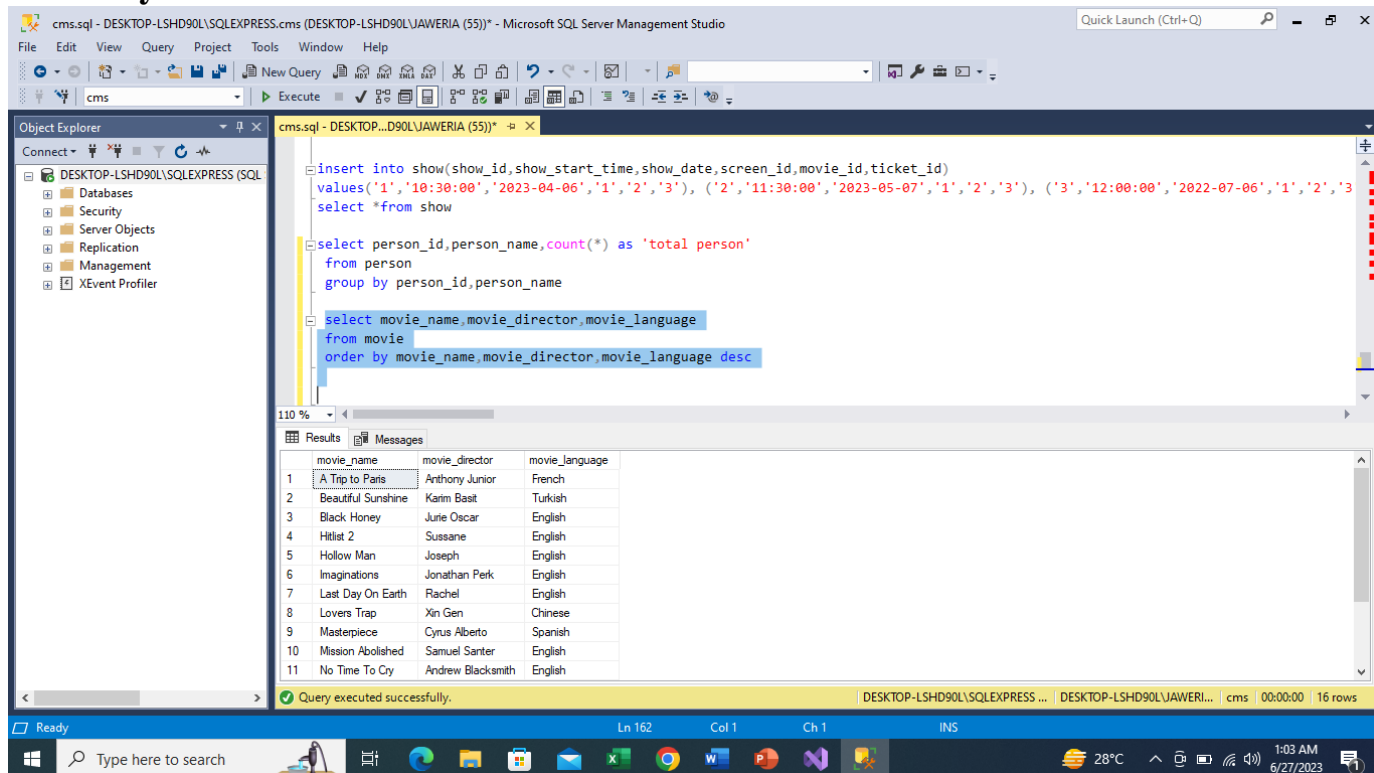
```
select * from seat
insert into ticket(ticket_id,ticket_price,person_id,seat_id)
values('1','700.00','1','1'), ('2','800.00','2','2'), ('3','900.00','3','4'), ('4','1000.00','4','3'), ('5','1200.00','5','6'), ('6','1300.00','6','5'), ('7','1800.00','7','8'), ('8','2000.00','8','7'), ('9','1900.00','9','10'), ('10','2300.00','10','11'), ('11','2500.00','11','9'), ('12','2600.00','12','11'), ('13','2700.00','13','12')
select * from ticket
insert into show(show_id,show_start_time,show_date,screen_id,movie_id,ticket_id)
values('1','10:30:00','2023-04-06','1','2','3'), ('2','11:30:00','2023-05-07','1','2','3'), ('3','12:00:00','2022-07-06','1','2','3')
select * from show
select person_id,person_name,count(*) as 'total person'
from person
group by person_id,person_name
```

The Results pane shows the output of the final query, which is a group by statement. The results are as follows:

person_id	person_name	total person
1	Maira	1
2	Jiya	1
3	Saira	1
4	Hamza	1
5	Hamid	1
6	Haziq	1
7	shooja	1
8	Maria	1
9	Jasia	1
10	Saira	1
11	Hamza	1

The status bar at the bottom indicates that the query was executed successfully and returned 16 rows.

## Order by:



```
insert into show(show_id,show_start_time,show_date,screen_id,movie_id,ticket_id)
values('1','10:30:00','2023-04-06','1','2','3'), ('2','11:30:00','2023-05-07','1','2','3'), ('3','12:00:00','2022-07-06','1','2','3')
select *from show

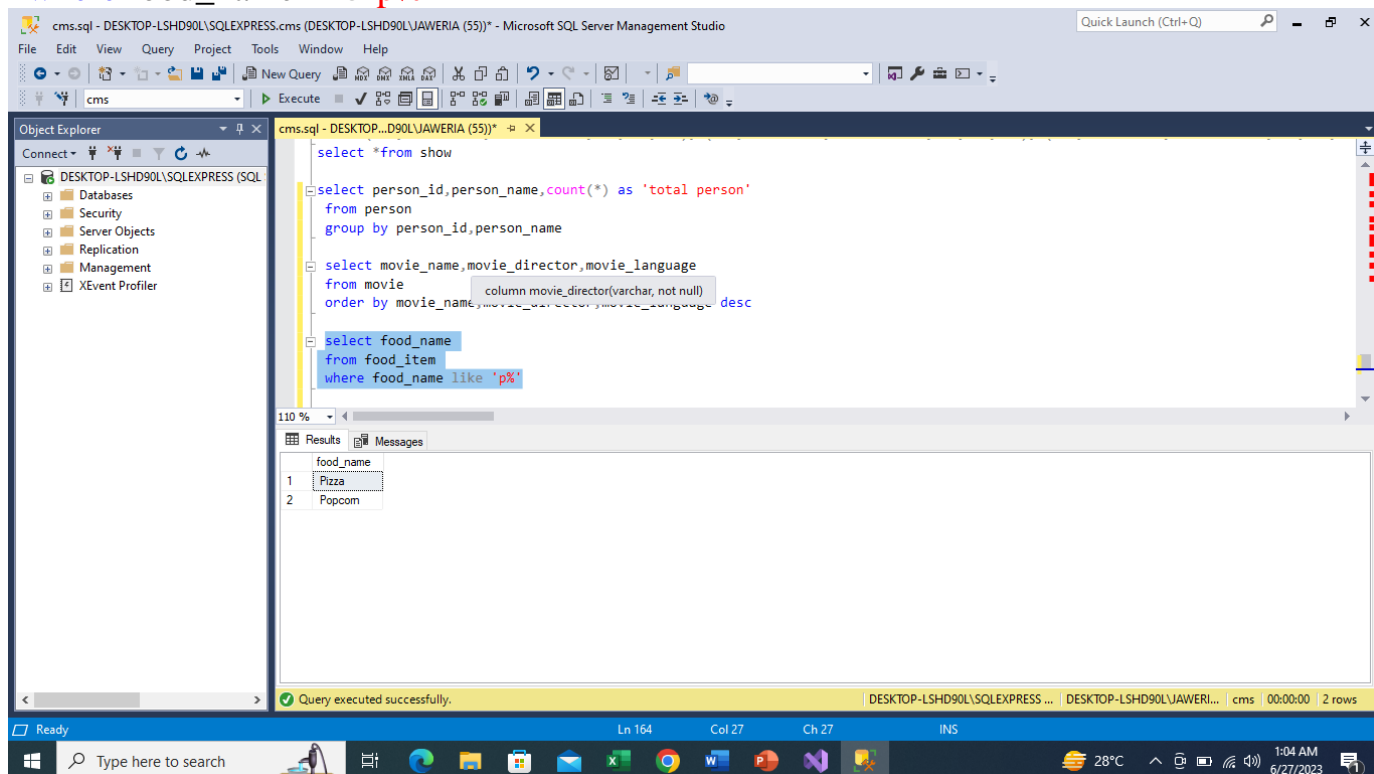
select person_id,person_name,count(*) as 'total person'
from person
group by person_id,person_name

select movie_name,movie_director,movie_language
from movie
order by movie_name,movie_director,movie_language desc
```

	movie_name	movie_director	movie_language
1	A Trip to Paris	Anthony Junior	French
2	Beautiful Sunshine	Karim Basit	Turkish
3	Black Honey	Jurie Oscar	English
4	Httlat 2	Sussane	English
5	Hollow Man	Joseph	English
6	Imaginations	Jonathan Perk	English
7	Last Day On Earth	Rachel	English
8	Lovers Trap	Xin Gen	Chinese
9	Masterpiece	Cyrus Alberto	Spanish
10	Mission Abolished	Samuel Santer	English
11	No Time To Cry	Andrew Blacksmith	English

## Like:

```
select food_name
from food_item
where food_name like 'p%'
```



```
select *from show

select person_id,person_name,count(*) as 'total person'
from person
group by person_id,person_name

select movie_name,movie_director,movie_language
from movie
column movie_director(varchar, not null)
order by movie_name,movie_director,movie_language desc

select food_name
from food_item
where food_name like 'p%'
```

	food_name
1	Pizza
2	Popcorn

## Update:

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
select food_name
from food_item
where food_name like 'p%'

select person_name
from person
where person_name like 'h%'

update food_item
set food_name= 'green tea'
where food_id = 4
select* from food_item
```

The Results pane displays the output of the query, showing a table with 11 rows and 3 columns: food\_id, food\_unit\_price, and food\_name.

food_id	food_unit_price	food_name
1	50.00	Lays
2	40.00	Fries
3	150.00	Coffee
4	130.00	green tea
5	350.00	Pizza
6	50.00	Ice cream
7	40.00	Popcom
8	130.00	Sandwich
9	150.00	Donut
10	350.00	Club Sandwiches
11	250.00	Loaded Fries

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-LSHD90L\SQLEXPRESS ... | DESKTOP-LSHD90L\JAWERIA ... | cms | 00:00:00 | 16 rows".

## Join:

Display which person order which food\_item

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
where person_name like 'h%'

update food_item
set food_name= 'green tea'
where food_id = 4
select* from food_item
--join

select person.person_name, food_item.food_name
from person inner join p_order_f
on person.person_id = p_order_f.person_id
inner join food_item
on p_order_f.food_id = food_item.food_id
```

The Results pane displays the output of the query, showing a table with 10 rows and 2 columns: person\_name and food\_name.

person_name	food_name
Jiya	Lays
Hina	Fries
Hadia	Coffee
Hamid	Popcom
Hadiq	Pizza
Saira	Donut
Maria	Brownie
Hadiq	Nuttela Shake
Saira	Donut
Maria	Chicken Patty

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-LSHD90L\SQLEXPRESS ... | DESKTOP-LSHD90L\JAWERIA ... | cms | 00:00:00 | 10 rows".

# Display which movie belongs to which genre

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains two SQL queries. The first query joins the person, p\_order\_f, and food\_item tables. The second query joins the movie and movie\_genre tables. The results pane shows the output of the second query, displaying a list of movies and their corresponding genres.

```
select person.person_name, food_item.food_name
from person inner join p_order_f
on person.person_id = p_order_f.person_id
inner join food_item
on p_order_f.food_id = food_item.food_id

select movie.movie_name, genre.genre_name
from movie inner join movie_genre
on movie.movie_id = movie_genre.movie_id
inner join genre
on movie_genre.genre_id = genre.genre_id
```

	movie_name	genre_name
1	Lovers Trap	romance
2	Hollow Man	mystery
3	Mission Abolished	thriller
4	Imaginations	adventure
5	A Trip to Paris	adventure
6	Masterpiece	mystery
7	Priceless Tag	crime
8	The Biggest Illusion	science fiction
9	The Real Independence	action
10	Black Honey	adventure

Query executed successfully.