REPORT

On

CINEMA DATABASE PROJECT

For

FUNDAMENTAL DATABASE (CS2231)

Section 3

Submitted by

RAMA SABBAGH

ID: 443011958

HANADI ALSHAWESH

ID:443011994

Supervisor:

Dr. Aisha Alsiyami

DEPARTMENT OF SOFTWARE ENGINEERING

COLLEGE OF INFORMATION SYSTEM

UMM AL-QURA UNIVERSITY

Jan 2024

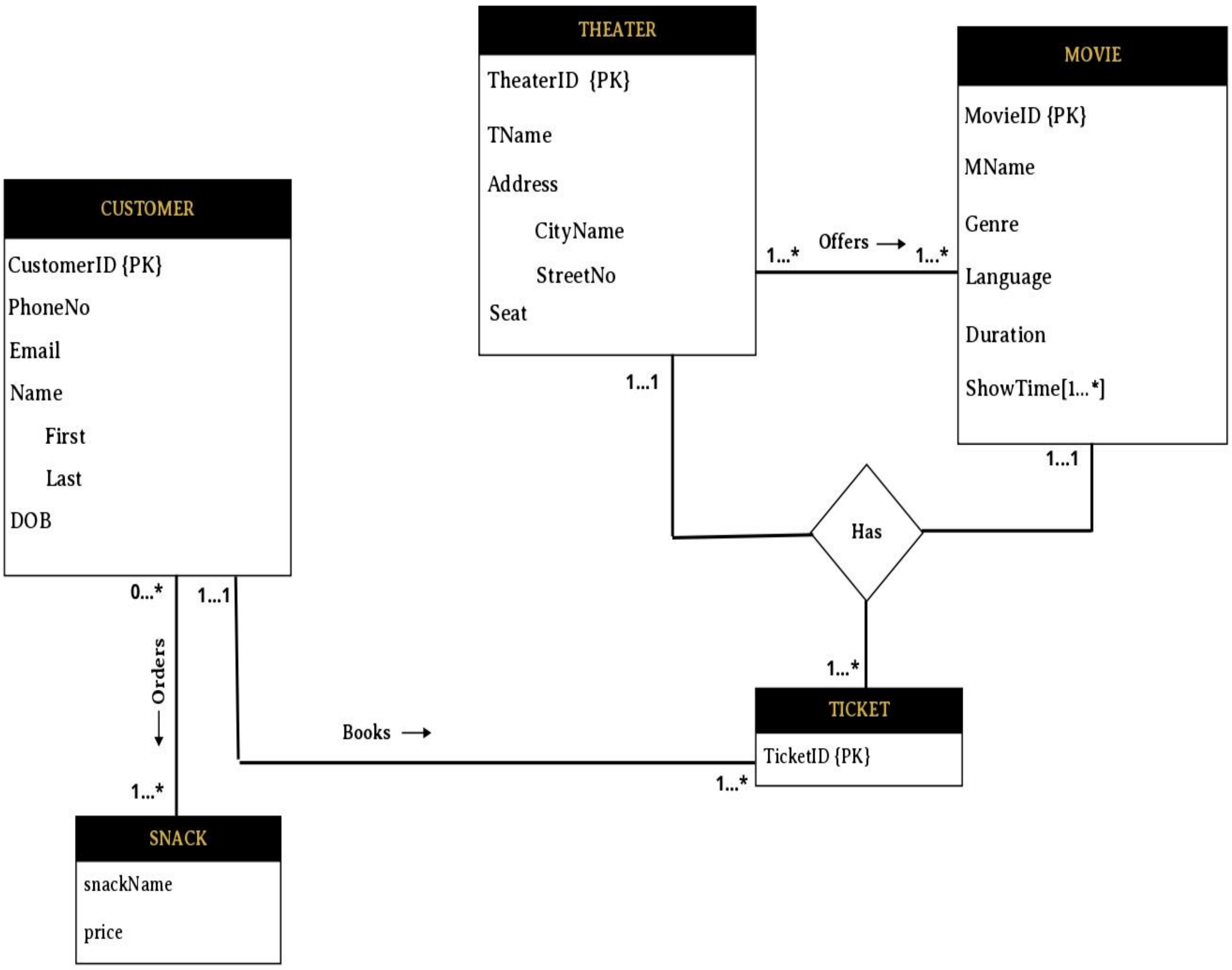
Group work report:

Cinema Database	Rama	Hanadi
Business Rules	**50%**	**50%**
Chen Notation	**50%**	**50%**
UML Notation	**50%**	**50%**
Mapping	**50%**	**50%**
Normalization	**50%**	**50%**
Schema	**50%**	**50%**
Query	**50%**	**50%**

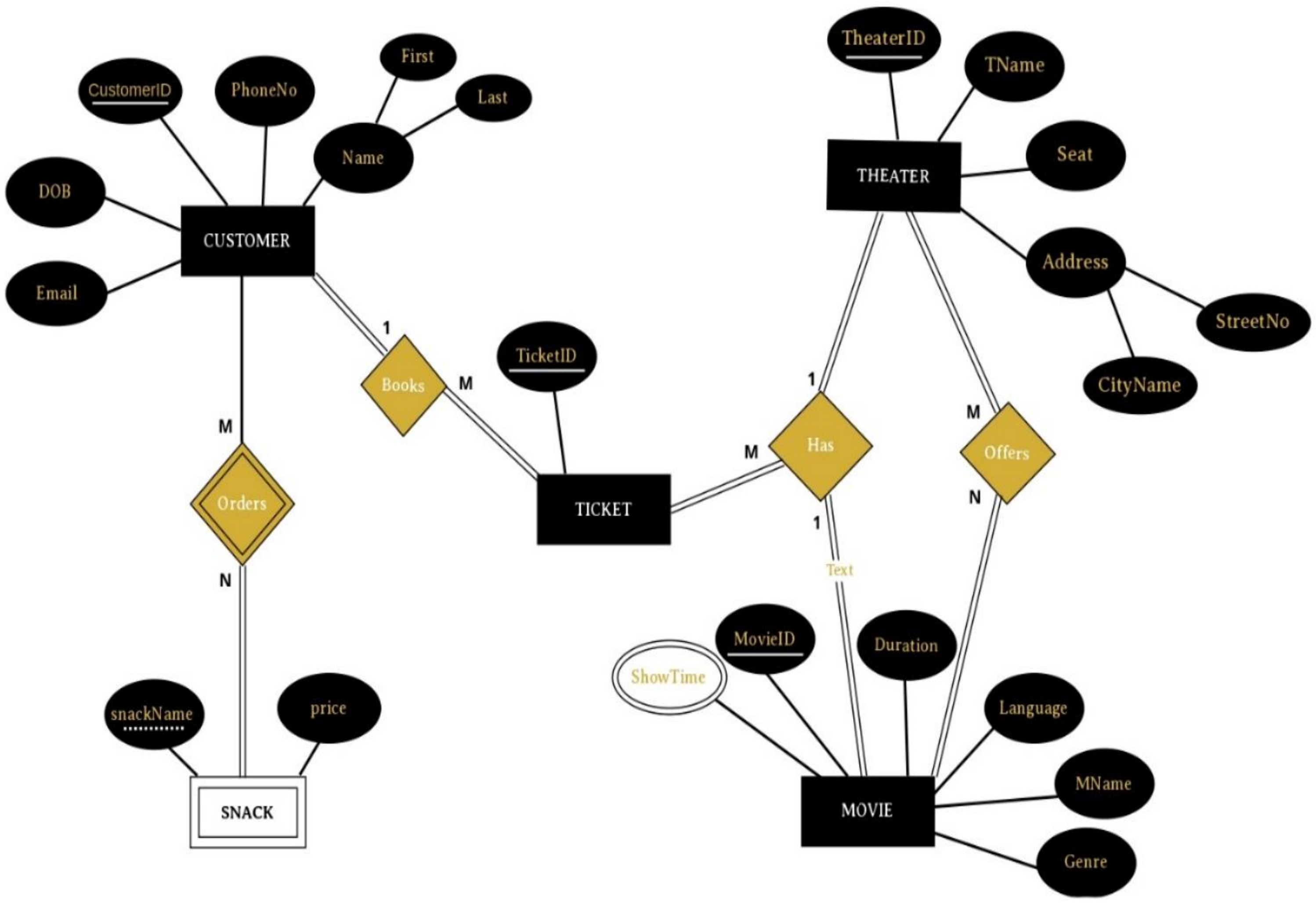
Business rules (Database Relationships):

- The cinema has many theaters, a theater offers many movies, and each movie can be offered by many theaters. Each theater has a theater ID, name, seat, and address consisting of city name, and street number.
- The database will store each customer ID, email, phone number, date of birth, and name consisting of first name and last name.
- Each customer may choose to order multiple snacks, and each snack must be ordered by at least one customer. The snack includes popcorn and soft drinks.
- Each customer can book up to many tickets, while each ticket can be booked by one customer.
- Each movie has a unique movie ID, name, genre, language, duration, and at least one Showtime.
- Every ticket has a designated movie at a specific theater. Each movie has a relationship with one theater, and a theater has connections with multiple tickets. Concurrently, one movie can also have associations with multiple tickets. Every single ticket must have a unique ticket ID.

UML Notation



Chen Notation



Mapping

Step1: Mapping of Regular Entity types

Customer

CustomerID	Fname	Lname	Email	DOB	PhoneNo

Ticket

TicketID

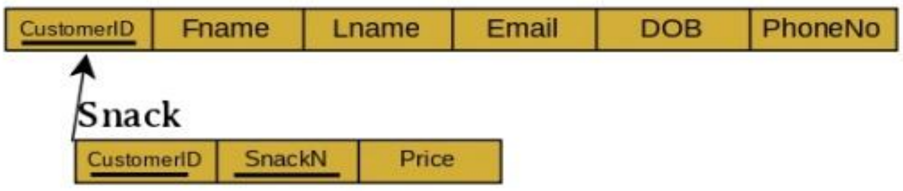
Movie

MovieID	Duration	Language	Mname	Genre

Theater

TheaterID	T_name	Seat	City_N	Street_No

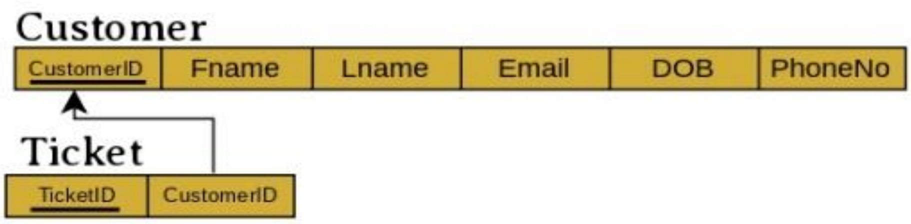
Step2: Mapping of Week Entity types



Step3 : Mapping of Binary 1:1 Relationship Types

There are no 1:1 Relationships in the ERD.

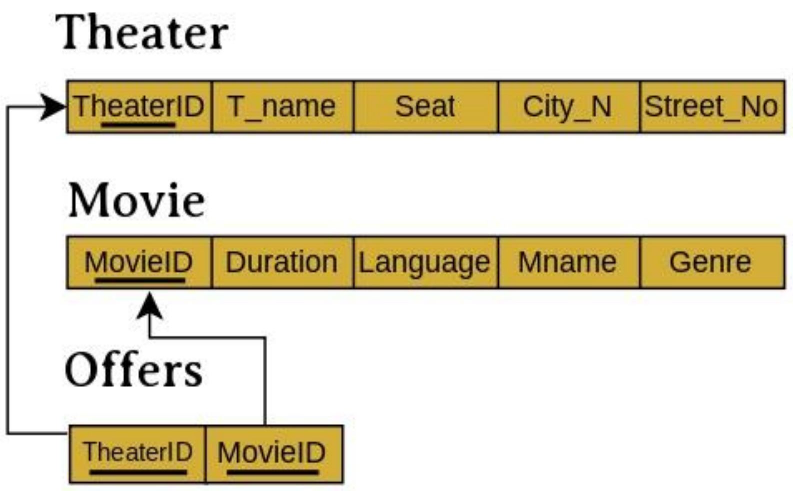
Step4 : Mapping of Binary 1:N Relationship Types



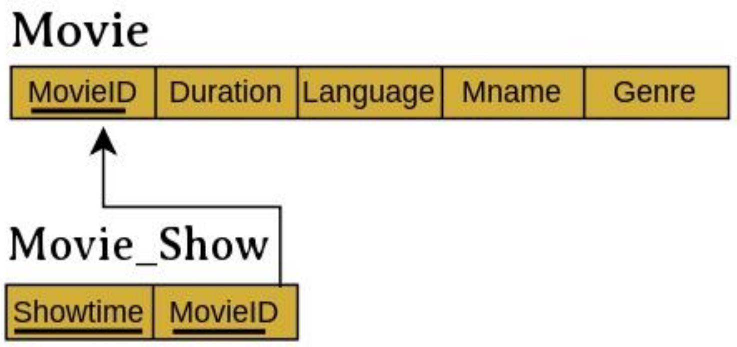
Step5 : Mapping of Binary M:N Relationship Types

(Customer_Snack)

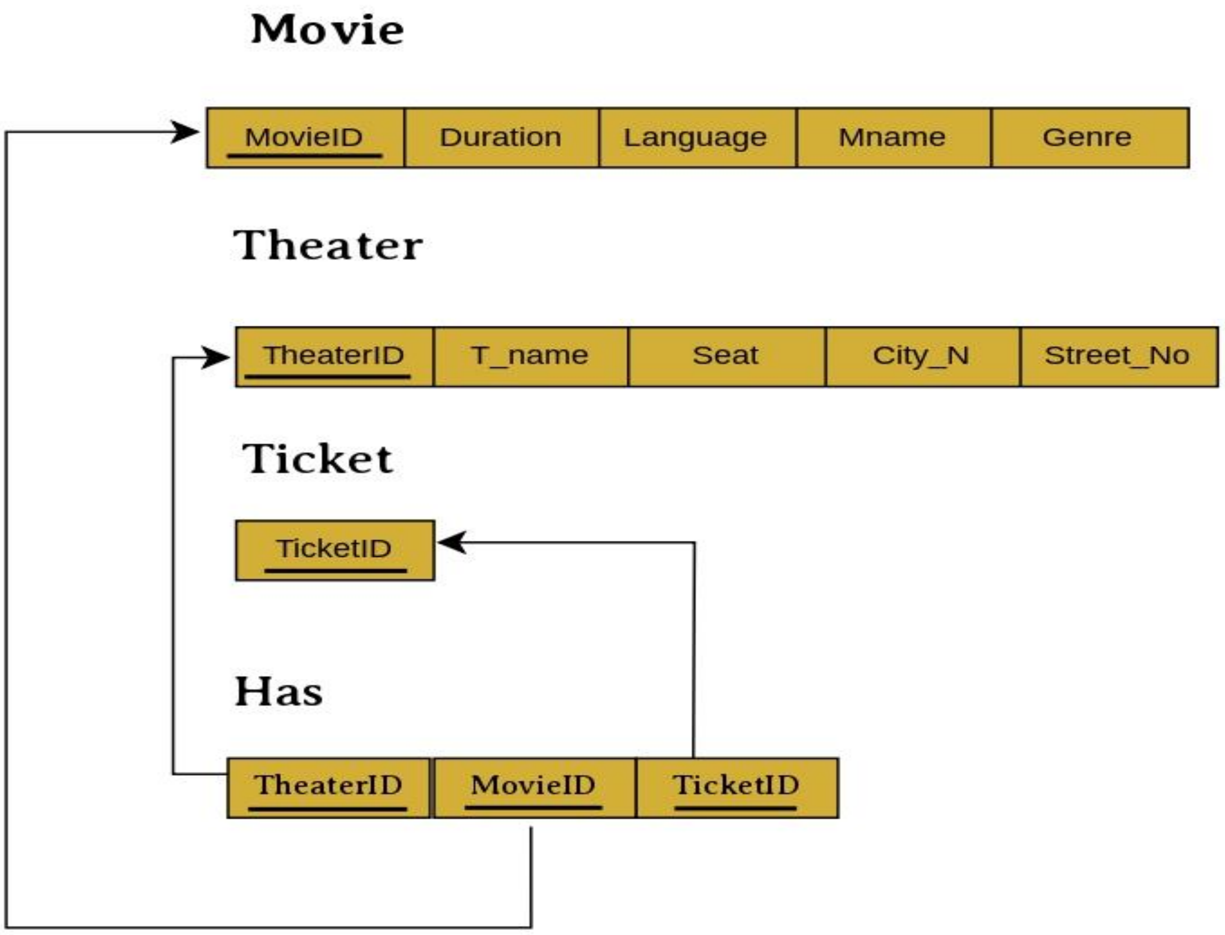
Done at step 2



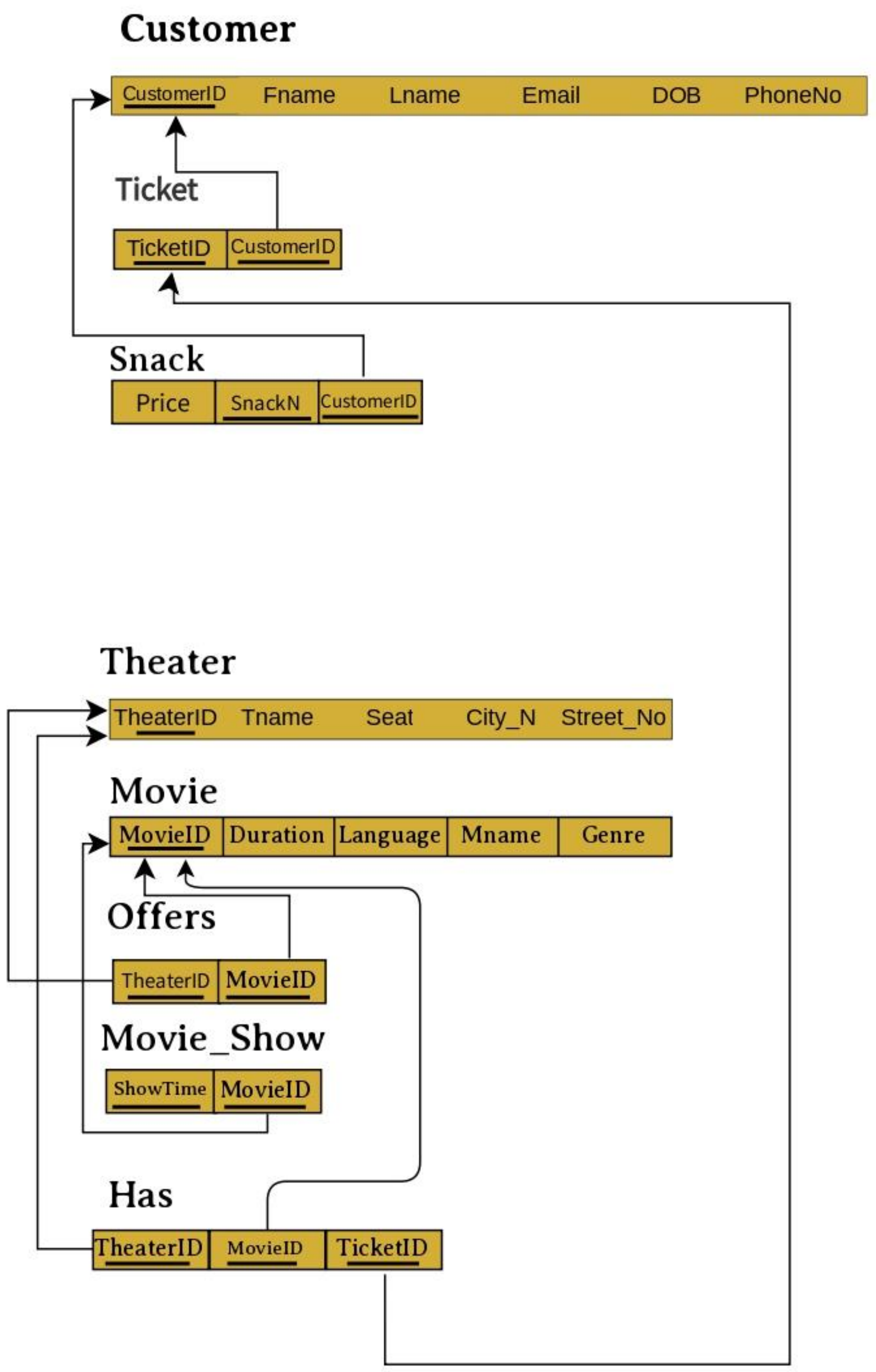
Step6 : Mapping of MultiValues Atribute



Step 7 : Mapping of Ternary Relationship



Final Mapping



Normalization

1NF: Repeated group removal

- No change because there is no multivalued or repeating group

Customer

CustomerID	Fname	Lname	Email	DOB	PhoneNo

Ticket

TicketID	CustomerID

Snack

Price	SnackN	CustomerID

Theater

TheaterID	Tname	Seat	City_N	Street_No

Movie

MovieID	Duration	Language	Mname	Genre

Offers

TheaterID	MovieID

Movie_Show

ShowTime	MovieID

Has

TheaterID	MovieID	TicketID

2NF: No partial dependencies

Customer

CustomerID	Fname	Lname	Email	DOB	PhoneNo

Ticket

TicketID	CustomerID

Snack

Price	SnackN	CustomerID

Theater

TheaterID	Tname	Seat	City_N	Street_No

Movie

MovieID	Duration	Language	Mname	Genre

Offers

TheaterID	MovieID

Movie_Show

ShowTime	MovieID

Has

TheaterID	MovieID	TicketID

3NF: No transitive dependencies

Customer

CustomerID	Fname	Lname	Email	DOB	PhoneNo

Ticket

TicketID	CustomerID

Snack

Price	SnackN	CustomerID

Theater

TheaterID	Tname	Seat	City_N	Street_No

Movie

MovieID	Duration	Language	Mname	Genre

Offers

TheaterID	MovieID

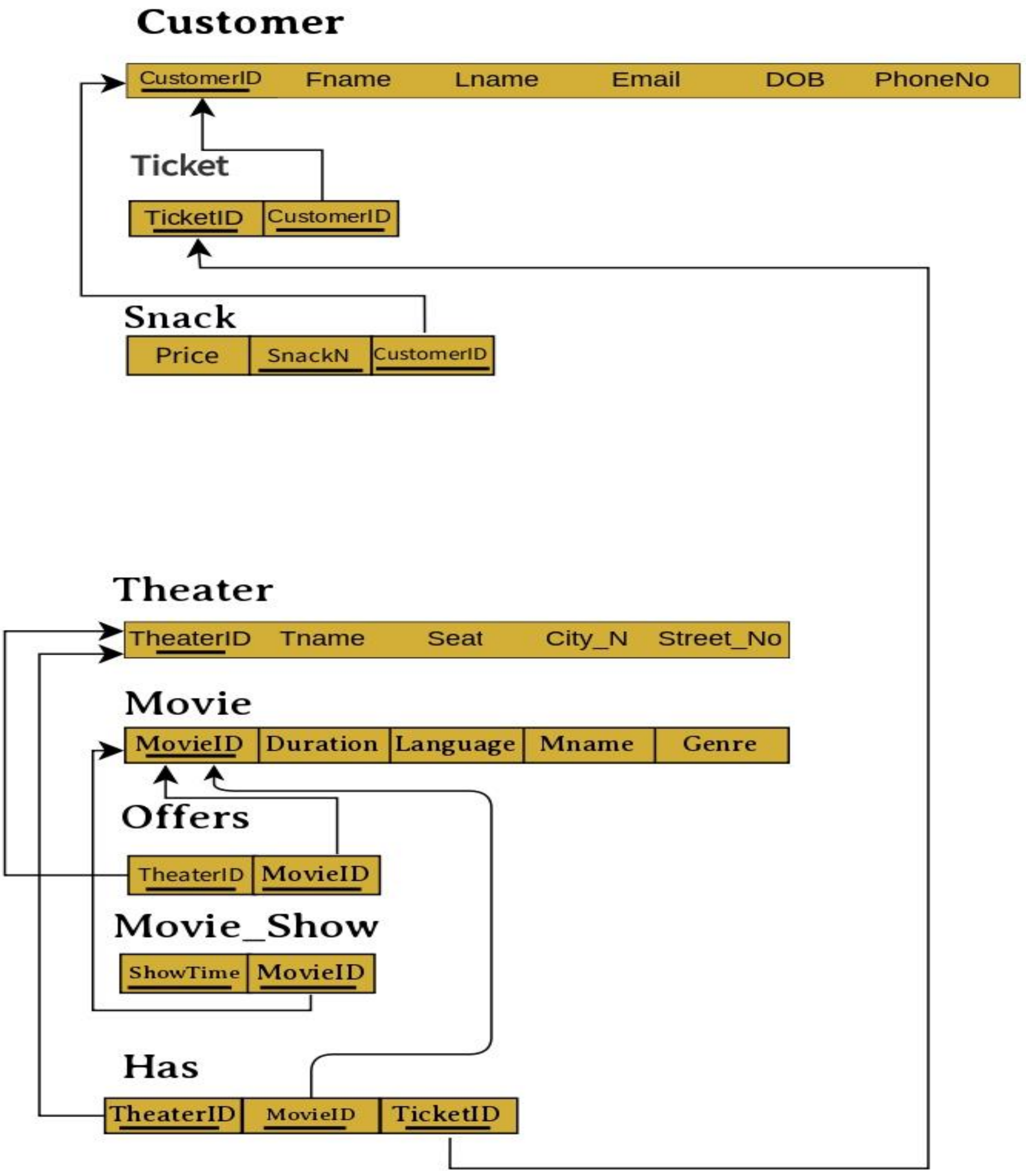
Movie_Show

ShowTime	MovieID

Has

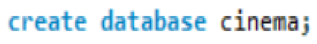
TheaterID	MovieID	TicketID

Mapping after Normalization



- In your MySQL Workbench create the project.
- Database/schema using CREATE SCHEMA.
- Create the schema tables using CREATE TABLE.

1- Create Schema

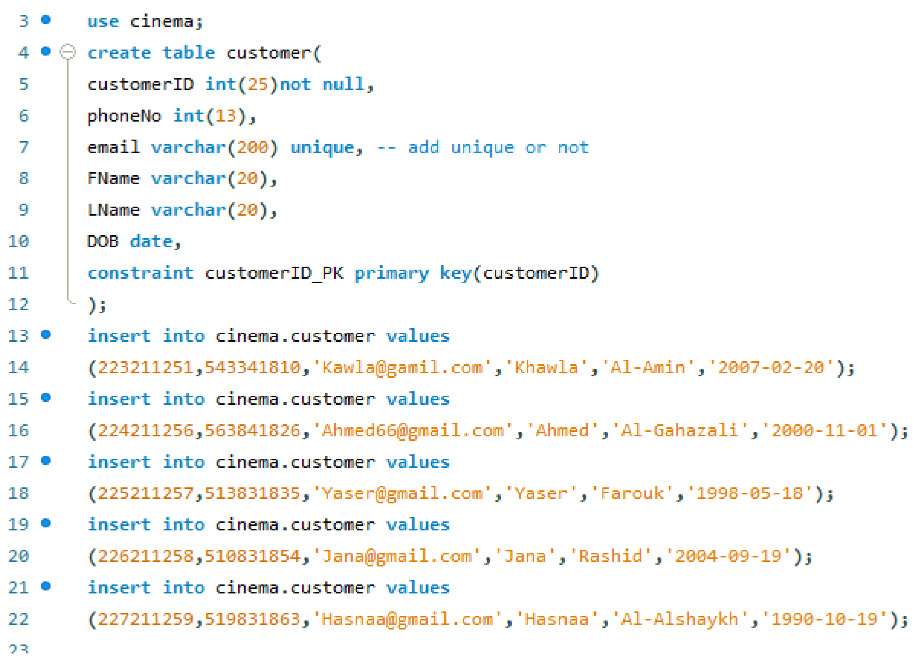


```
create database cinema;
```

2- SQL tables and commands

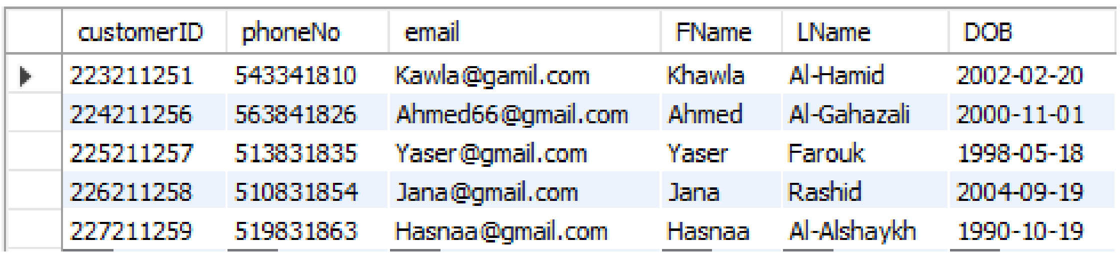
Creating & inserting the table:

2.1-Customer table:



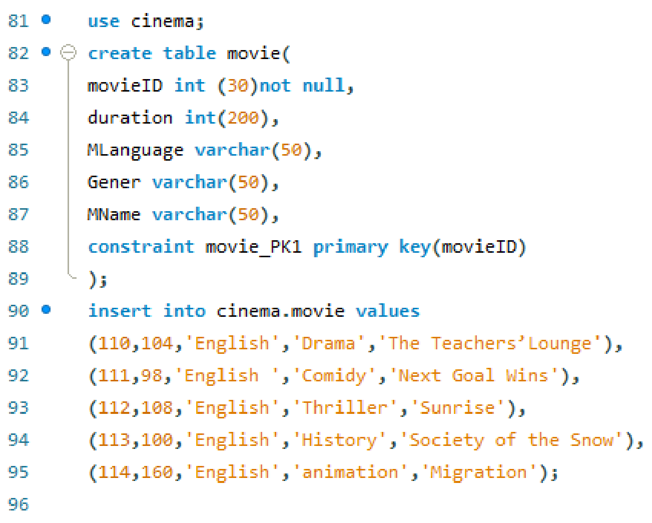
```
use cinema;
create table customer(
customerID int(25)not null,
phoneNo int(13),
email varchar(200) unique, -- add unique or not
FName varchar(20),
LName varchar(20),
DOB date,
constraint customerID_PK primary key(customerID)
);
insert into cinema.customer values
(223211251,543341810,'Kawla@gamil.com','Khawla','Al-Amin','2007-02-20');
insert into cinema.customer values
(224211256,563841826,'Ahmed66@gmail.com','Ahmed','Al-Gahazali','2000-11-01');
insert into cinema.customer values
(225211257,513831835,'Yaser@gmail.com','Yaser','Farouk','1998-05-18');
insert into cinema.customer values
(226211258,510831854,'Jana@gmail.com','Jana','Rashid','2004-09-19');
insert into cinema.customer values
(227211259,519831863,'Hasnaa@gmail.com','Hasnaa','Al-Alshaykh','1990-10-19');
```

Add Customer data:



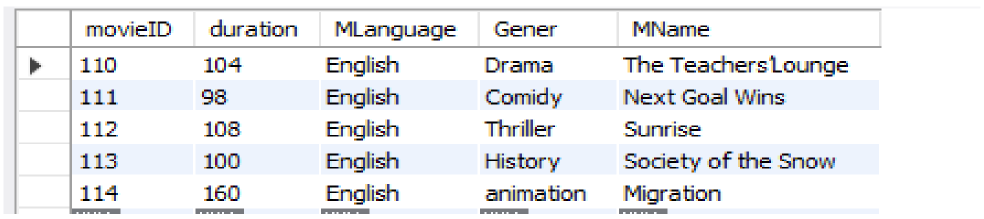
customerID	phoneNo	email	FName	LName	DOB
223211251	543341810	Kawla@gamil.com	Khawla	Al-Hamid	2002-02-20
224211256	563841826	Ahmed66@gmail.com	Ahmed	Al-Gahazali	2000-11-01
225211257	513831835	Yaser@gmail.com	Yaser	Farouk	1998-05-18
226211258	510831854	Jana@gmail.com	Jana	Rashid	2004-09-19
227211259	519831863	Hasnaa@gmail.com	Hasnaa	Al-Alshaykh	1990-10-19

2.2-Movie table:



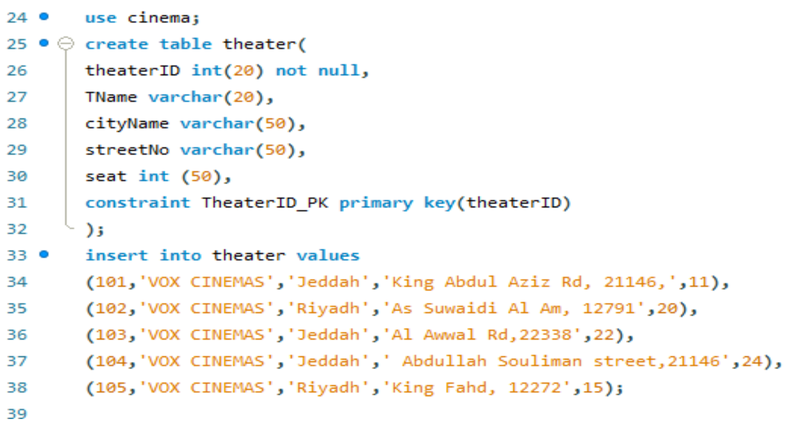
```
use cinema;
create table movie(
movieID int (30)not null,
duration int(200),
MLanguage varchar(50),
Gener varchar(50),
MName varchar(50),
constraint movie_PK1 primary key(movieID)
);
insert into cinema.movie values
(110,104,'English','Drama','The Teachers’Lounge'),
(111,98,'English ','Comidy','Next Goal Wins'),
(112,108,'English','Thriller','Sunrise'),
(113,100,'English','History','Society of the Snow'),
(114,160,'English','animation','Migration');
```

Add Movie data:



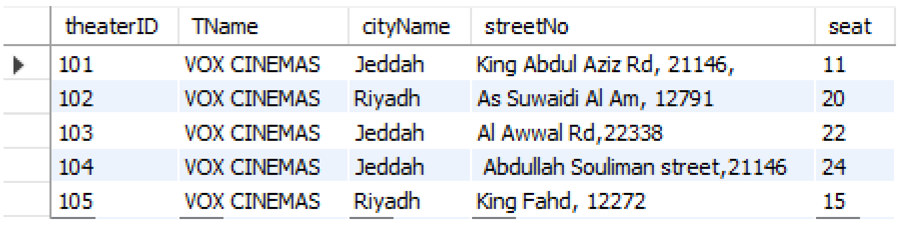
movieID	duration	MLanguage	Gener	MName
110	104	English	Drama	The Teachers’Lounge
111	98	English	Comidy	Next Goal Wins
112	108	English	Thriller	Sunrise
113	100	English	History	Society of the Snow
114	160	English	animation	Migration

2.3-Theater table:



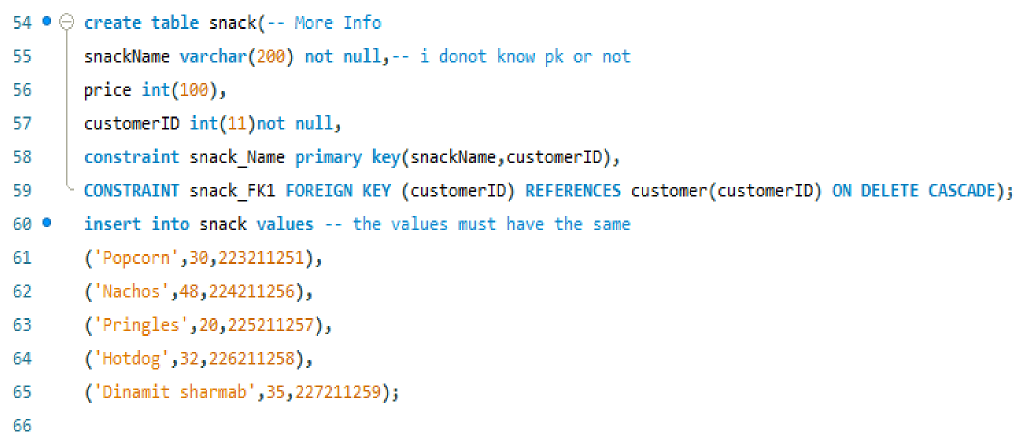
```
use cinema;
create table theater(
theaterID int(20) not null,
TName varchar(20),
cityName varchar(50),
streetNo varchar(50),
seat int (50),
constraint TheaterID_PK primary key(theaterID)
);
insert into theater values
(101,'VOX CINEMAS','Jeddah','King Abdul Aziz Rd, 21146,',11),
(102,'VOX CINEMAS','Riyadh','As Suwaidi Al Am, 12791',20),
(103,'VOX CINEMAS','Jeddah','Al Awwal Rd,22338',22),
(104,'VOX CINEMAS','Jeddah',' Abdullah Souliman street,21146',24),
(105,'VOX CINEMAS','Riyadh','King Fahd, 12272',15);
```

Add Theater data:



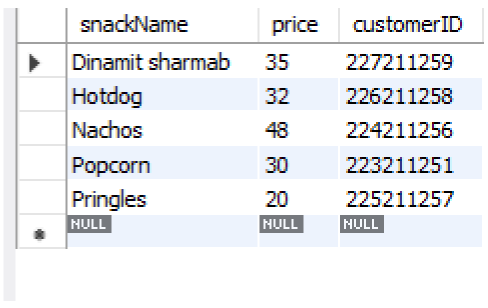
theaterID	TName	cityName	streetNo	seat
101	VOX CINEMAS	Jeddah	King Abdul Aziz Rd, 21146,	11
102	VOX CINEMAS	Riyadh	As Suwaidi Al Am, 12791	20
103	VOX CINEMAS	Jeddah	Al Awwal Rd,22338	22
104	VOX CINEMAS	Jeddah	Abdullah Souliman street,21146	24
105	VOX CINEMAS	Riyadh	King Fahd, 12272	15

2.4-Snack table:



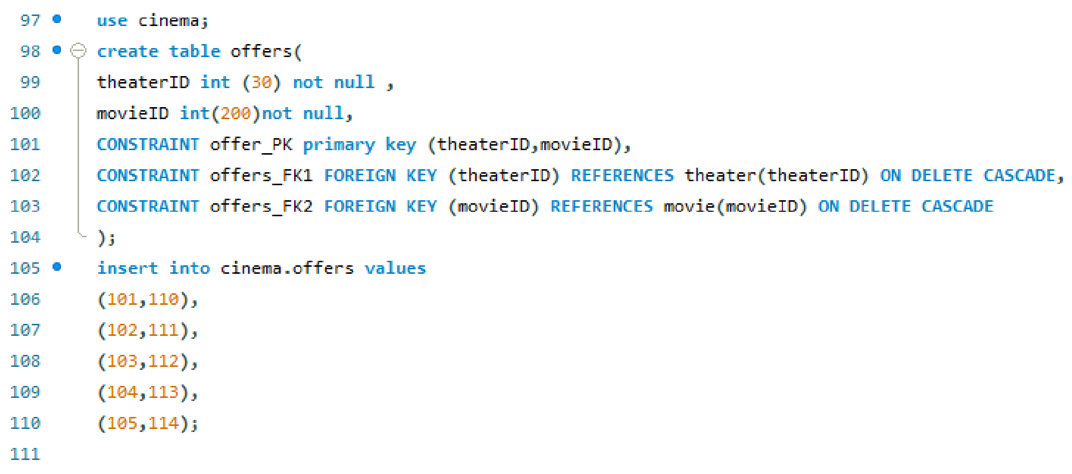
```
create table snack(-- More Info
snackName varchar(200) not null,-- i donot know pk or not
price int(100),
customerID int(11)not null,
constraint snack_Name primary key(snackName,customerID),
CONSTRAINT snack_FK1 FOREIGN KEY (customerID) REFERENCES customer(customerID) ON DELETE CASCADE);
insert into snack values -- the values must have the same
('Popcorn',30,223211251),
('Nachos',48,224211256),
('Pringles',20,225211257),
('Hotdog',32,226211258),
('Dinamit sharmab',35,227211259);
```

Add Snack data:



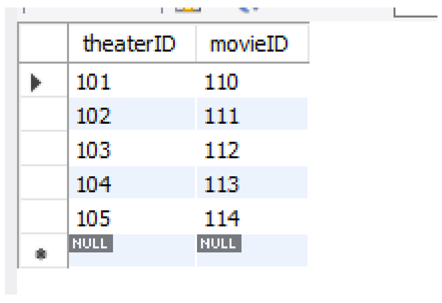
snackName	price	customerID
Dinamit sharmab	35	227211259
Hotdog	32	226211258
Nachos	48	224211256
Popcorn	30	223211251
Pringles	20	225211257
NULL	NULL	NULL

2.5-Offers table:



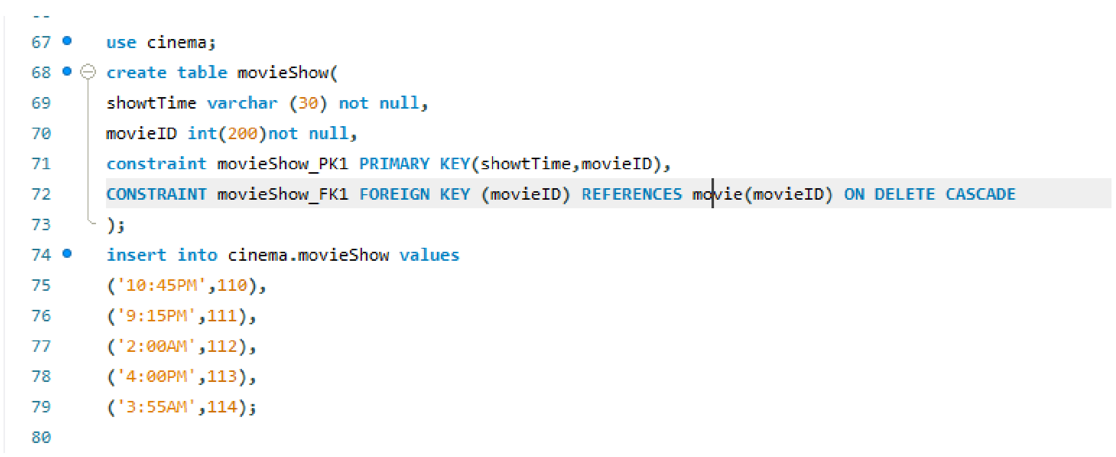
```
use cinema;
create table offers(
theaterID int (30) not null ,
movieID int(200)not null,
CONSTRAINT offer_PK primary key (theaterID,movieID),
CONSTRAINT offers_FK1 FOREIGN KEY (theaterID) REFERENCES theater(theaterID) ON DELETE CASCADE,
CONSTRAINT offers_FK2 FOREIGN KEY (movieID) REFERENCES movie(movieID) ON DELETE CASCADE
);
insert into cinema.offers values
(101,110),
(102,111),
(103,112),
(104,113),
(105,114);
```

Add Offers data:



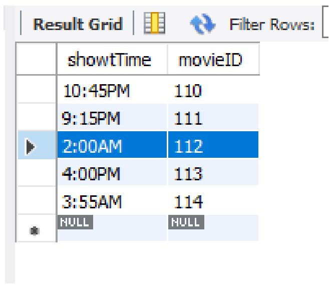
theaterID	movieID
101	110
102	111
103	112
104	113
105	114
NULL	NULL

2.6-movie_Show table:



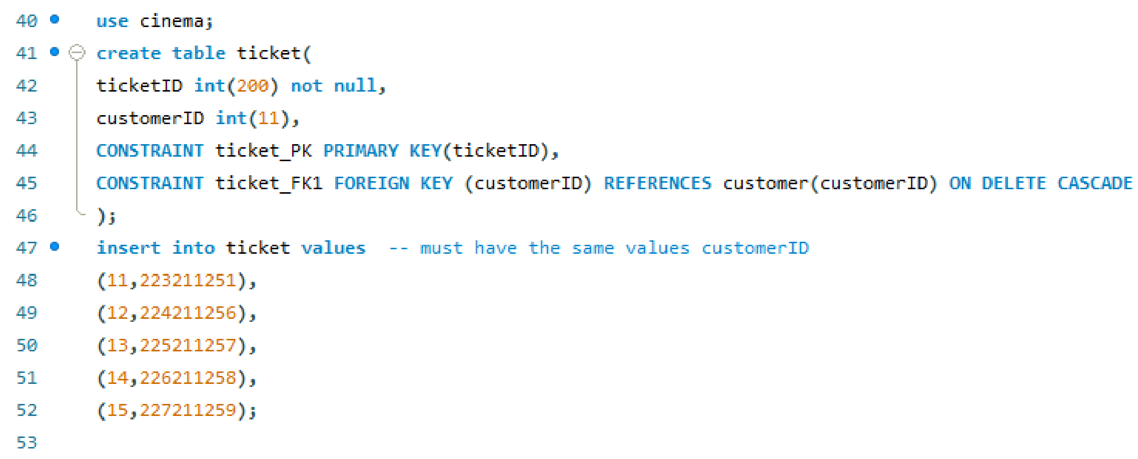
```
--
use cinema;
create table movieShow(
showTime varchar (30) not null,
movieID int(200)not null,
constraint movieShow_PK1 PRIMARY KEY(showTime,movieID),
CONSTRAINT movieShow_FK1 FOREIGN KEY (movieID) REFERENCES movie(movieID) ON DELETE CASCADE
);
insert into cinema.movieShow values
('10:45PM',110),
('9:15PM',111),
('2:00AM',112),
('4:00PM',113),
('3:55AM',114);
```

Add MovieShow data:



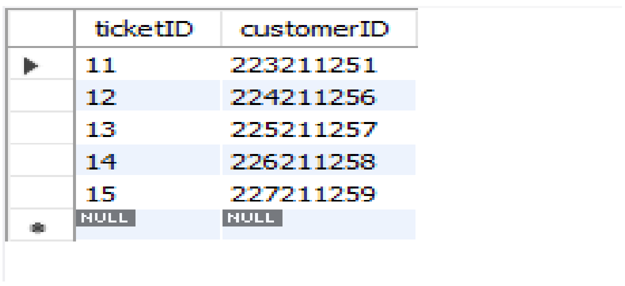
showTime	movieID
10:45PM	110
9:15PM	111
2:00AM	112
4:00PM	113
3:55AM	114
NULL	NULL

2.7- Ticket table:



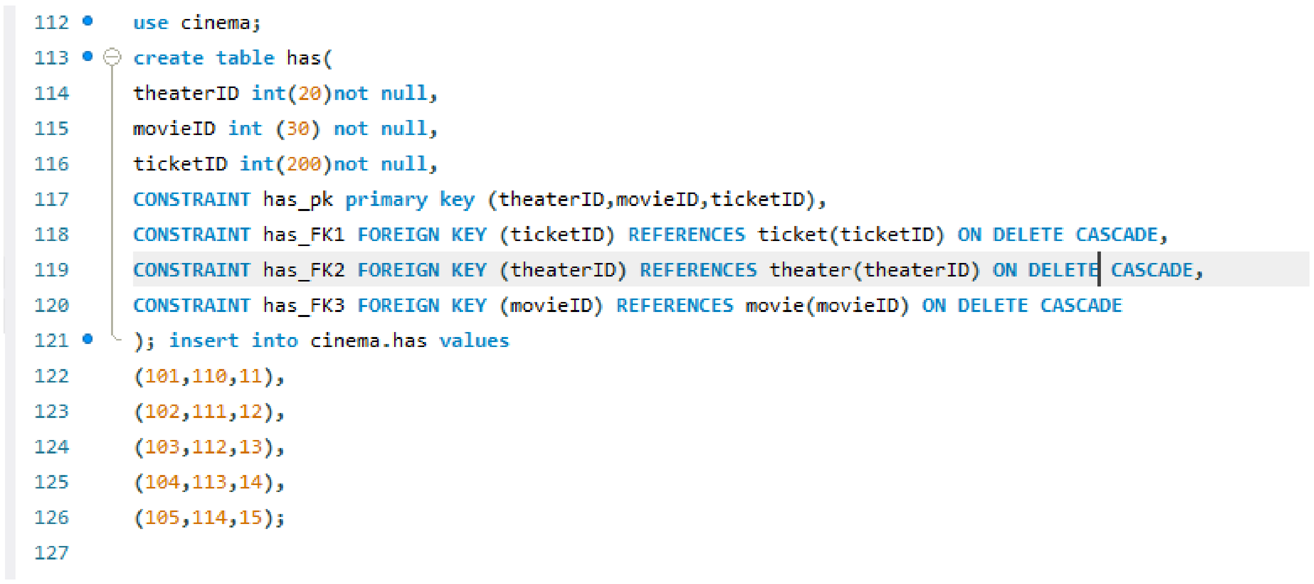
```
use cinema;
create table ticket(
ticketID int(200) not null,
customerID int(11),
CONSTRAINT ticket_PK PRIMARY KEY(ticketID),
CONSTRAINT ticket_FK1 FOREIGN KEY (customerID) REFERENCES customer(customerID) ON DELETE CASCADE
);
insert into ticket values  -- must have the same values customerID
(11,223211251),
(12,224211256),
(13,225211257),
(14,226211258),
(15,227211259);
```

Add Ticket data:



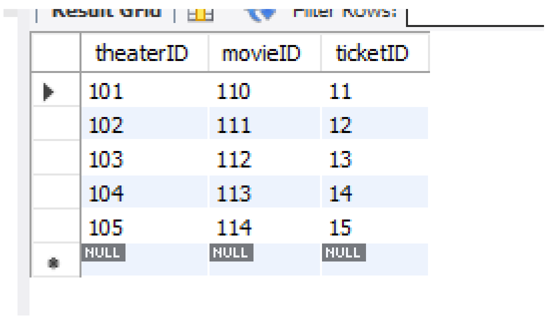
ticketID	customerID
11	223211251
12	224211256
13	225211257
14	226211258
15	227211259
NULL	NULL

2.8-Has table:



```
use cinema;
create table has(
theaterID int(20)not null,
movieID int (30) not null,
ticketID int(200)not null,
CONSTRAINT has_pk primary key (theaterID,movieID,ticketID),
CONSTRAINT has_FK1 FOREIGN KEY (ticketID) REFERENCES ticket(ticketID) ON DELETE CASCADE,
CONSTRAINT has_FK2 FOREIGN KEY (theaterID) REFERENCES theater(theaterID) ON DELETE CASCADE,
CONSTRAINT has_FK3 FOREIGN KEY (movieID) REFERENCES movie(movieID) ON DELETE CASCADE
); insert into cinema.has values
(101,110,11),
(102,111,12),
(103,112,13),
(104,113,14),
(105,114,15);
```

Add Has data:

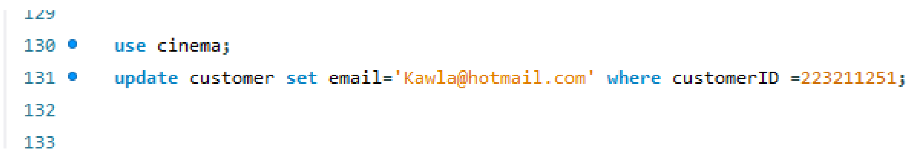


theaterID	movieID	ticketID
101	110	11
102	111	12
103	112	13
104	113	14
105	114	15
NULL	NULL	NULL

3.Updating &Deleting:

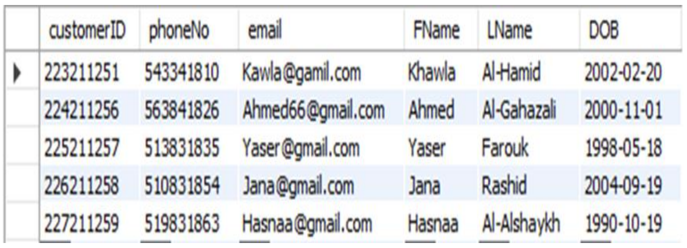
Update customer table

This SQL statement is updating the email address for a customer with the ID 223211251 in the "customer" table. The new email address being set is 'Kawla@hommail.com'.

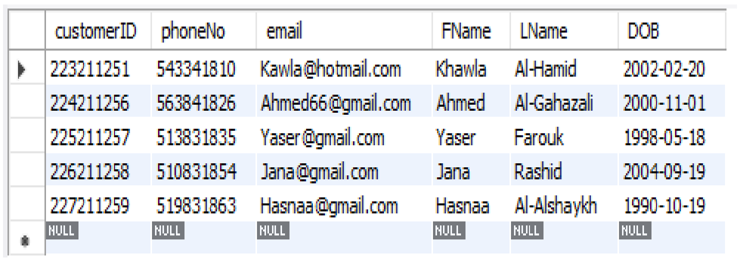


```sql
use cinema;
update customer set email='Kawla@hotmail.com' where customerID =223211251;
```

Before: **After:**



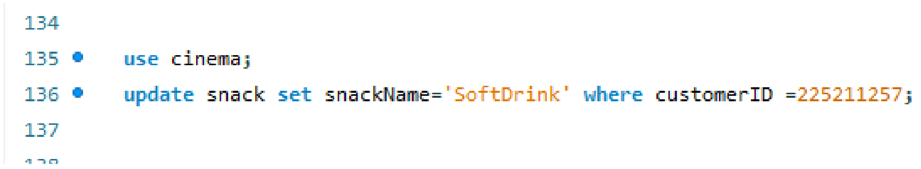
customerID	phoneNo	email	FName	LName	DOB
223211251	543341810	Kawla@gamil.com	Khawla	Al-Hamid	2002-02-20
224211256	563841826	Ahmed66@gmail.com	Ahmed	Al-Gahazali	2000-11-01
225211257	513831835	Yaser@gmail.com	Yaser	Farouk	1998-05-18
226211258	510831854	Jana@gmail.com	Jana	Rashid	2004-09-19
227211259	519831863	Hasnaa@gmail.com	Hasnaa	Al-Alshaykh	1990-10-19



customerID	phoneNo	email	FName	LName	DOB
223211251	543341810	Kawla@hotmail.com	Khawla	Al-Hamid	2002-02-20
224211256	563841826	Ahmed66@gmail.com	Ahmed	Al-Gahazali	2000-11-01
225211257	513831835	Yaser@gmail.com	Yaser	Farouk	1998-05-18
226211258	510831854	Jana@gmail.com	Jana	Rashid	2004-09-19
227211259	519831863	Hasnaa@gmail.com	Hasnaa	Al-Alshaykh	1990-10-19
NULL	NULL	NULL	NULL	NULL	NULL

3.1 Updating &Deleting:

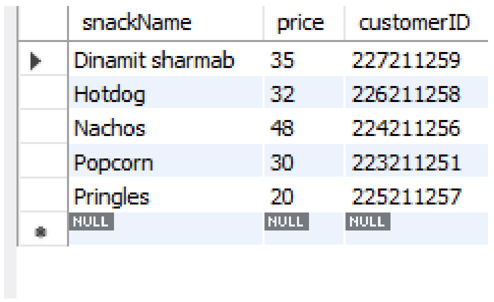
Update snack table



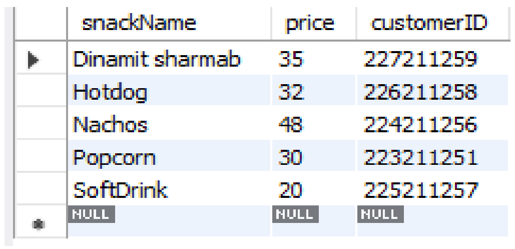
```sql
use cinema;
update snack set snackName='SoftDrink' where customerID =225211257;
```

update the snack Name from Pringels to 'Soft Drink' in the "snack" table for a record where the customer ID is 225211257.

Before: **After:**

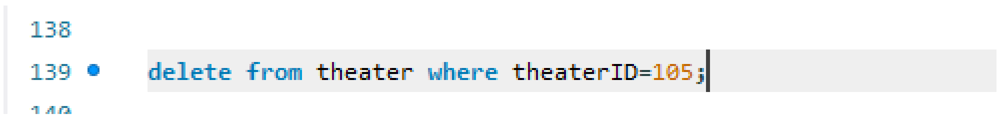


snackName	price	customerID
Dinamit sharmab	35	227211259
Hotdog	32	226211258
Nachos	48	224211256
Popcorn	30	223211251
Pringles	20	225211257
NULL	NULL	NULL



snackName	price	customerID
Dinamit sharmab	35	227211259
Hotdog	32	226211258
Nachos	48	224211256
Popcorn	30	223211251
SoftDrink	20	225211257
NULL	NULL	NULL

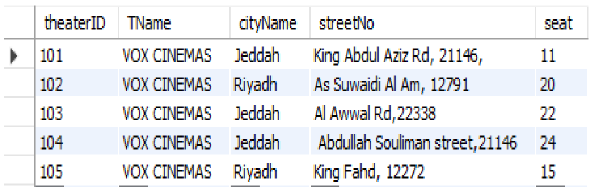
Delete theater row:



```sql
delete from theater where theaterID=105;
```

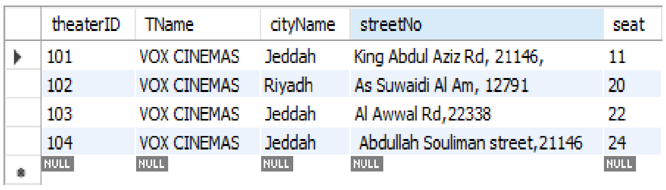
Deletes the record with theater ID 105 from the "theater" table.

Before:



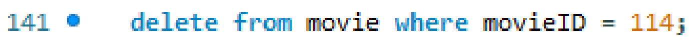
	theaterID	TName	cityName	streetNo	seat
▶	101	VOX CINEMAS	Jeddah	King Abdul Aziz Rd, 21146,	11
	102	VOX CINEMAS	Riyadh	As Suwaidi Al Am, 12791	20
	103	VOX CINEMAS	Jeddah	Al Awwal Rd,22338	22
	104	VOX CINEMAS	Jeddah	Abdullah Souliman street,21146	24
	105	VOX CINEMAS	Riyadh	King Fahd, 12272	15

After:



	theaterID	TName	cityName	streetNo	seat
▶	101	VOX CINEMAS	Jeddah	King Abdul Aziz Rd, 21146,	11
	102	VOX CINEMAS	Riyadh	As Suwaidi Al Am, 12791	20
	103	VOX CINEMAS	Jeddah	Al Awwal Rd,22338	22
	104	VOX CINEMAS	Jeddah	Abdullah Souliman street,21146	24
*	NULL	NULL	NULL	NULL	NULL

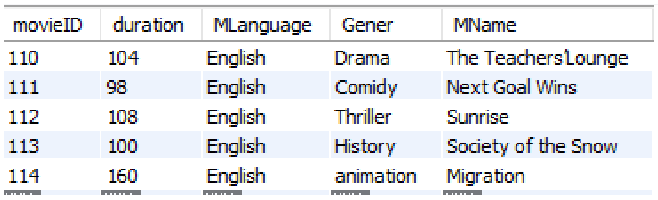
Delete Movie row:



```sql
delete from movie where movieID = 114;
```

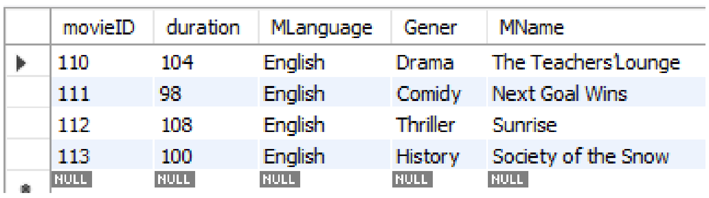
Deletes the record with movie ID 114 from the "movie" table.

Before:



movieID	duration	MLanguage	Gener	MName
110	104	English	Drama	The Teachers Lounge
111	98	English	Comidy	Next Goal Wins
112	108	English	Thriller	Sunrise
113	100	English	History	Society of the Snow
114	160	English	animation	Migration

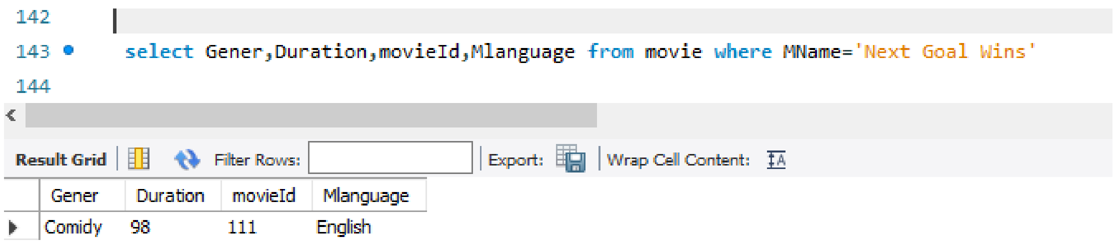
After:



	movieID	duration	MLanguage	Gener	MName
▶	110	104	English	Drama	The Teachers Lounge
	111	98	English	Comidy	Next Goal Wins
	112	108	English	Thriller	Sunrise
	113	100	English	History	Society of the Snow
*	NULL	NULL	NULL	NULL	NULL

Query data using Select;

1- SELECT with WHERE



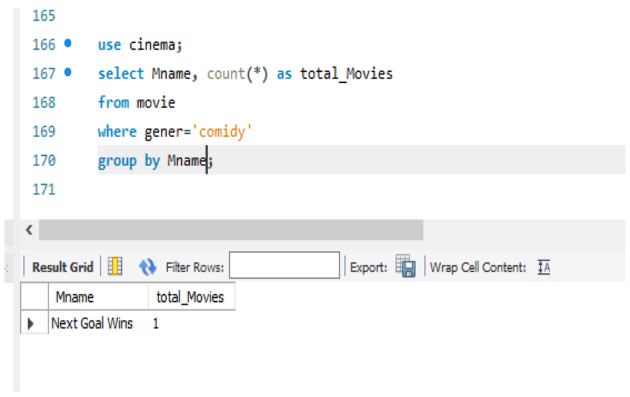
```
select Gener,Duration,movieId,Mlanguage from movie where MName='Next Goal Wins'
```

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

Gener	Duration	movieId	Mlanguage
Comidy	98	111	English

Select the columns 'Gener', 'Duration', 'movieId', and 'Mlanguage' from the 'movie' table where the movie name ('MName') is 'Next Goal Wins.'

2-SELECT with GROUP By, WHERE and COUNT



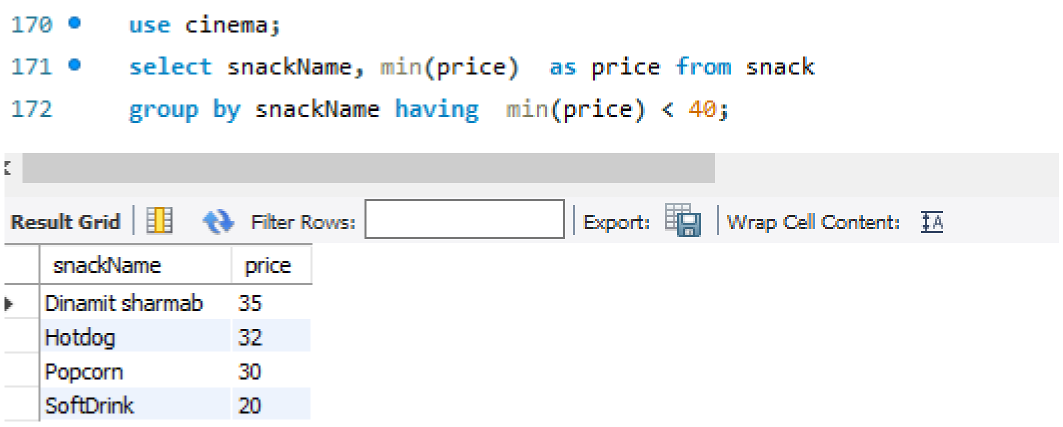
```
use cinema;
select Mname, count(*) as total_Movies
from movie
where gener='comidy'
group by Mname;
```

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

Mname	total_Movies
Next Goal Wins	1

Select the 'name' column and counts the occurrences (as 'total Movies') from the 'movie' table where the genre is comedy. It then groups the results by movie name ('Mname').

3-SELECT with HAVING



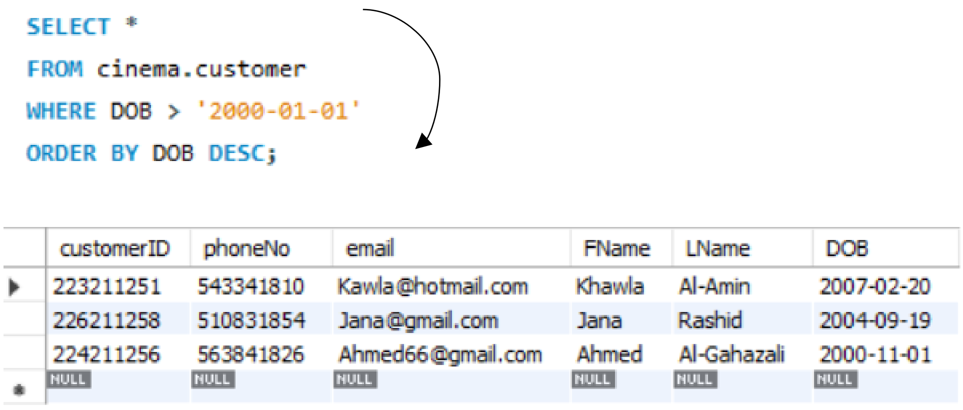
```
use cinema;
select snackName, min(price)  as price from snack
group by snackName having  min(price) < 40;
```

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

snackName	price
Dinamit sharmab	35
Hotdog	32
Popcorn	30
SoftDrink	20

Select the 'snackName' and the 'minimum price ('price')from the 'snack' table. It groups the resutls by 'snackName' and filters the grouped data, only including rows where the minimum price is less than 40.

4-SELECT with ORDER BY

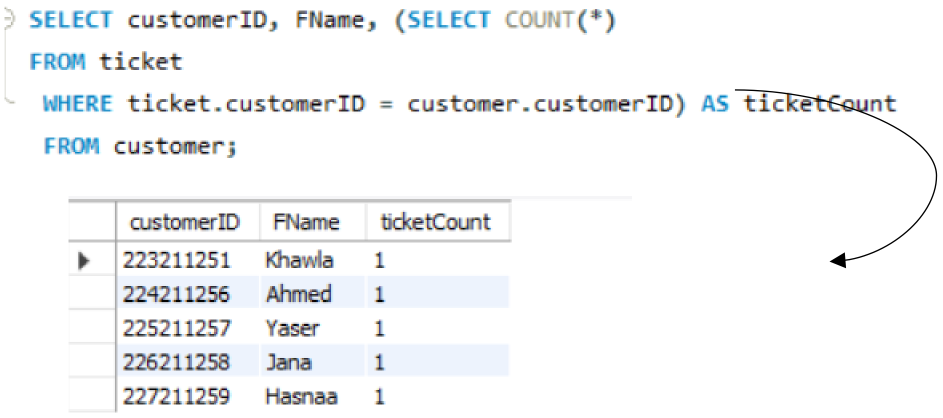


```
SELECT *
FROM cinema.customer
WHERE DOB > '2000-01-01'
ORDER BY DOB DESC;
```

customerID	phoneNo	email	FName	LName	DOB
223211251	543341810	Kawla@hotmail.com	Khawla	Al-Amin	2007-02-20
226211258	510831854	Jana@gmail.com	Jana	Rashid	2004-09-19
224211256	563841826	Ahmed66@gmail.com	Ahmed	Al-Gahazali	2000-11-01
NULL	NULL	NULL	NULL	NULL	NULL

select all columns from the 'customer' table in the 'cinema' database where the date of birth ('DOB') is after'2000-01-01'. The results are then ordered in descending order based on the date of birth.

5-SELECT with SUBQURY

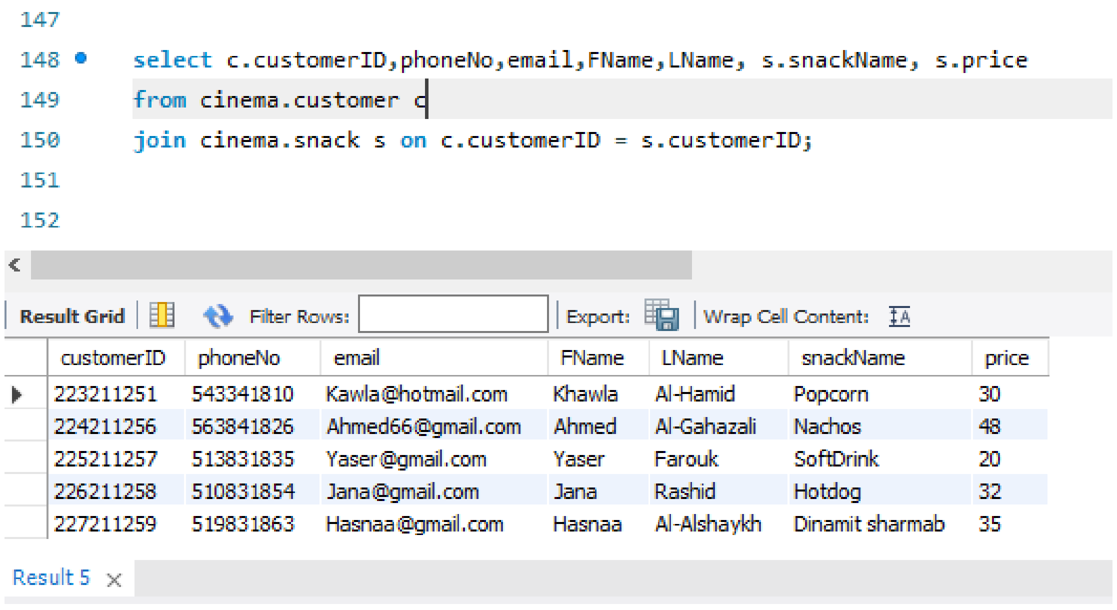


```
SELECT customerID, FName, (SELECT COUNT(*)
FROM ticket
 WHERE ticket.customerID = customer.customerID) AS ticketCount
 FROM customer;
```

	customerID	FName	ticketCount
▶	223211251	Khawla	1
	224211256	Ahmed	1
	225211257	Yaser	1
	226211258	Jana	1
	227211259	Hasnaa	1

select 'customerID' and 'FName' columns from the 'customer' table. It also includes a calculated column 'ticketCount,' which represents the count of tickets associated with each customer using a subquery.

6-SElECT WITH Inner-join operation



```
select c.customerID,phoneNo,email,FName,LName, s.snackName, s.price
from cinema.customer c
join cinema.snack s on c.customerID = s.customerID;
```

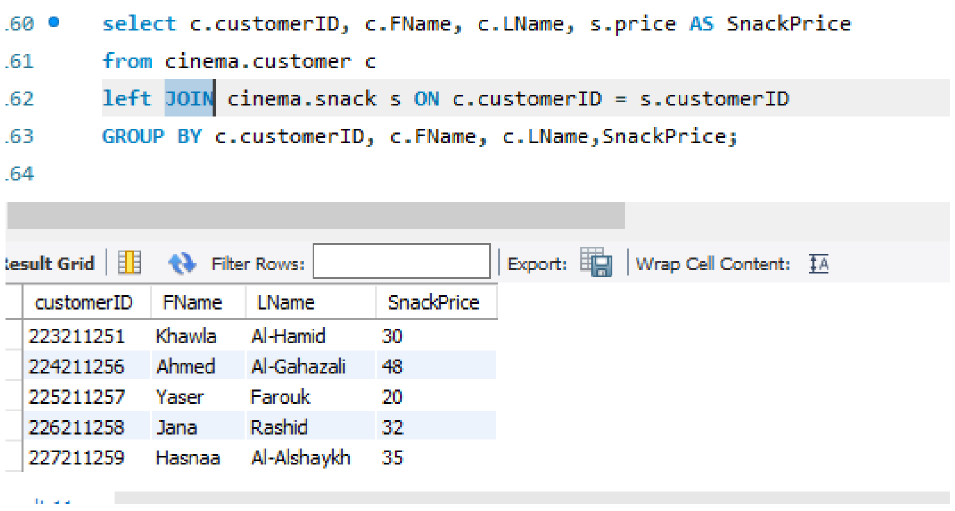
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	customerID	phoneNo	email	FName	LName	snackName	price
▶	223211251	543341810	Kawla@hotmail.com	Khawla	Al-Hamid	Popcorn	30
	224211256	563841826	Ahmed66@gmail.com	Ahmed	Al-Gahazali	Nachos	48
	225211257	513831835	Yaser@gmail.com	Yaser	Farouk	SoftDrink	20
	226211258	510831854	Jana@gmail.com	Jana	Rashid	Hotdog	32
	227211259	519831863	Hasnaa@gmail.com	Hasnaa	Al-Alshaykh	Dinamit sharmab	35

Result 5 ×

Select customer information (ID, email, first name, last name) along with snack details (snack name, price) from the 'customer' and 'snack' tables in the 'cinema' database. It uses a join condition linking customer IDs between the two tables.

7-SElECT WITH Left- join operation

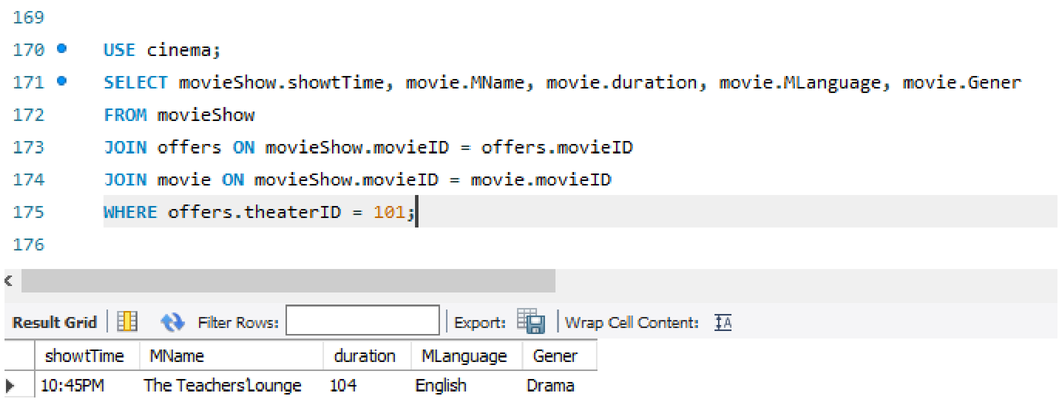


```
select c.customerID, c.FName, c.LName, s.price AS SnackPrice
from cinema.customer c
left JOIN cinema.snack s ON c.customerID = s.customerID
GROUP BY c.customerID, c.FName, c.LName,SnackPrice;
```

customerID	FName	LName	SnackPrice
223211251	Khawla	Al-Hamid	30
224211256	Ahmed	Al-Gahazali	48
225211257	Yaser	Farouk	20
226211258	Jana	Rashid	32
227211259	Hasnaa	Al-Alshaykh	35

Select customer ID, first name, last name, and the price of snacks (renamed as 'SnackPrice') for each customer from the 'customer' table. It uses a left join with the 'snack' table based on customer IDs. The results are then grouped by customer ID, first name, last name, and Snack Price.

8-USE inner join operation &WHERE

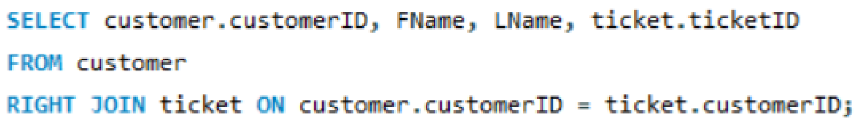


```
USE cinema;
SELECT movieShow.showtTime, movie.MName, movie.duration, movie.MLanguage, movie.Gener
FROM movieShow
JOIN offers ON movieShow.movieID = offers.movieID
JOIN movie ON movieShow.movieID = movie.movieID
WHERE offers.theaterID = 101;
```

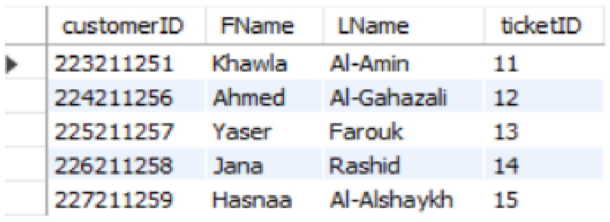
showtTime	MName	duration	MLanguage	Gener
10:45PM	The Teachers'Lounge	104	English	Drama

selects the show time, movie name, duration, language, and genre from the 'movie show' and 'movie' tables in the 'cinema' database. It joins these tables using the movie ID and filters the results based on a specific theater ID (101) from the 'offers' table.

9- SElECT WITH Right -join operation



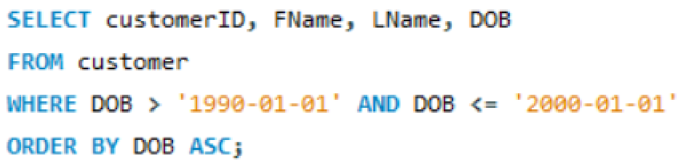
```
SELECT customer.customerID, FName, LName, ticket.ticketID
FROM customer
RIGHT JOIN ticket ON customer.customerID = ticket.customerID;
```



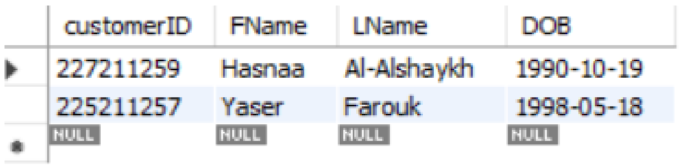
customerID	FName	LName	ticketID
223211251	Khawla	Al-Amin	11
224211256	Ahmed	Al-Gahazali	12
225211257	Yaser	Farouk	13
226211258	Jana	Rashid	14
227211259	Hasnaa	Al-Alshaykh	15

selects customer ID, first name, last name, and ticket ID from the 'customer' and 'ticket' tables. It performs a right join, including all records from the 'ticket' table and matching records from the 'customer' table based on the customer ID.

10- Using WHERE & ORDER BY



```
SELECT customerID, FName, LName, DOB
FROM customer
WHERE DOB > '1990-01-01' AND DOB <= '2000-01-01'
ORDER BY DOB ASC;
```



customerID	FName	LName	DOB
227211259	Hasnaa	Al-Alshaykh	1990-10-19
225211257	Yaser	Farouk	1998-05-18
NULL	NULL	NULL	NULL

selects customer ID, first name, last name, and date of birth (DOB) from the 'customer' table. It filters the results to include only records where the DOB is greater than '1998-01-01' and less than or equal to '2000-01-01'. The final results are then sorted in ascending order based on the date of birth.