विश्वजीवनामृतं ज्ञानम्

ABV-INDIAN INSTITUTE OF INFORMATION
TECHNOLOGY AND MANAGEMENT
GWALIOR-474 015

DATABASE MANAGEMENT SYSTEM

MINI PROJECT

Movie-Ticket-Booking-System *by*

AMAN KUMAR

(2020IMT-007)

ANSH RUSIA

(2020IMT-012)

SHUBHAJEET PRADHAN

(2020IMT-097)

VARUN KUMAR TIWARI

(2020IMT-112)

under the guidance and supervision of

Dr. DEBANJAN SADHYA (Asstt. Professor)

Contents

1 Introduction

1.1 Description of the movie ticket booking system

- As the name suggests the movie ticket management system is a database management system for a multiplex. The database is designed to accommodate multiple theater rooms at same time to have a hassle free experience for the customer and the staff.

- The project is highly flexible and is well efficient for managing all information about the customer. The key focus is: well management of data and easy retrieval of information. The speed and accuracy should be maintained in a proper way.

- Due to faster output of data the system becomes efficient. Their is no manual searching of files and hence loss of data due to human error is less.

1.2 Benefits of our Database design

- Less human error

- Strength and strain of manual labour can be reduced

- Data redundancy can be avoided to some extent

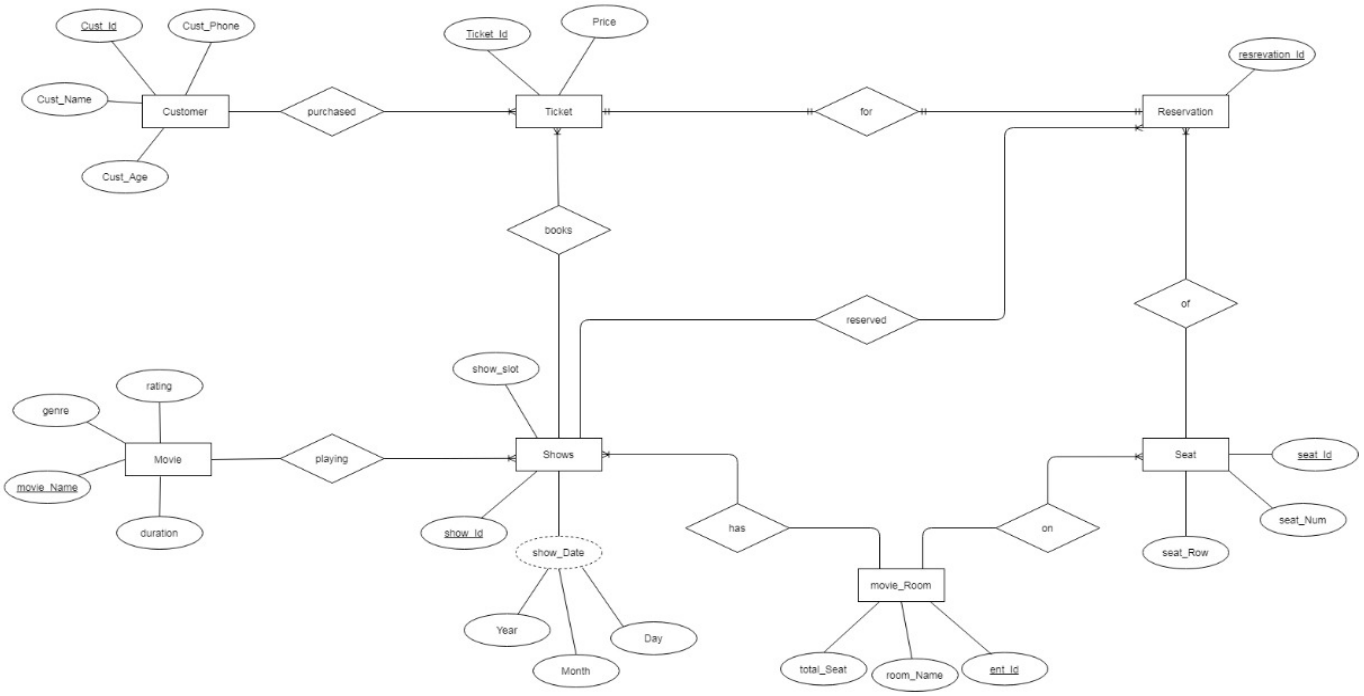
- Data consistency

- Easy to handle

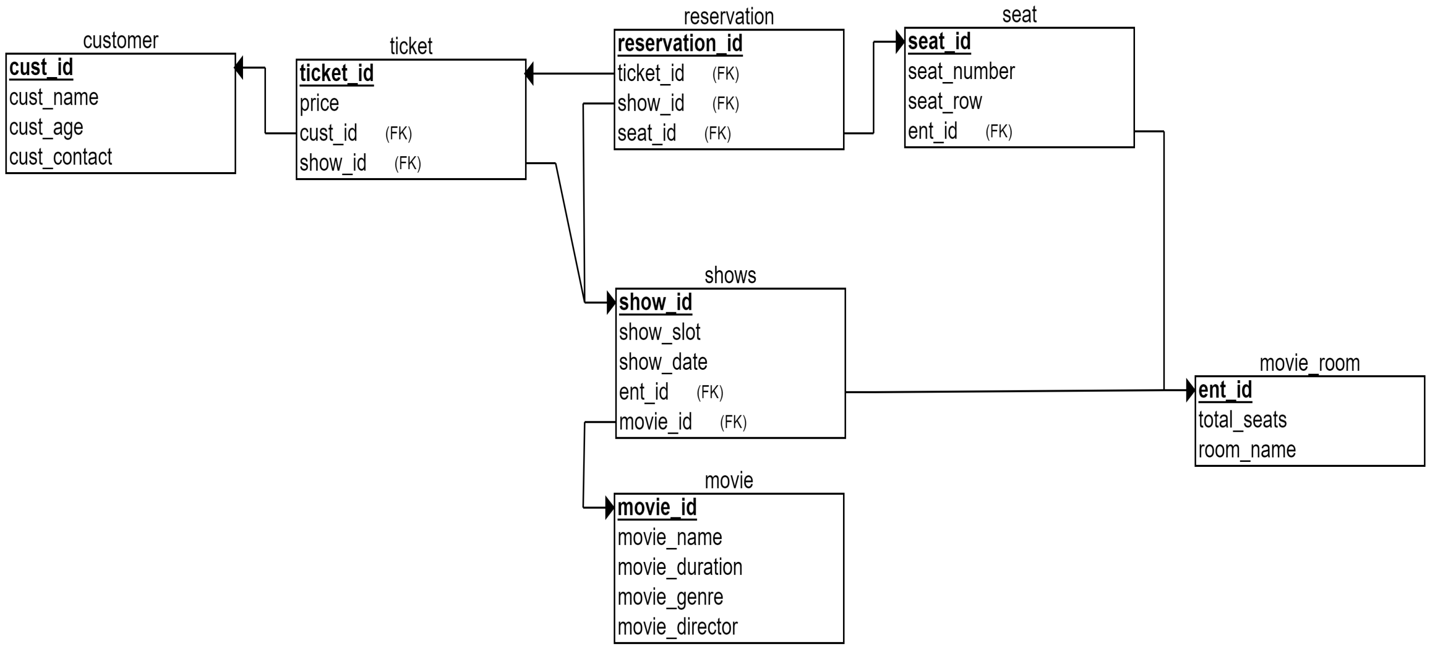
- Easy data updating

- Easy record keeping

2 ER diagram



3 Schema of the Databases



4 Entity sets and Relationship sets

4.1 Entity sets

There are 7 entity sets in our database:- the Customer entity, the Ticket entity, the Reservation entity, the Seat entity, the Shows entity, the Movie entity and the Movie Room entity. We have a separate primary key for every entity so as to prevent any redundancy in the data. The primary keys for respective entities are:-

- Customer entity - cust_Id
- Movie entity - movie_Name
- Movie Room entity - ent_Id
- Reservation entity - reservation_Id
- Seat entity - seat_Id
- Shows entity - show_Id
- Ticket entity - ticket_Id

4.2 Relationship sets

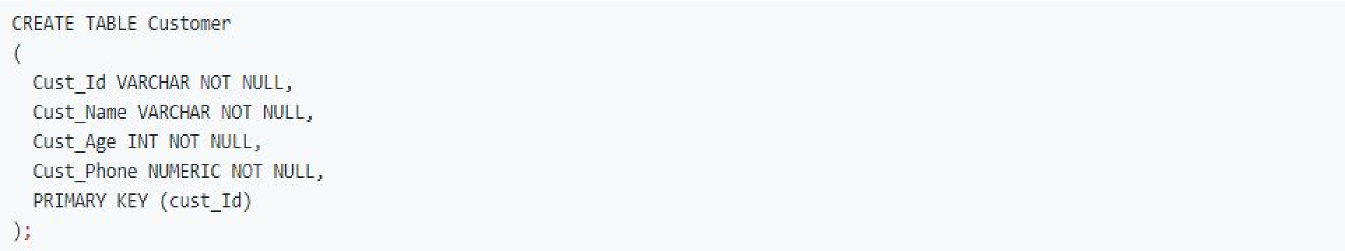
We have 8 relationship sets in our database which are described below :-

- Customer **purchased** Ticket (One to Many relationship)
- Ticket **for** Reservation (One to One relationship)
- Reservation **of** Seat (One to Many relationship)
- Reservation **reserved** Shows (One to Many relationship)
- Ticket **books** Shows (One to Many relationship)
- Seat **on** movie_Room (One to Many relationship)
- Shows **has** movie_Room (One to Many relationship)
- Shows **playing** Movie (One to Many relationship)

5 Tables

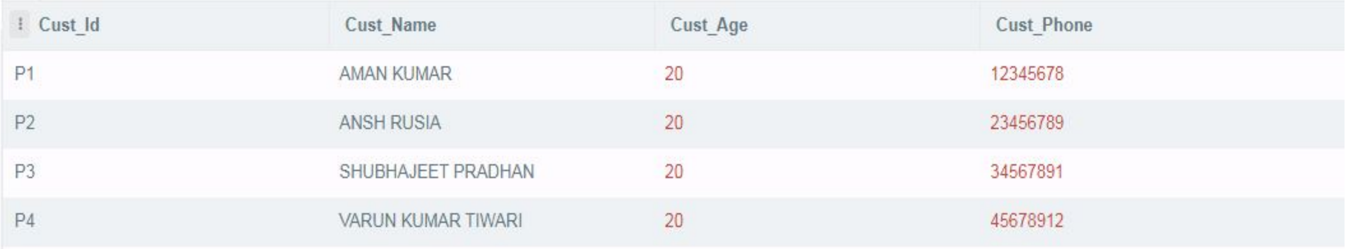
5.1 Customer Table

5.1.1 Query



```
CREATE TABLE Customer
(
  Cust_Id VARCHAR NOT NULL,
  Cust_Name VARCHAR NOT NULL,
  Cust_Age INT NOT NULL,
  Cust_Phone NUMERIC NOT NULL,
  PRIMARY KEY (cust_Id)
);
```

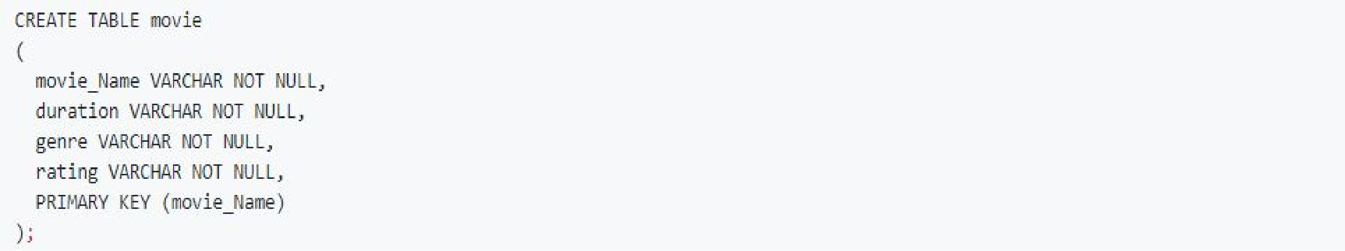
5.1.2 Output Table



Cust_Id	Cust_Name	Cust_Age	Cust_Phone
P1	AMAN KUMAR	20	12345678
P2	ANSH RUSIA	20	23456789
P3	SHUBHAJEET PRADHAN	20	34567891
P4	VARUN KUMAR TIWARI	20	45678912

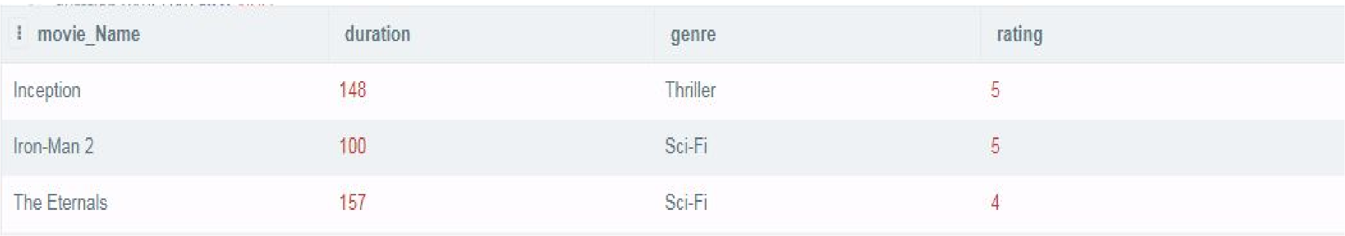
5.2 Movie Table

5.2.1 Query



```
CREATE TABLE movie
(
  movie_Name VARCHAR NOT NULL,
  duration VARCHAR NOT NULL,
  genre VARCHAR NOT NULL,
  rating VARCHAR NOT NULL,
  PRIMARY KEY (movie_Name)
);
```

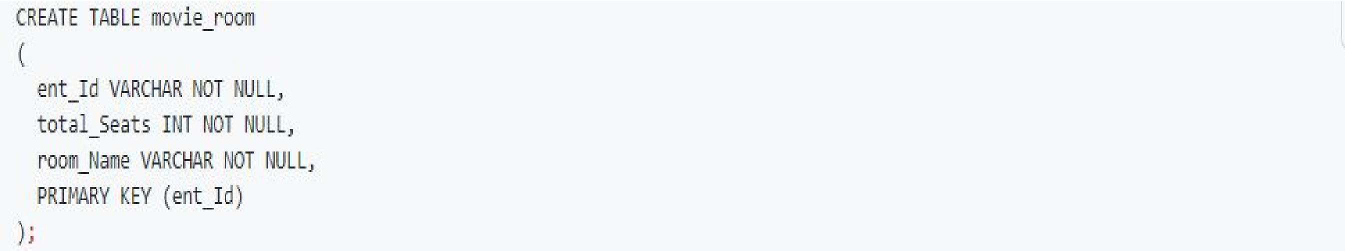
5.2.2 Output Table



movie_Name	duration	genre	rating
Inception	148	Thriller	5
Iron-Man 2	100	Sci-Fi	5
The Eternals	157	Sci-Fi	4

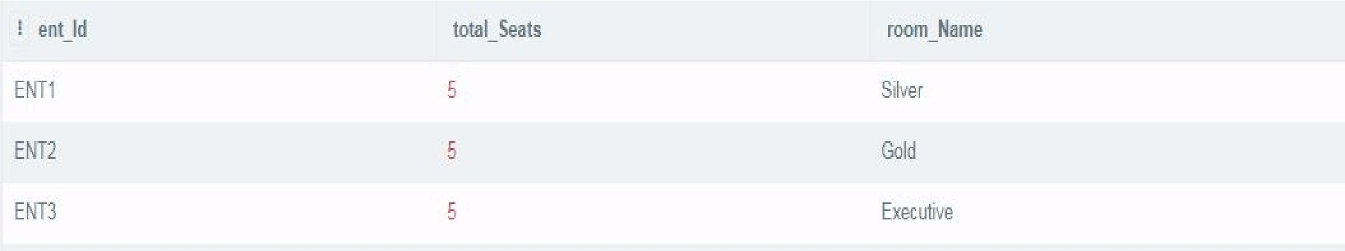
5.3 Movie Room Table

5.3.1 Query



```
CREATE TABLE movie_room
(
  ent_Id VARCHAR NOT NULL,
  total_Seats INT NOT NULL,
  room_Name VARCHAR NOT NULL,
  PRIMARY KEY (ent_Id)
);
```

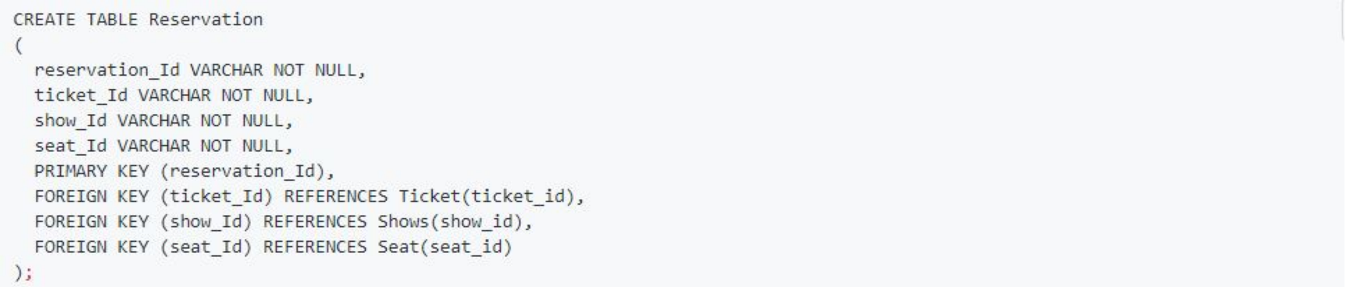
5.3.2 Output Table



ent_Id	total_Seats	room_Name
ENT1	5	Silver
ENT2	5	Gold
ENT3	5	Executive

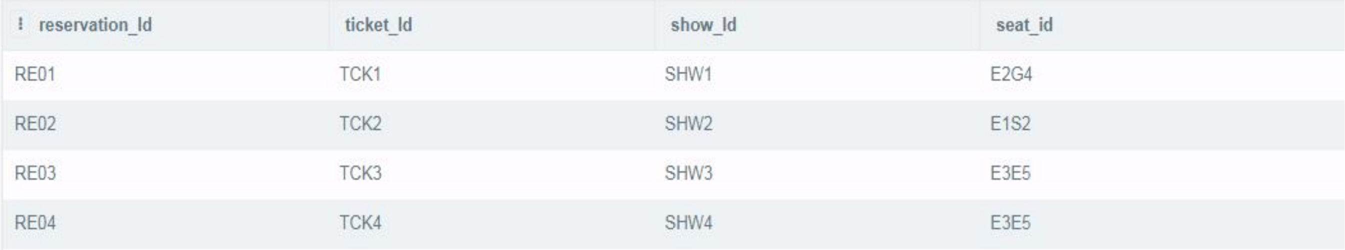
5.4 Reservation Table

5.4.1 Query



```
CREATE TABLE Reservation
(
  reservation_Id VARCHAR NOT NULL,
  ticket_Id VARCHAR NOT NULL,
  show_Id VARCHAR NOT NULL,
  seat_Id VARCHAR NOT NULL,
  PRIMARY KEY (reservation_Id),
  FOREIGN KEY (ticket_Id) REFERENCES Ticket(ticket_id),
  FOREIGN KEY (show_Id) REFERENCES Shows(show_id),
  FOREIGN KEY (seat_Id) REFERENCES Seat(seat_id)
);
```

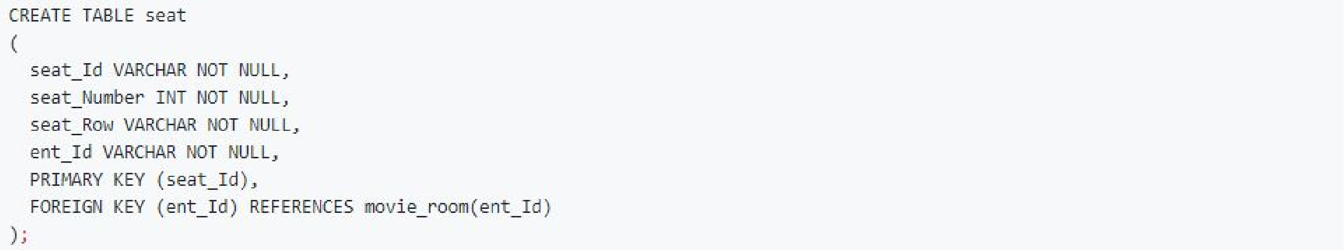
5.4.2 Output Table



reservation_Id	ticket_Id	show_Id	seat_id
RE01	TCK1	SHW1	E2G4
RE02	TCK2	SHW2	E1S2
RE03	TCK3	SHW3	E3E5
RE04	TCK4	SHW4	E3E5

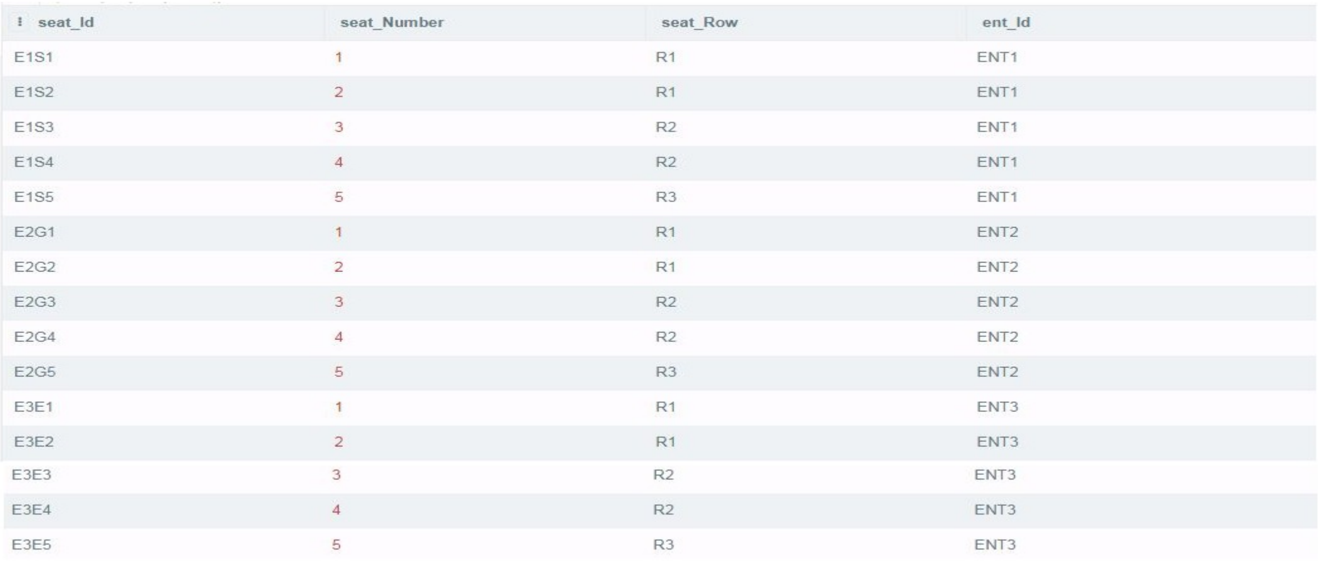
5.5 Seat Table

5.5.1 Query



```
CREATE TABLE seat
(
  seat_Id VARCHAR NOT NULL,
  seat_Number INT NOT NULL,
  seat_Row VARCHAR NOT NULL,
  ent_Id VARCHAR NOT NULL,
  PRIMARY KEY (seat_Id),
  FOREIGN KEY (ent_Id) REFERENCES movie_room(ent_Id)
);
```

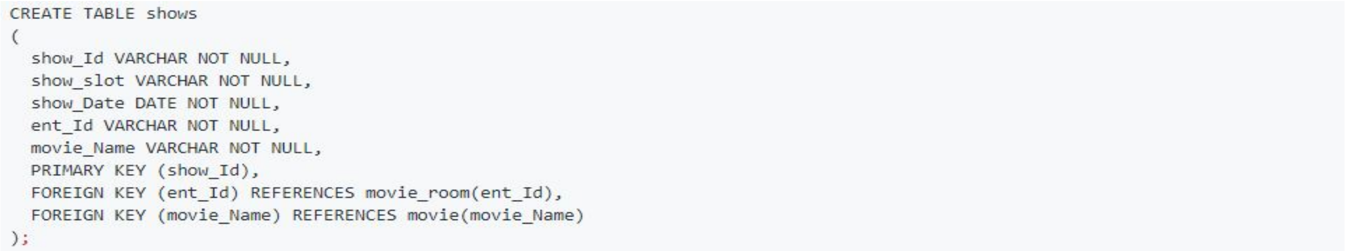
5.5.2 Output Table



seat_Id	seat_Number	seat_Row	ent_Id
E1S1	1	R1	ENT1
E1S2	2	R1	ENT1
E1S3	3	R2	ENT1
E1S4	4	R2	ENT1
E1S5	5	R3	ENT1
E2G1	1	R1	ENT2
E2G2	2	R1	ENT2
E2G3	3	R2	ENT2
E2G4	4	R2	ENT2
E2G5	5	R3	ENT2
E3E1	1	R1	ENT3
E3E2	2	R1	ENT3
E3E3	3	R2	ENT3
E3E4	4	R2	ENT3
E3E5	5	R3	ENT3

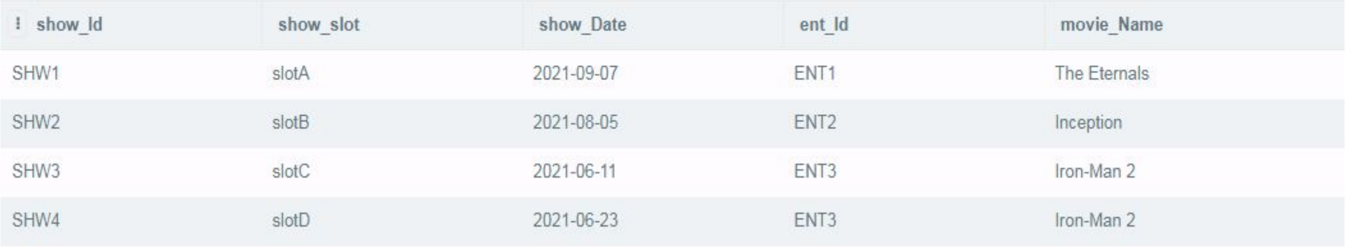
5.6 Shows Table

5.6.1 Query



```
CREATE TABLE shows
(
  show_Id VARCHAR NOT NULL,
  show_slot VARCHAR NOT NULL,
  show_Date DATE NOT NULL,
  ent_Id VARCHAR NOT NULL,
  movie_Name VARCHAR NOT NULL,
  PRIMARY KEY (show_Id),
  FOREIGN KEY (ent_Id) REFERENCES movie_room(ent_Id),
  FOREIGN KEY (movie_Name) REFERENCES movie(movie_Name)
);
```

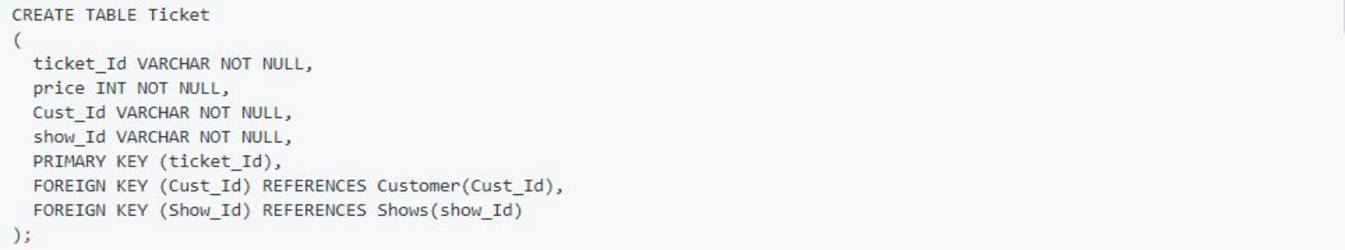
5.6.2 Output Table



show_Id	show_slot	show_Date	ent_Id	movie_Name
SHW1	slotA	2021-09-07	ENT1	The Eternals
SHW2	slotB	2021-08-05	ENT2	Inception
SHW3	slotC	2021-06-11	ENT3	Iron-Man 2
SHW4	slotD	2021-06-23	ENT3	Iron-Man 2

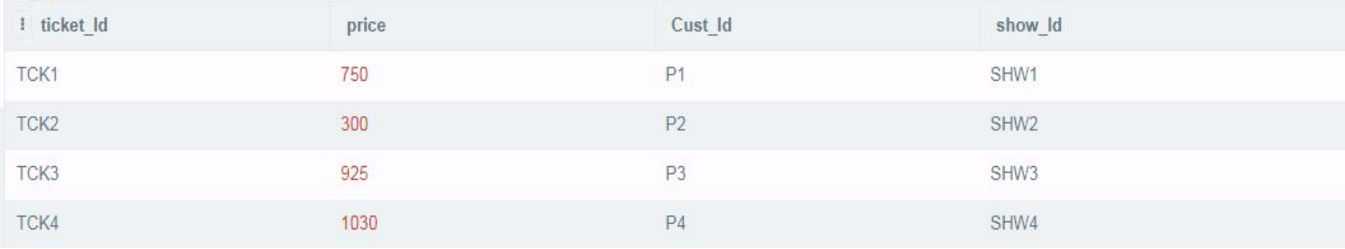
5.7 Ticket Table

5.7.1 Query



```
CREATE TABLE Ticket
(
  ticket_Id VARCHAR NOT NULL,
  price INT NOT NULL,
  Cust_Id VARCHAR NOT NULL,
  show_Id VARCHAR NOT NULL,
  PRIMARY KEY (ticket_Id),
  FOREIGN KEY (Cust_Id) REFERENCES Customer(Cust_Id),
  FOREIGN KEY (Show_Id) REFERENCES Shows(show_Id)
);
```

5.7.2 Output Table



ticket_Id	price	Cust_Id	show_Id
TCK1	750	P1	SHW1
TCK2	300	P2	SHW2
TCK3	925	P3	SHW3
TCK4	1030	P4	SHW4

6 Normalisation

6.1 Customer Table

Functional dependencies in this table are:
F.D. = {cust_id → cust_age cust_name cust_phone}
Candidate key = cust_id

- No multivalued attribute, hence the relation is in 1NF.
- Since there is only one attribute in the candidate key, all the non-key attributes are fully functional dependent on the primary key, and hence the relation is in 2NF form.
- Since there are no transitive dependencies present (no non-prime attribute derives other non-prime attributes), the relation is in 3NF.
- {cust_id → cust_name, cust_age, cust_phone} Since the left side of the FD is the super key, the relation follows BCNF.

6.2 Movie Table

Functional dependencies in this table are :
F.D. = { movie_name → room_name total_seats}
Candidate key = movie_name

- No multivalued attribute, hence the relation is in 1NF.
- Since there is only one attribute in the candidate key, all the non-key attributes are fully functional dependent on the primary key, and hence the relation is in 2NF form.
- Since there are no transitive dependencies present (no non-prime attribute derives other non-prime attributes), the relation is in 3NF.
- {movie_name → duration, genre, rating} Since the left side of the FD is the super key, the relation follows BCNF.

6.3 Movie Room Table

Functional dependencies in this table are :
F.D. = { ent_id → room_name total_seats}
Candidate key = ent_id

- No multivalued attribute, hence the relation is in 1NF.
- Since there is only one attribute in the candidate key, all the non-key attributes are fully functional dependent on the primary key, and hence the relation is in 2NF form.
- Since there are no transitive dependencies present (no non-prime attribute derives other non-prime attributes), the relation is in 3NF.
- {ent_id → total_seats, room_name} Since the left side of the FD is the super key, the relation follows BCNF.

6.4 Reservation Table

Functional dependencies in this table are :
F.D. = { reservation_id → ticket_id, show_id, seat_id}
Candidate key = reservation_id

- No multivalued attribute, hence the relation is in 1NF.
- Since there is only one attribute in the candidate key, all the non-key attributes are fully functional dependent on the primary key, and hence the relation is in 2NF form.
- Since there are no transitive dependencies present (no non-prime attribute derives other non-prime attributes), the relation is in 3NF.
- {reservation_id → ticket_id, show_id, seat_id} Since the left side of the FD is the super key, the relation follows BCNF.

6.5 Seat Table

Functional dependencies in this table are :
F.D. = { seat_id → seat_number, seat_row, ent_id}
Candidate key = seat_id

- No multivalued attribute, hence the relation is in 1NF.
- Since there is only one attribute in the candidate key, all the non-key attributes are fully functional dependent on the primary key, and hence the relation is in 2NF form.
- Since there are no transitive dependencies present (no non-prime attribute derives other non-prime attributes), the relation is in 3NF.
- {seat_id → seat_number, seat_row, ent_id} Since the left side of the FD is the super key, the relation follows BCNF.

6.6 Shows Table

Functional dependencies in this table are :
F.D. = { show_id → show_slot, show_date, ent_id, movie_name}
Candidate key = show_id

- No multivalued attribute, hence the relation is in 1NF.
- Since there is only one attribute in the candidate key, all the non-key attributes are fully functional dependent on the primary key, and hence the relation is in 2NF form.
- Since there are no transitive dependencies present (no non-prime attribute derives other non-prime attributes), the relation is in 3NF.
- {show_id → show_slot, show_date, ent_id, movie_id} Since the left side of the FD is the super key, the relation follows BCNF

6.7 Ticket Table

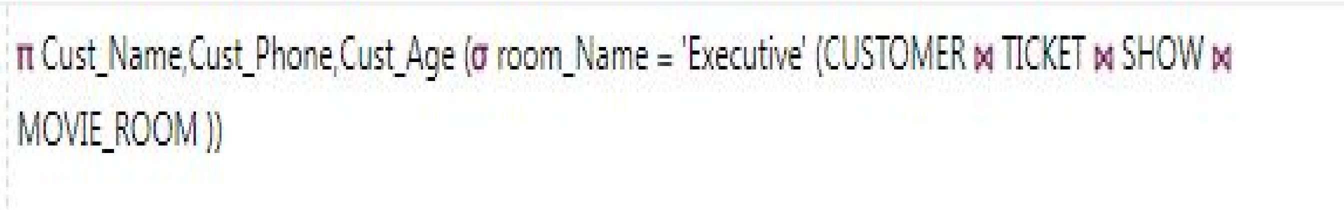
Functional dependencies in this table are :
F.D. = { ticket_id → price, cust_id, show_id}
Candidate key = ticket_id

- No multivalued attribute, hence the relation is in 1NF.
- Since there is only one attribute in the candidate key, all the non-key attributes are fully functional dependent on the primary key, and hence the relation is in 2NF form.
- Since there are no transitive dependencies present (no non-prime attribute derives other non-prime attributes), the relation is in 3NF.
- {ticket_id → price, cust_id, show_id} Since the left side of the FD is the super key, the relation follows BCNF.

7 Queries and Results

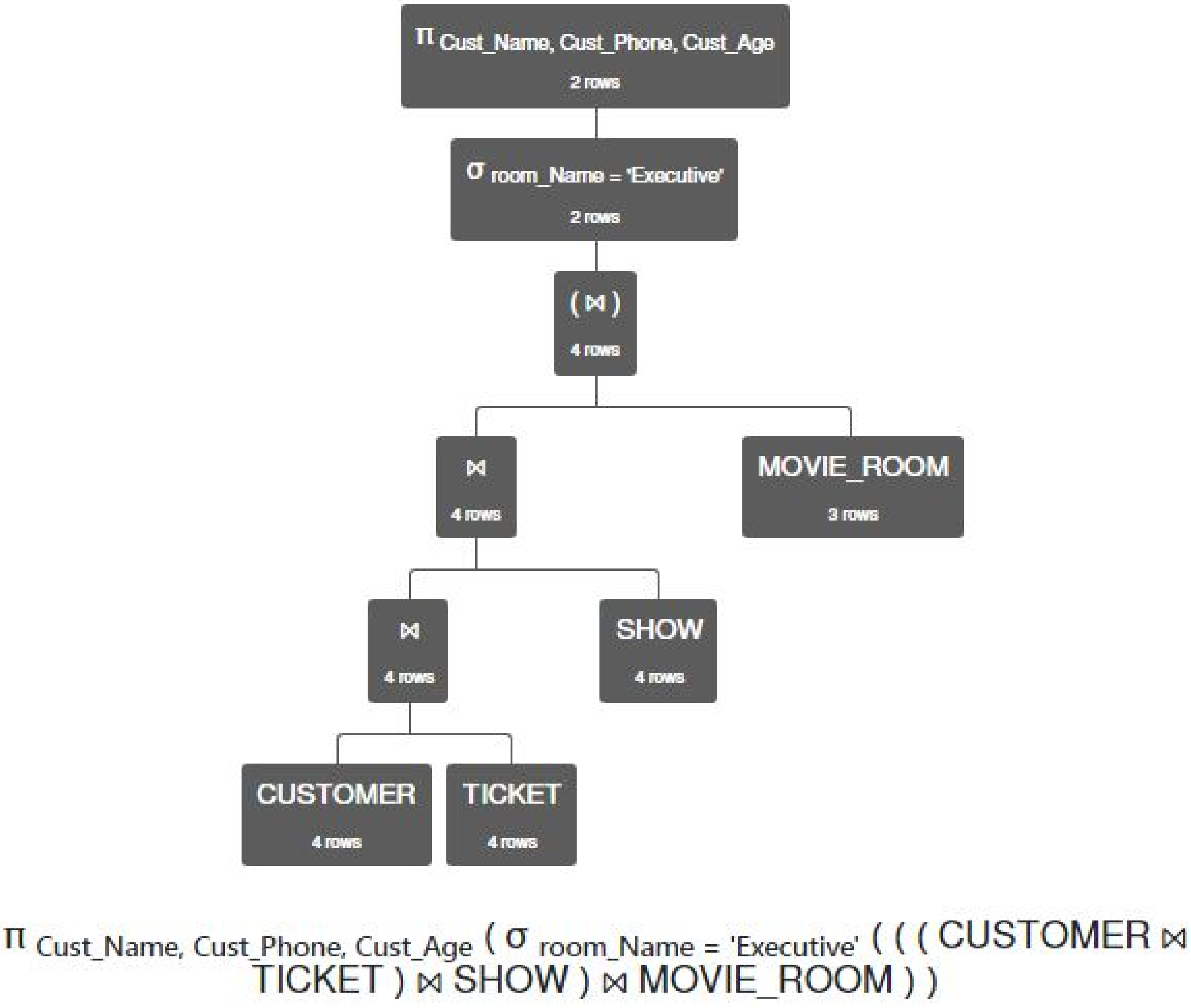
7.1 Show all the details of people who booked movie for only "Executive Class".

7.1.1 Relational Algebra Expression



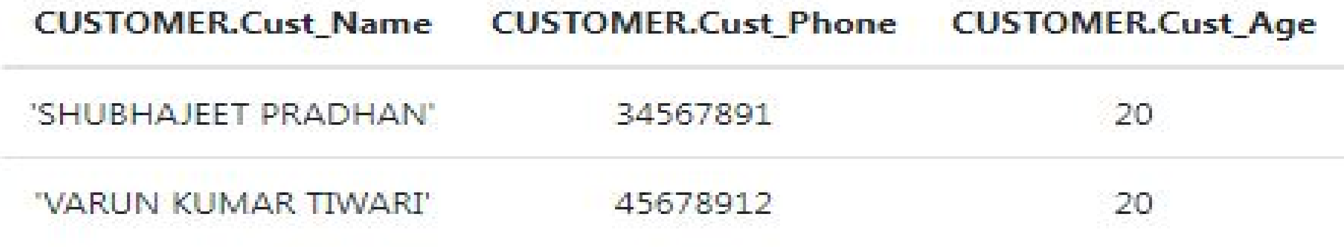
$\pi_{Cust\_Name, Cust\_Phone, Cust\_Age} (\sigma_{room\_Name = 'Executive'} (CUSTOMER \bowtie TICKET \bowtie SHOW \bowtie MOVIE\_ROOM))$

7.1.2 WorkFlow



$\pi_{Cust\_Name,\ Cust\_Phone,\ Cust\_Age} (\sigma_{room\_Name = 'Executive'} (((CUSTOMER \bowtie TICKET) \bowtie SHOW) \bowtie MOVIE\_ROOM))$

7.1.3 R.A. Result Table

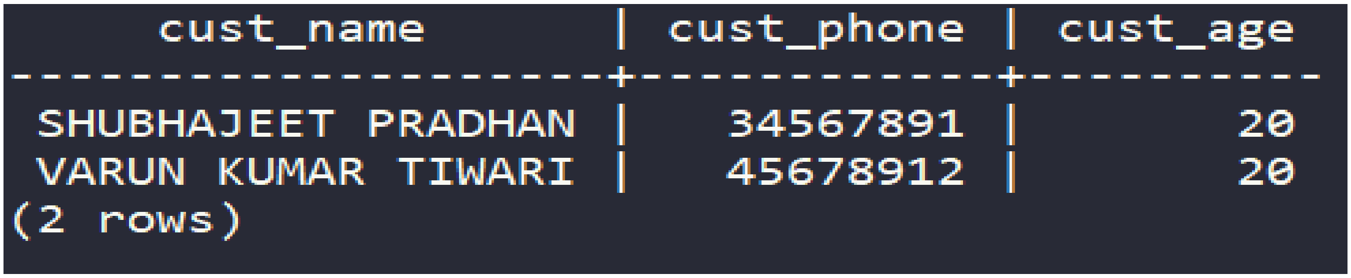


CUSTOMER.Cust_Name	CUSTOMER.Cust_Phone	CUSTOMER.Cust_Age
'SHUBHAJEET PRADHAN'	34567891	20
'VARUN KUMAR TIWARI'	45678912	20

7.1.4 SQL Query

```sql
SELECT cust_name, cust_phone, cust_age
FROM customer NATURAL JOIN ticket NATURAL JOIN shows NATURAL JOIN movie_room
WHERE room_name = 'Executive';
```

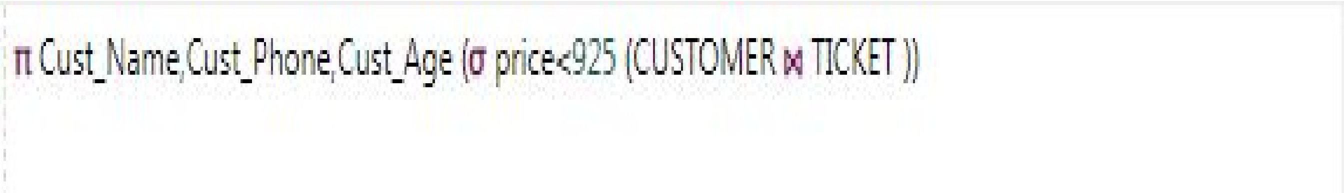
7.1.5 SQL Result Table



```
     cust_name       | cust_phone | cust_age
---------------------+------------+----------
 SHUBHAJEET PRADHAN |   34567891 |       20
 VARUN KUMAR TIWARI |   45678912 |       20
(2 rows)
```

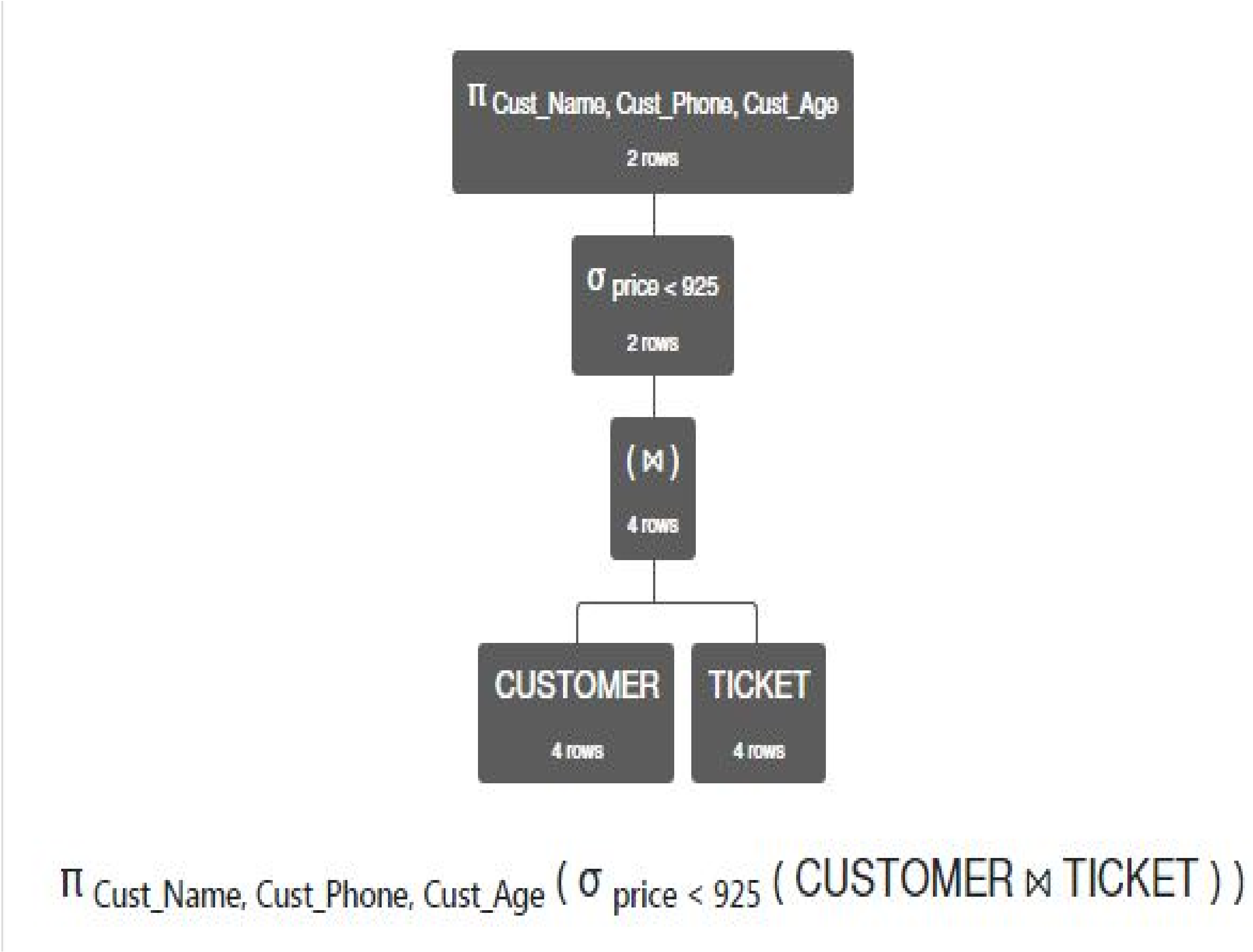
7.2 Show all the details of peoples who booked for a movie whose price is less than 925.

7.2.1 Relational Algebra Expression



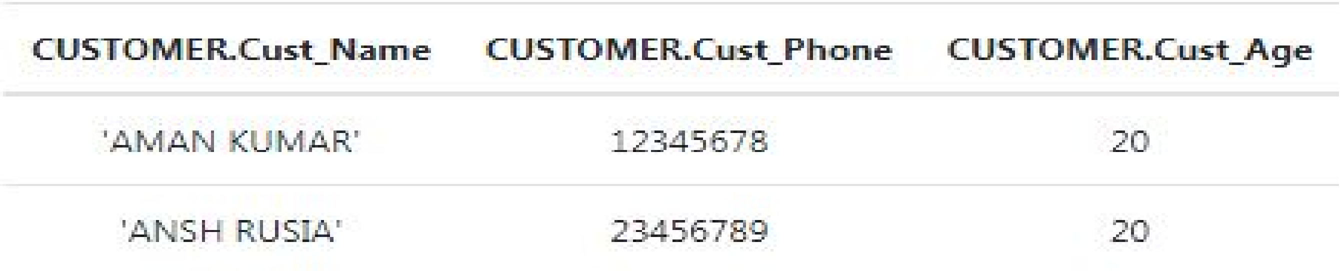
π Cust_Name,Cust_Phone,Cust_Age (σ price<925 (CUSTOMER $\bowtie$ TICKET))

7.2.2 WorkFlow



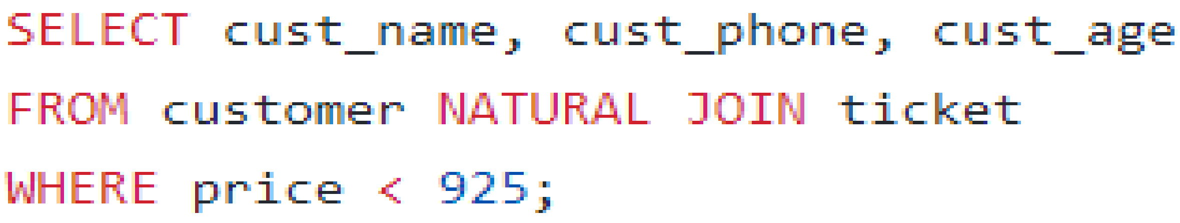
$\pi_{Cust\_Name,\ Cust\_Phone,\ Cust\_Age}\ (\ \sigma_{price\ <\ 925}\ (\ CUSTOMER \bowtie TICKET\ )\ )$

7.2.3 R.A. Result Table



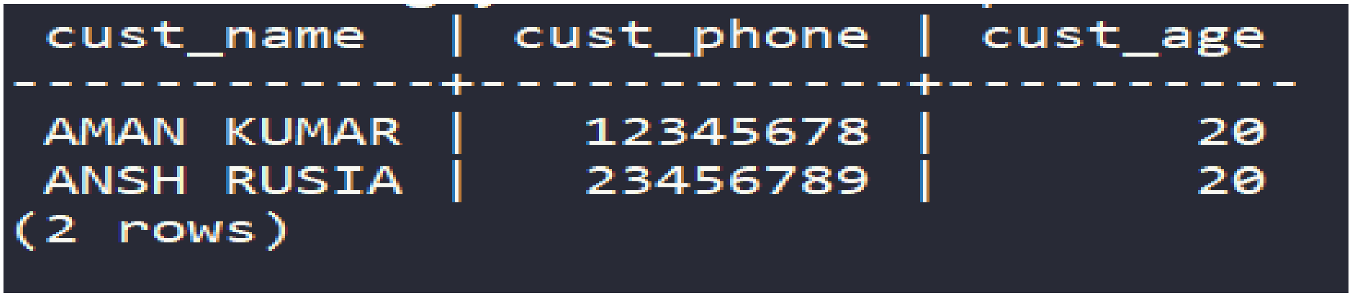
CUSTOMER.Cust_Name	CUSTOMER.Cust_Phone	CUSTOMER.Cust_Age
'AMAN KUMAR'	12345678	20
'ANSH RUSIA'	23456789	20

7.2.4 SQL Query



```
SELECT cust_name, cust_phone, cust_age
FROM customer NATURAL JOIN ticket
WHERE price < 925;
```

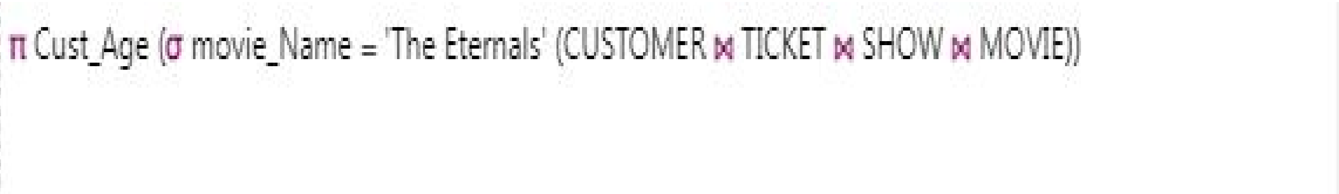
7.2.5 SQL Result Table



```
 cust_name   | cust_phone | cust_age
-------------+------------+----------
 AMAN KUMAR  |   12345678 |       20
 ANSH RUSIA  |   23456789 |       20
(2 rows)
```

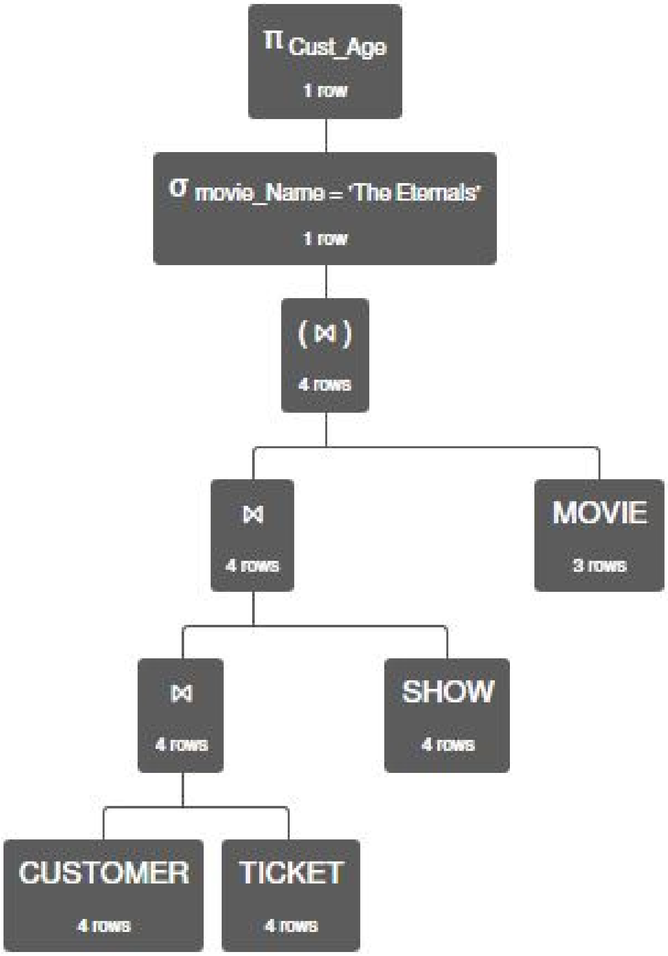
7.3 Show the age of all customers who are watching the movie "The Eternals".

7.3.1 Relational Algebra Expression



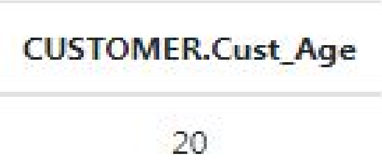
π Cust_Age (σ movie_Name = 'The Eternals' (CUSTOMER $\bowtie$ TICKET $\bowtie$ SHOW $\bowtie$ MOVIE))

7.3.2 WorkFlow



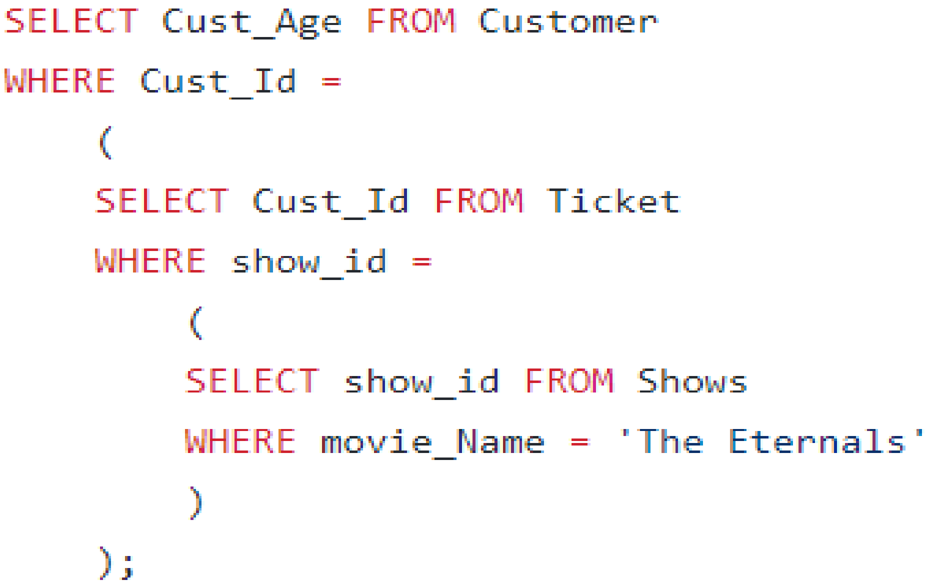
$\pi_{\text{Cust\_Age}}\ (\ \sigma_{\text{movie\_Name = 'The Eternals'}}\ (\ (\ (\ \text{CUSTOMER} \bowtie \text{TICKET}\ ) \bowtie \text{SHOW}\ ) \bowtie \text{MOVIE}\ )\ )$

7.3.3 R.A. Result Table



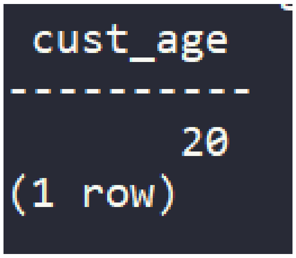
CUSTOMER.Cust_Age
20

7.3.4 SQL Query



```
SELECT Cust_Age FROM Customer
WHERE Cust_Id =
    (
    SELECT Cust_Id FROM Ticket
    WHERE show_id =
        (
        SELECT show_id FROM Shows
        WHERE movie_Name = 'The Eternals'
        )
    );
```

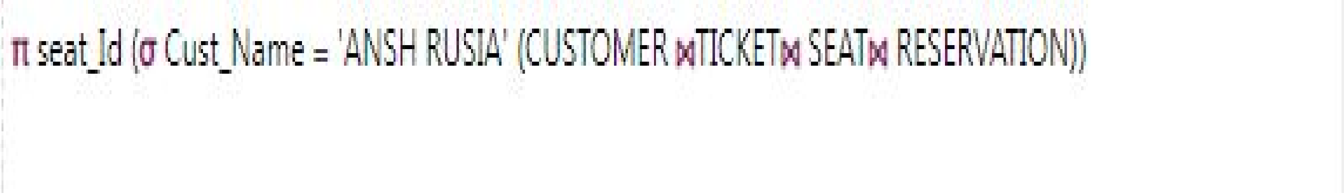
7.3.5 SQL Result Table



```
 cust_age
----------
       20
(1 row)
```

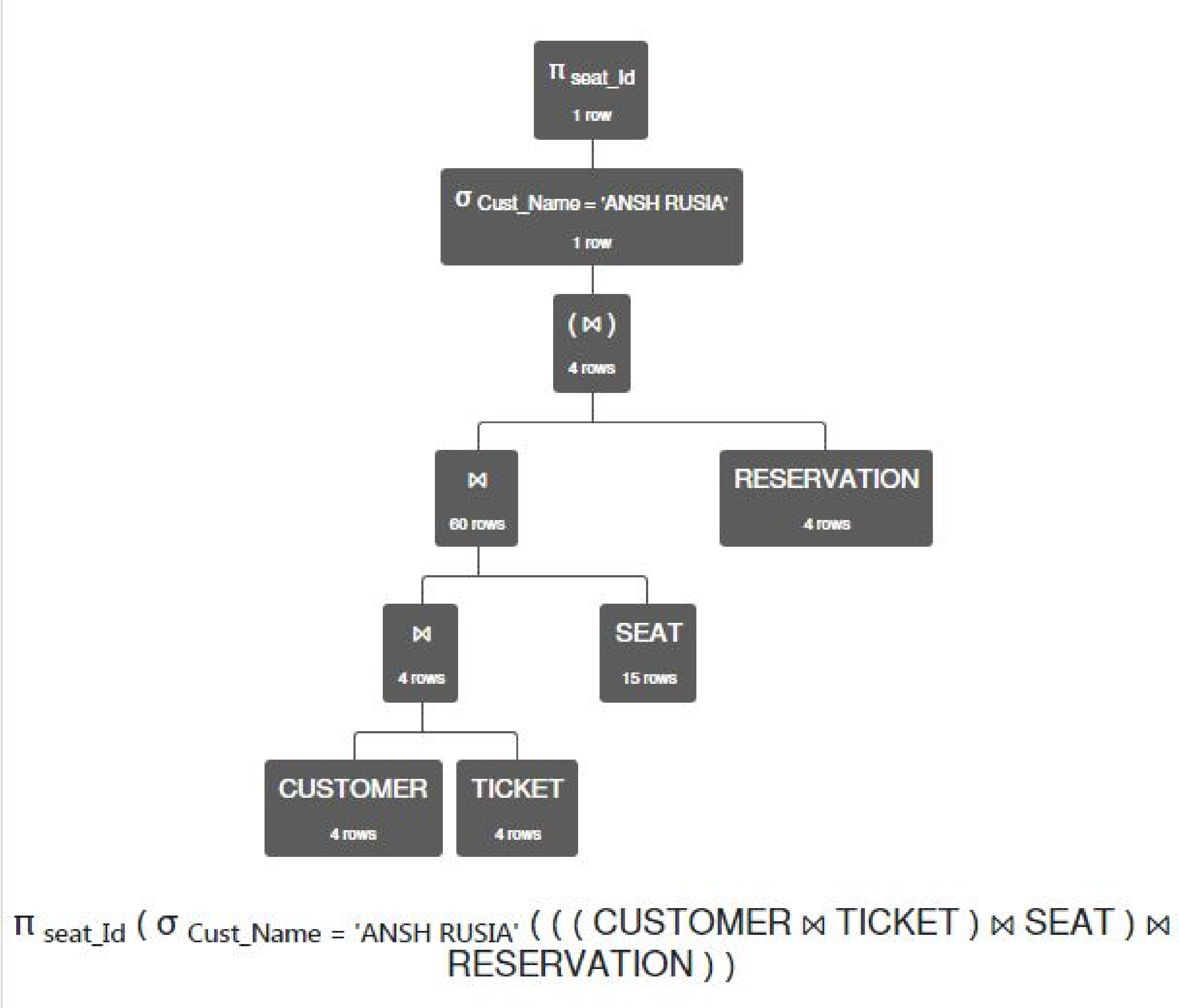
7.4 Show seat id of customer whose name is "ANSH RUSIA".

7.4.1 Relational Algebra Expression



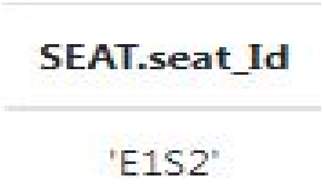
π seat_Id (σ Cust_Name = 'ANSH RUSIA' (CUSTOMER ⋈TICKET⋈ SEAT⋈ RESERVATION))

7.4.2 WorkFlow



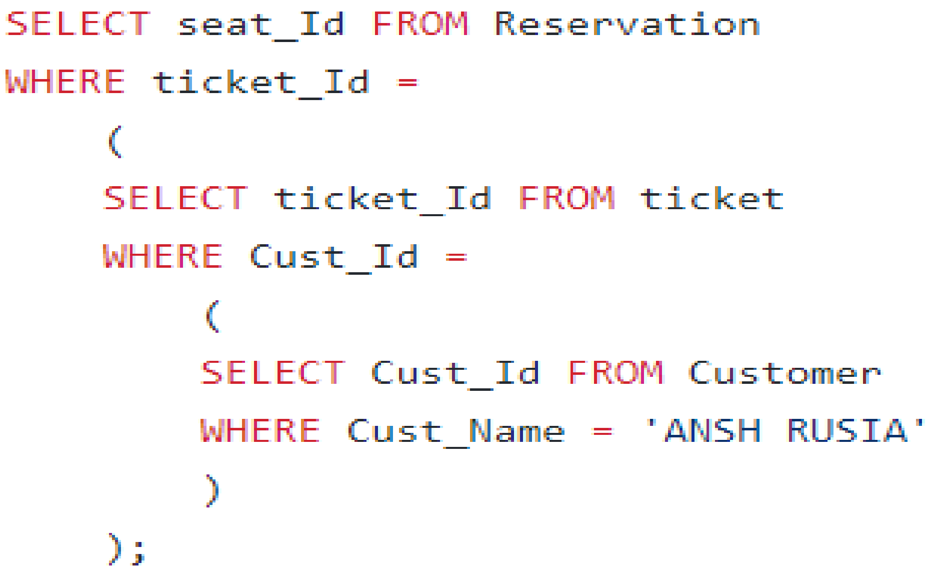
$\pi_{\text{seat\_Id}} (\sigma_{\text{Cust\_Name = 'ANSH RUSIA'}} (((\text{CUSTOMER} \bowtie \text{TICKET}) \bowtie \text{SEAT}) \bowtie \text{RESERVATION}))$

7.4.3 R.A. Result Table



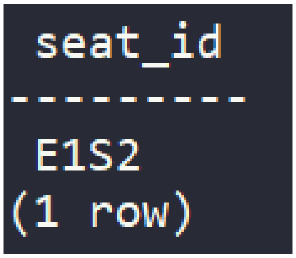
SEAT.seat_Id
'E1S2'

7.4.4 SQL Query



```
SELECT seat_Id FROM Reservation
WHERE ticket_Id =
        (
        SELECT ticket_Id FROM ticket
        WHERE Cust_Id =
                (
                SELECT Cust_Id FROM Customer
                WHERE Cust_Name = 'ANSH RUSIA'
                )
        );
```

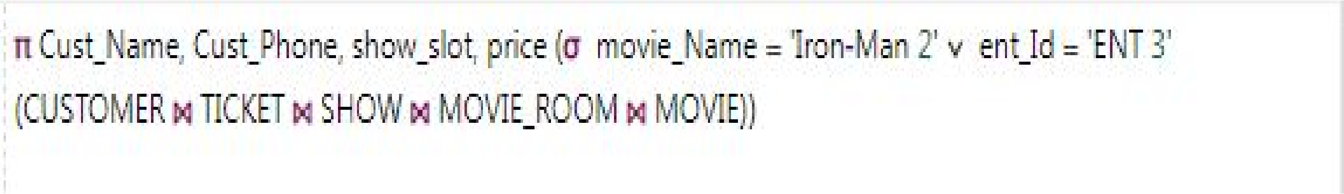
7.4.5 SQL Result Table



```
 seat_id
----------
 E1S2
(1 row)
```

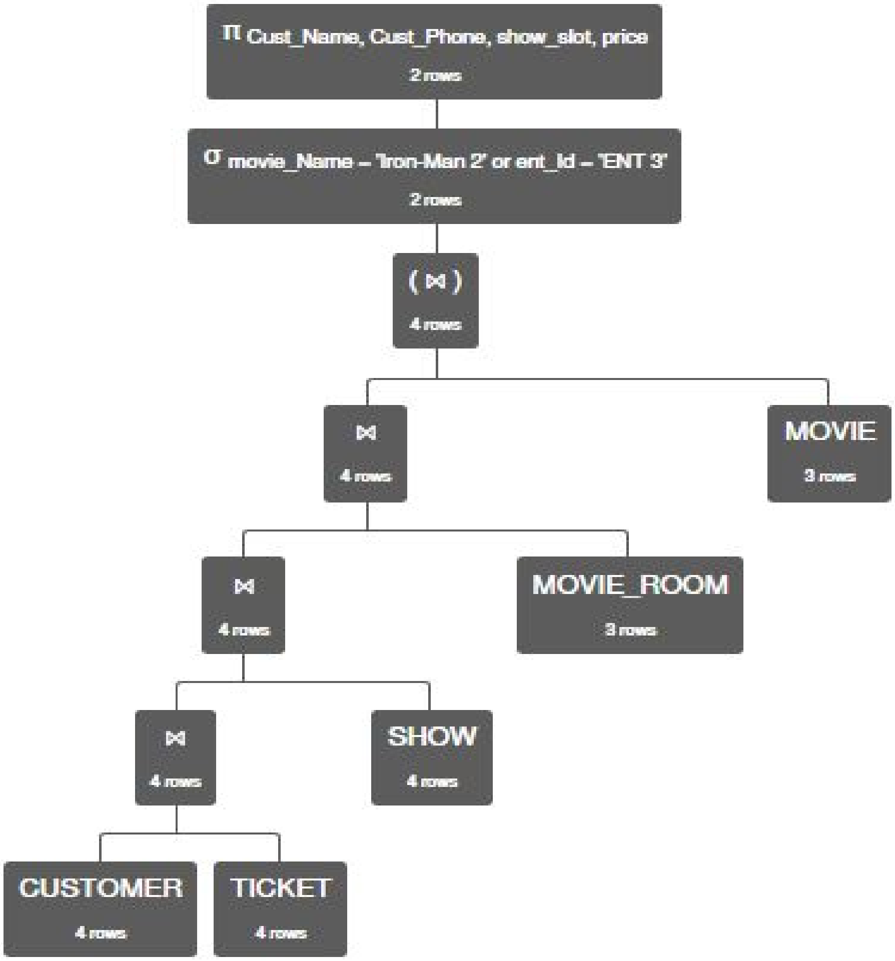
7.5 Show all the Customers who is watching "Iron-Man 2" in "ENT 3"

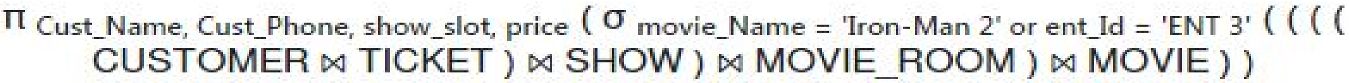
7.5.1 Relational Algebra Expression



π Cust_Name, Cust_Phone, show_slot, price (σ movie_Name = 'Iron-Man 2' ∨ ent_Id = 'ENT 3'
(CUSTOMER ⋈ TICKET ⋈ SHOW ⋈ MOVIE_ROOM ⋈ MOVIE))

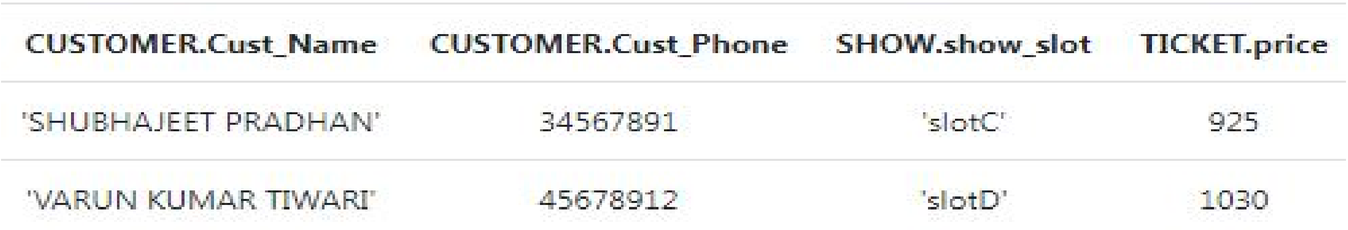
7.5.2 WorkFlow





π Cust_Name, Cust_Phone, show_slot, price (σ movie_Name = 'Iron-Man 2' or ent_Id = 'ENT 3' ((((
CUSTOMER ⋈ TICKET) ⋈ SHOW) ⋈ MOVIE_ROOM) ⋈ MOVIE))

7.5.3 R.A. Result Table

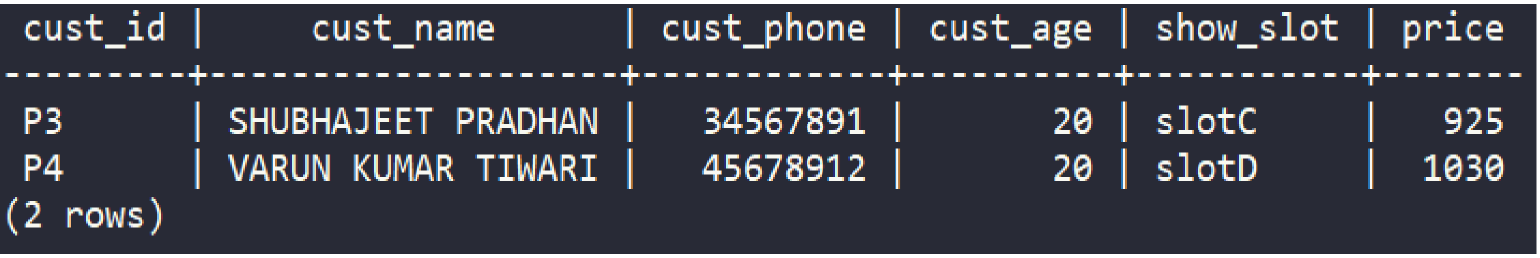


CUSTOMER.Cust_Name	CUSTOMER.Cust_Phone	SHOW.show_slot	TICKET.price
'SHUBHAJEET PRADHAN'	34567891	'slotC'	925
'VARUN KUMAR TIWARI'	45678912	'slotD'	1030

7.5.4 SQL Query

```
SELECT cust_id, cust_name, cust_phone, cust_age, show_slot, price
FROM customer NATURAL JOIN ticket NATURAL JOIN shows
WHERE movie_name = 'Iron-Man 2' AND ent_id = 'ENT3';
```

7.5.5 Result Table



```
 cust_id |      cust_name       | cust_phone | cust_age | show_slot | price
---------+----------------------+------------+----------+-----------+-------
 P3      | SHUBHAJEET PRADHAN |   34567891 |       20 | slotC     |   925
 P4      | VARUN KUMAR TIWARI |   45678912 |       20 | slotD     |  1030
(2 rows)
```