

Entertainment booking System

DBMS - IT214

S7_G9

Instructor: Prof. Minal Bhise

Mentor TA: Manoj Kumar

Team Members:

201901138 : Aditya Ramanuj

201901149 : Dev Tejot

201901215 : Siddharth Jadav

201901444 : Devyanish Koul



Dhirubhai Ambani Institute of
Information and Communication
Technology

Section 1

Final version of SRS

[A].Understand & Complete the Description of the Case Study/Problem Domain

1) Search online, Do the thinking and expand the problem domain for the short project description is given to you. Make sure that the Final SRS document has all points/sections as per the below-given sequence & numbering.

■ Introduction

- This system is basically concerned with entertainment booking system.
- In this project we try to educate viewers on how a entertainment booking system works.
- It store and retrieve information about the various transactions related to booking system.
- track of all its customers and thus schedule their shows accordingly.
- It maintains records of customers in the different shows on different dates.

■ Real world working flow

- In online booking system, engine is an application on social media pages to capture and process direct online reservations.
- With Most of people today now booking their reservations online and had to become reliant on online applications and software to believeer then reservations.
- By doing online workflow includes benefits of reaching large, new audience and increasing bookings
- But there is issue also that those reservations requires a commission fee to be paid to the third-party. When this fees starts to going up over 20%, it becomes incredibly frustrating for owners that are independent operators that sees their profits might melt away.
- If there is no third party applications or any online booking systems, then people will face huge crowds and it takes much more time for money transactions .
- if sometimes the situation occurs that we reaches any malls or theatre and if they are closed, then it wastes our time and transportation cost.

■ Purpose

- Main purpose is to remove long queues and to remove croud, in most places there are not enough ticket counters to handle the peak time crowd. But it is not practicable to increase the number of ticket counters just to serve at peak time.
- The purpose of the report is to collect and analyse all of the various concepts that have come up to define the system.
- To obtain a better knowledge of the project, we will predict and sort out how we hope this product will be used.
- The purpose of this document is to provide a detailed overview of our software product.

■ Intended Audience

- Through this project we present a comprehensive solution for ticket booking in system. Entertainment booking System, an online ticket selling software that is easy to understand, easy to use and offers the simplicity of fast point-and-click service to the customers.

■ Scope

- The developed system will facilitate online booking, keep customer records, provides an online menu on show schedules, show time, their prices and alternative payment method and will have page dedicated to customer queries and replies.

- The system includes reservation cancellation, specific seat reservation, classes and other management issues.
- All the manual work should be converted and computerized so that the manual work for the employees will be reduced.
- The database should be stored in computer rather than storing all the records manually.
- Introduced a new online booking system is not only technological innovation, but also will improve the services solve the difficult ticketing.

■ Description

- Surfing of data is easy with system.
- The wait time of the customer will be reduced and no need to wait in the lines.
- Accuracy of the information.
- it is a fast process compare to manual process.
- data is efficient.
- A reserving gadget allows you to do manner greater than simply passively receive bookings and bills online.
- Reservation technology has advanced to the factor that it's come to be a critical hub for coping with each component of your business, from advertising to distribution to operations.
- One of the most important benefits of a web reserving gadget is that you could live open for commercial enterprise 24/7, regardless of your hours of operation.
- Similar to how internet site permits visitors to discover extra approximately commercial enterprise without getting preserve of a brochure, a booking system makes it in order that they don't have to wait till they are able to get a preserve of you to reserve their spot.
- Everybody books Movie ticket online at some point. There is a movie reservations system and channel manager. Which is faster and cheaper.
- In addition to traditional booking systems, we also have like bookmyshow, Townscript etc. for Online ticket booking in only one platform.
- Although these sites expand a platform's reach, they also create some potential problems.
- Using a couple of such online market ought to cause overbooking. To keep away from that, it's excellent to have all reservation data in a single location.
- This document contains the problem statement that the current system is facing which is hampering the growth opportunities of the company
- The following SRS contains the detail product perspective from booking system.
- These booking systems bring activity business owners into the 21st century
- We can maximize reservations. By maximizing our reservations, it won't leave unused activity spots or rentals on the table.
- We can get paid quicker. It requires customers to prepay for activities and rentals. This puts money into our pocket faster.
- We're not tied to a phone. It means we can capture more potential business because reservations and information are always available. We would never get caught in a dreaded game of telephone tag.
- It becomes easy to manage our calendar. Through this we always know the status of our business, and our schedule can be easily managed from a convenient location.
- It helps to grow our business. We can quickly determine our most popular sellers, even the most popular add-ons by using an online entertainment booking system with some robust insights. \
 - It reduces workloads for the staff and optimizes customer service. The platforms can make sure that bookings are synced and the availability is updated with each booking processing.

Normal user:

- Can register for the site
- Search the hotel details basing on the criteria.
- Book the hotel room

Hotel Agent:

- Can register for the site

- Can add/update the details of the hotel.

Administrator:

- Will approve the new hotel details added to the application
- Can delete the user/hotel details.

Goals:

To complete the Online booking flow, registration of the users, adding details of the database.

- A reserving gadget allow you to do manner greater than simply passively receive bookings and bills online.
- Reservation technology has advanced to the factor that it's come to be a critical hub for coping with each component of your business, from advertising to distribution to operations.
- One of the most important benefits of a web reserving gadget is that you could live open for commercial enterprise 24/7, regardless of your hours of operation.
- Similar to how internet site permits visitors to discover extra approximately commercial enterprise without getting preserve of a brochure, a booking system makes it in order that they don't have to wait till they are able to get a preserve of you to reserve their spot.
- Everybody books Movie ticket online at some point. There is a movie reservations system and channel manager. Which is faster and cheaper.
- In addition to traditional booking systems, we also have like bookmyshow, Townscript etc. for Online ticket booking in only one platform.
- Although these sites expand a platform's reach, they also
- create some potential problems.
- Using a couple of such online market ought to cause
- overbooking. To keep away from that, it's excellent to have all reservation data in a single location.
- This document contains the problem statement that the current system is facing which is hampering the growth opportunities of the company
- The following SRS contains the detail product perspective from booking system.
- These booking systems bring activity business owners into the 21st century
- We can maximize reservations. By maximizing our reservations, it won't leave unused activity spots or rentals on the table.
- We can get paid quicker. It requires customers to prepay for activities and rentals. This puts money into our pocket faster.
- We're not tied to a phone. It means we can capture more
- potential business because reservations and information are always available. We would never get caught in a dreaded game of telephone tag.
- It becomes easy to manage our calendar. Through this we always know the status of our business, and our schedule can be easily managed from a convenient location.
- It helps to grow our business. We can quickly determine our most popular sellers, even the most popular add-ons by using
- an online entertainment booking system with some robust insights.
- It reduces workloads for the staff and optimizes customer service. The platforms can make sure that bookings are synced and the availability is updated with each booking processing.
- As EASY BOOKING is having manual booking system, they are facing some problems issuing booking requests of customers. All the necessary booking stuffs are being done in hard copy. So it become much difficult for staffs to keep the records updated all the time.
- As for example, if the customers need to change the check in date it become difficult for them to find out the customers booking details for updating as there are so many customers booking records. Again, regarding current system customers cannot give feedback online and also staff cannot reply to them promptly.
- Besides tourists from other South East Asian countries need to call directly for booking purposes. So they cannot get the chance to view their apartment rooms or hotels rooms before they make book.

[B]. Document the Requirements Collection/ Fact Finding Phase(10-20 pages)

1) Background Reading/s

- In this section we collected data from various sites and blogs, below are the main points.
- Releasing dates
- Ticket Rates
- Discount policy
- Ticket booking reports
- Handle customer complaints
- Membership
- Online payments
- Coupons and gift cards

▪ Better plans :

- Having bookings managed online means you can more efficiently manage your staff numbers as there isn't the requirement of someone manually managing bookings.
- Customers can easily book their desired slot at any time or from any place, assuring their spot at your venue is secure. They can also sign any waivers required, as well as pre-ordering any food or required items.
- most consumers use the website and booking system now a days that is optimised for mobile devices, ensuring it follows responsive design guidelines.
- Keep track of your partnership commissions.
- Track your growth with real-time reporting .
- Avoid double bookings with channel management.

▪ Some Websites.

- <https://vertabelo.com/blog/a-database-model-for-a-hotel-reservation-booking-app-and-channel-manager/>
- <https://www.checkfront.com/blog/what-is-an-online-booking-system>
- https://www.utdallas.edu/~chung/RE/Presentations07S/Team_1_Doc/Documents/SRS4.0.doc

2) Interview/s :

▪ Interview1

- Role Play interview plan
- System: Entertainment booking system
- Interviewee: Aditya Ramanuj (Role Play)
 - Designation : Manager

- Interviewer:
 - Aditya Ramanuj
 - Dev Tejot
 - Siddharth Jadav
 - Devyanish Koul
- Date: 5/10/2021
- Time: 4:30
- Place: Google Meet
- Purpose of interview
 - Problem regarding online Booking system
- Agenda:
 - Initial ideas
 - Problem facing after cancellation of ticket
 - Handle Customer's Complaints

■ Interview Summary1

- Role Play interview plan
- System: Entertainment booking system
- Interviewee: Aditya Ramanuj (Role Play)
 - Designation : Manager
- Interviewer:
 - Aditya Ramanuj
 - Dev Tejot
 - Siddharth Jadav
 - Devyanish Koul
- Date: 5/10/2021
- Time: 4:30
- Place: Google Meet
- Summary of Interview
 - Direct booking from web.
 - Instant Refund from cancellation of ticket
 - Arrange Customer's Complaints.

■ Interview2

- Role Play interview plan
- System: Entertainment booking system
- Interviewee: Aditya Ramanuj (Role Play)
 - Designation : Manager
- Interviewer:
 - Aditya Ramanuj
 - Dev Tejot
 - Siddharth Jadav
 - Devyanish Koul
- Date: 5/10/2021
- Time: 5:30
- Place: Google Meet
- Purpose of interview
 - Problem regarding online Booking system
- Agenda:
 - Problem in offline booking
 - Why we need a system
 - Handle Customer's Complaints

■ Interview Summary2

- Role Play interview plan
- System: Entertainment booking system
- Interviewee: Aditya Ramanuj (Role Play)
 - Designation : Manager
- Interviewer:
 - Aditya Ramanuj
 - Dev Tejot
 - Siddharth Jadav
 - Devyanish Koul
- Date: 5/10/2021
- Time: 5:30
- Place: Google Meet
- Summary of Interview
 - In offline booking we need to wait in line and waste of time.
 - It makes easy to book tickets and easy to manage all data and surfing.
 - Arrange Customer's Complaints.

■ Interview3

- Role Play interview plan
- System: Entertainment booking system
- Interviewee: Aditya Ramanuj (Role Play)
 - Designation : Manager
- Interviewer:
 - Aditya Ramanuj
 - Dev Tejot
 - Siddharth Jadav
 - Devyanish Koul
- Date: 5/10/2021
- Time: 4:30
- Place: Google Meet
- Purpose of interview
 - Problem regarding online Booking system
- Agenda:
 - Initial ideas
 - Problem facing after cancellation of ticket
 - Handle Customer's Complaints

■ Interview Summary3

- Role Play interview plan
- System: Entertainment booking system
- Interviewee: Aditya Ramanuj (Role Play)
 - Designation : Manager
- Interviewer:
 - Aditya Ramanuj
 - Dev Tejot
 - Siddharth Jadav
 - Devyanish Koul
- Date: 5/10/2021
- Time: 4:30
- Place: Google Meet
- Summary of Interview
 - Direct booking from web.

- Instant Refund from cancellation of ticket

Arrange Customer's Complaints.

■ Interview4

- Role Play interview plan
- System: Entertainment booking system
- Interviewee: Aditya Ramanuj (Role Play)
 - Designation : Manager
- Interviewer:
 - Aditya Ramanuj
 - Dev Tejot
 - Siddharth Jadav
 - Devyanish Koul
- Date: 5/10/2021
- Time: 4:30
- Place: Google Meet
- Purpose of interview
 - Problem regarding online Booking system
- Agenda:
 - Initial ideas
 - Problem facing after cancellation of ticket
 - Handle Customer's Complaints

■ Interview Summary4

- Role Play interview plan
- System: Entertainment booking system
- Interviewee: Aditya Ramanuj (Role Play)
 - Designation : Manager
- Interviewer:
 - Aditya Ramanuj
 - Dev Tejot
 - Siddharth Jadav
 - Devyanish Koul
- Date: 5/10/2021
- Time: 4:30
- Place: Google Meet
- Summary of Interview
 - Direct booking from web.
 - Instant Refund from cancellation of ticket

Arrange Customer's Complaints.

4) Questionnaire:

- Here we have collected 71 responses for our SRS project through google form. Following are the outcomes of our survey with some observations and google forms structure.



Entertainment Booking System Review

Form description

Which age group do you belong to *

15-18

18+

Do you book the tickets online ? *

Yes

No

Which application do you use for booking? .. *

Bookmyshow

Paytm

amazonpay

Do you want an application which contain all booking platform in one ? *

Yes

No

Maybe



Maybe

Do you want information of seats which are available. *

Yes

No

Maybe

Do you want any information about food facilities. *

Yes

No

Maybe

Do you want any updates regarding movies or concerts. *

Yes

No

Maybe

Are you satisfied with the current application you are using for booking tickets. *

1

2

3

4

5

Do you want to give any special Suggestions.

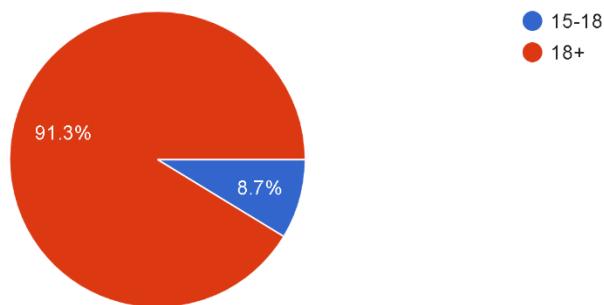
Long answer text



[C]. Create Fact-Finding Chart

Which age group do you belong to

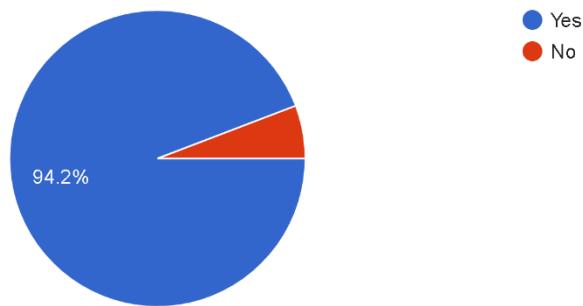
69 responses



Majority of the people responding belong to the age group 18+ as expected because of high use of smartphones.

Do you book the tickets online ?

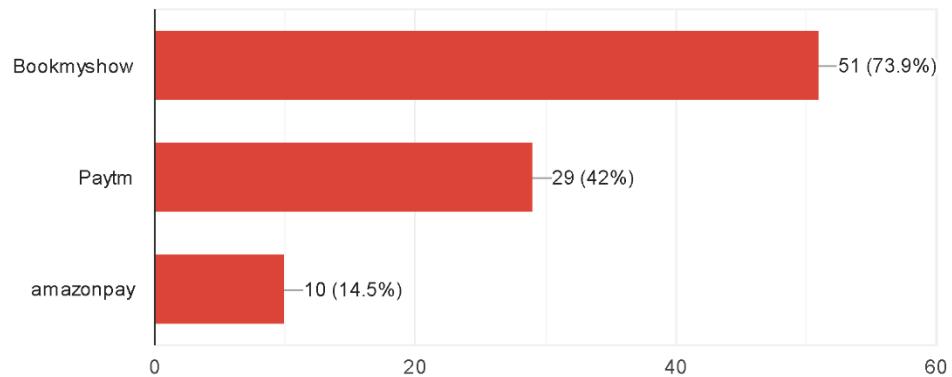
69 responses



Majority of the people use online booking system.

Which application do you use for booking? ..

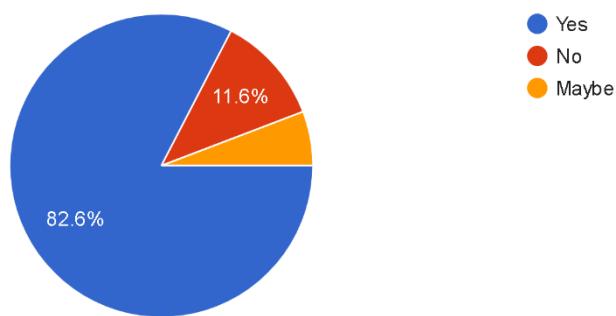
69 responses



Majority of the people use bookmyshow for book tickets.

Do you want an application which contain all booking platform in one ?

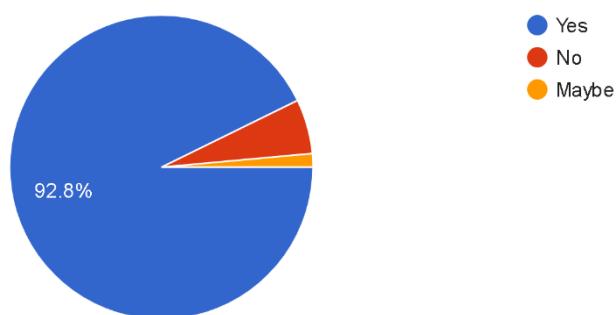
69 responses



82% customer want single platform for booking

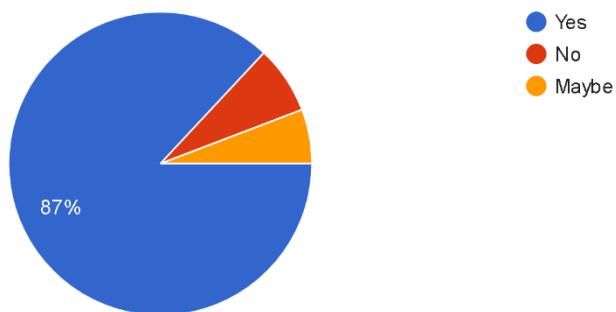
Do you want information of seats which are available.

69 responses



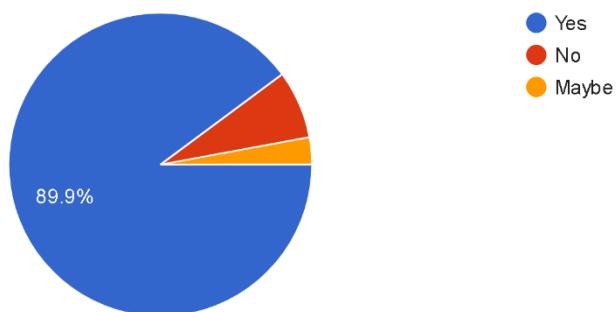
Do you want any information about food facilities.

69 responses



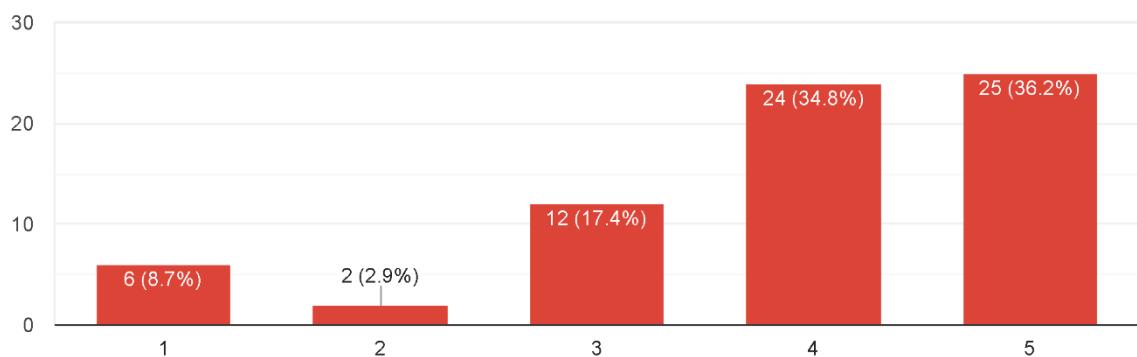
Do you want any updates regarding movies or concerts.

69 responses



Are you satisfied with the current application you are using for booking tickets.

69 responses



Majority of the people like the service provided by the app.

IT214 S7_G9

Short answer-based Question:

- **Question**
 - Do you want to give any special Suggestions.
- **Answers :**
 - Entertainment booking system should show
 - 1. Seats availability for cinema.
 - 2. Upcomming releases.
 - 3. Most popular.
 - 4. Food ordering.
 - 5. movie review etc
 - There are a few sites that have all in one platform like godaddy. So just ensure that you're creating something apart from that. Focus on providing value.
 - Entertainment Booking System should have security and data integrity.

4) Observation:

- System: Entertainment booking system
- Observation by:
 - Aditya Ramanuj
 - Dev Tejot
 - Siddharth Jadav
 - Devyanish Koul
- Date : 08/10/2021
- Time : 4:30
- Duration :30min
- Place : Google Meet
- Observstion:
 - The major problem in the payroll system shall be either resolved in two hours maintenance window.
 - The online booking system shall perform automatic backups once per batch.
 - System should assure data integrity.
 - System should include restore and recover functions in order to prevent data loss.

[C]. Create Fact-Finding Chart

Objective	Technique	Subject(s)	Time Constraint
To get background on government strategy and decisions	Background reading	Articles, Journals, and reference databases	0.5 day
To determine the features of application	Questionnaire	Google form	2 days
Problems with the lack of management			

Satisfaction % of decisions of			
Summary	Observation	All above	2 hours

[D]. List Requirements

Interface Requirement

- **Login Page :** When this function is selected by the theatre admin, he can log in through his username and password.
- **Registration Theatre :** When this function is selected by the theatre admin, he is offered a screen where he can register his theatre with the movie ticket booking system so that tickets can be booked for that theatre on the application
- Screen displaying information about product that the shop having :
 - **Add Movie :**
User can add movie that are currently running in the theatre.
 - **Remove Movie :**
User can remove movie that are no longer running in the theatre.
 - **Update Theatre :**
User can update the theatre seats for the tickets that have beenbooked/cancelled at the ticket window
- If the customers select the buy button then another screen of shopping cart will be opened.
 - **Book Ticket :**
User can book tickets for various movies from the application.
 - **Cancel Ticket :**
User can cancel his previously booked tickets.
 - **Make Payment :**
User can make the payment for the ticket that he wishes to book

Details of Public (feedback) :

- We need the details of the people that given feedback, so that we can manage and give the service that they want and those services they have suggested if available.

Multiple Payment options :

- The software should be given many types of payment systems, so they can reserve seats of their choice.
- Premium membership has access to special offers and discounts for the user who purchased.

Strong servers :

- We need strong type of servers if could handle theatres in large numbers, that could handle the requests of huge number of people and also the payment system should run smoothly and fast that if someone had cancelled their request that reserved seat should be free within minutes. Multiple transactions could run on single sort of time.

Database :

- The system require Data Base for the store the any transaction of the system like MYSQL etc.
- The system also require DNS(domain name space) for the naming on the internet.
- At the last user need web browser for interact with the system.

Application Efficiency :

- In software search results should enable users to find the most recent and relevant booking options and Users have to book or pay for the tickets only in a sort of time manner after the tickets added to the cart.
- System should move the window to the payments system only when the mandatory fields like date, movie time interval, location(if bus booking system) has been mentioned
- The booking confirmation should sent to the specified contact details.
- If customer wants to buy the product then he/she must be registered, unregistered user can't go to the shopping cart.
- The customer after login or registration can make order or cancel order of the product from the shopping cart.

Privacy

- Customer logins to the system by entering valid user id and password for the shopping, passes through login window.
- Use of captcha and encryption to avoid bots from booking tickets
- The transaction process will be fully secured, as send booking confirmation to the user's contact.
- The system must automatically log out all customers after a period of inactivity.
- The system does not have to store cookies that containing the user password.
- Sensitive data will be encrypted before being sent over insecure connections like the internet.
- The system should be available at all times, meaning the user can access it using a web browser or software.

If there is any type of data corruption in between then, a replacement page will be shown.

[E]. User Classes and Characteristics

- The users of website are administratotrs members who can maintain the website.
- The managers are assumed to have basic knowledge of computers and internet.
- The admins should have more knowledge about internal parts of the system, and they have potential to solve the small problems like power failure restart the system etc.
- User interface should be simple and it helps online user guide is sufficient to educate the users on how to use without any problems and difficulties

[F]. Operating Environment

- Hardware : i3 Processor or Android 5 or Higher
- Software : Windows 7 or Android OS
- Connectivity Requirement : Wi-Fi or Mobile internet
- External Interface Requirements
 - Theater database : Provide seat details
 - Login page
 - Aadhar Card Database

[G]. Product Functions

- The user(Admin, Customer) can login in to the system to perform different operations.
- The Admin can use the system to modify data to database.
- The customer and theatre owner can modify their own data/
- Time to time update regarding seat availability
- Discounts on tickets
- Get feedback from the customers.
- Provide reviews of the movie.

[H]. Privileges

Advantages

1. Business is open around the clock

- Using an online booking system means that your business is open 24 hours a day, seven days a week.

2. Maximum reservation can be done

- If any customer need to cancel the booking then spot automatically opens online by giving another customer the chance to book it.
- By maximize the reservations any spots or rentals on table will not empty.

3. Fast and quick transactions

- Customers can acquire to prepay for activities and rentals, fast and secure.

4. Analysis about business

- An online system provides dashboards of analytics that help to grow business.
- We can quickly determine popular sellers, the most requested time slots, and more add-ons by online booking system.

5. Easily to manage add-ons

- we can add more guides, activities, or grow hours to business.
- We can add additional perks to customers, so can make extra revenue in business.

Disadvantages

1. Need internet access

- In remote areas like small villages, this type of system doesn't work.
- Also need internet access to check reservations and to add bookings that are made online.

[I]. Assumptions

- Accuracy of the information of the user is the responsibility of the user itself.
- Need quick update in database.
- User must have connected to internet to use the system.
- The platform must be capable to run system.
- User can give comments and feedback about his experience on site.

[J]. Business Constraints

- Creation of digital infrastructure, good internet connection and fast accessible database server that can handle large amounts of traffic at a time.

- A Commercial database is used for maintaining the database and the application server takes care of the site.
- An online booking system provides you with a dashboard of analytics that help you grow your business.

Section 2

Final Noun Analysis

I. Noun (& Verb) Analysis.

Noun	Verb
Accuracy	Add
Activity	Advanced
Addition	Allow
Availability	Are
Benefits	Be
Bills	Becomes
Bookings	Booking
Books	Bring
Brochure	Can
Business	Capture
Calendar	Caught
Century	Cause
Channel	Come
Company	Compare
Component	Contains
Couple	Coping
Customer	Create
Data	Could
Detail	Determine
Distribution	Discover
Document	Do

Enterprise	Dreaded
Everybody	Expand
Factor	Facing
Gadget	Following
Game	Get
Growth	Getting
Hours	Grow
Hub	Hampering
Information	Has
Lines	Have
Location	Helps
Manager	Is
Manner	Keep
Market	Know
Money	Live
Movie	Make
Online	Makes
Operation	Manage
Opportunities	Maximizing
Order	Means
Owners	Need
Perspective	Optimizes
Phone	Ought

Platform	Paid
Pockets	Permits
Point	Prepay
Problem	Preserve
Process	Puts
Product	Reach
Rentals	Receive
Reservation	Reduces
Reserve	Reserving
Schedule	Synced
Sellers	Tried
Service	Updated
Site	Using
Spot	Van
Statement	Wait
Status	Will
System	Would
Table	
Tag	
Technology	
Telephone	
Ticket	

Accepted Noun & Verbs list

ENTITY SET	ATTRIBUTE	RELATION
Activity	LOGIN	In
Availability	BOOKING	contains
Company	THEATRE	connects
Customer	CUSTOMER	has
Document	ADMIN	Has
Gadget	USER	Has
Location	BOOKING	On
Manager	ADMIN	As
Market	USERID	With
Money	CUSTOMER	Has
Movie	THEATRE	In
Phone	SEATS	On
Platform	MOVIE	In
Product	OFFERS	Has
Reservation	ADMIN	As
Schedule	CUSTOMER	Has
System	ADMIN	With
Table		
Telephone	USERID	
Ticket	SELLS	Want
Visitors	OFFERS	Provide
Prepay	MOVIE	Manages
Grow	CUSTOMER	Can

Hampering		
Advanced	MOVIE	Has
Allow		
Booking	RELEASE DATE	Can

Rejected Noun & Verbs list

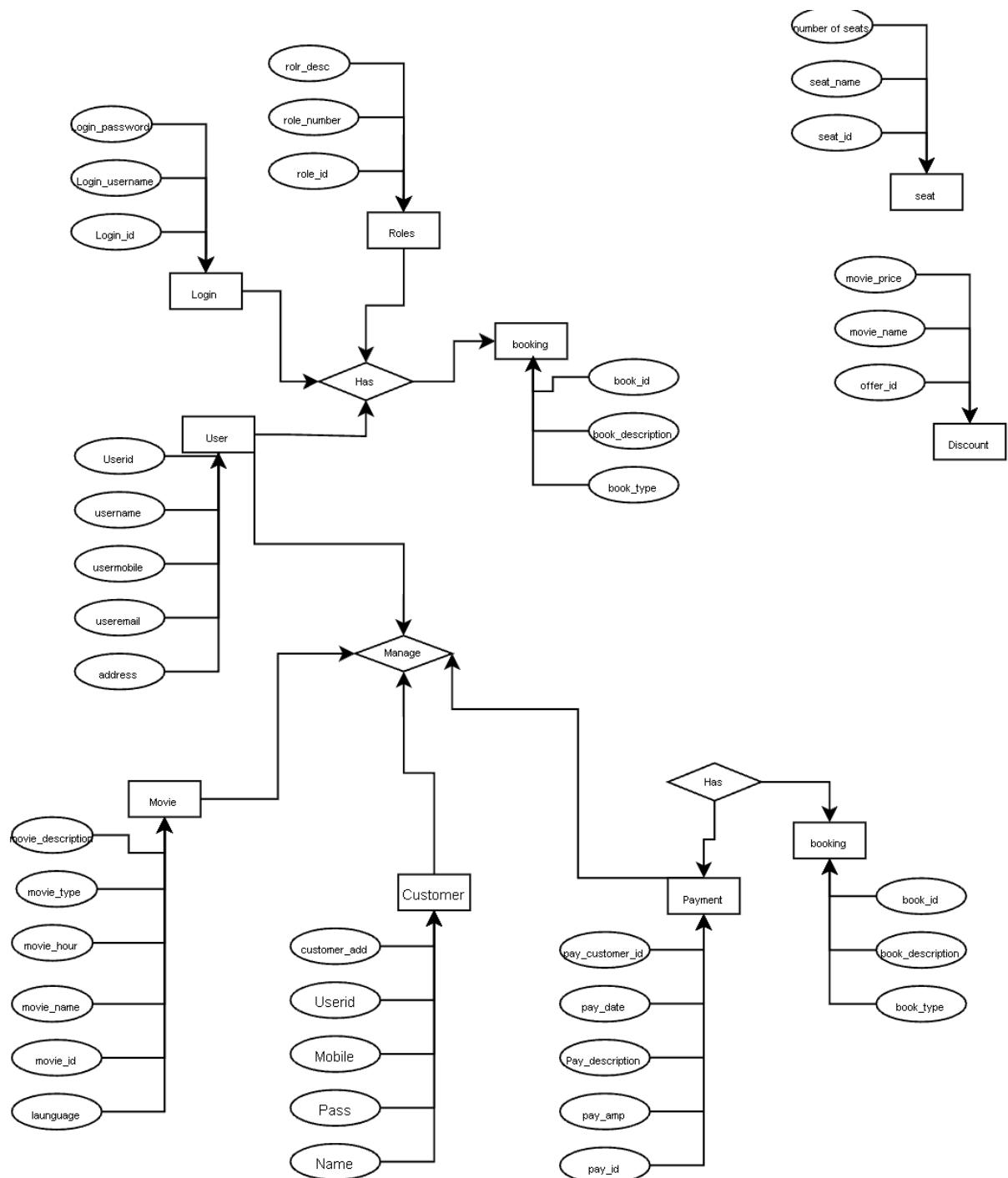
NOUN	REASON	VERB	REASON
Benefits	VAGUE	Allow	
Bills	GENERAL	Are	VAGUE
Bookings	VAGUE	Be	GENERAL
Books	GENERAL	Becomes	GENERAL
Brochure	IRRELEVANT	Capture	
Business	GENERAL	Caught	VAGUE
Calendar	VAGUE	Cause	GENERAL
Century	IRRELEVANT	Come	IRRELEVANT
Channel	GENERAL	Dreaded	
Data	GENERAL	Expand	IRRELEVANT
Detail	VAGUE	Facing	GENERAL
Distribution		Following	
Enterprise	GENERAL	Know	IRRELEVANT
Everybody	IRRELEVANT	Live	VAGUE
Factor		Make	GENERAL
Growth	VAGUE	Makes	VAGUE
Hours		Manage	IRRELEVANT

Hub	VAGUE		
Information	VAGUE		
Lines	IRRELEVANT		
Perspective	GENERAL		
Reservation			
Reserve	VAGUE		
Schedule	GENERAL		
Sellers	VAGUE		
Service	GENERAL		
Visitors	IRRELEVANT		
	VAGUE		

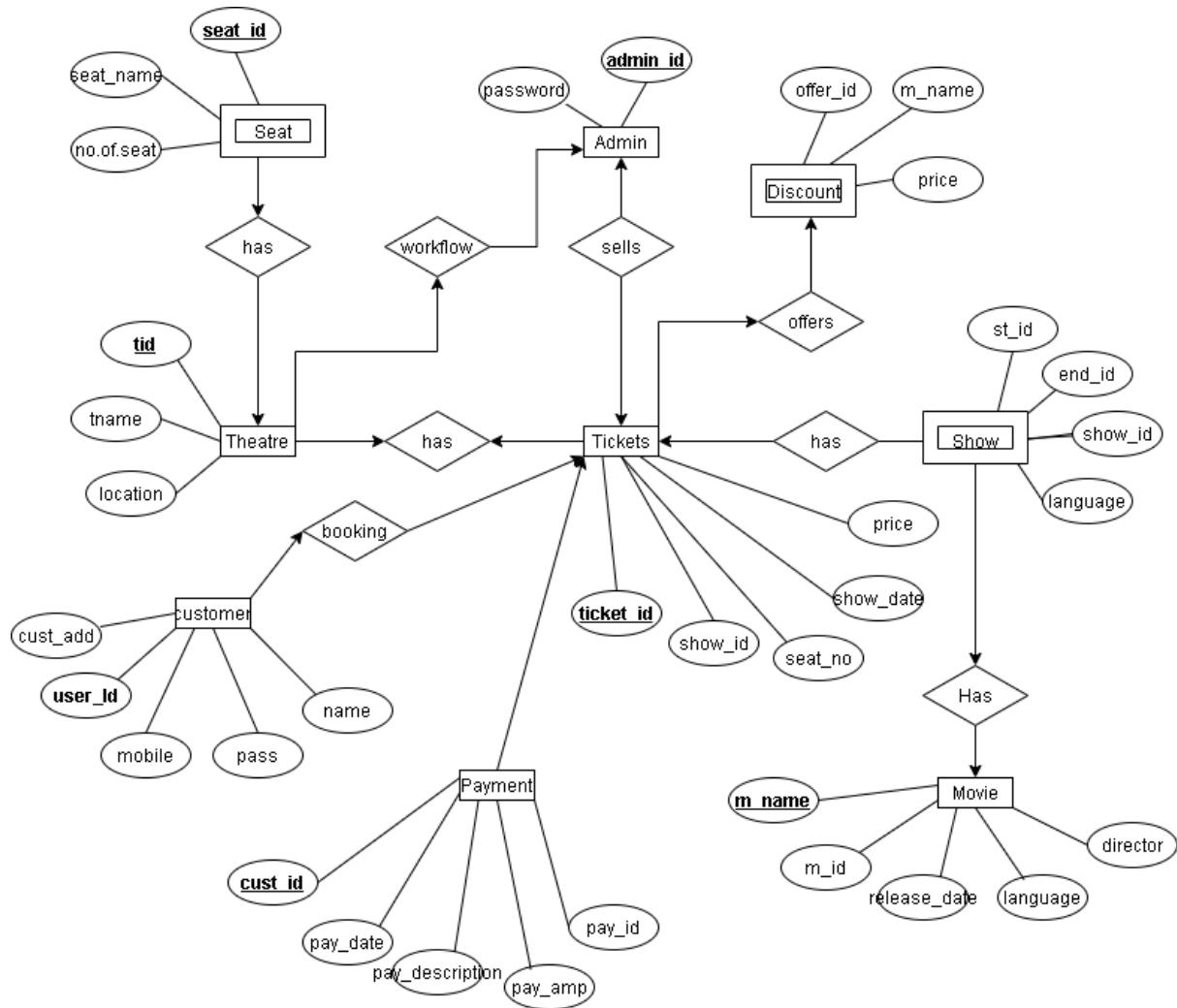
Section 3

Final ER-Diagrams (all versions)

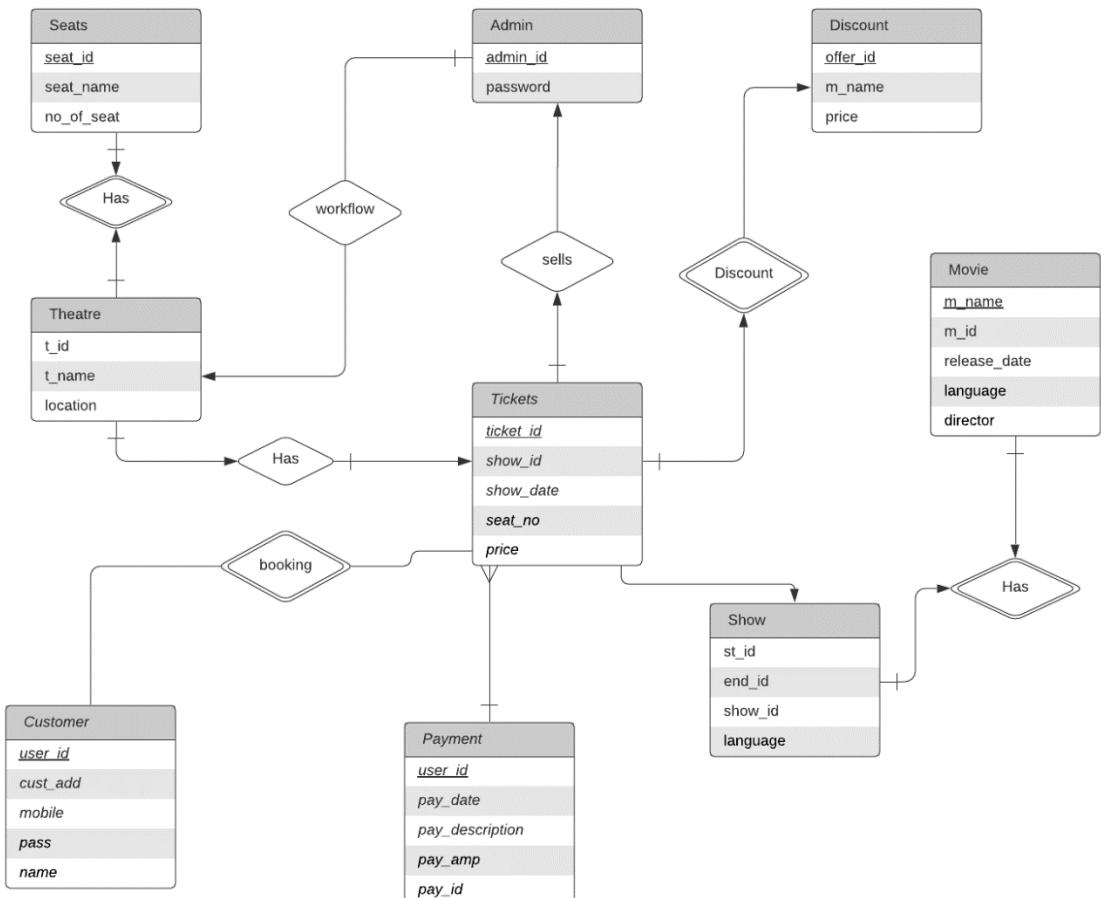
ER_DIAGRAM_V1



ER_DIAGRAM_V2



ER_DIAGRAM_V3



Section 4

Conversion of

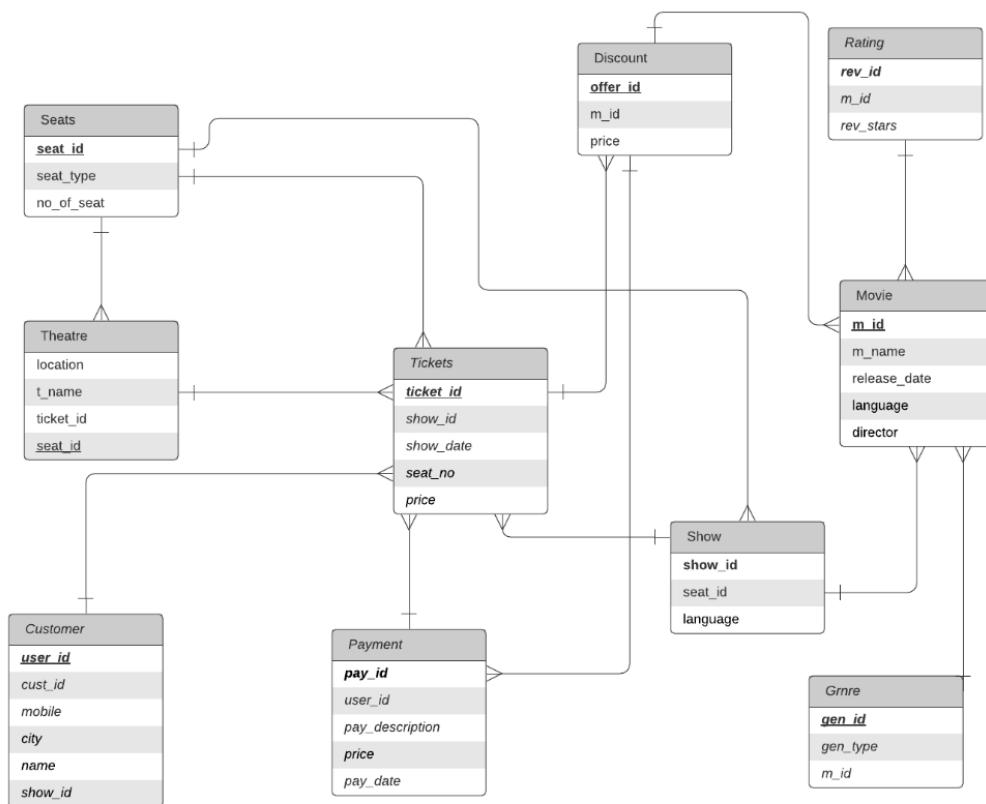
Final ER-Diagram to

Relational Model

Mapping E-R Model to Relational Model

- tickets(**ticket_id**, show_id, show_date, seat_no, price)
- customer(**user_id**, cust_id, mobile, city, name, show_id)
- payment(**pay_id**, user_id, pay_description, price, pay_date)
- show(**show_id**, seat_id, language)
- movie(**m_name**, m_id, release_date, language, director)
- theatre(**t_name**, location, ticket_id, seat_id)
- seats(**seat_id**, seat_type, no_of_seat)
- discount(**offer_id**, m_name, price, m_id)
- rating(**rev_id**, m_id, rev_stars)
- genre(**gen_id**, gen_type, m_name)

RELATIONAL SCHEMA



Section 5

Normalization and Schema Refinement

Normalization and Schema Refinement

Tickets

- tickets(ticket_id, show_id, show_date, seat_no, price)
- Primary Key: ticket_id(ticket_id, show_id, show_date, seat_no, price)
- Foreign Key: null
- Update anomaly: We need to update the price multiple times on weekends.
- Insert Anomaly : We cannot insert ticket_id until the confirmation of atleast one payment is not done.
- Delete anomaly : Loss of the information about the user if we delete the ticket_id.
- Redundancy: To avoid redundancy we can distribute or store tickets at different location in the same city.
- Transitive relation:

ticket_id -> user_id

user_id -> show_id

ticket_id ->show_id

Customer

- customer(user_id, cust_id, mobile, city, name, show_id)
- Primary Key: user_id(user_id, cust_id, mobile, city, name, show_id)
- foreign key : null

Payment

- payment(pay_id, user_id, pay_description, price, pay_date)
- primary key : pay_id(pay_id, user_id, pay_description, price, pay_date)
- foreign key : user_id
- Update anomaly: We need to update the prices of tickets in payment section.
- Insert Anomaly : For upcoming new movies and shows, we have to insert the price of tickets.
- Delete anomaly : Loss of the information about the pay_id, if we delete the user_id .
- Partial dependency : partial dependency : price ->pay_date

Here we have partial dependency between price and pay date because price is decided by pay date which is proper subset of candidate key(PK). So for getting rid of it we need to breakdown payment table in two parts.

Payment1 (pay_id,pay_date)

Payment2 (pay_id,price,user_id,pay_description)

Show

- show(show_id, seat_id, language)
- primary key : show_id(show_id, seat_id, language)
- foreign key : seat_id

Movie

- movie(m_id,m_name , release_date, language, director)
 - primary key : m_id(m_id,m_name , release_date, language, director)
 - foreign key : null
-
- Update anomaly : We need to update the m_id if the movie is in demand.
 - Insert Anomaly : We have to insert m_id for upcoming movies.
 - Delete anomaly : Loss of the information about the movies, if we delete the m_id.
 - Redundancy : release date is same for many different movies, So 2NF is

Movie_database-1 (m_id,release_date)

Movie_database-2 (m_name,release_date,language,director)

Now there is no redundancy left in 2NF, so there is no need to go to 3NF.

Theatre

- theatre(t_name, location, ticket_id, seat_id)
 - primary key : t_name(t_name, location, ticket_id, seat_id)
 - foreign key : seat_id
-
- Redundancy : If 1 movie is broadcasted in multiple theatre and if multiple movies are broadcasted in 1 theatre, then it will cause redundancy.

Hence 1NF form is

theatre(ticket_id,t_name,seat_id,location)

Theatre_info(t_name,location)

partial dependency : t_name \rightarrow ticket_id, seat_id

For removing partial dependency, we need to convert into a second normalized form.

Theatre_info (t_name,location)

Theatre(ticket_id,seat_id)

Now there is no redundancy in above relation, so there is no need to go to 2NF and 3NF.

Seats

- seats (seat_id, seat_type, no_of_seat)
- primary key: seat_id(seat_id, seat_type, no_of_seat)
- foreign key : null
- Transitive relation :

Ticket_id -> seat_id

Seat_id -> seat_type

Ticket_id -> seat_type

Discount

- discount(offer_id, price, m_id)
- primary key :offer_id(offer_id, price, m_id)
- foreign key : m_id
- Update anomaly :We need to update the discount price in festive seasons multiples times on weekends.
- Insert Anomaly : We cannot insert offer_id until the confirmation of at least one payment is not done.
- Delete anomaly : Loss of the information about the offer_id if we delete the m_id.
- Redundancy : Many movies and shows have same discount offers so it will cause redundancy.

So 1NF form is

Discount_info(m_id,price)

Foreign key : m_id

Partial dependency :

partial dependency (offer_id-> current price)

For removing partial dependency we need to convert into 2NF.

So 2NF is

Discount_info (m_id,price)

offer(offer_id,price)

Now there is no redundancy in above relation, so there is no need to go to 2NF and 3NF.

- Transitive relation :

m_id -> offer_id

offer_id -> price

m_id -> price

Rating

- rating(rev_id, m_id, rev_stars)
- primary key : rev_id
- foreign key : m_id
- Update anomaly :We need to update the rating of the current movies and shows every day.
- Insert Anomaly : we have to insert the rating for upcoming movies and shows.

Genre

- genre(gen_id, gen_type, m_id)
- primary key:gen_id
- foreign key:m_id

Section6: SQL: Final DDL Scripts, Insert statements, 40 SQL Queries with Snapshots of output of each query.

DDL Script

```
show
CREATE TABLE "final".show
(
    show_id bigint NOT NULL,
    seat_id bigint,
    language character varying,
    CONSTRAINT show_id PRIMARY KEY (show_id),
    CONSTRAINT seat_id FOREIGN KEY (seat_id)
        REFERENCES "final".seats (seat_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);
```

pgAdmin 4

Browser

- final
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (10)
 - customer
 - discount
 - genre
 - movie
 - payment
 - rating
 - seats
 - show
 - theater
 - tickets
 - Trigger Functions
 - Types
 - Views
 - public
 - Subscriptions
- Practice_db
- Project
- postgres
- Login/Group Roles
- Tablespaces

Query Editor

```
1 SELECT * FROM final.show
2 ORDER BY show_id ASC
```

Messages

Successfully run. Total query runtime: 254 msec.
89 rows affected.

Data Output

	show_id	seat_id	language
1	100	1	Somali
2	101	2	Khmer
3	102	3	Finnish
4	103	4	Belarusian
5	104	5	Hungarian
...

Notifications

Recorded time	Event	Process ID
No data found		

Explain

Use Explain/Explain analyze button to generate the plan for a query.

Payload

Successfully run. Total query runtime: 254 msec. 89 rows affected.

```
theatre
CREATE TABLE "final".theatre
(
    location character varying,
    t_name character varying,
    ticket_id bigint,
    seat_id bigint,
    theatre_id bigint NOT NULL,
    CONSTRAINT theatre_id PRIMARY KEY (theatre_id),
    CONSTRAINT seat_id FOREIGN KEY (seat_id)
        REFERENCES "final".seats (seat_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT ticket_id FOREIGN KEY (ticket_id)
        REFERENCES "final".tickets (tickets_id) MATCH SIMPLE
        ON UPDATE CASCADE
```

ON DELETE CASCADE
NOT VALID

);

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under the schema 'final'. The central area is the 'Query Editor' containing two lines of SQL:

```
1 SELECT * FROM final.theatre
2 ORDER BY theatre_id ASC
```

The 'Messages' panel at the bottom right indicates the query was successfully run with a runtime of 523 msec and 88 rows affected.

The 'Data Output' panel displays the results of the query as a table:

	location	t_name	ticket_id	seat_id	theatre_id
1	Glacier Hill	PVR	1	1	101
2	Browning	INOX	2	2	102
3	Barnett	MOVETIME	3	3	103
4	Westridge	PVR	4	4	104
5	Carpenter	INOX	5	5	105
6	Talisman	INOX	6	6	106
7	Norway Maple	PVR	7	7	107
8	Everett	CARNIVAL	8	8	108
9	Brickson Park	CINEPOLIS	9	9	109
10	Bayside	INOX	10	10	110
11	Hooper	CARNIVAL	11	11	111

discount

```
CREATE TABLE "final".discout
(
    offer_id bigint NOT NULL,
    price bigint,
    m_id bigint,
    CONSTRAINT offer_id PRIMARY KEY (offer_id),
    CONSTRAINT m_id FOREIGN KEY (m_id)
        REFERENCES "final".movie (m_id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID
);
```

Screenshot of PgAdmin 4 showing the "final" schema in the Browser pane. The "Tables (10)" section is selected, and the "customer" table is highlighted.

The Query Editor pane contains the following SQL query:

```
1 SELECT * FROM final.discount
2 ORDER BY offer_id ASC
```

The Data Output pane shows the results of the query:

offer_id	price	m_id
1	400	650
2	180	651
3	400	652
4	400	653
5	400	654
6	180	655
7	400	656
8	400	657
9	400	658
10	100	659

The Messages pane indicates: Successfully run. Total query runtime: 122 msec. 88 rows affected.

customer

```
CREATE TABLE "final".customer
(
    user_id bigint NOT NULL,
    mobile bigint,
    city character varying,
    name character varying,
    show_id bigint,
    PRIMARY KEY (user_id)
);
```

Screenshot of PgAdmin 4 showing the "final" schema in the Browser pane. The "Tables (10)" section is selected, and the "customer" table is highlighted.

The Query Editor pane contains the following SQL query:

```
1 SELECT * FROM final.customer
2 ORDER BY user_id ASC
```

The Data Output pane shows the results of the query:

user_id	mobile	city	name	show_id
1	862557664	Huangzhang	Jeddy	110
2	3188303582	Zhongbeo	Ransom	111
3	9665138254	Pizarno	West	112
4	7603642399	Aygevan	Gentlee	113
5	6524094174	A-da-Gorda	Berrie	114
6	5003385447	Donggo	Reggis	115
7	6207746469	Ragey	Theobald	116
8	5267968773	Patrung	Lonnie	117
9	8781677893	Salinas	Maitilde	118
10	2658081095	Banawang	Moshe	119

The Messages pane indicates: Successfully run. Total query runtime: 132 msec. 88 rows affected.

movie

```
CREATE TABLE "final".movie
(
m_id bigint NOT NULL,
m_name character varying,
release_date timestamp without time zone,
language character varying,
director character varying,
PRIMARY KEY (m_id)
);
```

The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database schema, including the 'final' schema which contains tables like customer, discount, genre, movie, payment, rating, seats, show, theater, and tickets. The 'Tables (10)' section is currently selected. The main area shows the SQL Query Editor with the following query:

```
1 SELECT * FROM final.movie
2 ORDER BY m_id ASC
```

The Messages panel indicates the query was successfully run with a total runtime of 159 msec and 88 rows affected. The Data Output panel displays the results of the query, showing 10 rows of movie data with columns: m_id, m_name, release_date, language, and director. The Explain panel provides a brief note about using the Explain/Explain Analyze button to generate a query plan.

m_id	m_name	release_date	language
1	650 Down From the Mountain	2021-09-22 00:00:00	Somali
2	651 Leonard Cohen: I'm Your Man	2021-07-14 00:00:00	Khmer
3	652 Lightning Bug	2021-06-09 00:00:00	Finnish
4	653 Copycat	2021-10-29 00:00:00	Belarusian
5	654 Black Peter (Cern?k Petr)	2021-08-12 00:00:00	Hungarian
6	655 War and Peace (Jang Aur Aman)	2021-07-25 00:00:00	Zulu
7	656 Kiss, The	2021-09-19 00:00:00	Malagasy
8	657 800 Bullets (800 Balas)	2021-03-23 00:00:00	Indonesian
9	658 Dunston Checks In	2021-05-14 00:00:00	English
10	659 Stargate: Continuum	2021-09-16 00:00:00	Indonesian

payment

```
CREATE TABLE "final".payment
(
pay_id bigint NOT NULL,
user_id bigint,
pay_description character varying,
price bigint,
pay_date timestamp without time zone,
CONSTRAINT pay_id PRIMARY KEY (pay_id),
CONSTRAINT user_id FOREIGN KEY (user_id)
    REFERENCES "final".customer (user_id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID
);
```

pgAdmin 4

File Object Tools Help

Browser

- final
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (10)
 - customer
 - discount
 - genre
 - movie
 - payment
 - rating
 - seats
 - show
 - theater
 - tickets
 - Trigger Functions
 - Types
 - Views
- public
- Subscriptions
- Practice_db
- Project
- postgres
- Login/Group Roles
- Tablespaces

Query Editor Query History

```
1 SELECT * FROM final.payment
2 ORDER BY pay_id ASC
```

Messages

Successfully run. Total query runtime: 154 msec.
89 rows affected.

Data Output

pay_id	user_id	pay_description	price	pay_date
1	16000	netbanking	4000	2021-09-22 00:00:00
2	32090	paytm	4125	2021-07-14 00:00:00
3	48180	netbanking	4250	2021-06-09 00:00:00
4	64270	paytm	4375	2021-09-29 00:00:00
5	80360	debit card	4500	2021-06-12 00:00:00

Notifications

Recorded time	Event	Process ID	Payload
No data found			

Explain

Use Explain/Explain analyze button to generate the plan for a query.

seat

```
CREATE TABLE "final".seats
(
seat_id bigint NOT NULL,
seat_type character varying,
PRIMARY KEY (seat_id)
);
```

pgAdmin 4

File Object Tools Help

Browser

- final
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (10)
 - customer
 - discount
 - genre
 - movie
 - payment
 - rating
 - seats
 - show
 - theater
 - tickets
 - Trigger Functions
 - Types
 - Views
- public
- Subscriptions
- Practice_db
- Project
- postgres
- Login/Group Roles
- Tablespaces

Query Editor Query History

```
1 SELECT * FROM final.seats
2 ORDER BY seat_id ASC
```

Messages

Successfully run. Total query runtime: 154 msec.
89 rows affected.

Data Output

seat_id	seat_type
1	gold
2	silver
3	platinum
4	gold
5	platinum

Notifications

Recorded time	Event	Process ID	Payload
No data found			

Explain

Use Explain/Explain analyze button to generate the plan for a query.

tickets

```
CREATE TABLE "final".tickets
()
```

```

tickets_id bigint NOT NULL,
show_id bigint,
show_date timestamp without time zone,
seat_no bigint,
price bigint,
PRIMARY KEY (tickets_id)
);

```

pgAdmin 4

File Object Tools Help

Browser

- final
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Procedures
 - > Sequences
 - > Tables (10)
 - > customer
 - > discount
 - > genre
 - > movie
 - > payment
 - > rating
 - > seats
 - > show
 - > theater
 - > tickets
 - > Trigger Functions
 - > Types
 - > Views
 - > public
 - > Subscriptions
- > Practice_db
- > Project
- > postgres
- > Login/Group Roles
- > Tablespaces

Query Editor Query History

```

1 SELECT * FROM final.tickets
2 ORDER BY tickets_id ASC

```

Messages

Successfully run. Total query runtime: 118 msec.
88 rows affected.

Data Output

	tickets_id	show_id	show_date	seat_no	price
1	1	110	2021-07-14 00:00:00	25	100
2	2	111	2021-06-09 00:00:00	26	180
3	3	112	2021-10-29 00:00:00	27	100
4	4	113	2021-06-12 00:00:00	28	180
5	5	114	2021-07-25 00:00:00	29	100

Explain

Use Explain/Explain analyze button to generate the plan for a query.

Notifications

Recorded time	Event	Process ID
No data found		

Successfully run. Total query runtime: 118 msec. 88 rows affected.

rating

```

CREATE TABLE "final".rating
(
    rev_id bigint NOT NULL,
    m_id bigint,
    rev_stars bigint,
    CONSTRAINT rev_id PRIMARY KEY (rev_id),
    CONSTRAINT m_id FOREIGN KEY (m_id)
        REFERENCES "final".movie (m_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);

```

Screenshot of pgAdmin 4 showing the "final" database structure and a query execution.

Browser:

- final
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (10)
 - customer
 - discount
 - genre
 - movie
 - payment
 - rating
 - seats
 - show
 - theater
 - tickets
 - Trigger Functions
 - Types
 - Views
- public
- Subscriptions
- Practice_db
- Project
- postgres
- Login/Group Roles
- Tablespaces

Query Editor: Final_db/postgres@PostgreSQL_13

```
1 SELECT * FROM final.rating
2 ORDER BY rev_id ASC
```

Messages: Successfully run. Total query runtime: 147 msec. 88 rows affected.

Data Output:

rev_id	bright	m_id	rev_stars	bright
1	1	650	1	
2	2	651	2	
3	3	652	3	
4	4	653	4	
5	5	654	2	
.	

Explain: Use Explain/Explain analyze button to generate the plan for a query.

Notifications: Recorded time Event Process ID

No data found

Successfully run. Total query runtime: 147 msec. 88 rows affected.

genre

```
CREATE TABLE "final".genre
(
    gen_id bigint NOT NULL,
    gen_type character varying,
    m_id bigint,
    CONSTRAINT gen_id PRIMARY KEY (gen_id),
    CONSTRAINT m_id FOREIGN KEY (m_id)
        REFERENCES "final".movie (m_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);
```

Screenshot of pgAdmin 4 showing the "final" database structure and a query execution.

Browser:

- final
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (10)
 - customer
 - discount
 - genre
 - movie
 - payment
 - rating
 - seats
 - show
 - theater
 - tickets
 - Trigger Functions
 - Types
 - Views
- public
- Subscriptions
- Practice_db
- Project
- postgres
- Login/Group Roles
- Tablespaces

Query Editor: Final_db/postgres@PostgreSQL_13

```
1 SELECT * FROM final.genre
2 ORDER BY gen_id ASC
```

Messages: Successfully run. Total query runtime: 166 msec. 88 rows affected.

Data Output:

gen_id	gen_type	m_id
1	Romance Western	650
2	Documentary	651
3	Comedy Drama	652
4	Adventure Comedy Fantasy	653
5	Children Drama	654
6	Comedy Drama	655
7	Comedy Drama Romance	656
8	Action Drama Mystery	657
9	Action Crime Romance Thriller	658
10	Action Horror	659

Explain: Use Explain/Explain analyze button to generate the plan for a query.

Notifications: Recorded time Event Process ID

No data found

Successfully run. Total query runtime: 166 msec. 88 rows affected.

IT214 S7_G9

1) How many movies in movie table

Query:

```
select count(m_id) from movie;
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** PostgreSQL 13, Final_db
- Query Editor:** Query History
- Query:**

```
1 CREATE SCHEMA final;
2 SET SEARCH_PATH TO final;
3
4 select count(m_id) from movie;
```
- Data Output:** A table with one row showing the count of movies.

count
88

- Messages:** Successfully run. Total query runtime: 289 msec.
1 rows affected.

Tuples : 1

2) find number of tickets where price is greater than 200.

Query:

```
SELECT * FROM tickets
WHERE (price >= 200);
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** PostgreSQL 13, Final_db
- Query Editor:** Query History
- Query:**

```
1 CREATE SCHEMA final;
2 SET SEARCH_PATH TO final;
3
4 --1
5 select count(m_id) from movie;
6
7 --2
8 SELECT * FROM tickets
9 WHERE (price >= 200);
10
11 |
```
- Data Output:** A table showing ticket details for 43 rows.

tickets_id	show_id	show_date	seat_no	price
40	83	192 2021-02-20 00:00:00	107	250
41	84	193 2021-09-13 00:00:00	108	250
42	86	195 2021-03-22 00:00:00	110	250
43	87	196 2021-09-22 00:00:00	111	250

- Messages:** Successfully run. Total query runtime: 131 msec.
43 rows affected.

Tuple: 43

3) How many gold seats are there in seats table?

Query:

```
SELECT *  
FROM seats  
WHERE (seat_type = 'gold');
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema, including Schemas, Tables, Columns, and various system objects like Foreign Data Wrappers and Functions. The main area is a Query Editor titled 'Final_db/postgres@PostgreSQL 13*', containing the following SQL code:

```
1 CREATE SCHEMA final;
2 SET SEARCH_PATH TO final;
3
4 --1
5 select count(m_id) from movie;
6
7 --2
8 SELECT * FROM tickets
9 WHERE (price >= 200);
10
11 --3
12 SELECT *
13   FROM seats
14 WHERE (seat_type = 'gold');
15
```

The 'Data Output' pane shows the results of the last query, which is a table with two columns: seat_id and seat_type. The data is as follows:

seat_id	seat_type
31	gold
32	gold
33	gold
34	gold

The 'Explain' button is visible above the results, and a message box indicates the query was successfully run with a runtime of 176 msec and 34 rows affected.

Tuple : 34

4) Number of movies where language is “Malagasy”.

Query:

SELECT *

FROM show

```
WHERE (language = 'Malagasy');
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with nodes like Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (2), and Tables (10). The Tables node is expanded, showing customer, discount, genre, movie, payment, rating, seat, show, theater, and tickets. The right pane shows a query editor with the following SQL code:

```
Final db/postgres@PostgreSQL 13
Query Editor Query History
--2
SELECT * FROM tickets
WHERE (price >= 200);
--3
SELECT *
FROM seats
WHERE (seat_type = 'gold');
--4
SELECT *
FROM show
WHERE (language = 'Malagasy');
```

The Data Output section displays the results of the last query:

show_id	seat_id	language
1	106	7 Malagasy
2	116	17 Malagasy
3	118	19 Malagasy
4	120	21 Malagasy
5	163	64 Malagasy

The Explain section contains a note: "Use Explain/Explain analyze button to generate the plan for a query." The Messages section shows: "Successfully run. Total query runtime: 91 msec. 5 rows affected."

Tuples : 5

5) Find information about movie 'Lord of War'.

Query:

```
SELECT *
FROM movie
WHERE (m_name = 'Lord of War');
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes icons for file operations, dashboard, properties, statistics, dependencies, and dependents.
- Connections:** Shows a connection to "Final_db/postgres@PostgreSQL_13" with session ID "201901149_1".
- Query Editor:** Contains the SQL query:

```
10
11 --3
12 SELECT *
13 FROM seats
14 WHERE (seat_type = 'gold');
15
16 --4
17 SELECT *
18 FROM show
19 WHERE (language = 'Malagasy');
20
21 --5
22 SELECT *
23 FROM movie
24 WHERE (m_name = 'Lord of War');
```
- Data Output:** Displays the results of the last query:

m_id	m_name	release_date	language	director
667	Lord of War	2021-07-16 00:00:00	Hungarian	Christy Nerenberg
- Messages:** Shows a success message: "Successfully run. Total query runtime: 56 msec. 1 rows affected."
- Notifications:** Shows "No data found".

Tuple :1

6)nHow many seats are there?

Query:

```
select count(seat_id) from "Final".seats;
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes icons for file operations, dashboard, properties, statistics, dependencies, and dependents.
- Connections:** Shows a connection to "Final_db/postgres@PostgreSQL_13" with session ID "201901149_1".
- Query Editor:** Contains the SQL query:

```
223
224 select count(seat_id) from "Final".seats;
225
226
```
- Data Output:** Displays the results of the query:

count
89

Tuple: 1

7) Display which movie names are in the language the 'French'?

Query:

```
SELECT m_name from
```

```
"Final".movie  
WHERE (language = 'French');
```

```
225  
226 SELECT m_name from  
227 "Final".movie  
228 WHERE (language = 'French');  
229  
230
```

Data Output Explain Messages Notifications

m_name
Crew, The
Company of Heroes

Tuple: 2

8) Display the genre of the movie named with m_id '677'.

Query:

```
select gen_type from "Final".genre  
where m_id ='677'
```

```
229  
230 select gen_type from "Final".genre  
231 where m_id ='677'  
232
```

Data Output Explain Messages Notifications

gen_type
Drama Musical

Tuple:1

9) Display movie Information(m_id) which has rating less than three.

Query:

```
select  
m_id as id from "Final".rating  
where rev_stars < '3'
```

```
237
```

Data Output Explain Messages Notifications

id
650
651
654
657
659

Tuple: 5

10) Display the average rating (stars) of the movies .

Query:

```
select avg(rev_stars) from "Final".rating
```

240		
Data Output Explain Messages Notifications		
	avg numeric	🔒
1	3.188888888888889	

Tuple: 1

11) Write all the movies with release date as the particular date 11-10-2021.

Query:

```
select m_name from "Final".movie  
where release_date = '11-10-2021'
```

243		
Data Output Explain Messages Notifications		
	m_name character varying	🔒
1	Steamboy (Such??mub??i)	
2	Loving You	

Tuple: 2

12) Find the number of movies who have got a 5 star ratings.

Query:

```
select count(rev_stars) from "Final".rating  
where rev_stars='5'
```

241
242 select count(rev_stars) from "Final".rating
243 where rev_stars='5'
244
245
246

Data Output Explain Messages Notifications		
	count bigint	🔒
1	25	

Tuple:1

13) Find the details of the customer in the city named 'Luchenza'

Query:

```
select * from "Final".customer  
where city='Luchenza'
```

```
244  
245 select * from "Final".customer  
246 where city='Luchenza'  
247  
248
```

Data Output Explain Messages Notifications

	user_id [PK] bigint	mobile bigint	city character varying	name character varying	show_id bigint
1	54	6148143396	Luchenza	Malvina	163

Tuple : 1

14) Find the details of all the tickets whose ticket price is the maximum of all the tickets

Query:

```
select * from "Final".tickets  
where price = (select max(price) from "Final".tickets)
```

```
247  
248 select * from "Final".tickets  
249 where price = (select max(price) from "Final".tickets)  
250  
251
```

Data Output Explain Messages Notifications

	tickets_id [PK] bigint	show_id bigint	show_date date	seat_no bigint	price bigint
1	17	126	2021-12-11	41	400
2	25	134	2021-11-05	49	400
3	38	147	2021-12-30	62	400
4	40	149	2021-09-09	64	400
5	42	151	2021-08-26	66	400

Tuple: 5

15) Find the number of all the tickets which have been sold for more than Rs 200.

Query:

```
select count(tickets_id) from "Final".tickets  
where price>'200'
```

```
250  
251 select count(tickets_id) from "Final".tickets  
252 where price>'200'  
253  
254
```

Data Output Explain Messages Notifications

	count bigint
1	43

Tuple:1

16) Find the total sum of price of all the tickets sold.

Query:

```
select sum(price) from "Final".tickets
```

```
253
254 select sum(price) from "Final".tickets
255
Data Output Explain Messages Notifications
sum numeric
1 20920
```

Tuple:1

17) Find the number of all the movies whose star ratings is greater than the average ratings

Query:

```
select count(rev_stars) from "Final".rating
where rev_stars > (select avg(rev_stars) from "Final".rating)
```

```
255
256 select count(rev_stars) from "Final".rating
257 where rev_stars > (select avg(rev_stars) from "Final".rating)
258
259
Data Output Explain Messages Notifications
count bigint
1 40
```

Tuple:1

18) What are the details of the movie with the name of the director as 'Ailbert Trott'.

Query:

```
select * from "Final".movie
where director='Ailbert Trott'
```

```

259 select * from "Final".movie
260 where director='Albert Trott'
261
262

```

Data Output Explain Messages Notifications

m_id	[PK] bigint	m_name	character varying	release_date	date	language	character varying	director	character varying
1	672	Les disparus de Saint-Agil		2021-09-25		Northern Sotho		Albert Trott	

Tuple:1

19) What is the name of the director whose movie bags the movie id of 4 ?

Query:

```

select director from "Final".movie
join "Final".rating on "Final".movie.m_id = "Final".rating.m_id
Where "Final".movie.m_id = 678

```

266

Data Output Explain Messages Notifications

director
Christa Cleland

✓ Successfully run. Total query runtime: 85 msec. 1 rows affected.

Tuple:1

20) Display all the movies which released in the month of September?

Query:

```

select m_name from "Final".movie
where extract(month from release_date)=09;

```

```

274 select m_name from "Final".movie
275 where extract(month from release_date)=09;
276
277

```

Data Output Explain Messages Notifications

m_name
Lightning Bug
Copycat
Black Peter (Cern?e Petr)
MGS: Philanthropy
Fear

Tuple:5

21) Find the number of seats with with the seat type as gold?

Query:

```
SELECT count(seat_id) FROM "Final".seats  
WHERE (seat_type = 'gold');
```

```
267 SELECT count(seat_id) FROM "Final".seats  
268 WHERE (seat_type = 'gold');  
269
```

Data Output Explain Messages Notifications

	count
1	19

Tuple:1

22) Display the names and release dates of all the movies which released in the month of November.

Query:

```
select m_name,release_date from "Final".movie  
where extract(month from release_date)=11;
```

```
276  
277 select m_name,release_date from "Final".movie  
278 where extract(month from release_date)=11;  
279
```

Data Output Explain Messages Notifications

	m_name	release_date
1	Kiss, The	2021-11-28
2	North West Frontier	2021-11-20
3	Lord of War	2021-11-16
4	Public Eye, The (Follow Mel)	2021-11-29
5	Phenomena (a.k.a. Creepers)	2021-11-05

Tuple:5

23) Write the names of the customers and their mobile numbers whose user id range from 18 to 31.

Query:

```
select name, mobile  
from "Final".customer  
where user_id between '18' and '31'
```

```

279
280 select name, mobile
281 from "Final".customer
282 where user_id between '18' and '31'
283
284

```

Data Output Explain Messages Notifications

	name character varying	mobile bigint
1	Gwendolin	5564102323
2	Rik	5808462279
3	Evangelia	2466320054
4	Wynn	7806347838
5	Chloe	3408033250

Tuple:5

24) Find the price of all the tickets from the tickets table using the show table where the show_id > 184.

Query:

```

select sum( price )
from "Final".tickets
join "Final".show on "Final".show.show_id = "Final".tickets.show_id
where "Final".show.show_id>184

```

```

283
284 select sum( price )
285 from "Final".tickets
286 join "Final".show on "Final".show.show_id = "Final".tickets.show_id
287 where "Final".show.show_id>184
288

```

Data Output Explain Messages Notifications

	sum numeric
1	2950

Tuple:1

25) Write the details of all the customers who are watching the shows from the first show to the last show.

Query:

```

SELECT "Final".customer.name, "Final".customer.mobile, "Final".customer.city
FROM "Final".customer
INNER JOIN "Final".show ON "Final".show.show_id = "Final".customer.show_id

```

```

289 SELECT "Final".customer.name, "Final".customer.mobile,"Final".customer.city
290 FROM "final".customer
291 INNER JOIN "Final".show ON "Final".show.show_id = "Final".customer.show_id
292
293

```

Data Output Explain Messages Notifications

	name character varying	mobile bigint	city character varying
1	Jeddy	8625577664	Huangzhuang
2	Ransom	3188303582	Zhongbao
3	West	9665138254	Pizarro
4	Gerrilee	7603642999	Aygavan
5	Berrie	6324094174	A-da-Gorda

Tuple:5

26) Find number of customer where price is 400 or 200.

Query:

```

SELECT price
FROM payment
WHERE price = 200
OR price = 400;

```

```

204
205 SELECT price
206 FROM payment
207 WHERE price = 200
208 OR price = 400;
209
210

```

Data Output

price bigint
400
400
400

Notifications

Recorded time	Event	Process ID
No data found		

Explain

Use Explain/Explain analyze button to generate the plan for a query.

Tuple: 2

27) Write the names of all the movies from the table using the genre table whose genre is given as Comedy.

Query:

```

SELECT m_name
FROM "Final".movie
WHERE m_id IN (SELECT m_id FROM "Final".genre WHERE gen_type= 'Comedy')

```

```

292
293 SELECT m_name
294 FROM "Final".movie
295 WHERE m_id IN (SELECT m_id FROM "Final".genre WHERE gen_type= 'Comedy')
296

```

Data Output Explain Messages Notifications

m_name character varying
Journey of Natty Gann, The
Lord of War
Steamboy (Such??mub??)
Two Thousand Maniacs!
Louis Cyr: The Strongest Man in the World

Tuple:5

28) Find total price in payment table.

Query:

```
SELECT SUM(price)  
FROM payment;
```

The screenshot shows a database interface with the following details:

- Query: 211, 212, 213: SELECT SUM(price) FROM payment;
- Data Output: A table with one row showing the sum of price as 24200.
- Notifications: No notifications.
- Event: No data found.
- Process ID: Explain button is present with a tooltip: "Use Explain/Explain analyze button to generate the plan for a query."

Tuple: 1

29) Which customer ordered the most payment.

Query:

```
select name from customer where user_id in  
(select user_id from payment group by user_id order by count(user_id) desc limit 1);
```

The screenshot shows a database interface with the following details:

- Query: 214, 215, 216: select name from customer where user_id in (select user_id from payment group by user_id order by count(user_id) desc limit 1);
- Data Output: A table with one row showing the name as Antony.
- Notifications: No notifications.
- Event: No data found.
- Process ID: Explain button is present with a tooltip: "Use Explain/Explain analyze button to generate the plan for a query."

Tuple:1

30) Print the details of customers who had id 20.

Query:

```
select *  
from Customer  
where user_id='20';
```

The screenshot shows a database interface with the following details:

- Query: 219, 220, 221, 222, 223: select * from Customer where user_id='20';
- Data Output: A table with one row showing user_id 20, mobile 2466320054, city Urmanggudang, name Evangelia, and show_id 129.
- Notifications: No notifications.
- Event: No data found.
- Process ID: Explain button is present with a tooltip: "Use Explain/Explain analyze button to generate the plan for a query."

Tuple: 1

31) Write the names of the customers and their mobile numbers whose user id range from 50-60.

Query:

```
select name, mobile  
from "final".customer  
where user_id between '51' and '60'
```

The screenshot shows a database query execution interface. The code entered is:

```
224  
225 select name, mobile  
226 from "final".customer  
227 where user_id between '51' and '60'  
228  
229  
230
```

The Data Output section displays the results:

name	mobile
Kerry	4847923595
Susann	7837046702
Russ	9824490133

The Notifications panel shows a success message: "Successfully run. Total query runtime: 51 msec. 10 rows affected."

Tuple:10

32) Find the details of all the tickets whose ticket type is silver

Query:

```
select * from "Final".seat  
where seat_type='silver';
```

The screenshot shows a database query execution interface. The code entered is:

```
229  
230 select * from seats  
231 where seat_type='silver';  
232
```

The Data Output section displays the results:

seat_id	seat_type
25	25 silver
26	26 silver

The Notifications panel shows a success message: "Successfully run. Total query runtime: 218 msec. 26 rows affected."

Tuple:26

33) Display the movie with English language?

Query:

```
select * from "Final".movie  
where language='English'
```

Query Editor Query History

```
1 select * from "Final".movie
2 where language='English'
```

Data Output Explain Messages Notifications

m_id	m_name	release_date	language	director
658	Dunston Checks In	2021-08-01 00:00:00	English	Jo-ann Lovelock

34) Find location and theatre_id where t_name is PVR.

Query:

```
select location,theatre_id
from theatre
where t_name='PVR';
```

```
234 select location,theatre_id
235 from theatre
236 where t_name='PVR';
237
```

Data Output

location	theatre_id
Spelght	184
Pawling	188

Notifications

Recorded time Event Process ID

No data found

Explain

Use Explain/Explain analyze button to generate the plan for a query.

Successfully run. Total query runtime: 126 msec. 17 rows affected.

Tuple:17

35) Display the list of Comedy movies.

Query:

```
select * from "Final".genre
where gen_type='Comedy'
```

Query Editor Query History

```
1 select * from "Final".genre
2 where gen_type='Comedy'
```

Data Output Explain Messages Notifications

gen_id	gen_type	m_id
13	Comedy	662
18	Comedy	667
26	Comedy	675
42	Comedy	691
47	Comedy	696
57	Comedy	706
59	Comedy	708
70	Comedy	719
74	Comedy	723
86	Comedy	735

Tuple:10

36) Find details where t_name = INOX.

Query:

```
select * from "final".theatre  
where t_name='INOX'
```

The screenshot shows a database interface with a query editor and a results pane. The query is:

```
234 select location,theatre_id  
235 from theatre  
236 where t_name='INOX';  
237
```

The results table has columns 'location' and 'theatre_id'. The data is:

location	theatre_id
Spaght	184
Pewling	188

Notifications: No data found.

Explain: Use Explain/Explain analyze button to generate the plan for a query.

Recorded time: Event: Process ID: Payload: Successfully run. Total query runtime: 126 msec. 17 rows affected.

Tuple: 17

37) Display the list of movies whose price is less or equal to 250.

Query:

```
select * from "Final".discount  
where price <= 250;
```

The screenshot shows a database interface with a query editor and a results pane. The query is:

```
1 select * from "Final".discount  
2 where price <= 250;
```

The results table has columns 'offer_id', 'price', and 'm_id'. The data is:

offer_id	price	m_id
1	2	180
2	6	180
3	10	100
4	12	100
5	13	100
6	16	100
7	17	250
8	19	250
9	21	250
10	22	250
11	24	100
12	25	250
13	26	100
14	27	250
15	28	250
16	29	100
17	30	250
18	31	250
19	32	100
20	34	250

Notifications: 28°C Smoke ENG 20-11-2021

Tuple: 20

38) Find the number of all the movies whose star ratings is greater than 4

Query:

```
select count(rev_stars) from rating  
where rev_stars > '4'
```

The screenshot shows a database interface with a query editor and a results pane. The query is:

```
251  
252 select count(rev_stars) from rating  
253 where rev_stars > '4'  
254
```

The results table has a single column 'count'. The data is:

count
24

Notifications: No data found.

Explain: Use Explain/Explain analyze button to generate the plan for a query.

Tuple : 1

IT214 S7_G9

39) Trigger function to check the validity of show.

Query:

```
CREATE OR REPLACE FUNCTION "Final".check_show_id()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    VOLATILE
    COST 100
AS $BODY$
BEGIN

IF (new.show_id not in ( select show_id from Final.customer))
THEN RAISE NOTICE 'Show does not exists';
    RETURN NULL;
ELSE
RETURN NEW;
END IF;
END
$BODY$;
```

40) Trigger function to check the availability of customer and mobile number.

Query:

```
CREATE OR REPLACE FUNCTION "Final".check_mobile()
RETURNS trigger
LANGUAGE 'plpgsql'
VOLATILE
COST 100
AS $BODY$
BEGIN

IF (new.mobile>=999999999 AND new.mobile<=10000000000) THEN
IF (new.user_id in (select user_id from
Final.customer)) THEN
RETURN new;
ELSE
RAISE NOTICE 'Customer does not exists';
RETURN NULL;
END IF;
ELSE
RAISE NOTICE 'Please enter 10 digit mobile number';
RETURN NULL;
END IF;
END
$BODY$;
```

Section7: Project Code with output screenshots.

Front-end Development:

Code for view_tables:

```
import psycopg2

#connect to the db
con = psycopg2.connect(
    host = "localhost",
    database="Final_P",
    user = "postgres",
    password = "admin")

#cursor
cur = con.cursor()

#genre table

cur.execute("select gen_id, gen_type, m_id from final.genre")

rows = cur.fetchall()

for r in rows:
    print (f"gen_id {r[0]} gen_type {r[1]} m_id{r[2]} ")

#Payment table

cur.execute("select pay_id, user_id, pay_description, price, pay_date from
final.payment")

rows = cur.fetchall()

for r in rows:
    print (f"pay_id {r[0]} user_id {r[1]} pay_description{r[2]} price{r[3]}
pay_date{r[4]}")

#commit the transaction
con.commit()

#close the cursor
cur.close()
```

```
#close the connection
con.close()

File Edit Selection View Go Run Terminal Help
EXPLORER Testpy ...
PY EXTRA tempCodeRunnerFile.py Testpy
1 import psycopg2
2
3 #connect to the db
4 con = psycopg2.connect(
5     host = "localhost",
6     database="final_P",
7     user = "postgres",
8     password = "admin")
9
10 #cursor
11 cur = con.cursor()
12
13 # cur.execute("insert into final.seats (seat_id, seat_type) values (%s, %s)", (99, "asdf"))
14
15 #execute query
16 cur.execute("select gen_id, gen_type, m_id from final.genre")
17
18 rows = cur.fetchall()
19

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
gen_id 70 gen_type Comedy m_id719
gen_id 71 gen_type Adventure|Animation|Children|Comedy|Fantasy m_id720
gen_id 72 gen_type Horror|Drama|Thriller|Fantasy|Horror|Romance|Sci-Fi m_id721
gen_id 73 gen_type Horror|Sci-Fi m_id722
gen_id 74 gen_type Comedy m_id723
gen_id 75 gen_type Comedy|Drama m_id724
gen_id 76 gen_type Children|Drama m_id725
gen_id 77 gen_type Drama|Mystery|Thriller m_id726
gen_id 78 gen_type Horror|Sci-Fi|Thriller m_id727
gen_id 79 gen_type Action|Adventure|Animation m_id728
gen_id 80 gen_type Comedy|Drama m_id729
gen_id 81 gen_type Drama|Mystery|m_id730
gen_id 82 gen_type Crime|Drama|m_id731
gen_id 83 gen_type Action|Animation|Comedy m_id732
gen_id 84 gen_type Adventure|Animation|Children|Drama m_id733
gen_id 85 gen_type Action|Crime m_id734
gen_id 86 gen_type Comedy m_id735
gen_id 87 gen_type Documentary m_id736
gen_id 88 gen_type Drama m_id737
PS F:\Visual Studio\VS Files\Py Extra> |
```

In 21, Col 57 Spaces:4 UTF-8 CRLF Python

The screenshot shows the pgAdmin 4 interface. On the left is the Browser tree, which includes 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', 'Schemas (2)', 'final' (with 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Procedures', 'Sequences', 'Tables (10)', 'customer', 'discout', 'genre', 'movie', 'payment', 'rating', 'seats', 'show', 'theatre', 'tickets', 'Trigge Functions', 'Types', 'public', 'Subscriptions', and 'Final.db'), and 'Final_P'. The 'final.genre' table is selected. The main area shows the Query Editor with the following SQL:

```
1 SELECT * FROM final.genre
2 ORDER BY gen_id ASC
```

The Messages panel indicates: "Successfully run. Total query runtime: 394 msec, 88 rows affected."

The Data Output panel displays the results of the query:

gen_id	gen_type	m_id
82	character varying	731
83	Comedy Drama	732
84	Action Animation Comedy	733
85	Adventure Animation Children Drama	734
86	Action Crime	735
87	Horror Drama Thriller	736
88	Drama	737

The Explain panel contains the note: "Use Explain/Explain analyze button to generate the plan for a query."

The screenshot shows a Visual Studio Code window with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** Shows a file named "tempCodeRunnerFile.py" under "PY EXTRA".
- Code Editor:** Contains Python code for a payment table, including database connection, cursor handling, and commit operations.
- Terminal:** Displays the output of the executed Python script, listing numerous rows of payment data with columns: pay_id, user_id, pay_description, price, and pay_date.
- Status Bar:** Shows "Python 3.10.0 64-bit" and "Ln 11, Col 19 Spaces: 4 UTF-8 CRLF Python".

The screenshot shows a pgAdmin 4 window with the following details:

- File Bar:** File, Object, Tools, Help.
- Browser:** Shows the database structure with "Tables (10)" expanded, including "customer", "discount", "genre", "movie", "rating", "seats", "show", "theatre", and "tickets".
- Query Editor:** Contains a simple SQL query to select all from the payment table and order by pay_id ASC.
- Messages:** Displays a message indicating the query was successfully run with a runtime of 126 msec and 88 rows affected.
- Data Output:** Shows the results of the query as a table with columns: pay_id, user_id, pay_description, price, and pay_date.
- Explain:** A tooltip provides information on using the Explain/Explain analyze button to generate a query plan.
- Notifications:** Shows a table with columns: Recorded time, Event, Process ID, and Payload, with a note "No data found".

Code for insert data:

```
import psycopg2

#connect to the db
con = psycopg2.connect(
    host = "localhost",
    database="Final_P",
    user = "postgres",
    password = "admin")

#cursor
cur = con.cursor()

cur.execute("insert into final.seats (seat_id, seat_type) values (%s, %s)",
(99, "platinum") )

cur.execute("insert into final.customer (user_id, mobile, city, name, show_id)
values (%s, %s, %s, %s, %s)", (99, "585462495", 'mumbai', 'jett', 199 ) )

#commit the transaction
con.commit()

#close the cursor
cur.close()

#close the connection
con.close()
```

pgAdmin 4

File Object Tools Help

Browser

- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (2)
 - > final
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Procedures
 - > Sequences
 - > Tables (10)
 - > customer
 - > diecoute
 - > genre
 - > movie
 - > payment
 - > rating
 - > seats
 - > show
 - > theatre
 - > tickets
 - > Trigger Functions
 - > Types
 - > Views
 - > public
 - > Subscriptions
- > Final db

Dashboard Properties SQL Statistics Dependencies Dependents final.seats/Final... final.seats/Final... final.seats/Final... Final_P/postgr... Fir < > x

Query Editor Query History

```
1 SELECT * FROM final.customer
2 ORDER BY user_id ASC
```

Messages

Successfully run. Total query runtime: 119 msec.
89 rows affected.

Data Output

user_id	mobile	city	name	show_id
83	bright	character varying	Stepnia	192
84	84	4541584468	Buenavista	193
85	85	8314975192	Uryupinsk	194
86	86	5126832926	Romblon	195
87	87	6615211934	Yamblo	196
88	88	7399643812	Gustire	197
89	99	585462495	Jett	199

Explain

Use Explain/Explain analyze button to generate the plan for a query.

Notifications

Recorded time	Event	Process ID	Payload
No data found			

pgAdmin 4

File Object Tools Help

Browser

- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (2)
 - > final
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Procedures
 - > Sequences
 - > Tables (10)
 - > customer
 - > diecoute
 - > genre
 - > movie
 - > payment
 - > rating
 - > seats
 - > show
 - > theatre
 - > tickets
 - > Trigger Functions
 - > Types
 - > Views
 - > public
 - > Subscriptions
- > Final db

Dashboard Properties SQL Statistics Dependencies Dependents final.seats/Final... final.seats/Final... final.seats/Final... Final_P/postgres@PostgreSQL 13

Query Editor Query History

```
1 SELECT * FROM final.seats
2 ORDER BY seat_id ASC
```

Messages

Successfully run. Total query runtime: 136 msec.
89 rows affected.

Data Output

seat_id	seat_type
79	gold
80	gold
81	gold
82	gold
83	gold
84	gold
85	gold
86	gold
87	gold
88	gold
89	platinum

Explain

Use Explain/Explain analyze button to generate the plan for a query.

Notifications

Recorded time	Event	Process ID	Payload
No data found			

Code for delete data:

```
import psycopg2

#connect to the db
con = psycopg2.connect(
    host = "localhost",
    database="Final_P",
    user = "postgres",
    password = "admin")

#cursor
cur = con.cursor()

data = input('seat_id')
cur.execute("DELETE FROM \"final\".\"seats\" WHERE
\"seat_id\\"='{}';".format(data))

data = input('user_id')
cur.execute("DELETE FROM \"final\".\"customer\" WHERE
\"user_id\\"='{}';".format(data))

#commit the transaction
con.commit()

#close the cursor
cur.close()

#close the connection
con.close()
```

The screenshot shows a Visual Studio Code window with a dark theme. The main editor pane contains Python code for connecting to a PostgreSQL database and inserting data into tables named 'seats' and 'customer'. The code uses the `psycopg2` library. Below the editor are tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab shows command-line history for running the script. A sidebar on the right lists several PowerShell windows. The status bar at the bottom indicates the file is 16 lines long, has 2 spaces, is in UTF-8 encoding, and is using Python.

```

File Edit Selection View Go Run Terminal Help
File View,Tablepy Deletepy Insertpy x
Insertpy > ...
1 import psycopg2
2
3 #connect to the db
4 con = psycopg2.connect(
5     host = "localhost",
6     database="Final_P",
7     user = "postgres",
8     password = "admin")
9
10 #cursor
11 cur = con.cursor()
12
13
14 cur.execute("insert into final.seats (seat_id, seat_type) values (%s, %s)", (99, "platinum"))
15
16 #cur.execute("insert into final.customer (user_id, mobile, city, name, show_id) values (%s, %s, %s, %s)", (99, "585462495", 'mumbai', 'jett', 199))
17
18
19 #commit the transaction
20 con.commit()
21
22 #close the cursor
23 cur.close()
24
25 #close the connection

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS F:\Visual Studio\VS Files\Py Extra> python -u "f:\Visual Studio\VS Files\Py Extra\Insert.py"
PS F:\Visual Studio\VS Files\Py Extra> python -u "f:\Visual Studio\VS Files\Py Extra\Insert.py"
PS F:\Visual Studio\VS Files\Py Extra> []

Python 3.10.0 64-bit ④ 0 ⚡ 0

Ln 16, Col 2 Spaces: 4 UTF-8 CRLF Python ↻

The screenshot shows the pgAdmin 4 interface. The left sidebar is the Browser panel, listing various database objects like extensions, schemas, and tables. The central area is the Query Editor, which runs a query to select all from the 'final.seats' table and orders by 'seat_id' ascending. The Data Output panel displays the results, showing 88 rows where 'seat_id' ranges from 82 to 89 and 'seat_type' is consistently 'gold'. The Messages panel shows a success message: "Successfully run. Total query runtime: 129 msec. 88 rows affected." A tooltip in the Explain panel suggests using the Explain/Explain analyze button to generate a query plan.

pgAdmin 4

File Object Tools Help

Browser

- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- < Schemas (2)
 - < final
 - < Final_P
- > Sequences
- < Tables (10)
 - < customer
 - < discount
 - < genre
 - < movie
 - < payment
 - < rating
 - < seats
 - < show
 - < theatre
 - < tickets
- > Trigger Functions
- > Types
- > Views
- > public
- > Subscriptions

final.seats/Final_P/postgres@PostgreSQL 13

Query Editor Query History

```

1 SELECT * FROM final.seats
2 ORDER BY seat_id ASC

```

Messages

Successfully run. Total query runtime: 129 msec.
88 rows affected.

Data Output

seat_id	seat_type
82	gold
83	gold
84	gold
85	gold
86	gold
87	gold
88	gold

Notifications

Recorded time	Event	Process ID	Payload
No data found			

Explain

Use Explain/Explain analyze button to generate the plan for a query.

✓ Successfully run. Total query runtime: 129 msec. 88 rows affected.

The screenshot shows the Visual Studio Code interface with a dark theme. The main editor window contains Python code for connecting to a PostgreSQL database and performing a delete operation. The terminal below shows the command to run the script and the resulting output.

```

File Edit Selection View Go Run Terminal Help
ViewTablepy Deletepy Insertpy
Deletepy > ...
1 import psycopg2
2
3 #connect to the db
4 con = psycopg2.connect(
5     host = "localhost",
6     database="Final_P",
7     user = "postgres",
8     password = "admin")
9
10 #cursor
11 cur = con.cursor()
12
13
14 data = input('seat_id')
15 cur.execute("DELETE FROM \"final\".\"seats\" WHERE \"seat_id\"='{}';".format(data))
16
17 data = input('user_id')
18 cur.execute("DELETE FROM \"final\".\"customer\" WHERE \"user_id\"='{}';".format(data))
19
20 #commit the transaction
21 con.commit()
22
23 #close the cursor
24 cur.close()
25

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

PS F:\Visual Studio\VS Files\Py Extra> python -u "f:\Visual Studio\VS Files\Py Extra\Insert.py"
PS F:\Visual Studio\VS Files\Py Extra> python -u "f:\Visual Studio\VS Files\Py Extra\Insert.py"
PS F:\Visual Studio\VS Files\Py Extra> python -u "f:\Visual Studio\VS Files\Py Extra\Delete.py"
seat_id99
user_id99
PS F:\Visual Studio\VS Files\Py Extra>

```

Python 3.10.0 64-bit ② 0 ③ 0

The screenshot shows the pgAdmin 4 interface connected to a PostgreSQL database named 'Final_P'. The left sidebar shows the schema structure with various objects like extensions, languages, publications, schemas, and tables. The central area shows a query editor with the following SQL code:

```

1 SELECT * FROM final.customer
2 ORDER BY user_id ASC

```

The 'Messages' panel indicates the query was successfully run with a runtime of 116 msec and 88 rows affected. The 'Data Output' panel displays the results of the query, which are as follows:

user_id	mobile	city	name	show_id
82	82	6939297186	character varyng	Jamil
83	83	9965225590	Santo Tom??	Stepha
84	84	4541584468	Buenavista	Umeko
85	85	8314975192	Uryupinsk	Comellia
86	86	5126832926	Romblon	Rona
87	87	6615211934	Yamblo	Eduardo
88	88	7399643812	Guatire	Ester

The 'Notifications' and 'Recorded time' sections are empty. The 'Explain' button is visible in the bottom right corner of the message panel.