IT214

Dhirubhai Ambani
Institute of Information and Communication Technology

Team ID: S7 T8

Entertainment Event booking System

Instructor: Prof. Minal Bhise
Mentor Name: Manoj Sir

ID	Name
201901004	Axit Dhola
201901031	Sanny Dhameliya
201901049	Bhadrayu Bhalodia
201901066	Kenil Bhingradiya

Table of Contents

Section 1: SRS

INDEX

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

Database Name: - BOOK_SYS

1.1 Purpose: -

The primary purpose of this document is to manage a database of multiple events simultaneously, and in one place, so it is easy to track and manage for customers and managers in real-time.

There are many benefits of using booking systems for events, and online bookings are the future. Let the event booking system do all of the hard work.

The software will automatically update in real-time so that you can keep track of the number of presents and upcoming events, tickets that you have sold, and how many available slots are left; selection of tickets and payment can be done with ease.

In addition, developers can use this doc as a source and guide.

1.2 Intended Audience and Reading Suggestions:-

The main intention of this document is,

- **Database owner (Administrator):** They are the admin of the database, and they can add/remove events, managers, users.

- **Event managers:** These will be the primary users. They can add/remove customers, update any information of the customers, etc. In short, they are the manager of the database.

- **Event organizers:** They are the ones who organize new events or can add/edit old events. They can add or update event information in the database.
- **Customer:** They can only view information related to events, and if interested, they can book event tickets.

1.3 Product Scope:-

The database related product is helpful for,

- Provide facility for booking of all type of entertainment events
- Inquiry facility
- Provide instant ticket booking
- Manage user and maintain the database for all booking and transaction
- Reduction of traffic at offline booking centers
- Booking many tickets at the same time by different people

1.4 Description :

- The main motto of this database is to register different types of events and organize them easily using this platform. Also, for other users to search various events from one platform and book if they are interested. This database is also for managers to manage the registration process to manage and access easily.
- This platform brings event owners and customers nearer and able to contact them directly.
- Design must be able to handle large data uploading loads as well as keep data updating in real-time, so that will be very useful for users.
- In order to update the database information, which comes from sources like events organizers, movie theaters, which access the database and keep updating the number of users, seat availability.
- This platform is user-friendly, easy to learn, and reliable, and it also works on any multi-platforms, so this will work on all standard platforms like windows, android, iOS, Linux.
- The users can easily find available seats and reserve them. They can pay through various payment systems. Users can also cancel their bookings and also get a refund if canceled at the proper time.
- Using this database, event organizers easily find information about total confirmed tickets and details about all users who are willing to come to their events.
- Our database integrity is very powerful, so without any user concern, we will not store any private or financial data and not share it with others like the government or any other platforms.
- This module allows customers to make payments after the reservation. The customer will receive a ticket receipt by mail after the payment.
  They Make payment to selected service, Generate billing receipt and Update credit card details and also View payment details when he/she wants.
- With the help of these databases, for each customer, we can get to know about positive and negative points with the help of personalized feedback reports.
- Event organizers can host various events on platforms and they can easily find information about customers, how many seats are remaining.
- This module allows the administrator of this site to manage the different aspects of ticket booking-related activities. The administrator of the site can add or modify the records of concerts, movies, drams, plays, etc. they should be added, edit, view concerts records, movie details, and also View bookings, View cancellation reports, View payment reports, Add users and Change passwords.
- This module is for customers where customers can access the account module after login id and password. This module will display booked tickets, Billing receipts, ticket cancellation page, logout, etc. Users can Display booked receipts, Change passwords, and update profile information.

1.4.1 Product Perspective:-

- Product Perspective BOOK_SYS Online entertainment-related booking Reservation System is aimed to provide a Simple, Quick, and efficient Way to Book an entertainment-related Ticket.
- This System Should be User friendly, smooth, Fast, secure, reliable, and hassle-free to provide the user a pleasant Experience by Booking a Ticket.
- BOOK_SYS is intended to be a Stand-alone Online Portal and should be independent of any Other Software. It should be compatible With any Browser and Should be Platform Independent. SRS for BOOK_SYS System.

1.4.2 Product Functions:-

1. User Management
    - New User Registration
    - Existing User Login/Logout
    - Forgot the Password Recovery mechanism
        - Recovery Via Phone
        - Recovery Via Mail.

2. Enquiries
    - Seat Availability
    - Current event Information
    - Ticket Related Enquiries

3. Booking Facilities.
    - Booking a Ticket.
        - Booking a general Ticket.
        - Booking History.
        - Ticket Reprint Facility.
        - Booking Status/Current Status

    - Providing customer Details.
        - Personal Details.
        - ID details.
        - Type of seats

4. Payment Options.
    - Credit Cards.
    - Debit Cards.
    - Mobile/Net Banking.
    - Cash.

5. Cancellation and Refund Facilities.
    - Cancel a ticket.
    - Print Cancelled Ticket Facility.
    - Cancellation Status.
    - Canceled Ticket History.
    - Refunds Status.
    - Refunds History.

6. Event types
    - Event name
    - Event total cost
    - Event ticket price and availability
    - Event Date

1.4.3 USER CHARACTERISTICS

- The User Shall be familiar With the Procedure and Provisions to Book a ticket.
- The User shall have an average knowledge of Computer and Internet.
- The User should know the methods of transactions and Payment.

[B]. Document the Requirements Collection/ Fact-Finding Phase

2.1 Background readings

1. In this section, we collected data from various event booking sites and articles like
    a. www.bookmyshow.com
    b. https://pdfcoffee.com/srs-on-online-booking-system-pdf-free.html
    c. www.zoominfo.com
    d. https://www.accelevents.com/platform/
    e. https://play.google.com/store/apps/details?id=com.bt.bms (ratings and reviews)
2. We also collected customer reviews from different booking sites and also generated a google form.
3. For changes in anything then the website needs accurate and real-time data for Example
    - Available number of events
    - Number of customers region-wise.
    - Customers are fully vaccinated or not.
    - Food availability region-wise.
    - An available number of employees.

4. Some tips to improve situations
    - India's population is very high so Needs some fast servers that can handle many requests at a time.
    - There has to be total data of events with no chances of missing any kind of event-related data. It is very important for customer satisfaction.
    - Lack of understanding between customer and event organizer.
    - Batter communication between customer and help-center or organizer.

2.2 Interviews: -

Interview 1

- Role Play interview plan
- System: Entertainment booking System
- Interviewee: Dhruvil Gorasiya(Role Play)
    - Designation: Administrator of Event
- Interviewer:
    - Axit Dhola (Student of DAIICT)
    - Sanny Dhameliya (Student of DAIICT)
    - Bhadrayu Bhalodia (Student of DAIICT)
    - Kenil Bhingradiya (Student of DAIICT)

- Date: 04/10/2021
- Time: 16:30
- Duration: 30 minutes
- Place: Google Meet
- Purpose of Interview:
  - Preliminary meeting to identify problems and requirements regarding decisions of the Manager about Event.
- Agenda:
  - Problems with the lack of Management
  - Current Situation about booking System
  - Initial ideas
  - Follow-up actions

Interview Summary 1

- System: Entertainment booking System
- Interviewee: Dhruvil Gorasiya (Role Play)
  - Designation: Administrator of Event
- Interviewer:
  - Axit Dhola (Student of DAIICT)
  - Sanny Dhameliya (Student of DAIICT)
  - Bhadrayu Bhalodia (Student of DAIICT)
  - Kenil Bhingradiya (Student of DAIICT)
- Date: 04/10/2021
- Time: 16:30
- Duration: 30 minutes
- Place: Google Meet
- Summary of Interview
  - Preliminary meeting to identify problems and requirements regarding decisions of the Manager about Booking Management.
  - Also, users watch trailers of currently showing and upcoming movies, read reviews, synopsis, the cast of currently showing and forthcoming movie/ theatre/ concerts, and view and provide theatre/ venue review, user ratings, etc...
  - In the time of Covid-19, all things move on to online platforms and all the things like education, work, and all this stuff convert into online mode. Also, people promoted online work without the crowd of people.
  - The problem of lack of understanding between event managers and customers which now is solved by enjoying their event without buying tickets to spend several hours in a queue of people.

Interview 2

- Role Play interview plan
- System: Entertainment booking System
- Interviewee: Yash Sakaria (Role Play)
  - Designation: Event Organizer
- Interviewer:
  - Axit Dhola (Student of DAIICT)
  - Sanny Dhameliya (Student of DAIICT)

        - Bhadrayu Bhalodia (Student of DAIICT)
        - Kenil Bhingradiya (Student of DAIICT)
    - Date: 05/10/2021
    - Time: 17:30
    - Duration: 30 minutes
    - Place: Google Meet
    - Purpose of Interview:
        - Preliminary meeting to identify problems and requirements regarding decisions of the Manager about Event.
    - Agenda:
        - Problems with the lack of Management
        - Current Situation about booking System
        - Initial ideas
        - Follow-up actions

Interview Summary 2

- System: Entertainment booking System
- Interviewee: Yash Sakaria (Role Play)
    - Designation: Event organizer
- Interviewer:
    - Axit Dhola (Student of DAIICT)
    - Sanny Dhameliya (Student of DAIICT)
    - Bhadrayu Bhalodia (Student of DAIICT)
    - Kenil Bhingradiya (Student of DAIICT)
- Date: 05/10/2021
- Time: 17:30
- Duration: 30 minutes
- Place: Google Meet
- Summary of Interview
    - Preliminary meeting to identify problems and requirements regarding decisions of the Manager about Booking Management.
    - The users should be able to select movies/concerts/play at its nearby place so that they can select appropriate theatre and can also select show time, select ticket price, and set and carry out the efficient purchase of a ticket.
    - Make sure that the customer always gets the acknowledgment about his/her ticket for this event successfully booked.
    - Event organizer provides offers and rewards to the customers i.e. send alerts to the customers at an instance of time. So, customers can use the offers when they wish to do so.

Interview 3

- Role Play interview plan
- System: Entertainment booking System
- Interviewee:Vishabh maniya(Role Play)
    - Designation: Customer
- Interviewer:
    - Axit Dhola (Student of DAIICT)

        - Sanny Dhameliya (Student of DAIICT)
        - Bhadrayu Bhalodia (Student of DAIICT)
        - Kenil Bhingradiya (Student of DAIICT)
    - Date: 06/10/2021
    - Time: 18:30
    - Duration: 30 minutes
    - Place: Google Meet
    - Purpose of Interview:
        - Preliminary meeting to identify problems and requirements regarding decisions of the Manager about Event.
    - Agenda:
        - Problems with the lack of Management
        - Current Situation about booking System
        - Initial ideas
        - Follow-up actions

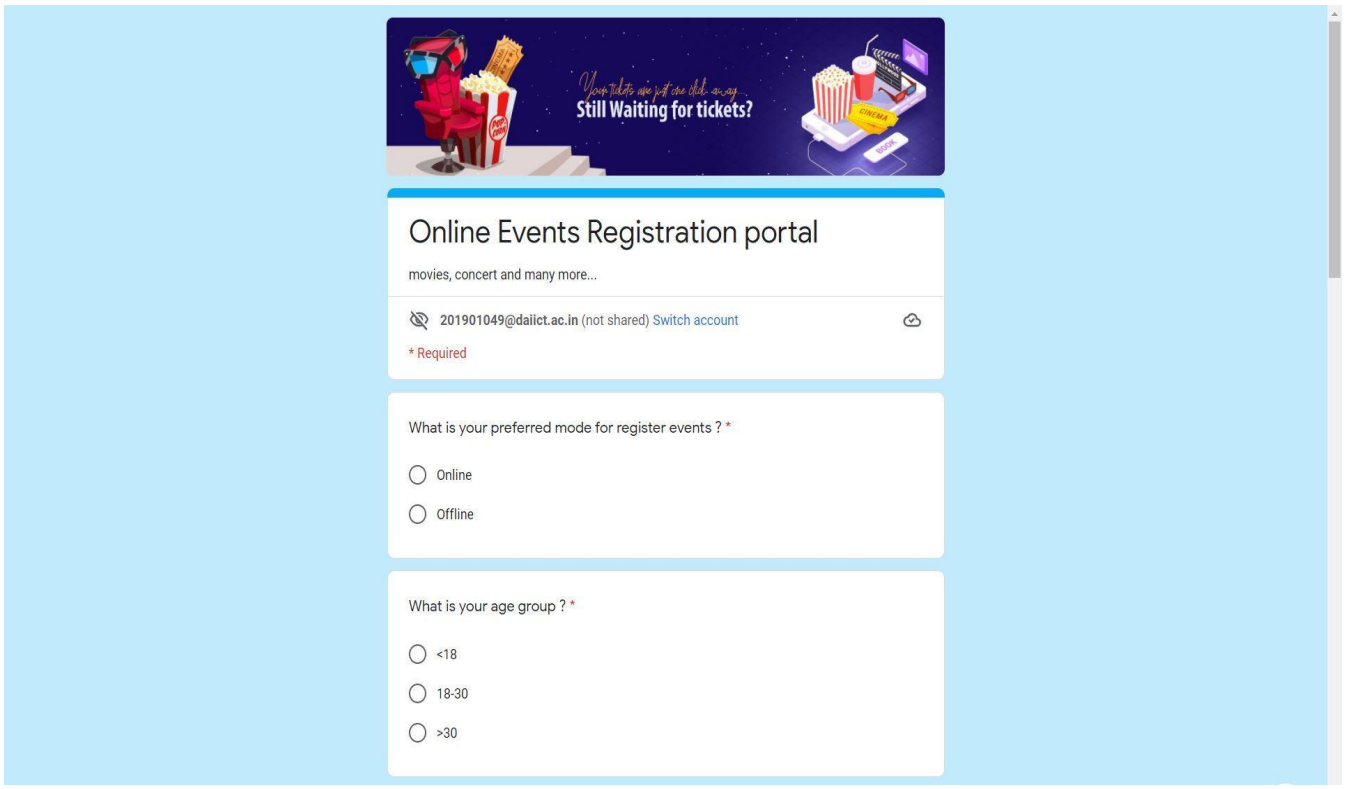
Interview Summary 3

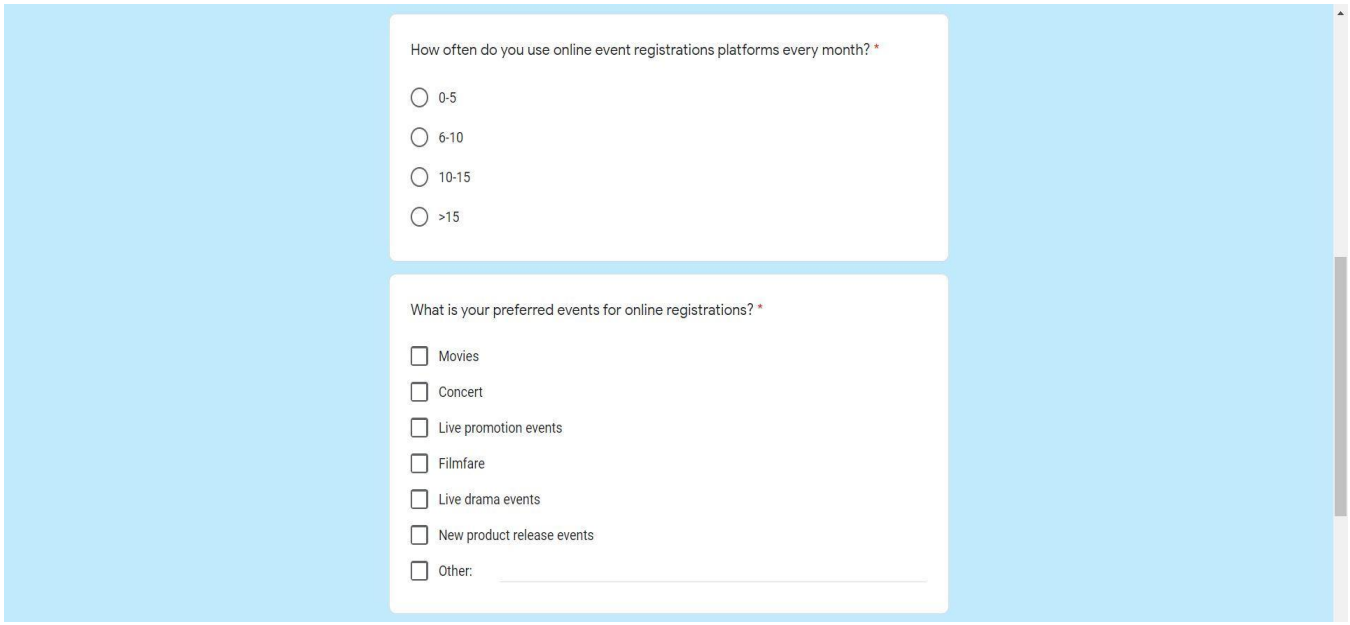
- System: Entertainment booking System
- Interviewee: Vishabh Maniya(Role Play)
    - Designation: Customer
- Interviewer:
    - Axit Dhola (Student of DAIICT)
    - Sanny Dhameliya (Student of DAIICT)
    - Bhadrayu Bhalodia (Student of DAIICT)
    - Kenil Bhingradiya (Student of DAIICT)
- Date: 06/10/2021
- Time: 18:30
- Duration: 30 minutes
- Place: Google Meet
- Summary of Interview
    - Additionally, users should be able to buy snacks at the desired time and receive alerts for forthcoming movies/concerts/theatre shows.
    - Also, users watch trailers of currently showing and upcoming movies, Read reviews, synopsis, the cast of currently showing and forthcoming movie/ theatre/ concerts, and View and provide theatre/ venue review, user ratings, etc...
    - Allow users to sort movies/ plays according to their genre and also provide movie/ play/ concert reviews, user ratings. See theatres/ venues on a map and get directions from the current location.
    - Further discussions are needed when more information is available.

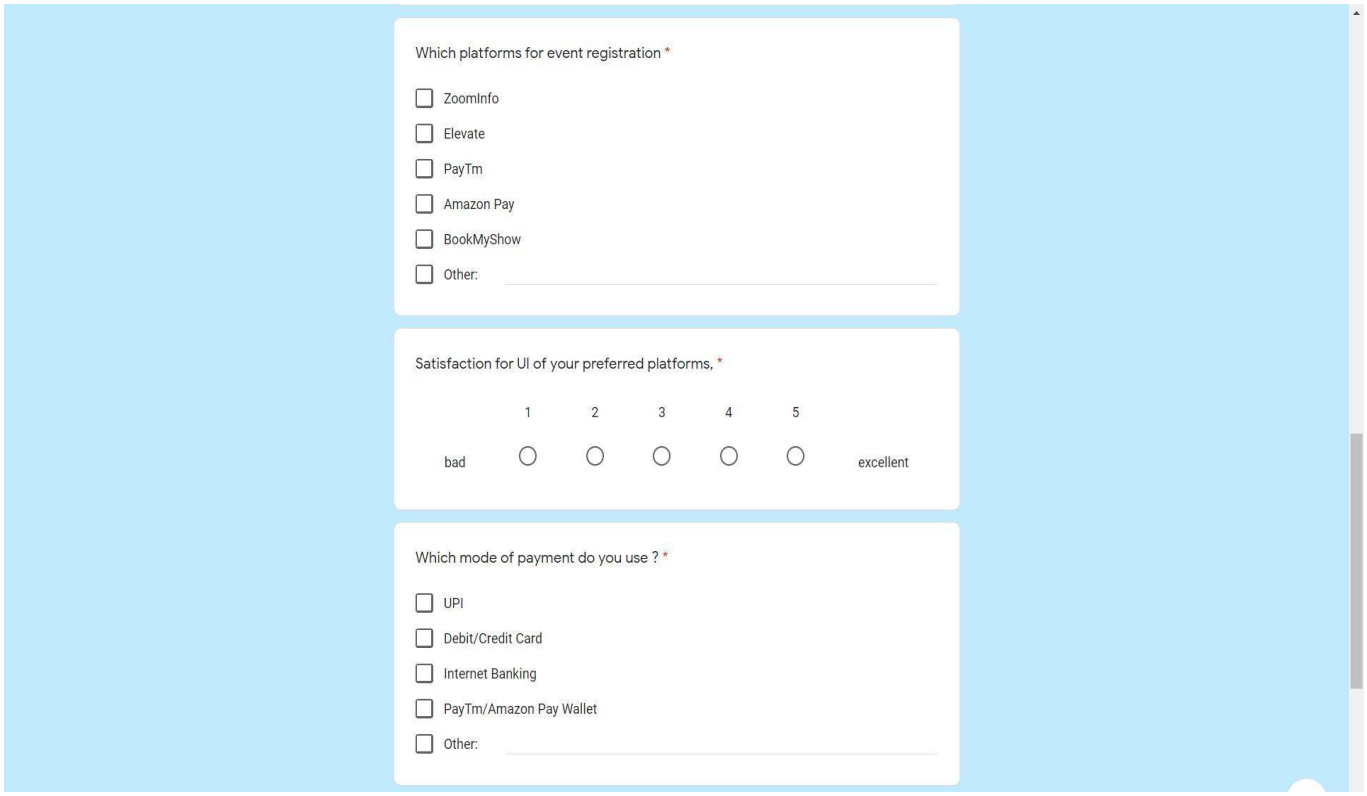
2.2.1 Combined Requirements gathered from the Interviews

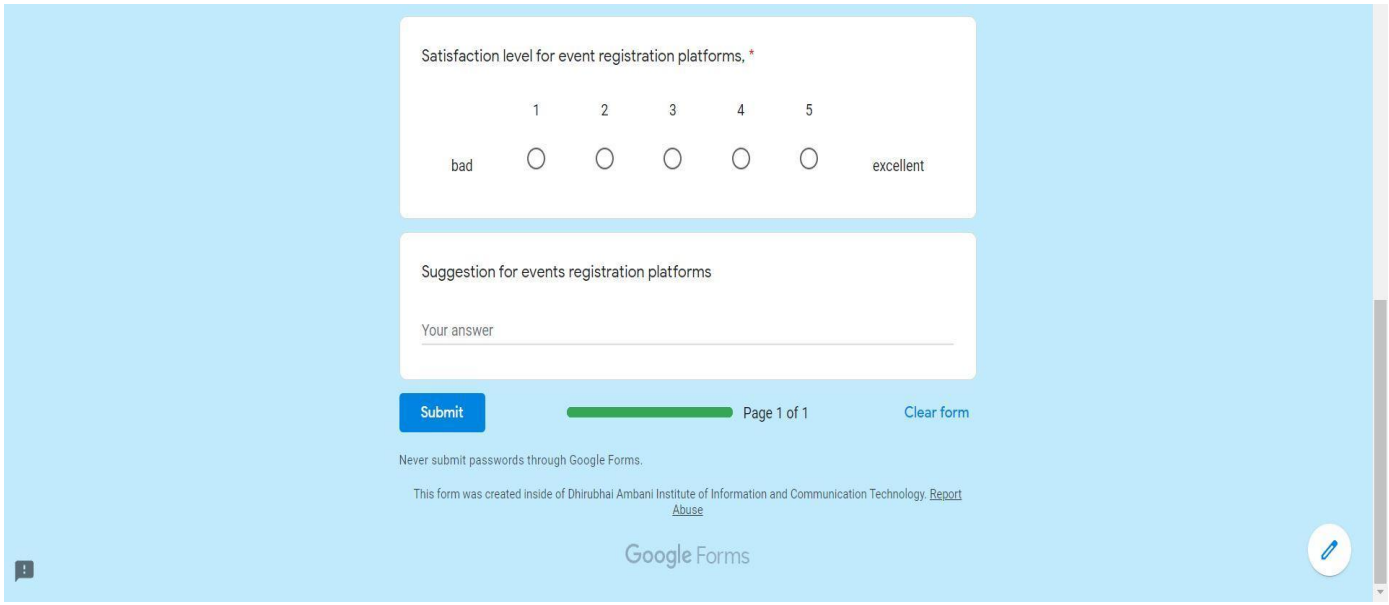
- As per all interviews, there has to be a good UI for the user interface with the platform.
- There is good privacy practice and robust data management and security.
- There should be multiple platform support so users can have accessibility from all types of devices and also be accessible on low-speed internet.
- There should be enough cloud storage and continuous flow to data read and write in the database.
- There should be real-time updates for the database without fail.
- There should be good communication between the customer and platform administrator for any interface difficulties(bugs).

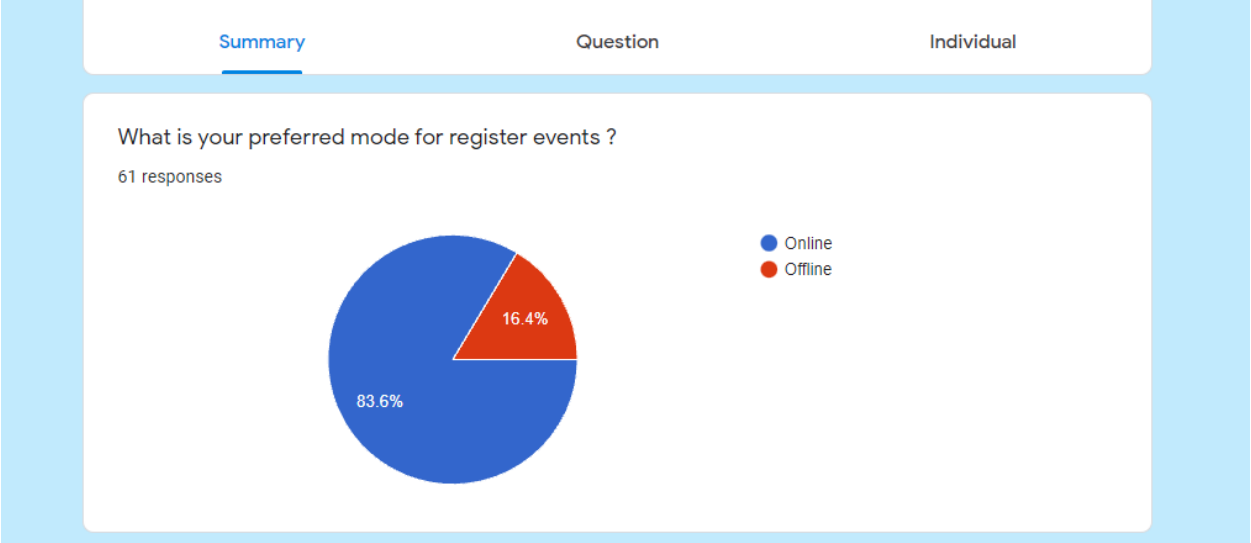
2.3 Questionnaires:-



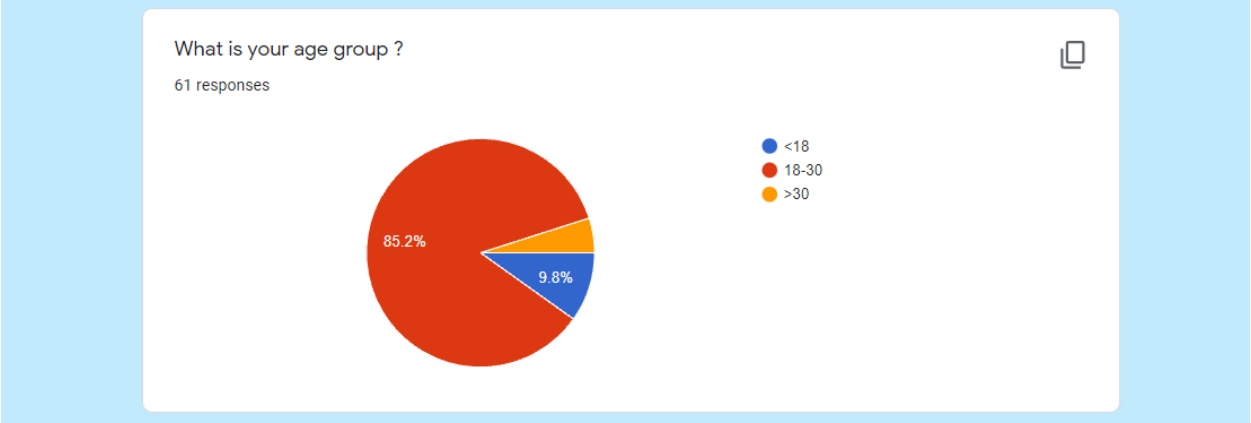



Which platforms for event registration *
ZoomInfo
Elevate
PayTm
Amazon Pay
BookMyShow
Other:
Satisfaction for UI of your preferred platforms, *
1 2 3 4 5
bad
excellent
Which mode of payment do you use ? *
UPI
Debit/Credit Card
Internet Banking
PayTm/Amazon Pay Wallet
Other:

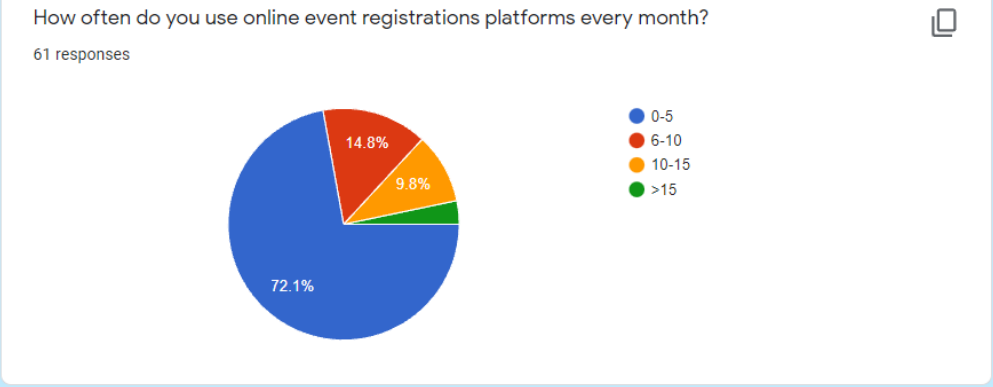

Satisfaction level for event registration platforms, *
1 2 3 4 5
bad
excellent
Suggestion for events registration platforms
Your answer
Submit
Page 1 of 1
Clear form
Never submit passwords through Google Forms.
This form was created inside of Dhirubhai Ambani Institute of Information and Communication Technology. Report Abuse
Google Forms



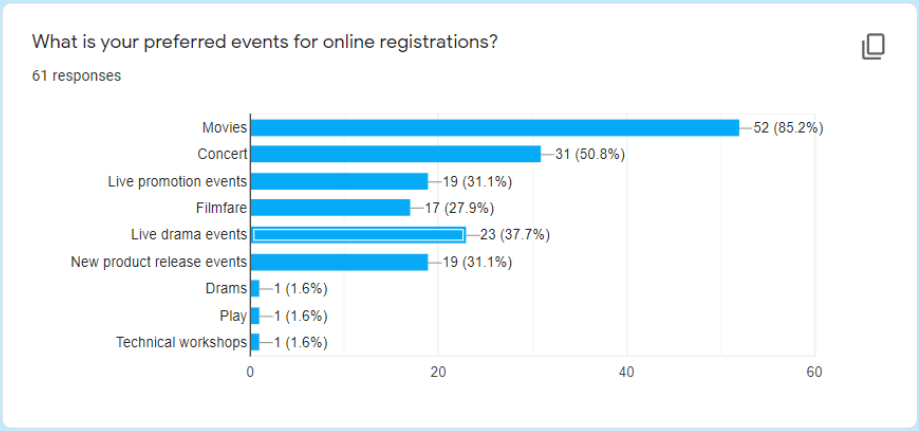
➢ Most customers prefer to book tickets online for different events.



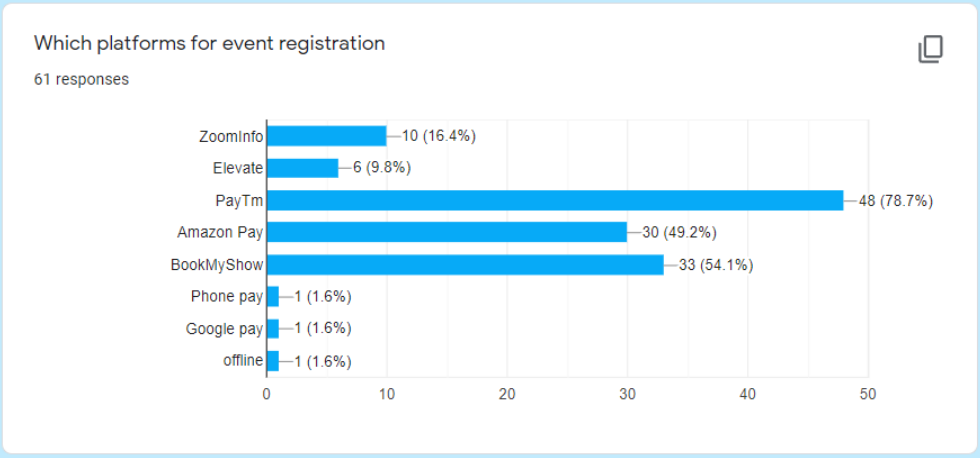
➢ Likely 86% of customers belong to the 18-30 age group.



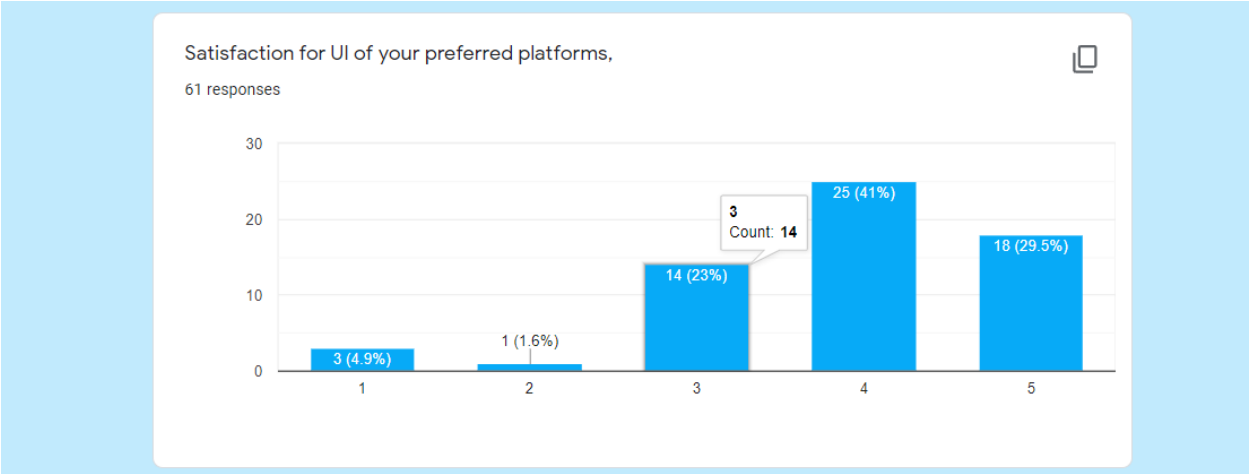
➢ Most of the customers go to around 0-5 events every month.



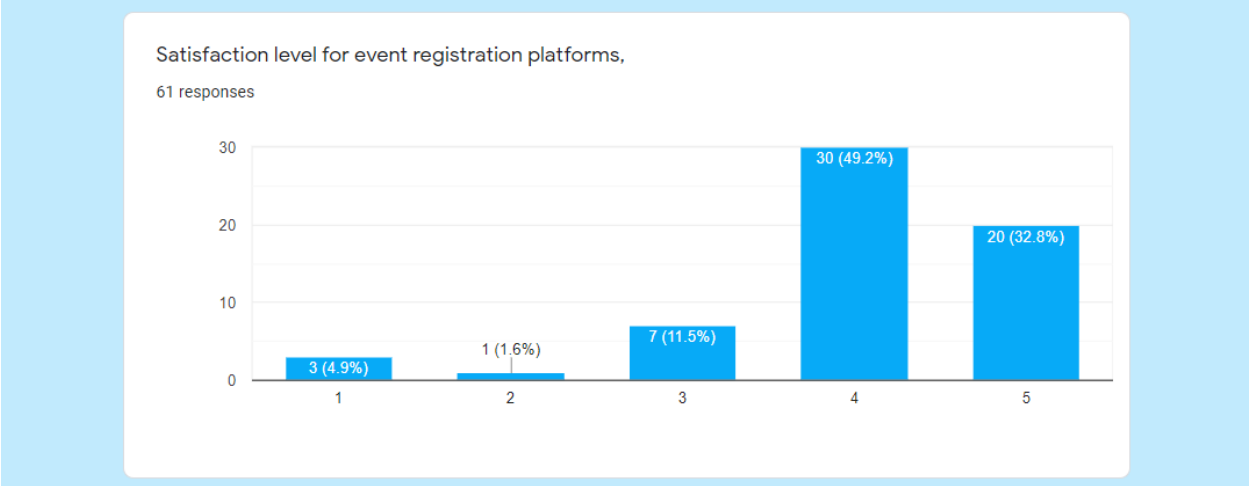
➢ As per the graph, most customers prefer movies and concert booking in online mode.



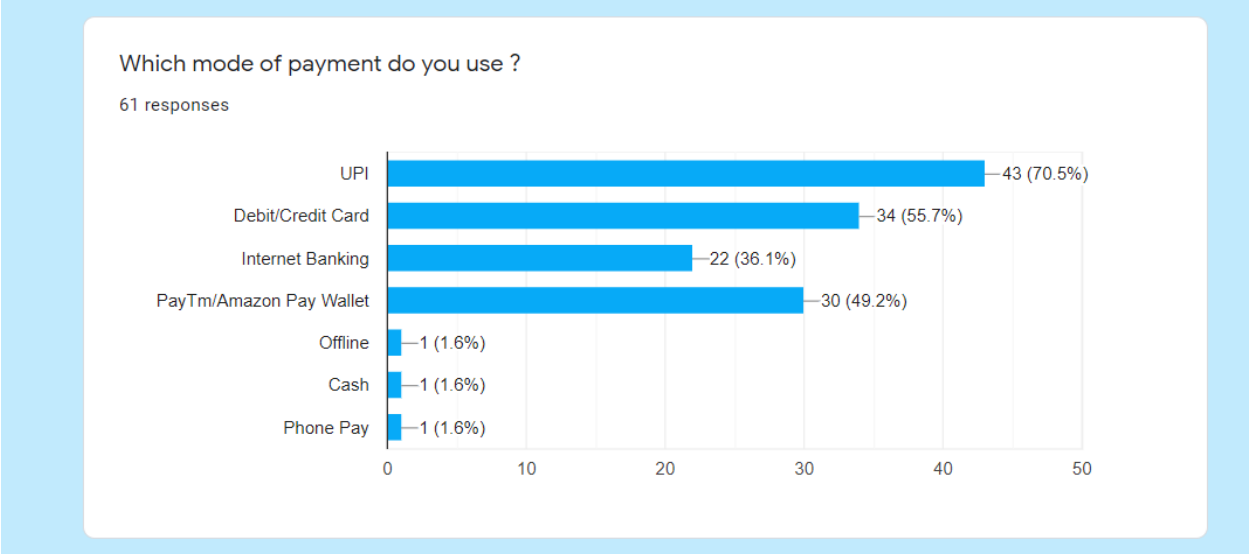
➢ As per the graph, most customers prefer PayTm, BookMyShow, and Amazon pay for booking in online mode.



➢ Platform UI is good as per review for customers.



➢ Most of the customers are satisfied with online bookings.



➢ In online mode, most customers preferred UPI, Debit/credit card, and internet banking for payment.

2.4 Observation:

- System: Entertainment booking System
- Observation by:
  - Axit Dhola (Student of DAIICT)
  - Sanny Dhameliya (Student of DAIICT)
  - Bhadrayu Bhalodia (Student of DAIICT)
  - Kenil Bhingradiya (Student of DAIICT)
- Date: 07/10/2021
- Time: 18:30
- Duration: 30 minutes
- Place: Google Meet

- Real-time update in the database of any events and availability.
- It is easy to learn and user-friendly interface.
- The data privacy policy is robust.
- Sometimes in many platforms unable to log in doesn't get OTP or invalid OTP.
- Sometimes platform does not load.
- In most platforms, customer care service is very bad, and wait for a long time to get a reply or chat/call with a representative.
- Many platforms take feedback from users and do nothing with it to solve their or common issues.
- After reviewing many public reviews of some platforms like BookMyShow user says before book ticket there is a 100% refund policy and after the ticket is booked the ticket says no cancellation is available. So, no transparency in their policy.

[C]. Create Fact-Finding Chart

Objective	Technique	Subjects	Time Commitment
Background knowledge of the platform	Background readings	Few Similar projects and requirements of this platform	1 day
Preliminary meeting to identify problems and requirements regarding decisions	Interview	Administrator of platform	30 minutes
Preliminary requirements of organizers	Interview	Event organizer	30 minutes
Preliminary requirements of users	Interview	customer	30 minutes
To get Customer feedback on the system	Questionnaires	A regular customer at a platform	1 hour
To get knowledge of event booking platform	Observation	Customer	30 minutes
To get knowledge User-friendly UI	Observation	Customer	30 m

[D]. List Requirements

➢ **NON Functional requirements:-**

- **Reliability**
  - The reliability of the overall program depends on the reliability of the individual components. The main pillar of system reliability is the backup of the database which is continually maintained and updated to reflect the most recent changes.
  - Thus the overall stability of the system depends on the stability of the container and its underlying operating system.
- **Availability**
  - The system must always be available, which means that the user can access it using a web browser, limited only by the downtime of the server on which the system is running. In the event of a hardware failure or database corruption, a replacement page will be displayed. Even in the event of a hardware failure or database corruption, database backups should be retrieved from the server and recorded by the administrator. Then the service will be restarted. This means 24X7 availability.
- **Maintainability**

    - A commercial database is used for database maintenance and the application server takes care of the site. In the event of an error, the program will be reset. In addition, the software design is carried out with consideration of modularity so that maintainability can be carried out efficiently.
- **Portability**
    - The application is based on HTML and scripting language. Thus, the end-user part is fully portable and any system using any web browser should be able to use the functionality of the system, including any hardware platform that is available or will be available in the future.
    - An end-user is using this system on any OS; either it is Windows or Linux.
    - The system shall run on PC, Laptops, and PDAs, etc.
- **Security**
    - The system uses SSL (secured socket layer) in all transactions that include any confidential customer information.
    - The system must automatically log out all customers after a period of inactivity.
    - The system should not leave any cookies on the customer's computer containing the user's password.
    - The system's back-end servers shall only be accessible to authenticated administrators.
    - Sensitive data will be encrypted before being sent over insecure connections like the internet.
- **Accessibility**
    - The system will be a web-based application that is going to be accessible on the web browser.
- **Back up**
    - We will take a backup in our system database. Toenable the administrator and the user to access the data from our system!
- **Performance**
    - The product shall be based on the web and has to be run from a web server.
    - The product shall take initial load time depending on internet connection strength which also depends on the media from which the product is run.
    - The performance shall depend upon hardware components of the client/customer
- **Accessibility**
    - The system shall provide handicap access.
    - The system shall provide multi-language support.
- **Supportability**
    - The source code developed for this system shall be maintained in the configuration management tool.

➢ **Functional Requirements: -**

- **Registration**
    - If a customer wants to book the ticket then he/she must be registered, an unregistered user can't book the ticket.
- **Login**
    - Customer logins to the system by entering valid user IDs and passwords for booking the ticket.
- **Search Movie**
    - The system shall have a search function. Customer or visitors can search movies based on movie name, date, time, and venue
- **Seat Viewing**
    - The customer shall be shown a 2D image of the seats from which the desired seats are selected.

- **Ticket canceling**
    - The customer shall be given an option to cancel the ticket one hour before the movie with some fine.
- **Payment**
    - For the customer, many types of secure billing will be prepaid by debit or credit card. The security will be provided by a third party like Pay-Pal etc.
- **Logout**
    - After the payment or browse the movie, the customer will log out.
- **Generate ticket**
    - After booking, the system can generate the portable document file (.pdf) and then send one copy to the customer's Email-address and another one as an SMS to the customer's phone.
- **Add movies**
    - The system shall have a feature for the admin to add movies and their details.
- **Remove movies**
    - The system shall have a feature for the admin to remove movies.

[E]. User Classes and Characteristics

- There are mainly four types of users
    - Event organizers
    - Event managers
    - Customers
    - Administrator
- The first one is event organizers who organize the events. The one who will come to put their event on a portal. They have access to all customers' data who register on this event and financial information.
- The second one is an event manager who manages all details about events like all requirements to show up on events, event details, etc. They have access to edit event details and customer names.
- The third one is customers, who want to participate in particular events. They have access to their profiles and all ongoing and upcoming event details.
- And the last one is the administrator who owns this platform. The administrator has access to the entire database.

[F]. Operating Environment

- Hardware:- i3 Processor or Android Version 5.0 or Higher
- Software:- Android OS or Windows 7 and up
- Connectivity Requirements:- Wi-Fi or Mobile Internet connectivity
- External Interface Requirements :
    - Third-party API:
        - Google's geographical locations API
        - Online bank's API
        - Different event platform API

[G]. Product Functions

- Providing real-time information about the below objects:-
    - Number of total upcoming events and their details
    - Total empty seat for every particular event
    - Food management
    - Your transaction history
    - Your account details
    - Completed events and live events detail
    - Your ticket status
    - Ticket details

[H]. Privileges

Data	**Privileges**
All User Details	Administrator, event organizer
Events details	Customer, Administrator, Manager, Organizer
Food details	Manager, administrator
Booking status(read,update,cancel)	Customer, administrator

[I]. Assumptions

- All required numerical information is in an accurate form.
- All third-party APIs should be working successively.
- Quick payment response from banks.
- Quick update in the database by all third-party providers.
- Each person must have a device with an internet connection.
- Users should have a basic idea about the internet and online payment and also should be familiar with computers.

[J]. Business Constraints

- The platform will be free to use to create new research projects and collaborate. To properly maintain the platform and bear the server costs, users must pay some predefined amount when their research paper gets published. Also, the project comes under an open-source license. So developers are welcomed to be a part of the development team.
- Creation of digital infrastructure, good internet connection, and fast, accessible database server that can handle large amounts of traffic at a time for this platform**.**

Section2: Noun Analysis.

1. Noun Analysis

Noun	**Verb**
Customer	Register
Event Organizer	Add/Remove events
Manager	Add/remove customer
Administrator	Manage platform
Name	search
Address	design
Gender	access
Phone No.	handle
Email ID	time
Age	organize
ID proof	Generate
Events	book
Food	Allow/prevent
Events Name	process
Login	contact
Password	upload
Tickets	update

Price	order
Payment	seats
Billing receipt	reserve
feedback	receive
Event Date	host
Event Time	edit
Refund	pay
Transaction ID	card
Customer ID	credit
Event ID	view
Event type ID	help
Ticket ID	negative
Manager ID	site
Vaccination Status	manage
Seat availability	add
Payment Method	modify
Destination	edit
Venue	change
Description	come

Motto	source
Different types	work
process	detail
owner	allow
Large data	selected
Information	points
Theaters	report
platform	record
Integrity	confirm
Financial data	personalized
Update	related
Credit card details	assign
Feedback reports	authorize
Different expects	
Event records	
View booking	
Cancellation report	
Payment report	
Account module	

Login ID	
Display	
Update profile	
Efficient	
Browser	
Software	
Procedure	
Internet	
resources	

2. Entity- Attributes:-

Candidate entity set	**Candidate attribute Set**	**Candidate relationship set**
customer	1. c_id 2. c_name 3. c_address 4. c_email 5. c_phoneno 6. c_age 7. password	login,add user
event	1. event_id 2. event_type_id 3. event_name 4. event_date 5. event_address 6. event_manager 7. event_start_time 8. event_end_time 9. event_size	Insert events and their details

event organizer	1. eo_id 2. event_id 3. eo_name 4. password	Details of event organizer
manager	1. m_id 2. m_name 3. m_phoneNo	Event managers
event_type	1. event_type_id 2. event_description	Preffed event types
ticket	1. t_id 2. c_id 3. t_price 4. seat_no 5. p_id	Ticket details
payment	1. p_id 2. c_id 3. P_type 4. f_id 5. offer_id 6. total_amount	Payment details and total payment details
food	1. f_id 2. f_name 3. f_price 4. f_quantity	Food details
offer	1. o_id 2. o_name 3. discount	Offer details
refund/cancellation	1. r_id 2. p_id 3. r_reason 4. r_amount 5. r_type	Return or cancellation detail

3. Rejected Nouns

Noun	**Reject Reason**
Large data	irrelevant
process	general
internet	Irrelevant
Data	Irrelevant
resources	redundant
ID proof	vague
deploy	duplicate
Description	Attribute
database	general
information	vague

4. Rejected Verbs

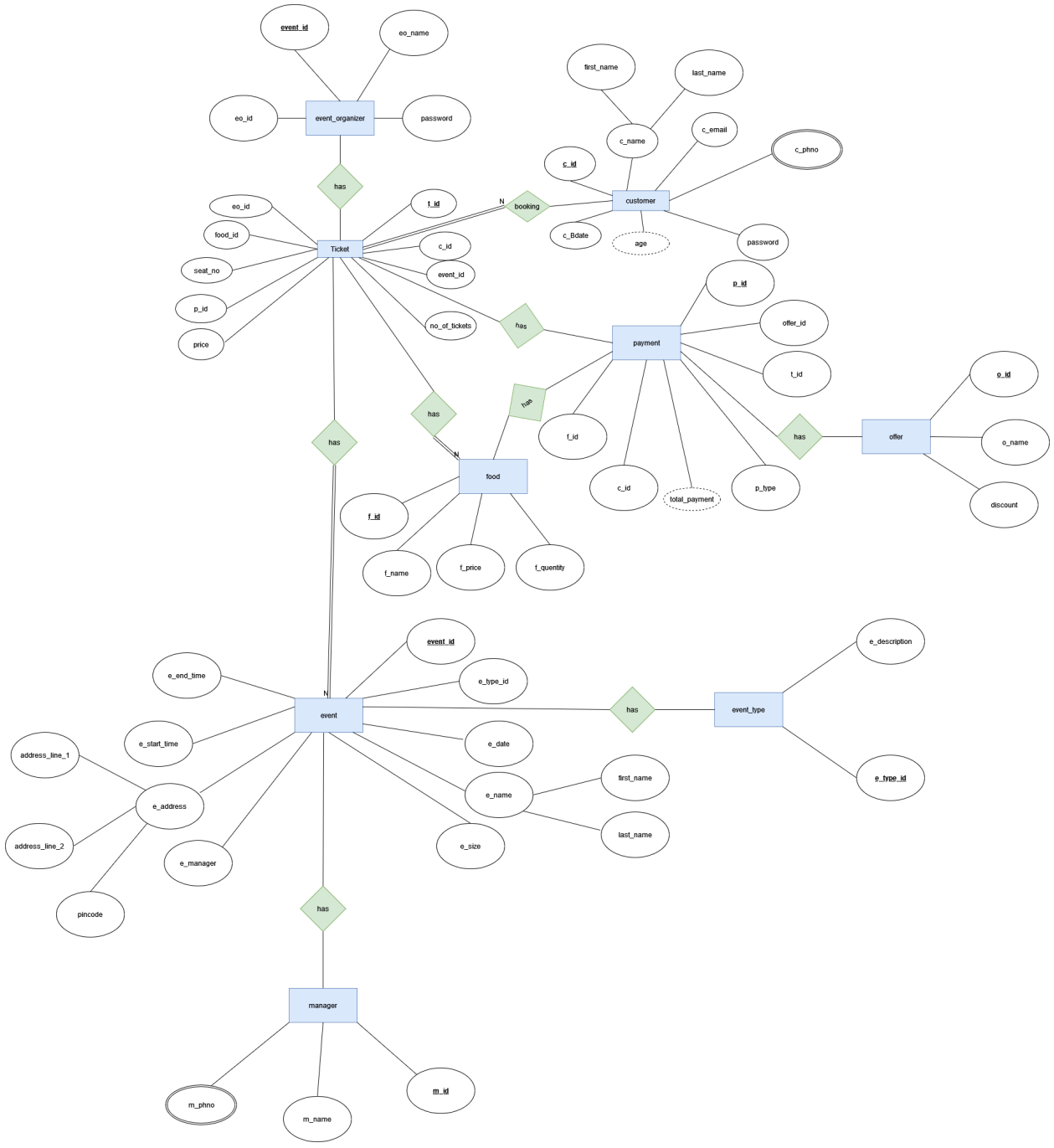
Verb	**Reject Reason**
edit	duplicate
modify	duplicate
access	duplicate
generate	general

assign	duplicate
authorize	general
view	general
help	general

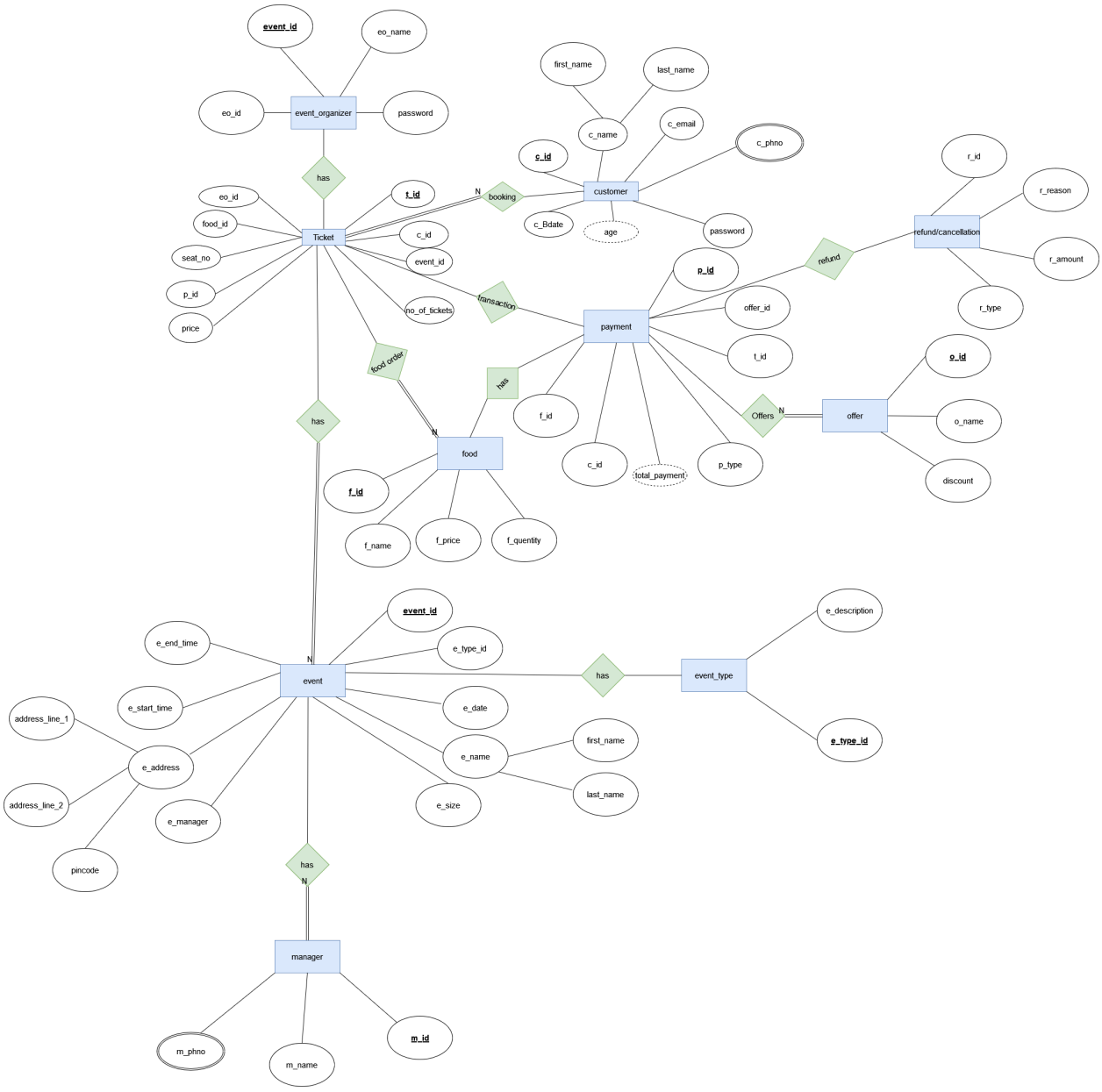
Section3: ER-Diagrams all versions

E-R Diagram:-

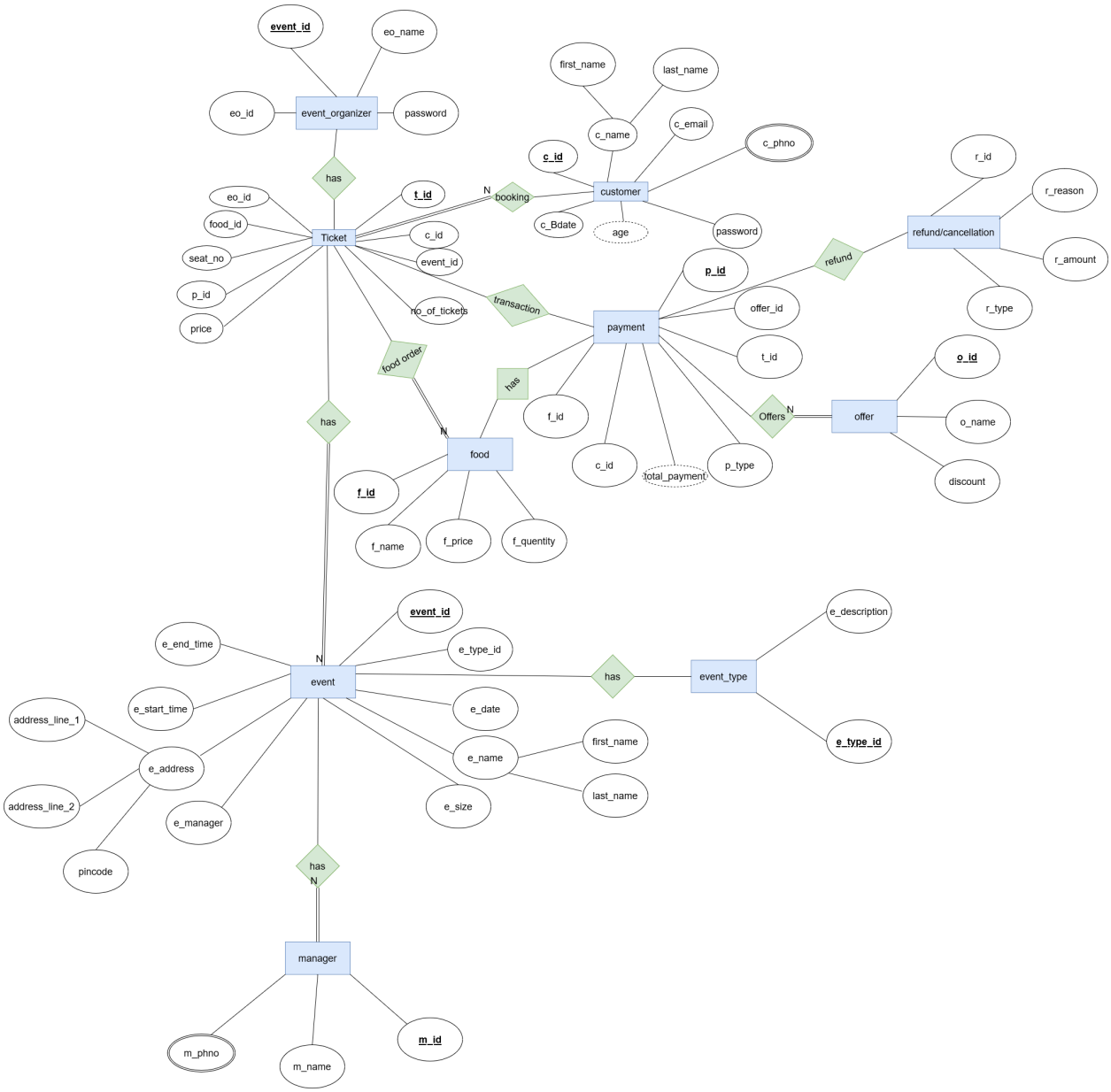
1. ER diagram (version 1)



2. ER diagram version 2

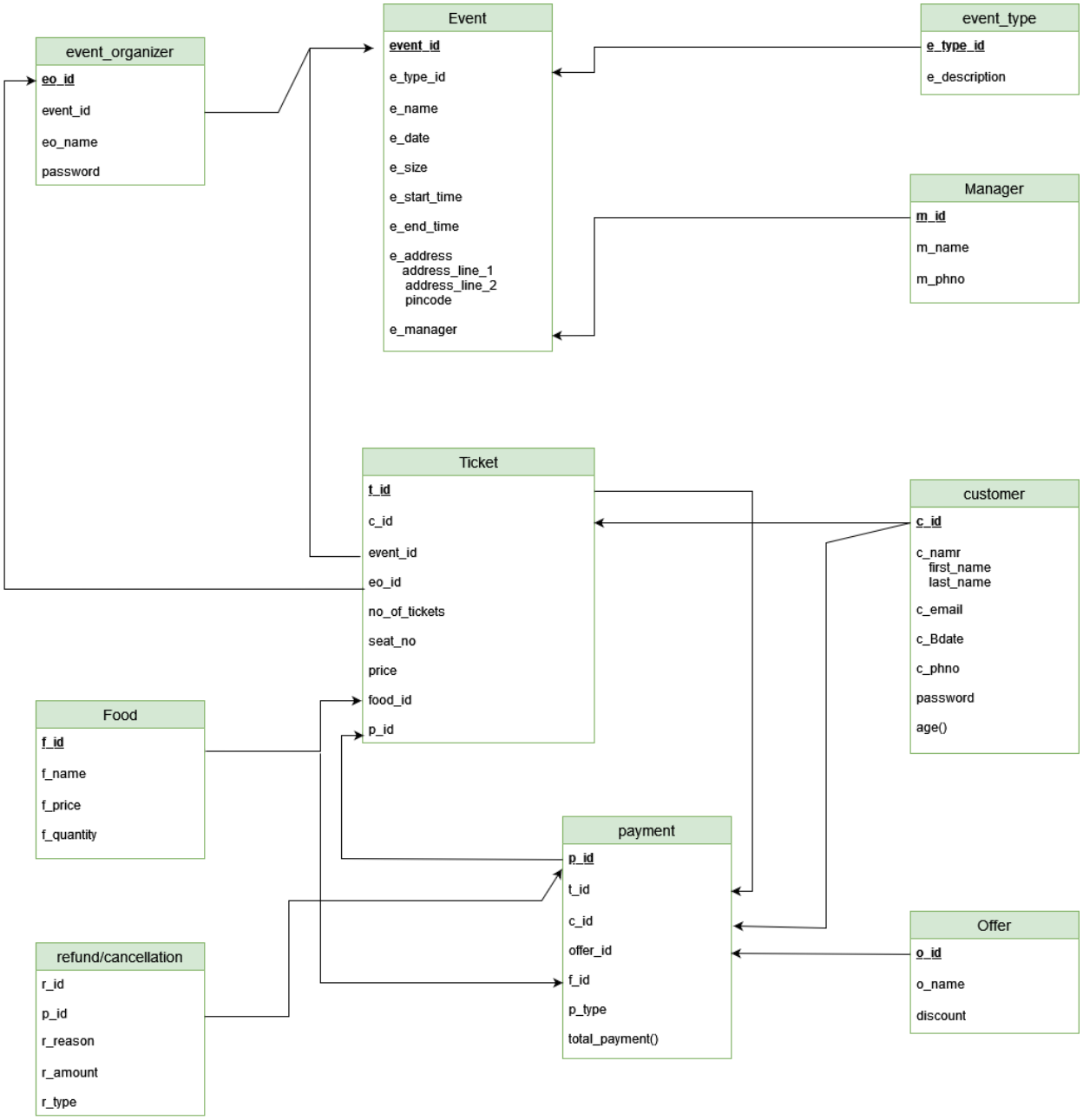


3. ER diagram final Version



Section4: Conversion of Final ER-Diagram to Relational Model.

4. RE relationship Diagram



Mapping E-R Model to Relational Model

- event_organizer(**eo_id**, event_id, eo_name, password)
  - Primary key:- eo_id
  - Foreign key:- event_id

- event(**event_id**, e_type_id, e_name, e_date, e_size, e_start_time, e_end_time, e_address, address_line1, address_line2, pincode, e_manager)
  - Primary key:- event_id
  - Foreign key:- event_type_id, e_manager

- event_type(**e_type_id**, e_desciption)
  - Primary key:- e_type_id

- manager(**m_id**, m_name, m_phno)
  - Primary key:- m_id

- customer (**c_id**, first_name, last_name, c_email, c_Bdate c_phno, password, age)
  - Primary key:- c_id

- ticket (**t_id**, c_id, event_id,eo_id, no_of_tickets, seat_no, price, food_id, p_id)
  - Primary key:- t_id
  - Foreign key:- c_id, event_id

- payment (**p_id**, t_id, c_id, offer_id, f_id, p_type, total_payment)
  - Primary key:- p_id
  - Foreign key:- c_id, t_id

- food (**f_id**, f_name, f_price, f_quantity)
  - Primary key:- f_id

- offer (**o_id**, o_name, discount)
  - Primary key:- o_id

- refund/cancellation (**r_id**, p_id, r_reason, r_amount, r_type)
  - Primary key:- r_id

- booking (**t_id**, c_id)
  - Primary key:- t_id

- food_order(**f_id** , t_id)
  - Primary key:- f_id

- transaction(**t_id** , p_id)
  - Primary key:- t_id

Section 5: Normalization and Schema Refinement

➔ Normalization & Schema Refinement

1) List all the Relations & Schemas with all details (Original Design of Database).

- event_organizer(**eo_id**, event_id, eo_name, password)
  - Primary key:- eo_id
  - Foreign key:- event_id
- event(**event_id**, e_type_id, e_name, e_date, e_size, e_start_time, e_end_time, e_address, address_line1, address_line2, pincode, e_manager)
  - Primary key:- event_id
  - Foreign key:- event_type_id, e_manager
- event_type(**e_type_id**, e_desciption)
  - Primary key:- e_type_id
- manager(**m_id**, m_name, m_phno)
  - Primary key:- m_id
- customer (**c_id**, first_name, last_name, c_email, c_Bdate, c_phno, age, password)
  - Primary key:- c_id
- ticket (**t_id**, c_id, event_id, eo_id, no_of_tickets, seat_no, price, food_id, p_id)
  - Primary key:- t_id
  - Foreign key:- c_id, event_id
- payment (**p_id**, t_id, c_id, p_type, f_id, offer_id, total_amount)
  - Primary key:- p_id
  - Foreign key:- c_id, t_id
- food (**f_id**, f_name, f_price, f_quantity)
  - Primary key:- f_id
- offer (**o_id**, o_name, discount)
  - Primary key:- o_id
- refund/cancellation (**r_id**, p_id, r_reason, r_amount, r_type)
  - Primary key:- r_id

- booking (**t_id**, c_id)
    - Primary key:- t_id
- food_order(**f_id** , t_id)
    - Primary key:- f_id
- transaction(**t_id** , p_id)
    - Primary key:- t_id

2) Identify and list all types of dependencies (PK, FK, Functional Dependencies) for each relation.

- event_organizer(eo_id -> eo_name, eo_id -> password)
- event(event_id -> e_name, event_id -> e_date, event_id -> e_size, event_id -> e_start_time, event_id -> e_end_time, event_id -> address_line1, event_id -> address_line2, event_id -> pincode)
- Event_type -> no dependencies
- manager(m_id ->m_name, m_id -> m_phno)
- customer (c_id -> c_first_name, c_id -> c_last_name, c_id -> c_address, c_id -> c_email, c_id ->c_phoneno,c_id -> c_Bdate, c_Bdate -> c_age, c_id -> password)
- ticket (t_id -> no_of_tickets, t_id -> price, seat_no, t_id -> p_id)
- payment (p_id -> p_type, p_id -> f_id, t_id -> total_amount)
- food (f_id -> f_name, f_id -> f_price)
- Offer -> no dependencies
- refund/cancellation (r_id -> r_reason, p_id -> r_amount, p_id -> r_type)

3) List of redundancies existing for every schema which is part of the database

➢ event_organizer(**eo_id**, event_id, eo_name, password)

- Primary key:- eo_id
- Foreign key:- event_id
- Redundancy:-
- Update anomaly:- It will set all foreign-keyed relation attributes to the updated value.
- Insert anomaly:- You can not insert a new event without assigning an event.
- Delete anomaly:- If deleted, you lost all data regarding that event organizer.
- 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
- 2NF:- There are no partial dependencies to this relation is already in 2NF.
- 3NF:- There are no transitive dependencies, so this relation is already in 3NF.

➢ event(**event_id**, e_type_id, e_name, e_date, e_size, e_start_time, e_end_time, e_address, address_line1, address_line2, pincode, e_manager)

- Primary key:- event_id
- Foreign key:- event_type_id, e_manager

- Redundancy:- There can be multiple event names with the same addresses or at the same time.

- Update anomaly:- It will set all foreign-keyed relation attributes to the updated value.
- Insert anomaly:- You can not insert a new event without assigning event type and event manager for that event.
- Delete anomaly:- if deleted, you lost all data regarding that event.

- 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
- 2NF:- There are no partial dependencies to this relation is already in 2NF.
- 3NF:- Here in this entity relationship, start time and end time is dependent on event name ane event name is dependent on event_id, so because of these, there is transitive dependency.
    - event(**event_id**, e_type_id, e_name, e_size, e_address, address_line1, address_line2, pincode, e_manager)
    - time(**e_id**, start_time, end_time)

➢ event_type(**e_type_id**, e_desciption)

- Primary key:- e_type_id
- Update anomaly:- It will not affect anything
- Insert anomaly:- It will not affect anything
- Delete anomaly:- If deleted, you lost all data regarding that event type.

- 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
- 2NF:- There are no partial dependencies to this relation is already in 2NF.
- 3NF:- There are no transitive dependencies, so this relation is already in 3NF.

- manager(**m_id**, m_name, m_phno)
    - Primary key:- m_id
    - Update anomaly:- It will not affect anything
    - Insert anomaly:-It will not affect anything
    - Delete anomaly:- If delete you lose all data regarding that manager
    - 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
    - 2NF:- There are no partial dependencies to this relation is already in 2NF.
    - 3NF:- There are no transitive dependencies, so this relation is already in 3NF.

- customer (**c_id**, c_name, c_address, c_email, c_phoneno,c_Bdate, c_age, password)
    - Primary key:- c_id
    - Update anomaly:- if you update the birth date, then age will automatically change.
    - Insert anomaly:-It will not affect anything.
    - Delete anomaly:- If delete you lose all data regarding that customer
    - 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
    - 2NF:- There are no partial dependencies to this relation is already in 2NF.
    - 3NF:- There are no transitive dependencies, so this relation is already in 3NF.

- ticket (**t_id**, c_id,event_id, eo_id, no_of_tickets, price, food_id, t_price, seat_no, p_id)
    - Primary key:- t_id
    - Foreign key:- c_id, event_id
    - Update anomaly:- It will not affect anything.
    - Insert anomaly:- You can not insert a new event without assigning customers and an event.
    - Delete anomaly:- If deleted, you lost all data regarding that ticket.
    - 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
    - 2NF:- There are no partial dependencies to this relation is already in 2NF.
    - 3NF:- There are no transitive dependencies, so this relation is already in 3NF.

- payment (**p_id**, c_id, p_type, f_id, offer_id, total_amount)
    - Primary key:- p_id
    - Foreign key:- c_id, t_id
    - Update anomaly:- you cannot update details of payment directly.
    - Insert anomaly:-You can not insert a new event without having a ticket.
    - Delete anomaly:- If deleted, you lost all data regarding that payment.
    - 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
    - 2NF:- There are no partial dependencies to this relation is already in 2NF.
    - 3NF:- There are no transitive dependencies, so this relation is already in 3NF.

- food (**f_id**, f_name, f_price, f_quantity)
    - Primary key:- f_id
    - Update anomaly:- It will not affect anything
    - Insert anomaly:- It will not affect anything.
    - Delete anomaly:- It will not affect anything

- 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
- 2NF:- There are no partial dependencies to this relation is already in 2NF.
- 3NF:- There are no transitive dependencies, so this relation is already in 3NF.

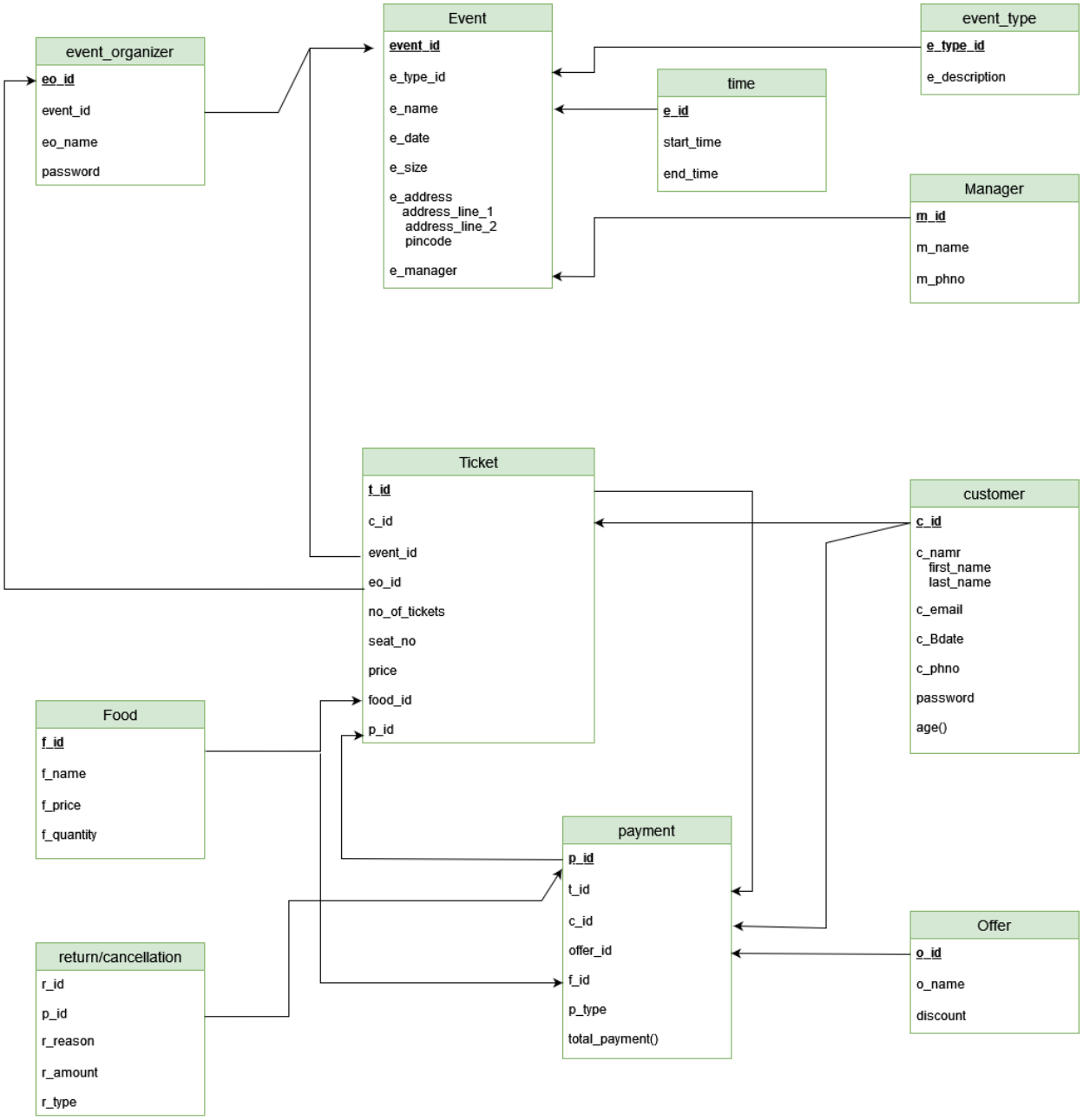
➢ offer (**o_id**, o_name, discount)

- Primary key:- o_id
- Update anomaly:- It will not affect anything.
- Insert anomaly:- It will not affect anything.
- Delete anomaly:- It will not affect anything.
- 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
- 2NF:- There are no partial dependencies to this relation is already in 2NF.
- 3NF:- There are no transitive dependencies, so this relation is already in 3NF.

➢ refund/cancellation (**r_id**, p_id, r_reason, r_amount, r_type)

- Primary key:- r_id
- Update anomaly:- you can not update anything here.
- Insert anomaly:-You can not insert a new event without having a payment id.
- Delete anomaly:-
- 1NF:- Our relation is already in 1NF, and there are no multiple attributes.
- 2NF:- There are no partial dependencies to this relation is already in 2NF.
- 3NF:- There are no transitive dependencies, so this relation is already in 3NF.

4) Final ER Diagram:-



5) List all final Relations & Schemas with all details (final Design of Database)

➢ event_organizer(**eo_id**, event_id, eo_name, password)

- Primary key:- eo_id

- Foreign key:- event_id

➢ event(**event_id**, e_type_id, e_name, e_date, e_size, e_address, address_line1, address_line2, pincode, e_manager)

- Primary key:- event_id
- Foreign key:- event_type_id, e_manager,e_name

➢ time(e_name, start_time, end_time)

- Primary key:- e_name

➢ event_type(**e_type_id**, e_desciption)

- Primary key:- e_type_id

➢ manager(**m_id**, m_name, m_phno)

- Primary key:- m_id

➢ customer (**c_id**, c_name, c_address, c_email, c_phoneno, c_age, password)

- Primary key:- c_id

➢ ticket (**t_id**, c_id, t_price, seat_no, p_id)

- Primary key:- t_id
- Foreign key:- c_id, event_id

➢ payment (**p_id**, c_id, p_type, f_id, offer_id, total_amount)

- Primary key:- p_id
- Foreign key:- c_id, t_id

➢ food (**f_id**, f_name, f_price, f_quantity)

- Primary key:- f_id

➢ offer (**o_id**, o_name, discount)

- Primary key:- o_id

➢ refund/cancellation (**r_id**, p_id, r_reason, r_amount, r_type)

- Primary key:- r_id

➢ booking (**t_id**, c_id)

- Primary key:- t_id

- food_order(**f_id** , t_id)
    - Primary key:- f_id
- transaction(**t_id** , p_id)
    - Primary key:- t_id

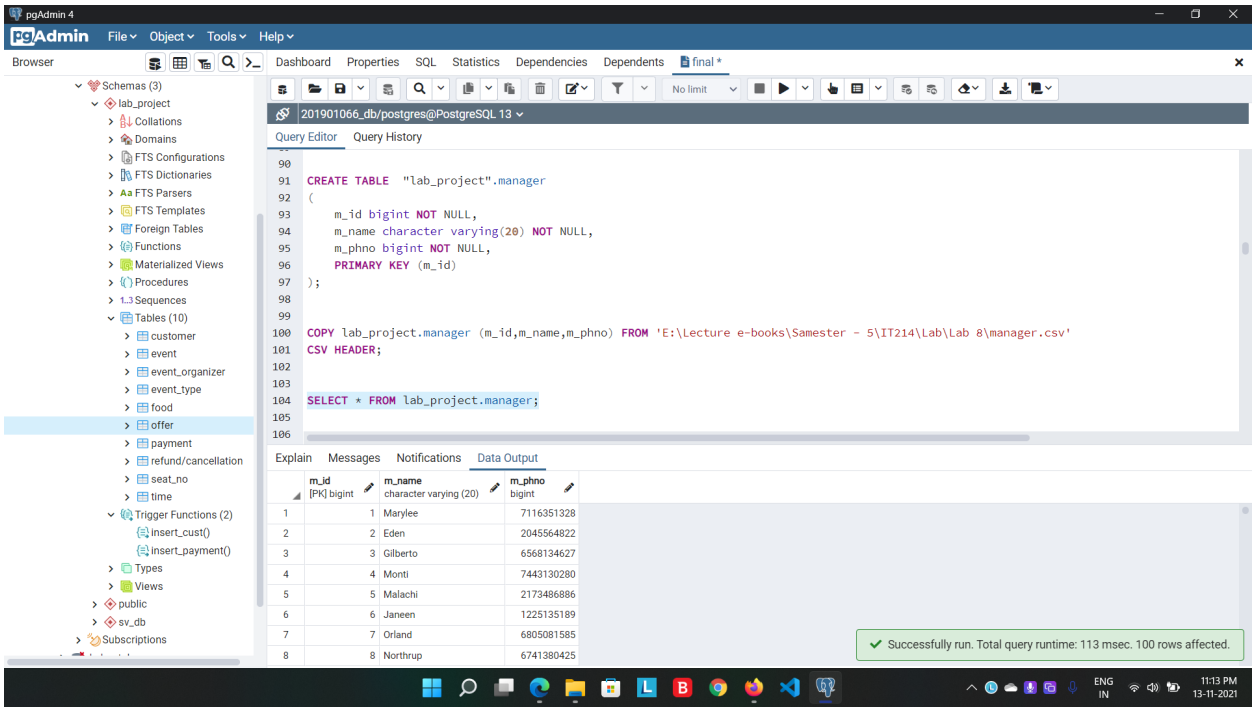
Section 6: SQL Final DDL Scripts, Insert statements, 40 SQL Queries with Snapshots of output of each query

6) DDL Scripts:-

1. Manager

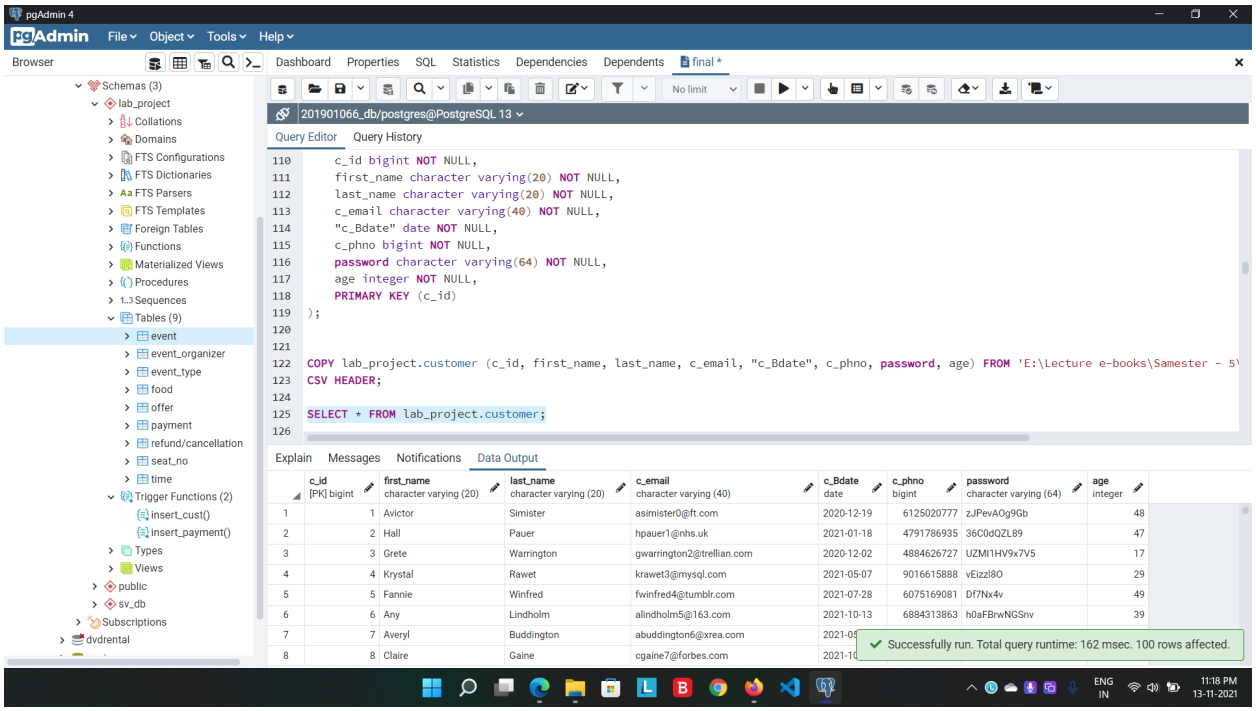
```
CREATE TABLE  "lab_project".manager

(

        m_id bigint NOT NULL,

        m_name character varying(20) NOT NULL,

        m_phno bigint NOT NULL,

        PRIMARY KEY (m_id)

);
```

COPY lab_project.manager (m_id,m_name,m_phno) FROM 'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\manager.csv' CSV HEADER;



2. Customer

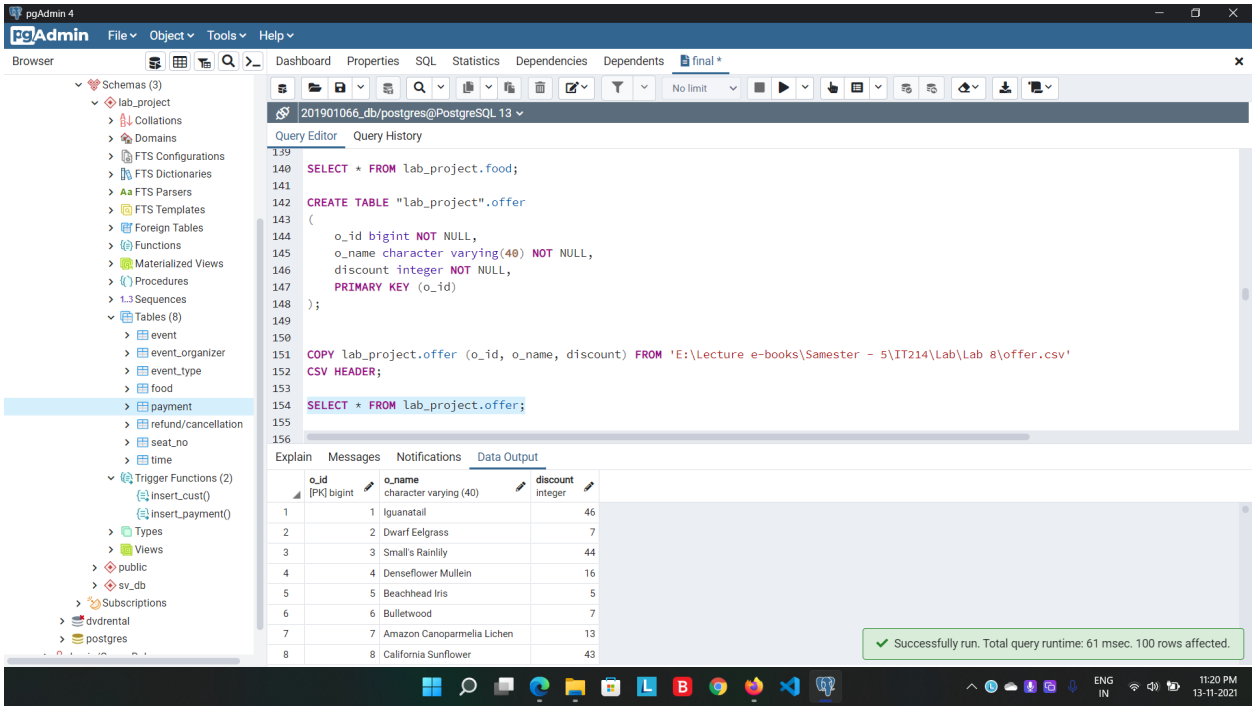
```
CREATE TABLE "lab_project".customer
(
        c_id bigint NOT NULL,
        first_name character varying(20) NOT NULL,
        last_name character varying(20) NOT NULL,
        c_email character varying(40) NOT NULL,
        "c_Bdate" date NOT NULL,
        c_phno bigint NOT NULL,
        password character varying(64) NOT NULL,
        age integer NOT NULL,
        PRIMARY KEY (c_id)
);
COPY lab_project.customer (c_id, first_name, last_name, c_email, "c_Bdate", c_phno, password, age) FROM
'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\customer.csv' CSV HEADER;
```



3. Offer

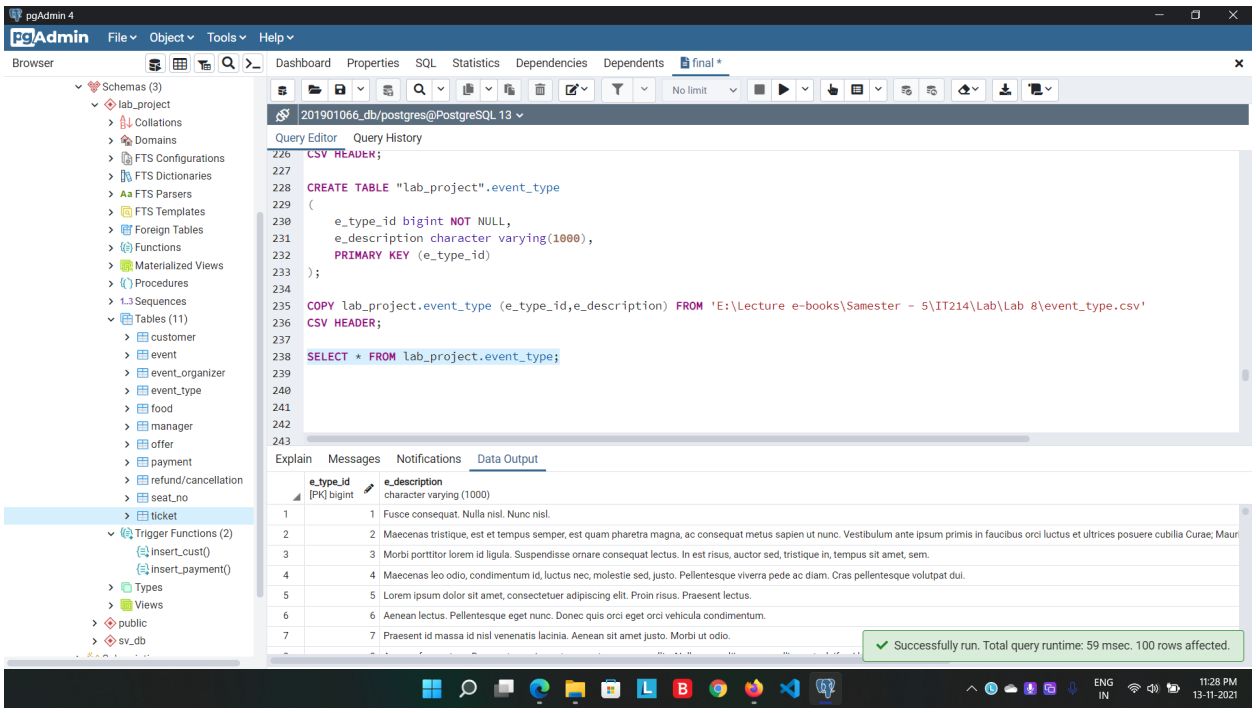
```
CREATE TABLE "lab_project".offer

(

        o_id bigint NOT NULL,

        o_name character varying(40) NOT NULL,

        discount integer NOT NULL,

        PRIMARY KEY (o_id)

);
```

```
COPY lab_project.offer (o_id, o_name, discount) FROM 'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\offer.csv' CSV HEADER;
```



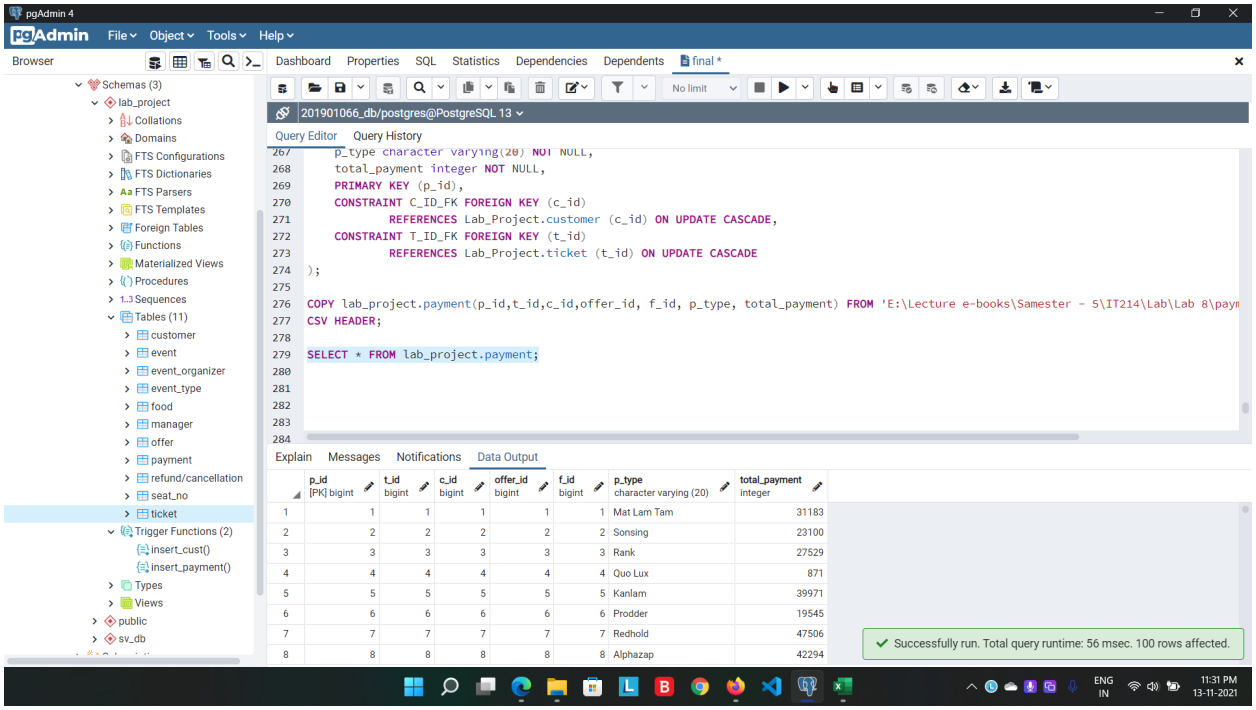
4. Event_type

```
CREATE TABLE "lab_project".event_type

(

        e_type_id bigint NOT NULL,

        e_description character varying(1000),

        PRIMARY KEY (e_type_id)

);

COPY lab_project.event_type (e_type_id,e_description) FROM 'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\event_type.csv' CSV HEADER;
```



5. Payment

```
CREATE TABLE "lab_project".payment

(

        p_id bigint NOT NULL,

        t_id bigint NOT NULL,

        c_id bigint NOT NULL,

        offer_id bigint,

        f_id bigint,

        p_type character varying(20) NOT NULL,

        total_payment integer NOT NULL,

        PRIMARY KEY (p_id),

        CONSTRAINT C_ID_FK FOREIGN KEY (c_id)

        REFERENCES Lab_Project.customer (c_id) ON UPDATE CASCADE,

        CONSTRAINT T_ID_FK FOREIGN KEY (t_id)

                REFERENCES Lab_Project.ticket (t_id) ON UPDATE CASCADE

);

COPY lab_project.payment(p_id,t_id,c_id,offer_id, f_id, p_type, total_payment) FROM 'E:\Lecture
e-books\Samester - 5\IT214\Lab\Lab 8\payment.csv' CSV HEADER;
```



6. Time

CREATE TABLE "lab_project".time

(

e_id bigint NOT NULL,

e_start_time time(0) without time zone NOT NULL,

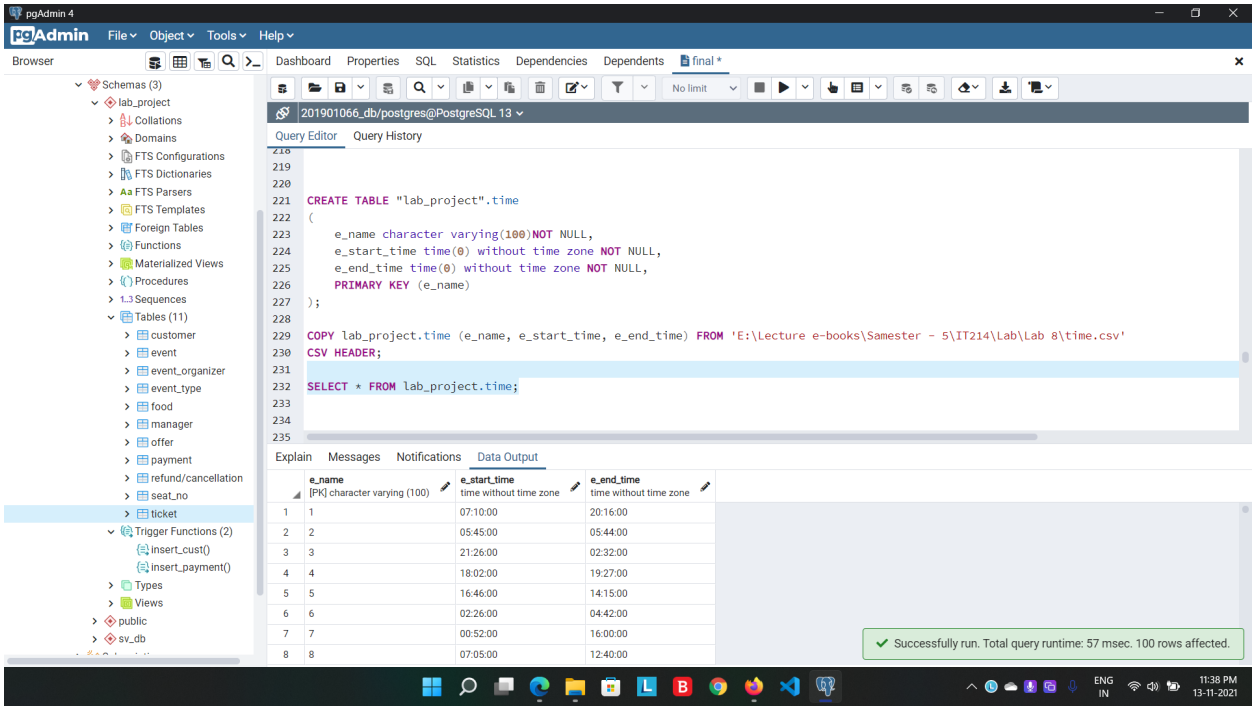
e_end_time time(0) without time zone NOT NULL,

PRIMARY KEY (e_name)

);

COPY lab_project.time (e_name, e_start_time, e_end_time) FROM 'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\time.csv'

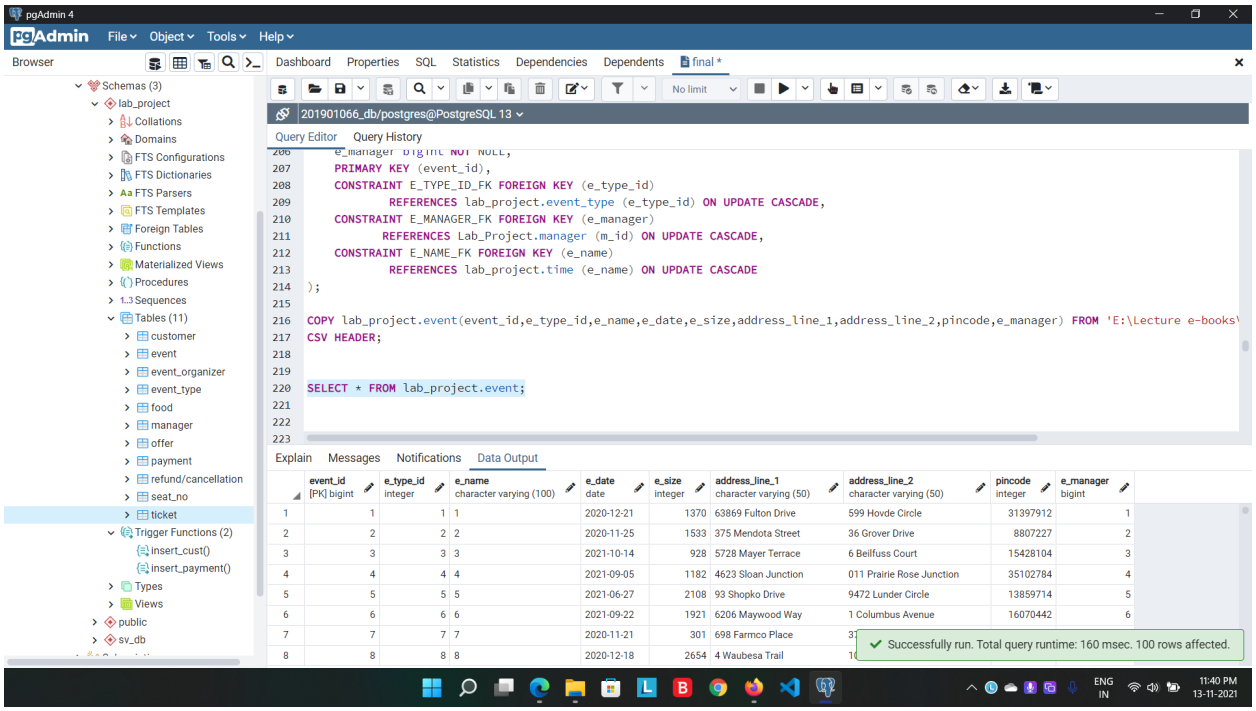
CSV HEADER;



7. Event

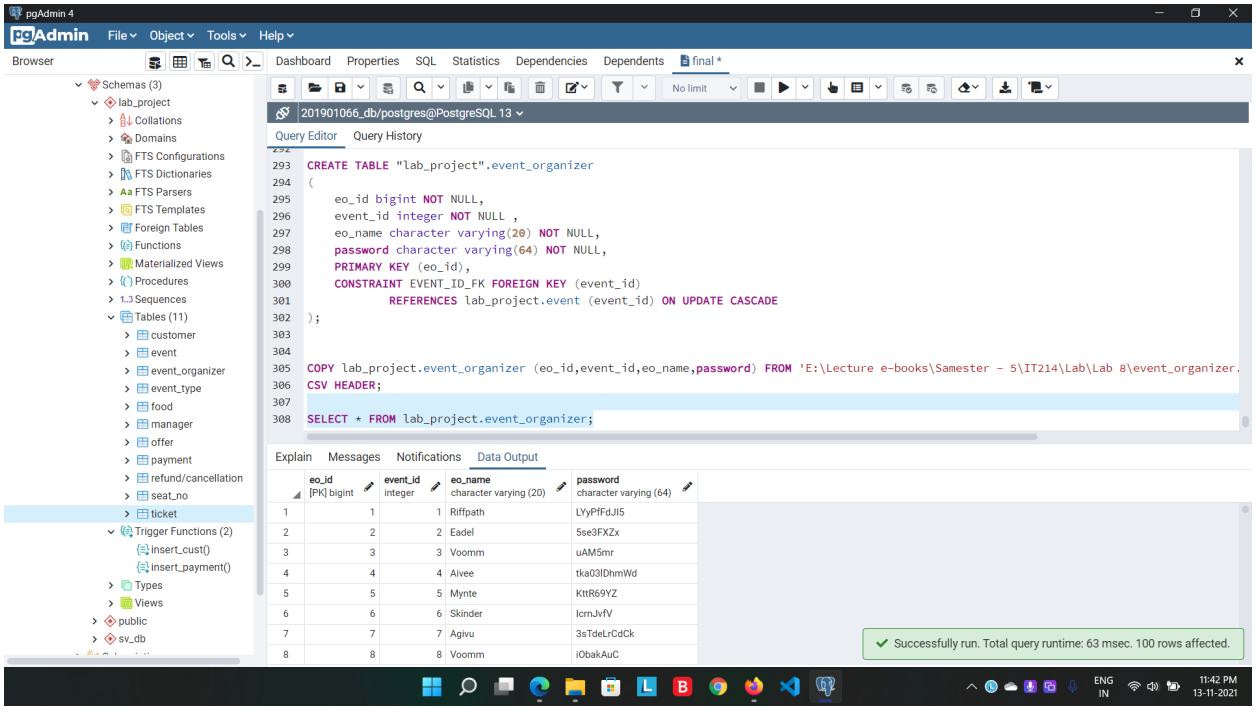
```
CREATE TABLE "lab_project".event
(
    event_id bigint NOT NULL,
    e_type_id integer NOT NULL,
    e_name character varying(100) NOT NULL,
    e_date date NOT NULL,
    e_size integer NOT NULL,
    address_line_1 character varying(50) NOT NULL,
    address_line_2 character varying(50),
    pincode integer NOT NULL,
    e_manager bigint NOT NULL,
    PRIMARY KEY (event_id),
```

```
    CONSTRAINT E_TYPE_ID_FK FOREIGN KEY (e_type_id)
        REFERENCES lab_project.event_type (e_type_id) ON UPDATE CASCADE,
  CONSTRAINT E_MANAGER_FK FOREIGN KEY (e_manager)
        REFERENCES Lab_Project.manager (m_id) ON UPDATE CASCADE,
  CONSTRAINT E_NAME_FK FOREIGN KEY (e_name)
            REFERENCES lab_project.time (e_name) ON UPDATE CASCADE
);

COPY
lab_project.event(event_id,e_type_id,e_name,e_date,e_size,address_line_1,address_line_2,pincode,e_manager)
FROM 'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\event.csv'

CSV HEADER;
```



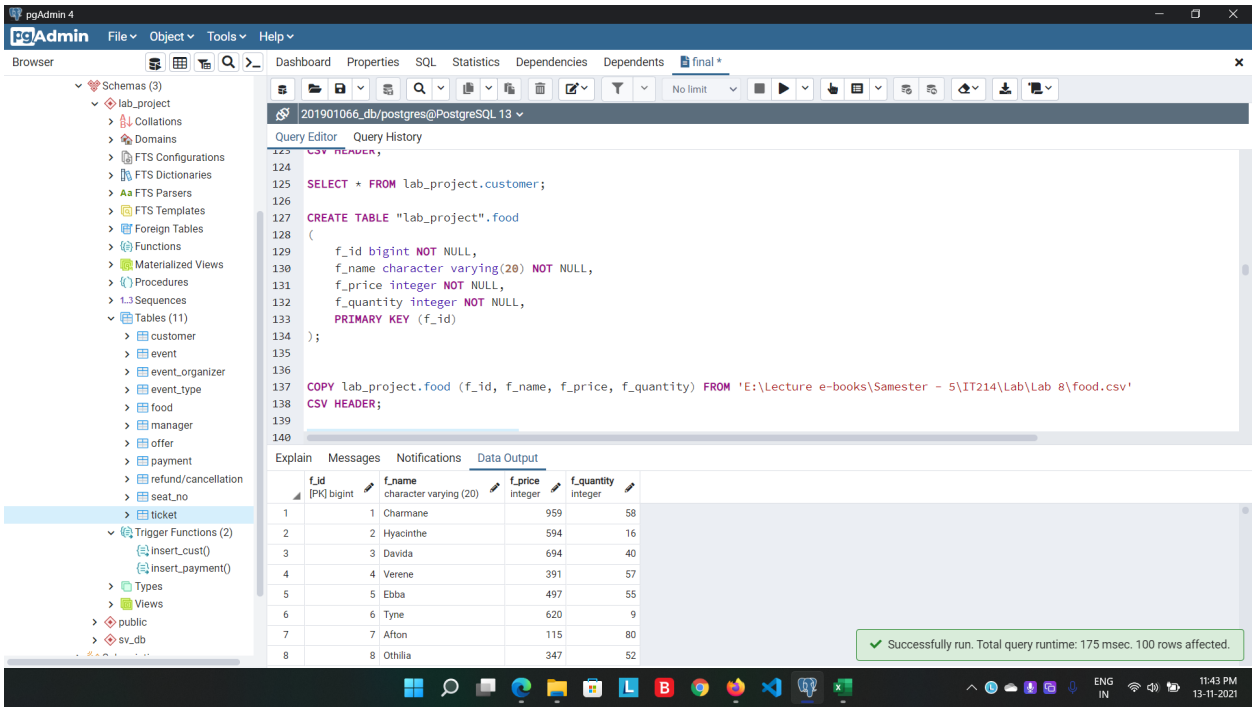
8. Event_organizer

```
CREATE TABLE "lab_project".event_organizer
(
        eo_id bigint NOT NULL,
        event_id integer NOT NULL ,
        eo_name character varying(20) NOT NULL,
        password character varying(64) NOT NULL,
        PRIMARY KEY (eo_id),
        CONSTRAINT EVENT_ID_FK FOREIGN KEY (event_id)
        REFERENCES lab_project.event (event_id) ON UPDATE CASCADE
);

COPY lab_project.event_organizer (eo_id,event_id,eo_name,password) FROM 'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\event_organizer.csv'
CSV HEADER;
```



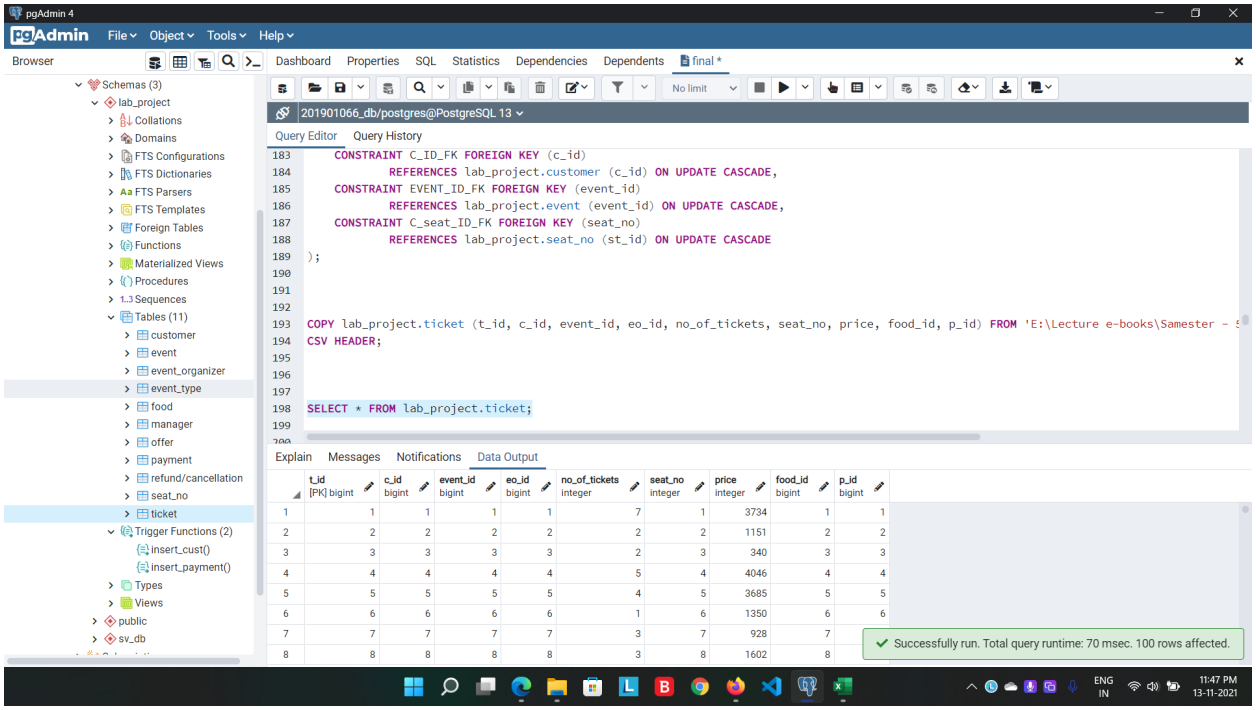
9. Food

```
CREATE TABLE "lab_project".food

(

        f_id bigint NOT NULL,

        f_name character varying(20) NOT NULL,

        f_price integer NOT NULL,

        f_quantity integer NOT NULL,

        PRIMARY KEY (f_id)

);

COPY lab_project.food (f_id, f_name, f_price, f_quantity) FROM 'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\food.csv'

CSV HEADER;
```



10. Ticket

```
CREATE TABLE "lab_project".ticket

(

        t_id bigint NOT NULL,

        c_id bigint NOT NULL,

        event_id bigint NOT NULL,

        eo_id bigint NOT NULL,

        no_of_tickets integer NOT NULL,

        seat_no integer NOT NULL,

        price integer NOT NULL,

        food_id bigint,

        p_id bigint NOT NULL,

        PRIMARY KEY (t_id),

        CONSTRAINT C_ID_FK FOREIGN KEY (c_id)

        REFERENCES lab_project.customer (c_id) ON UPDATE CASCADE,

        CONSTRAINT EVENT_ID_FK FOREIGN KEY (event_id)

        REFERENCES lab_project.event (event_id) ON UPDATE CASCADE,

   CONSTRAINT C_seat_ID_FK FOREIGN KEY (seat_no)

        REFERENCES lab_project.seat_no (st_id) ON UPDATE CASCADE

);

COPY lab_project.ticket (t_id, c_id, event_id, eo_id, no_of_tickets, seat_no, price, food_id, p_id) FROM
'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\ticket.csv'

CSV HEADER;
```



11. Refund/cancellation

CREATE TABLE "lab_project"."refund/cancellation"

(

r_id bigint NOT NULL,

p_id bigint NOT NULL,

r_reason character varying(200) NOT NULL,

r_amount bigint NOT NULL,

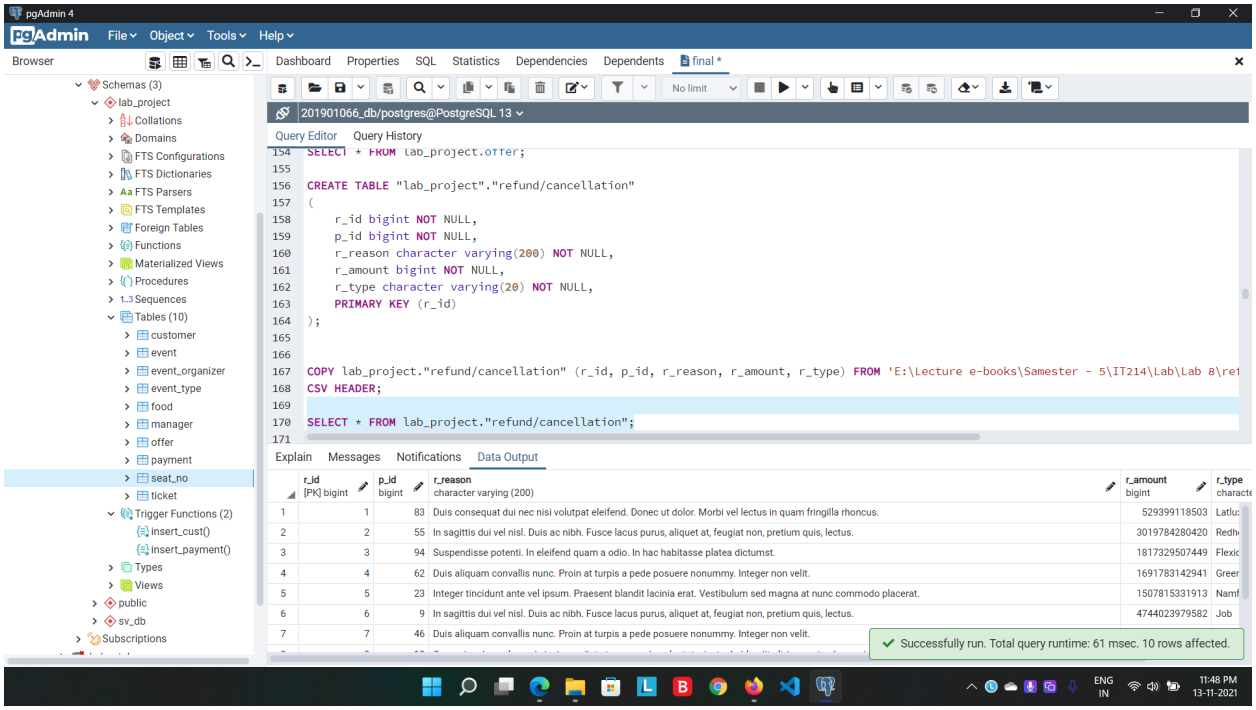
r_type character varying(20) NOT NULL,

PRIMARY KEY (r_id)

);

COPY lab_project."refund/cancellation" (r_id, p_id, r_reason, r_amount, r_type) FROM 'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\refund.csv'

CSV HEADER;

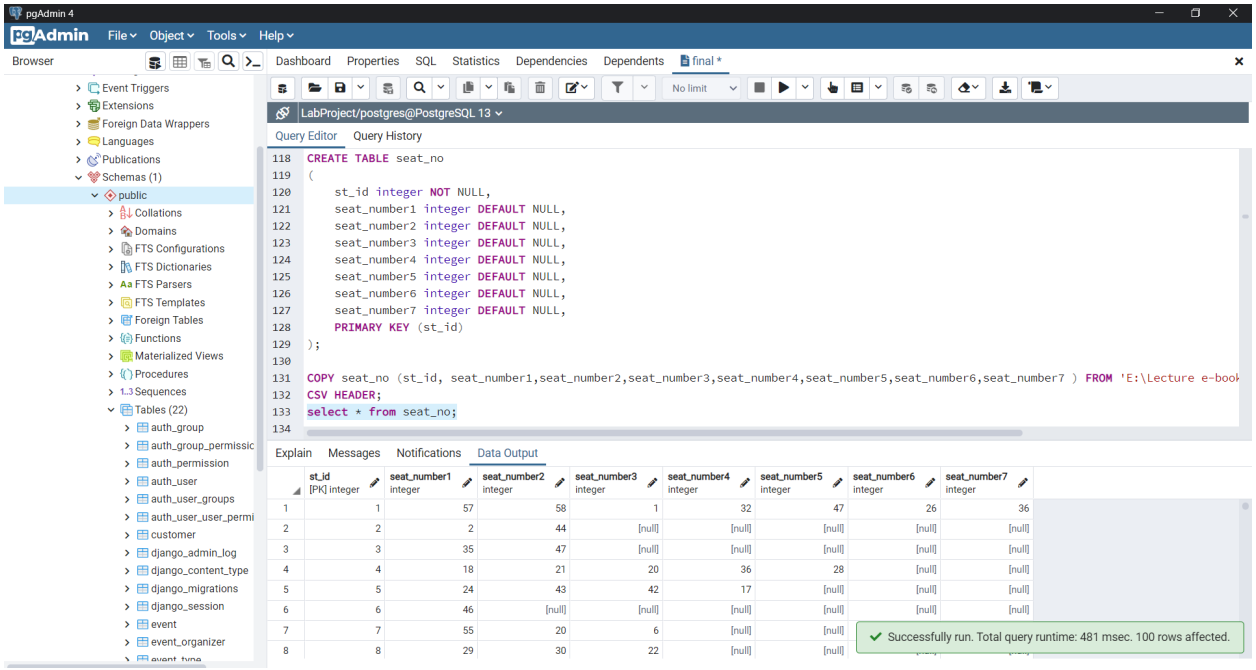


12. Seat_no

```
CREATE TABLE seat_no
(
        st_id integer NOT NULL,
        seat_number1 integer DEFAULT NULL,
   seat_number2 integer DEFAULT NULL,
   seat_number3 integer DEFAULT NULL,
   seat_number4 integer DEFAULT NULL,
   seat_number5 integer DEFAULT NULL,
   seat_number6 integer DEFAULT NULL,
   seat_number7 integer DEFAULT NULL,
   PRIMARY KEY (st_id)
);
```

COPY seat_no (st_id,
seat_number1,seat_number2,seat_number3,seat_number4,seat_number5,seat_number6,seat_number7) FROM
'E:\Lecture e-books\Samester - 5\IT214\Lab\Lab 8\seat_no.csv'

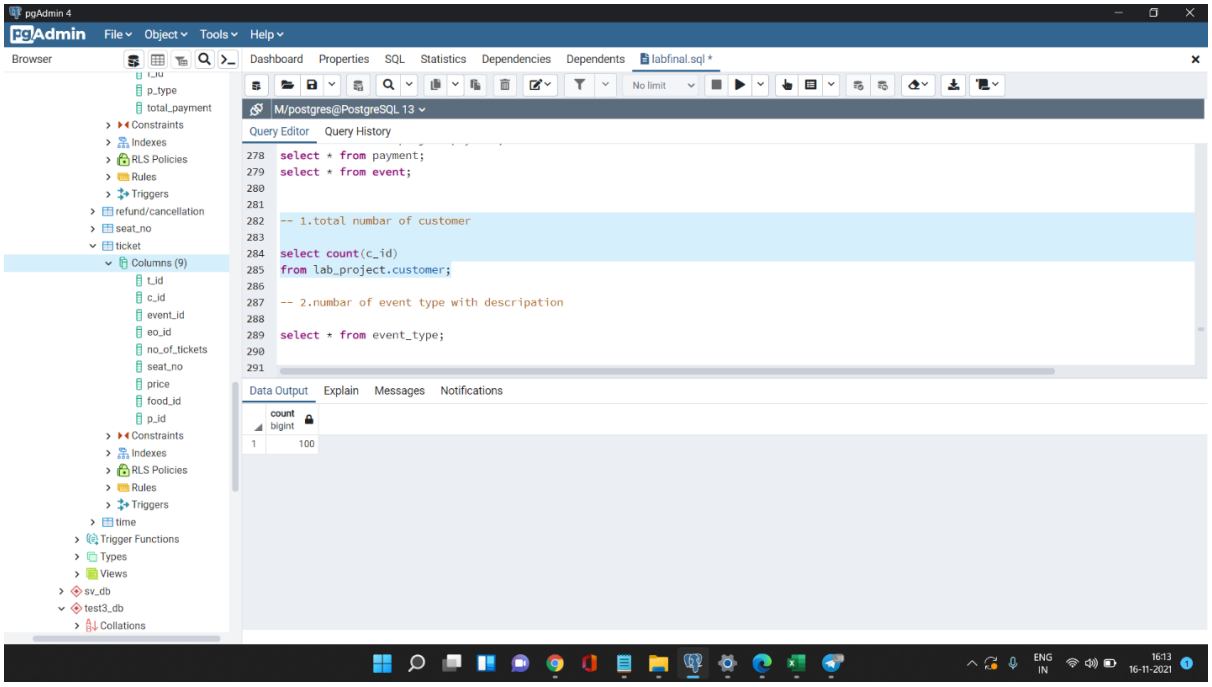
CSV HEADER;



Queries: -

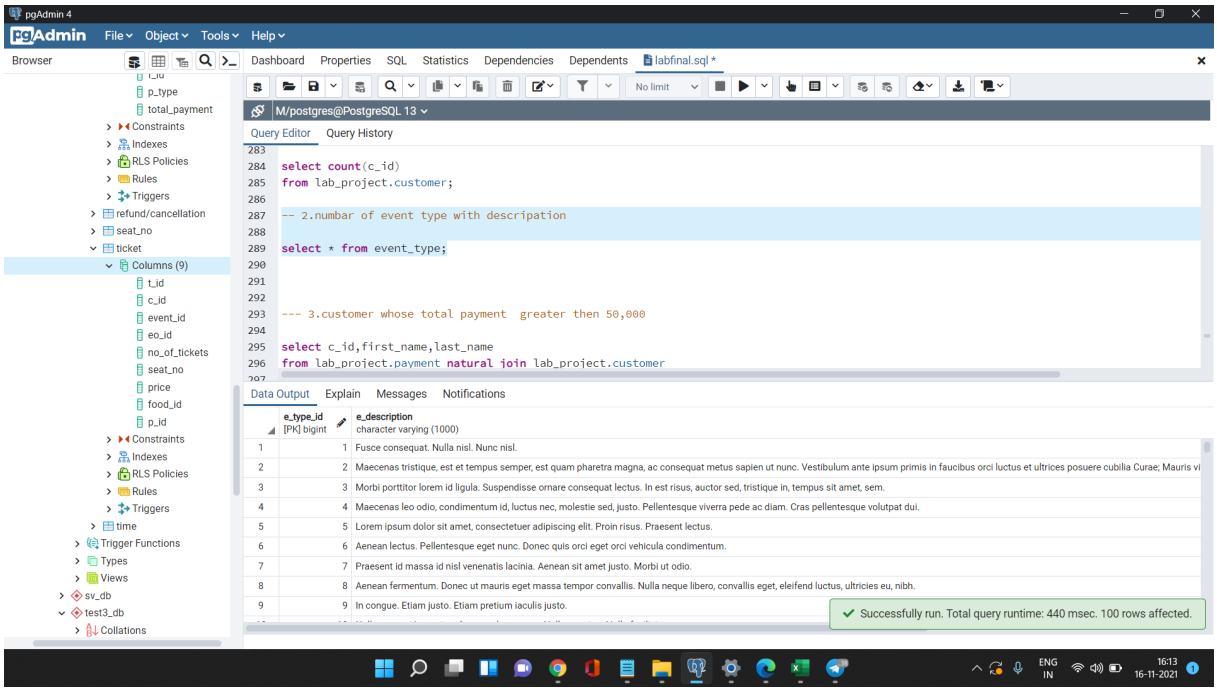
1. Total number of customers in the database.

select count(c_id)
from lab_project.customer;



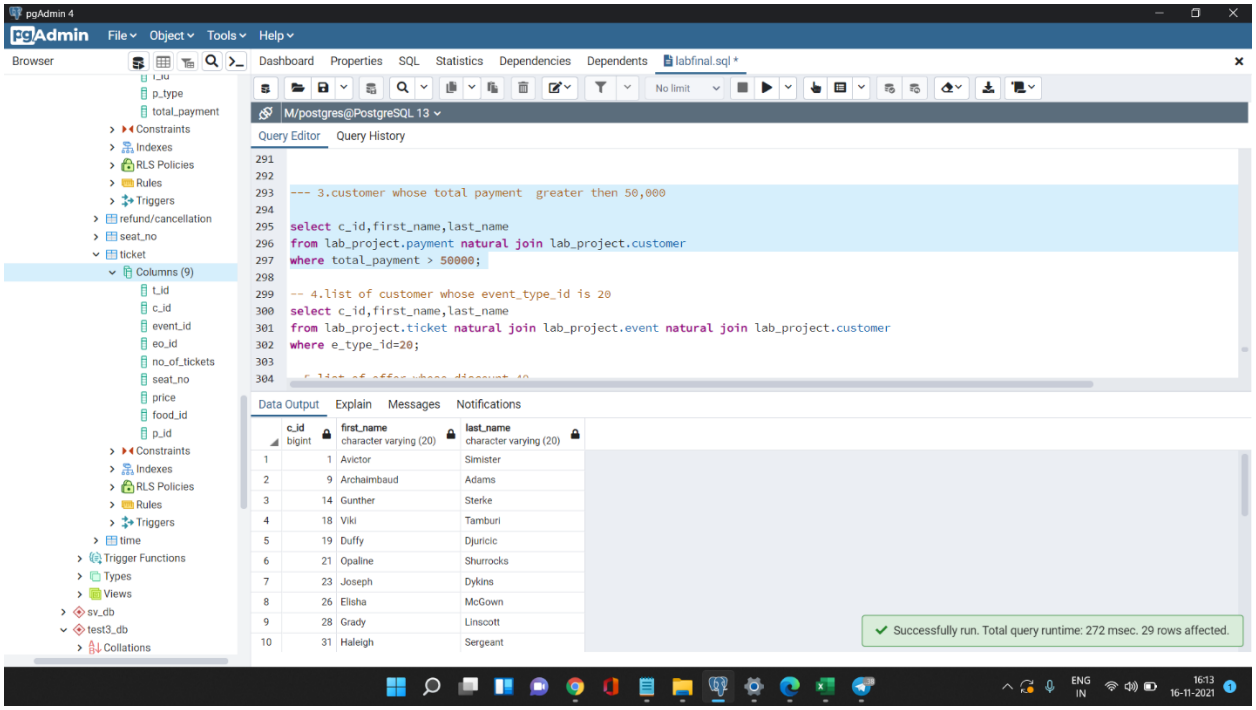
2. All different event types.

select * from event_type;



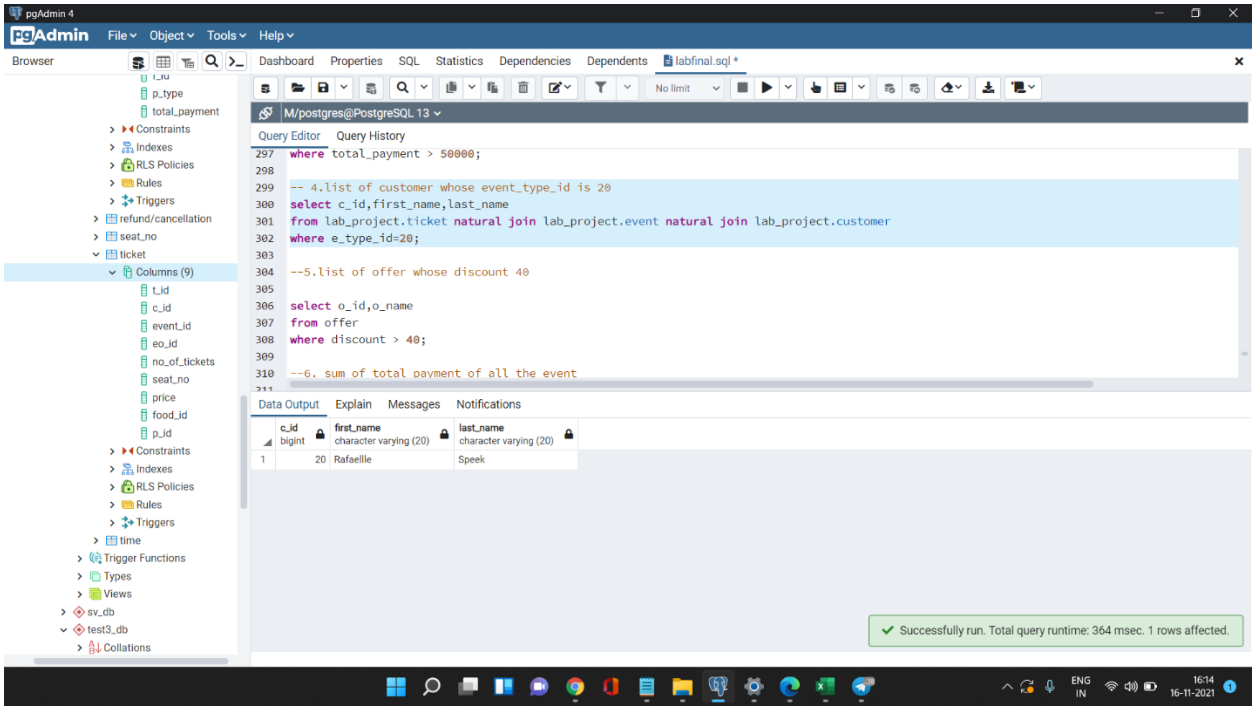
3. Customer details whose total payment is greater than 50000.

```
select c_id,first_name,last_name
from lab_project.payment natural join lab_project.customer
where total_payment > 50000;
```



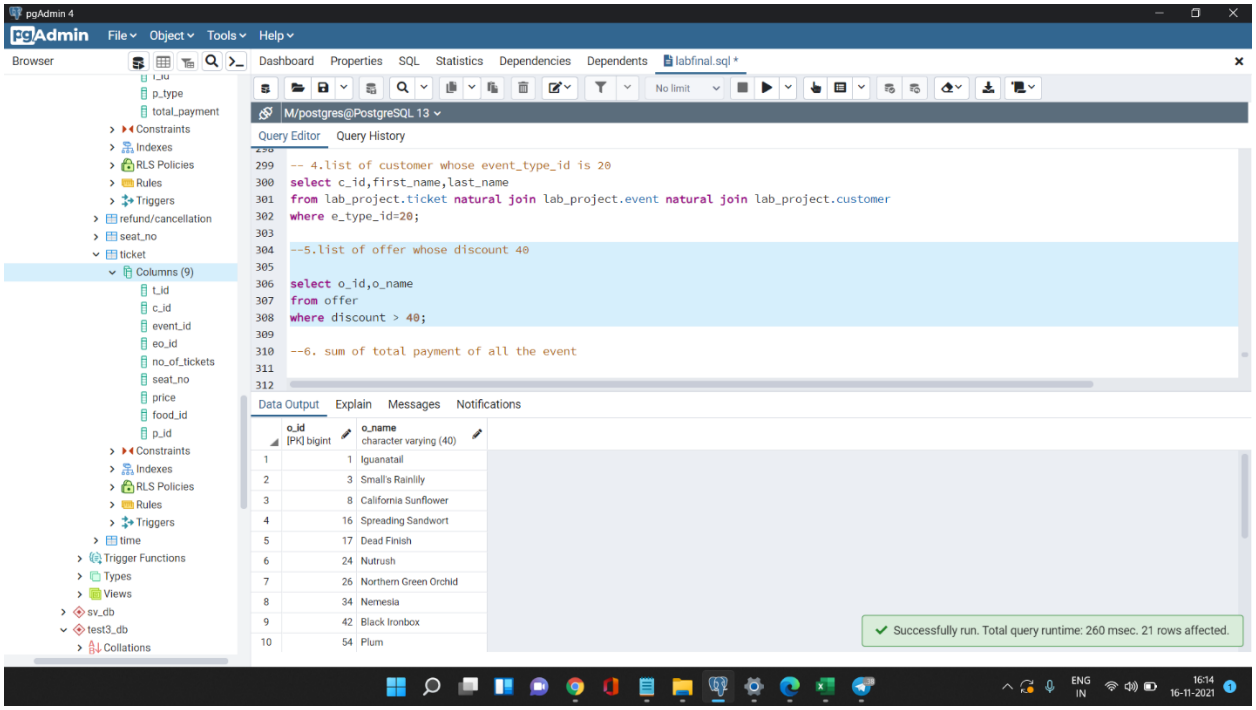
4. List of all customers whose event type id is 20.

```
select c_id,first_name,last_name
from lab_project.ticket natural join lab_project.event natural join lab_project.customer
where e_type_id=20;
```



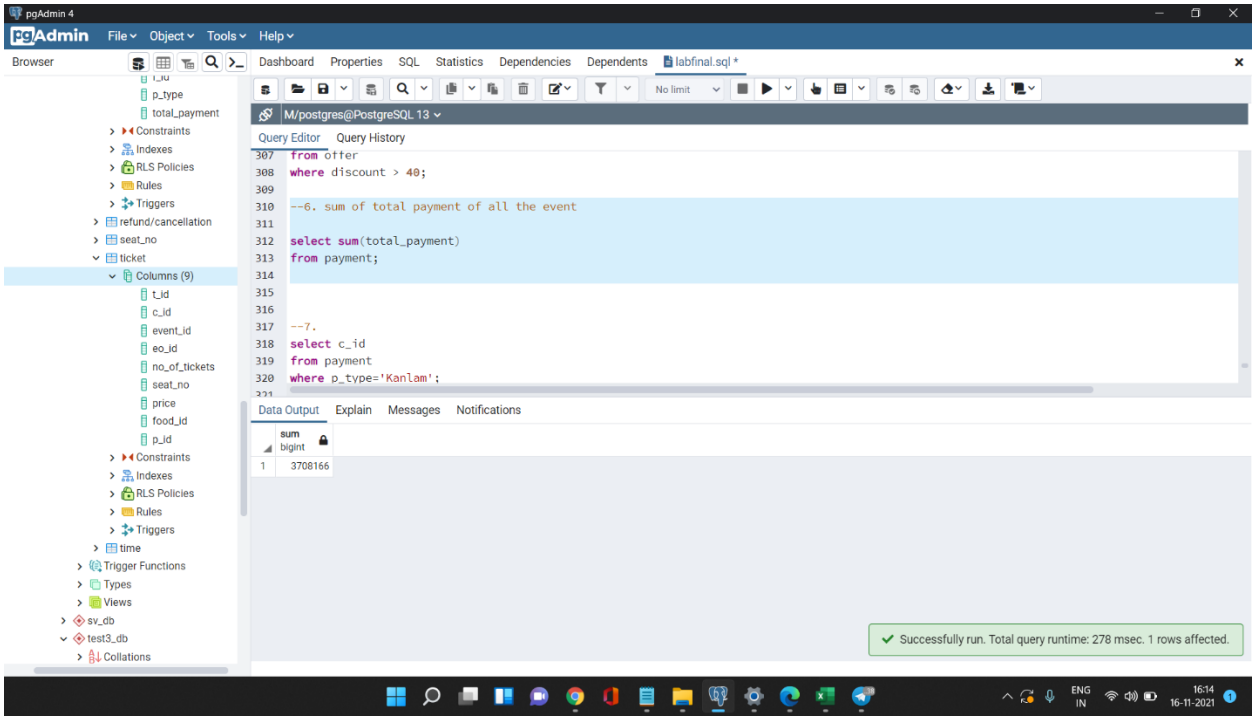
5. List of all offers which give discount more than 40%.

```
select o_id,o_name
from offer
where discount > 40;
```



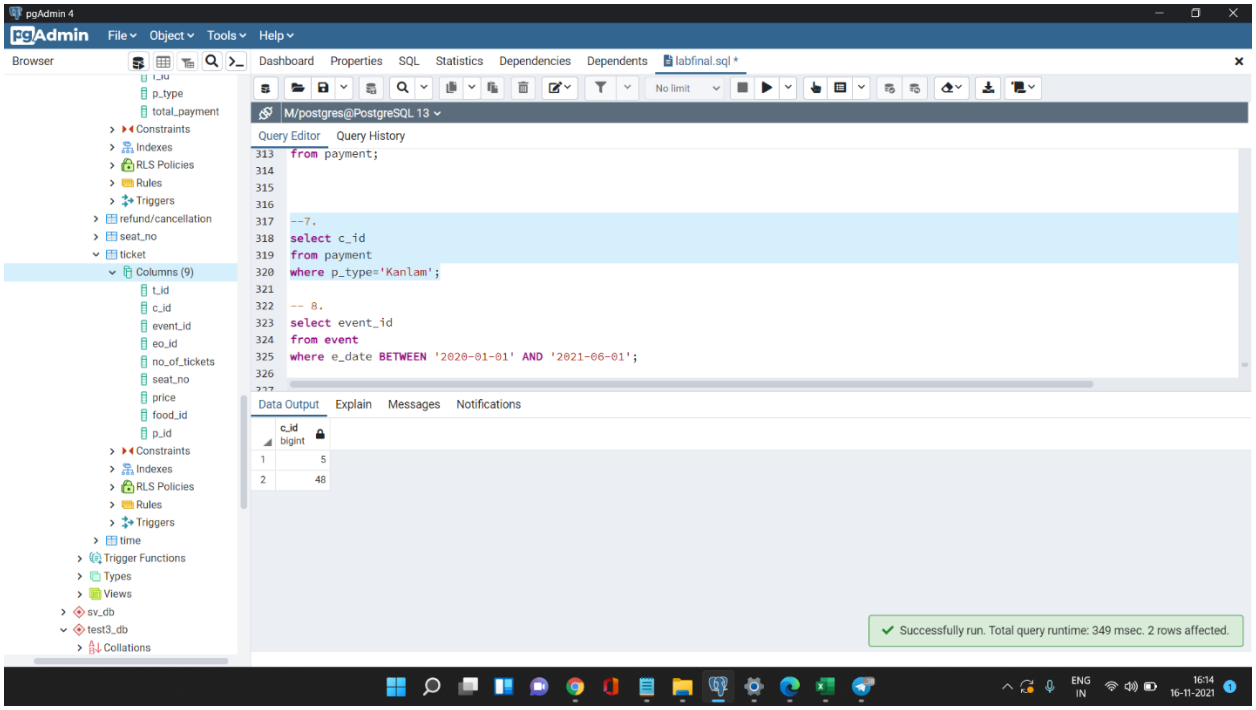
6. Total payments of all the event.

```
select sum(total_payment)
from payment;
```



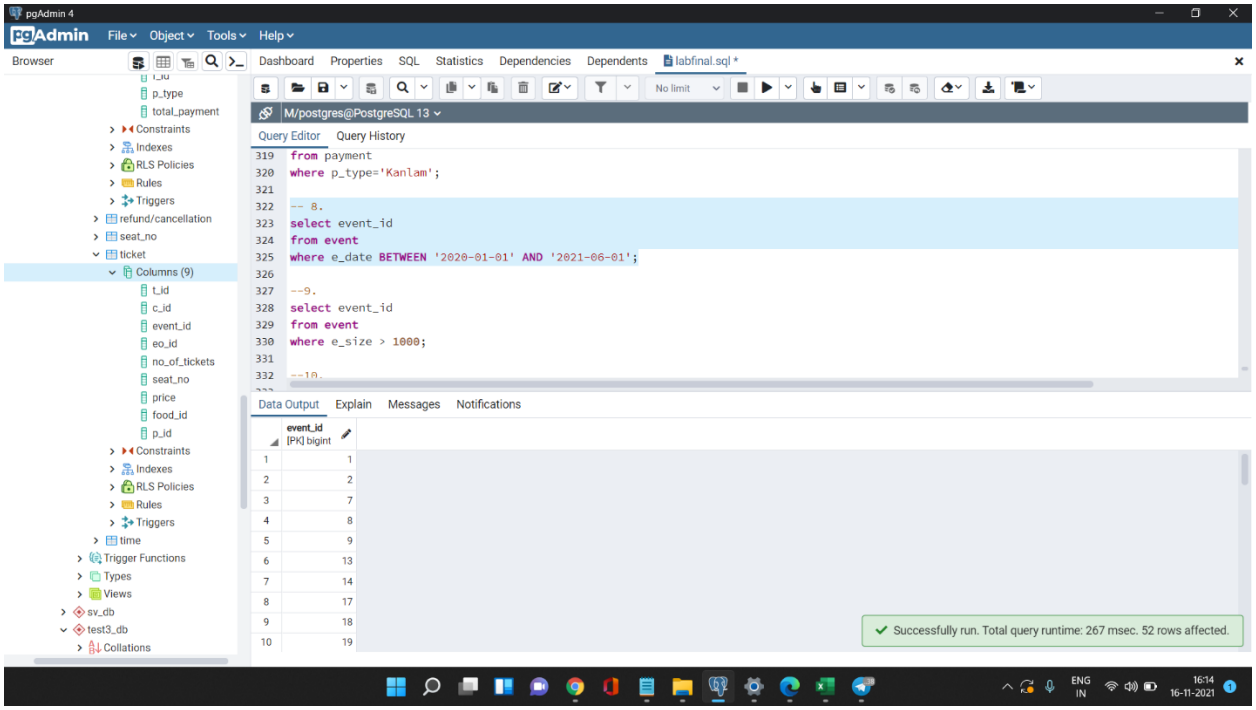
7. List of all customers whose payment type is kanlam(dummy data).

```
select c_id
from payment
where p_type='Kanlam';
```



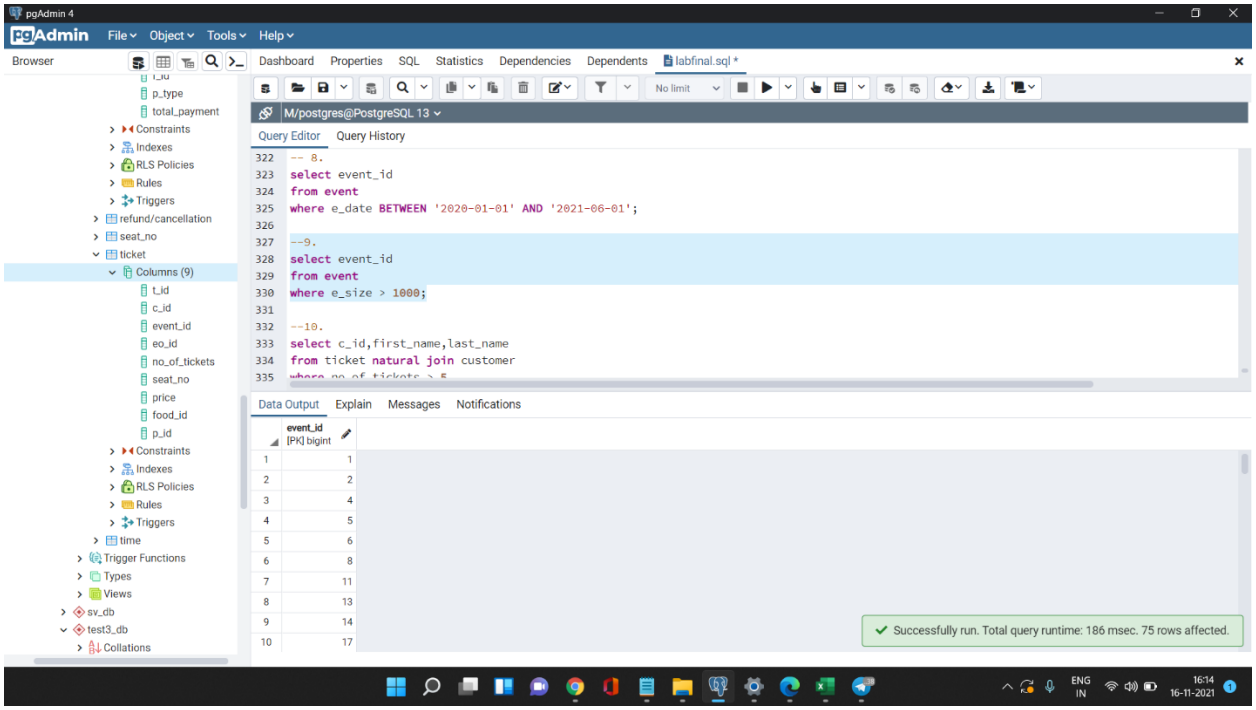
8. List of all events which took place between 2020-01-01 to 2021-06-01.

```
select event_id
from event
where e_date BETWEEN '2020-01-01' AND '2021-06-01';
```



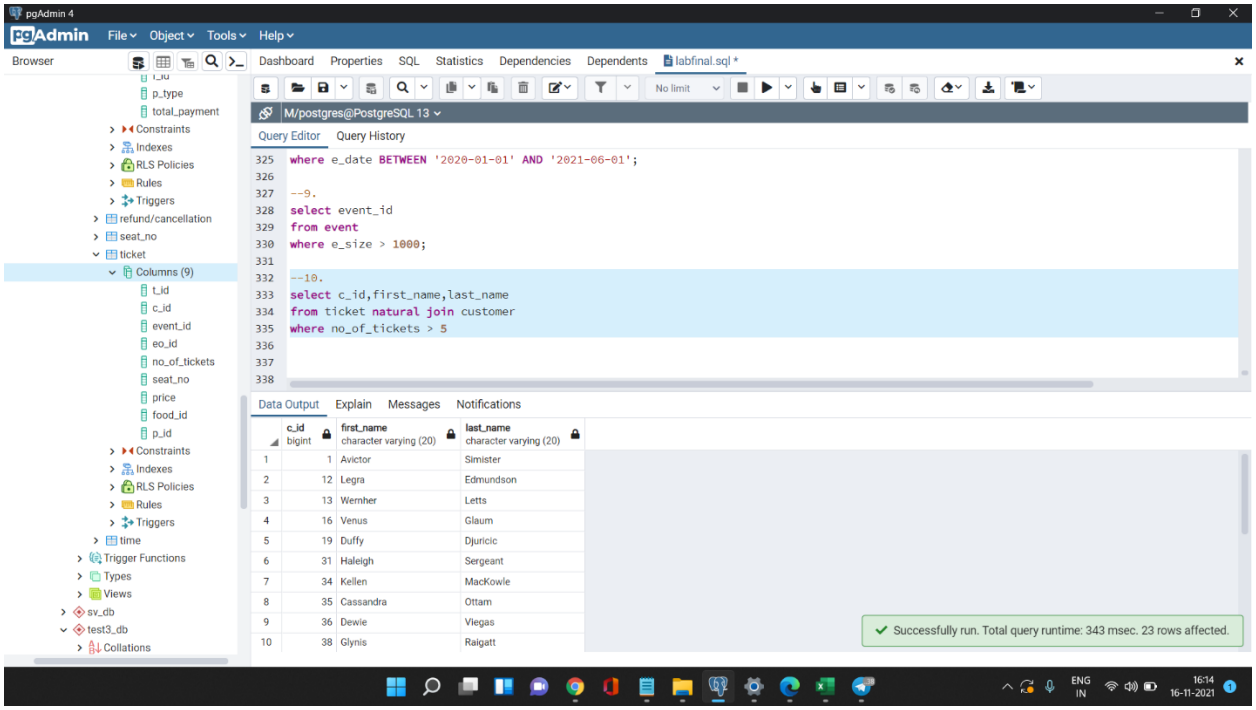
9. List of events with event size is greater than 1000.

```
select event_id
from event
where e_size > 1000;
```



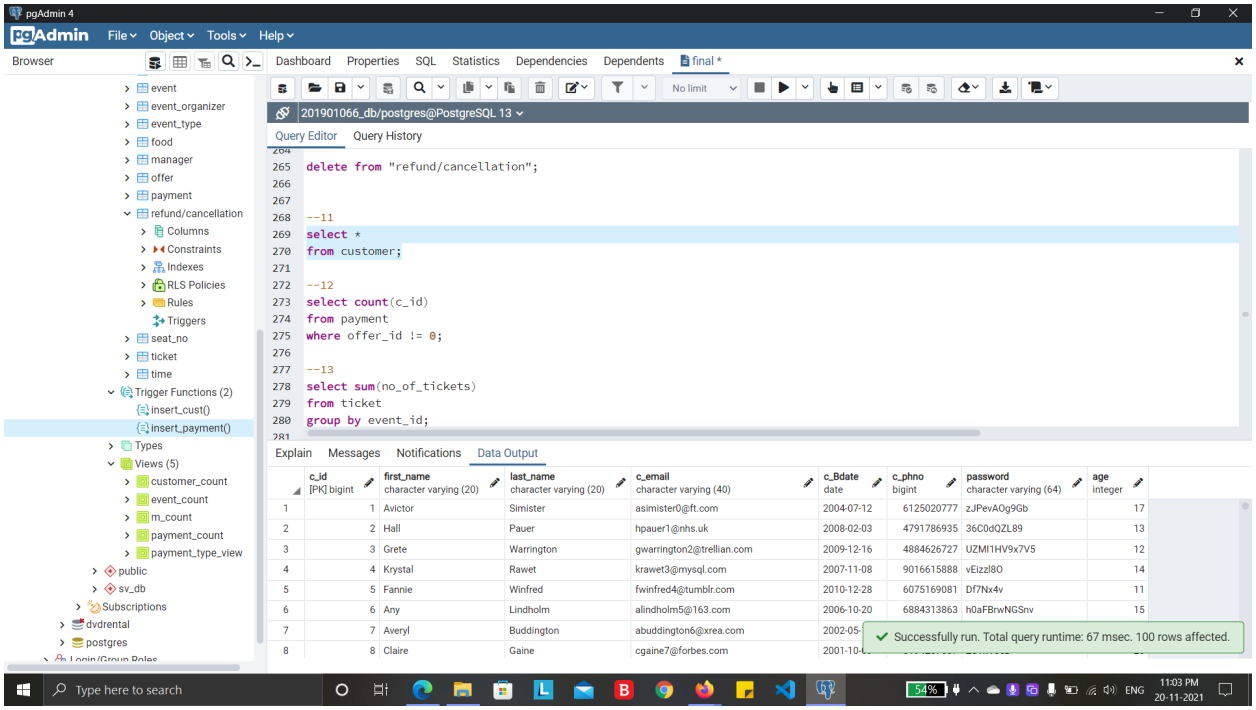
10. List of customers who booked more than five tickets at a time.

```
select c_id,first_name,last_name
from ticket natural join customer
where no_of_tickets > 5;
```



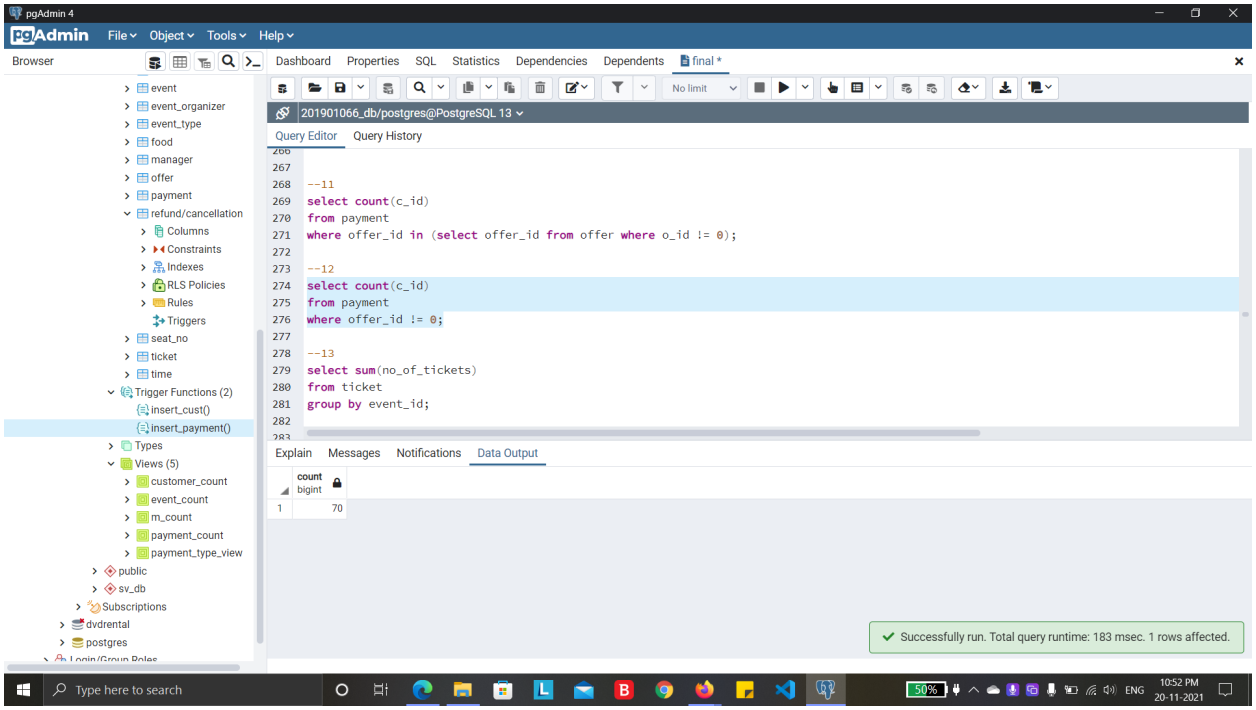
11. detail of all customers.

Select *
From customer;



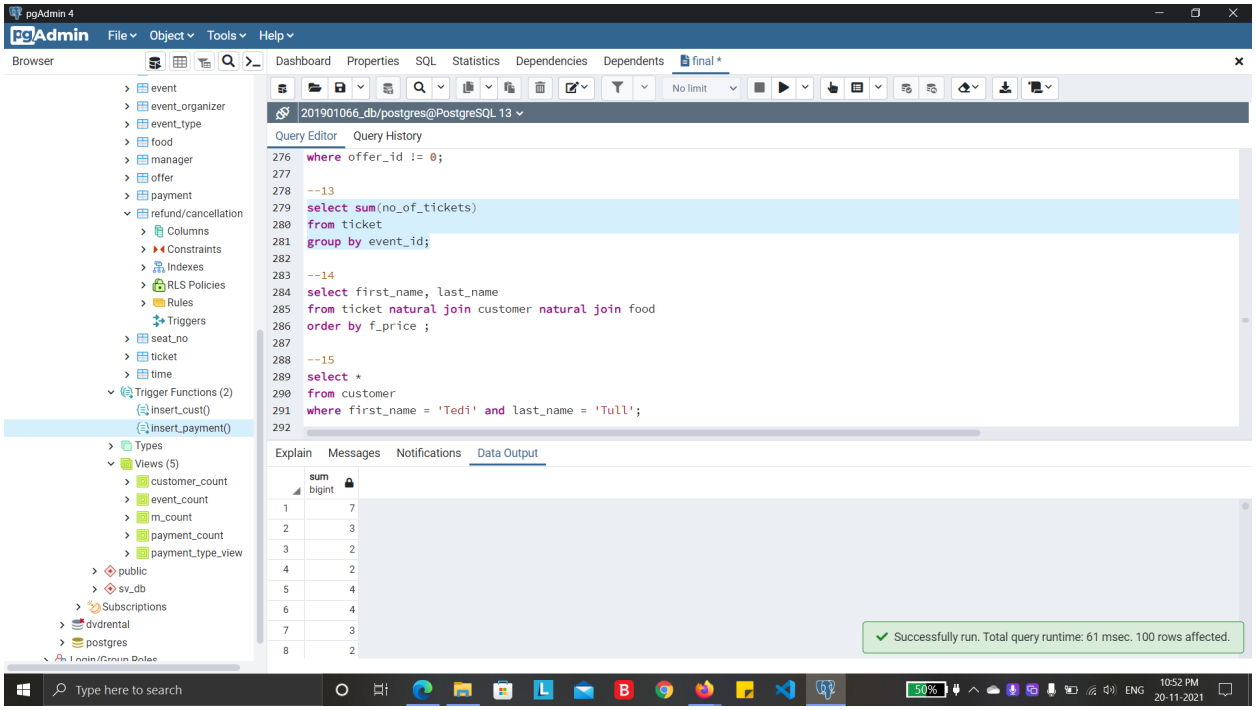
12. Number of customers who avail offers.

```
select count(c_id)
from payment
where offer_id != 0;
```



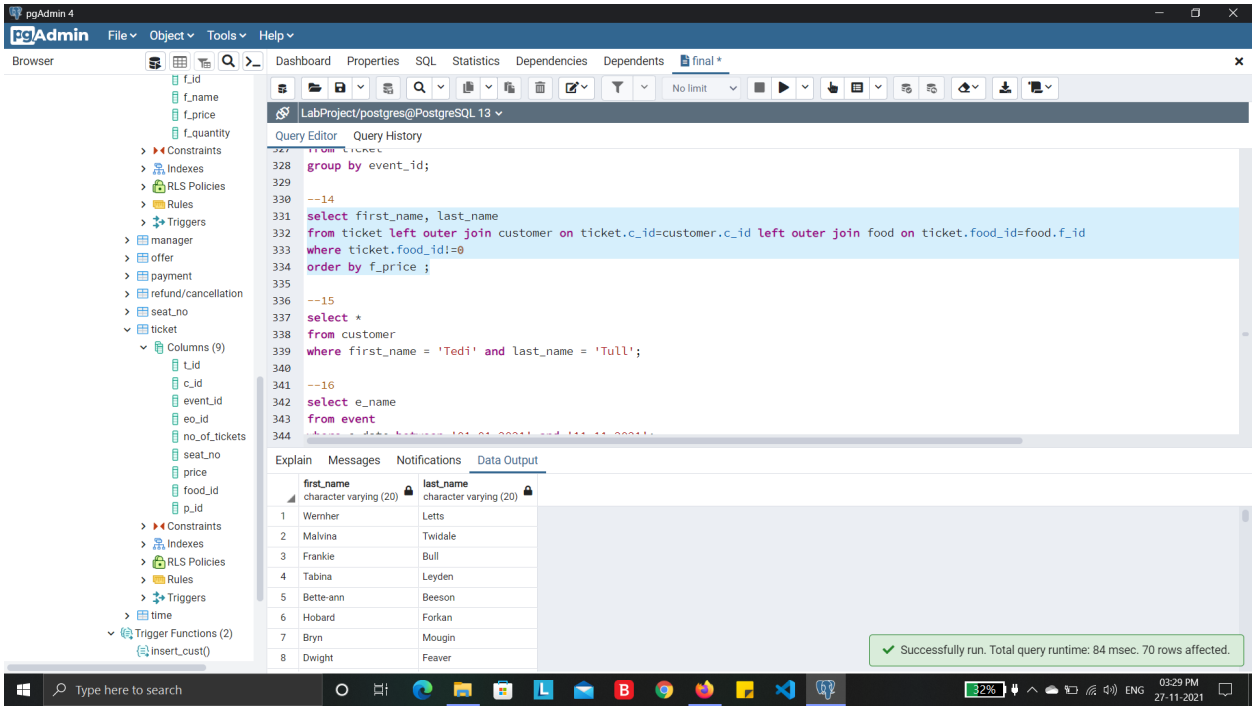
13. The total number of tickets for every event.

```
select sum(no_of_tickets)
from ticket
group by event_id;
```



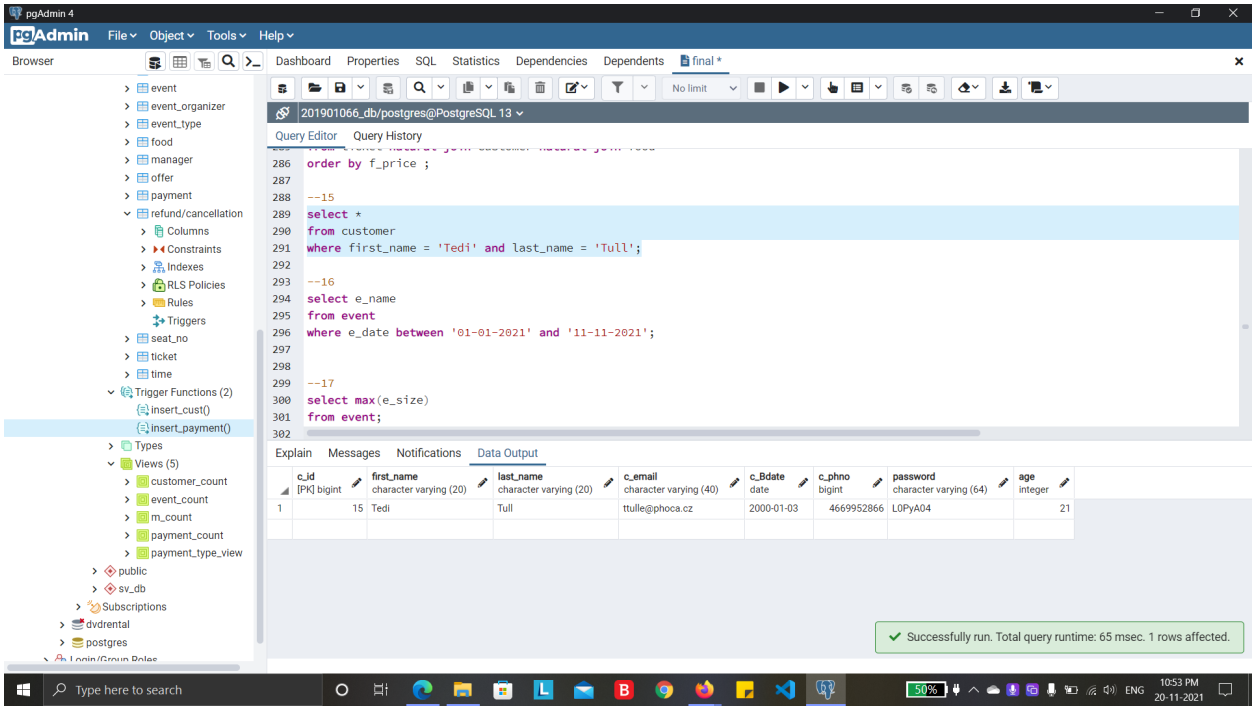
14. name of the customer who orders food,

select first_name, last_name
from ticket left outer join customer on ticket.c_id=customer.c_id left outer join food on ticket.food_id=food.f_id
where ticket.food_id!=0
order by f_price ;



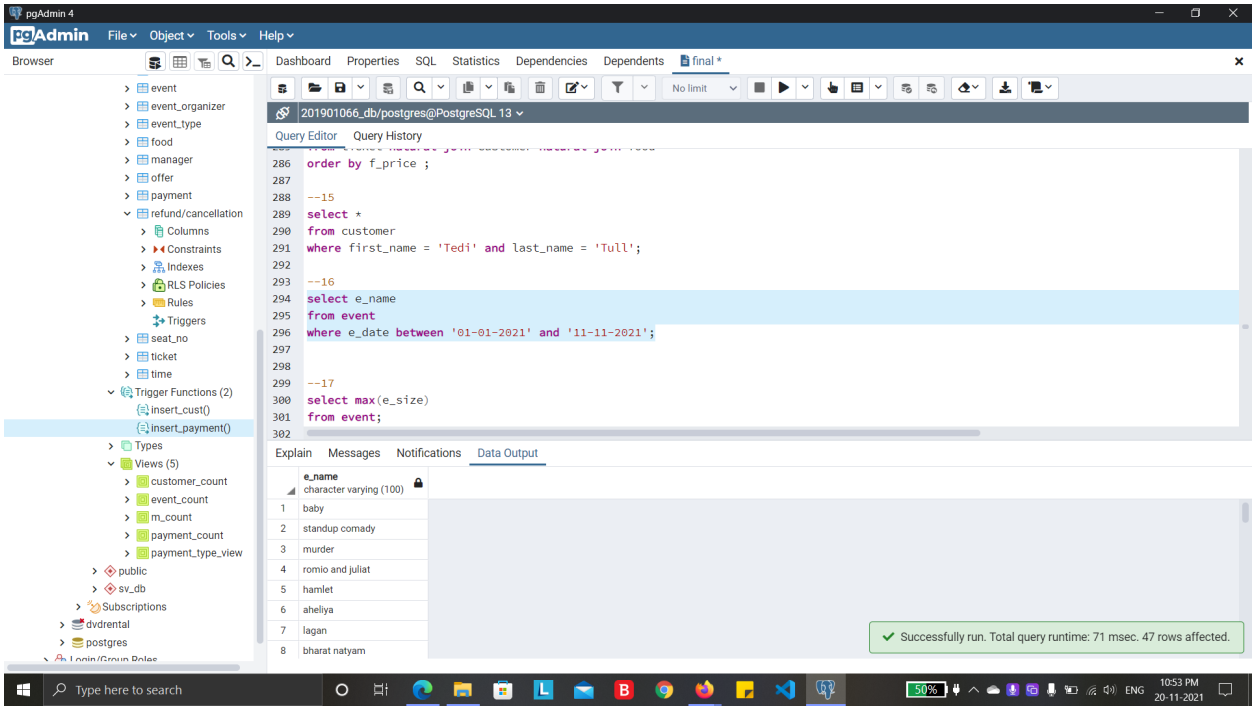
15. detail of customer whose first name is Tedi and last name is Tull.

```
select *
from customer
where first_name = 'Tedi' and last_name = 'Tull';
```



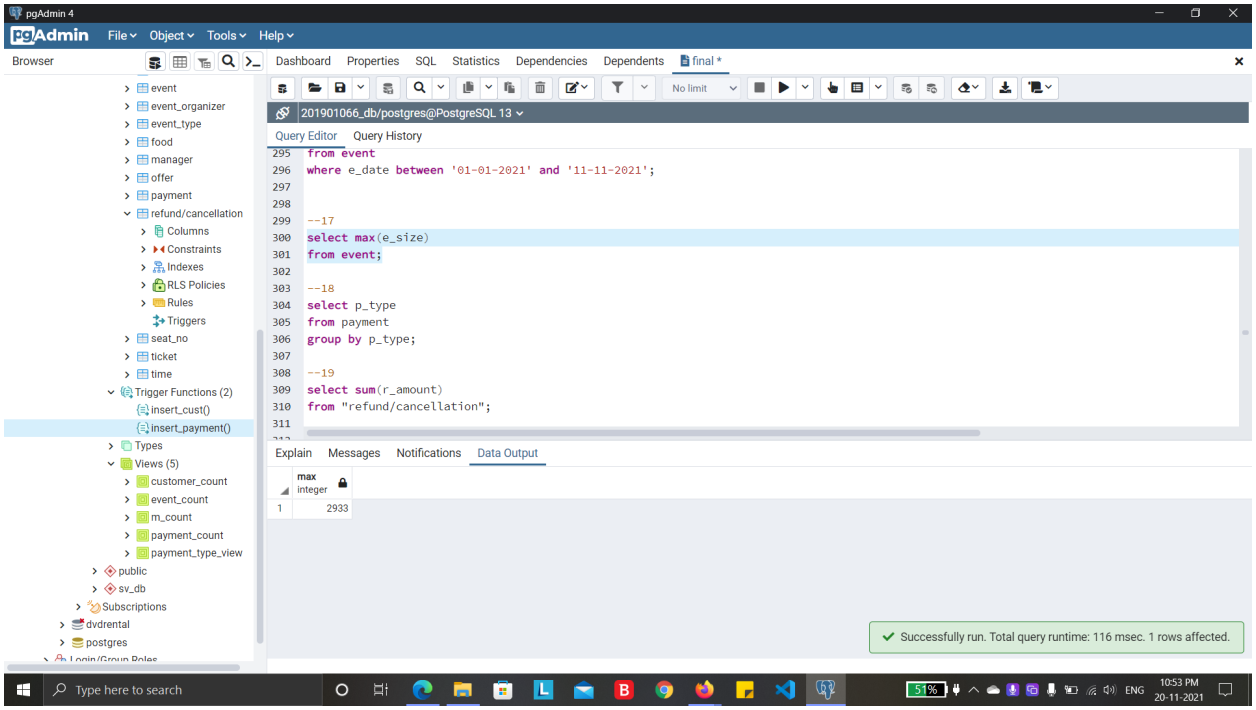
16. The event happens between 1-1-2021 to 11-11-2021.

```
select e_name
from event
where e_date between '01-01-2021' and '11-11-2021';
```



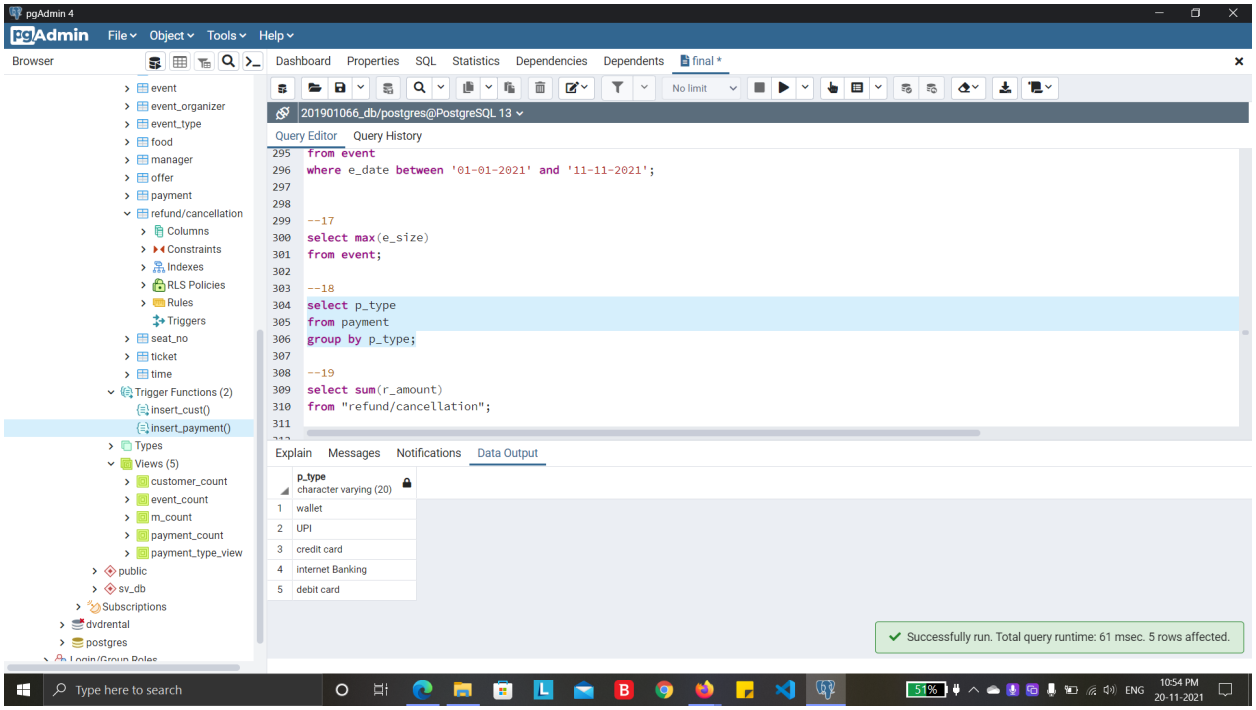
17. maximum number of capacity for the event,

select max(e_size)
from event;



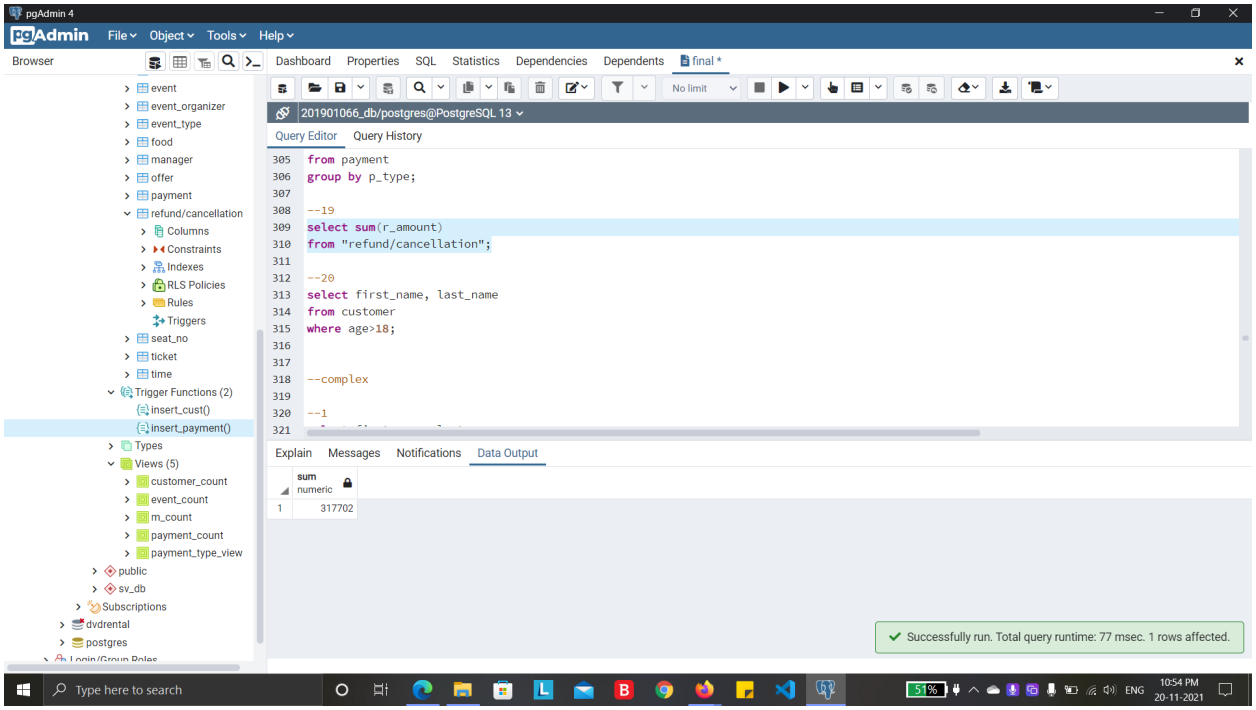
18. Total type of payment.

```
select p_type
from payment
group by p_type;
```



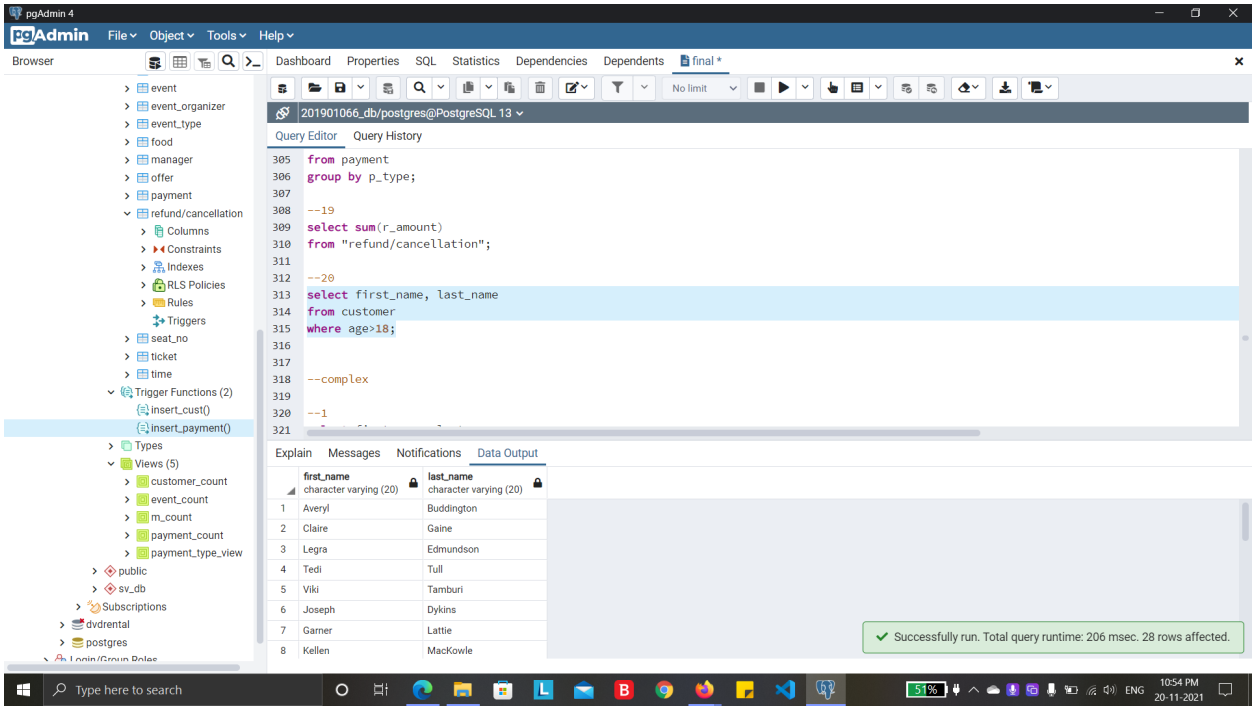
19. Sum of the total refunded amount.

```
select sum(r_amount)
from "refund/cancellation";
```



20. name of the customer whose age is greater than 18.

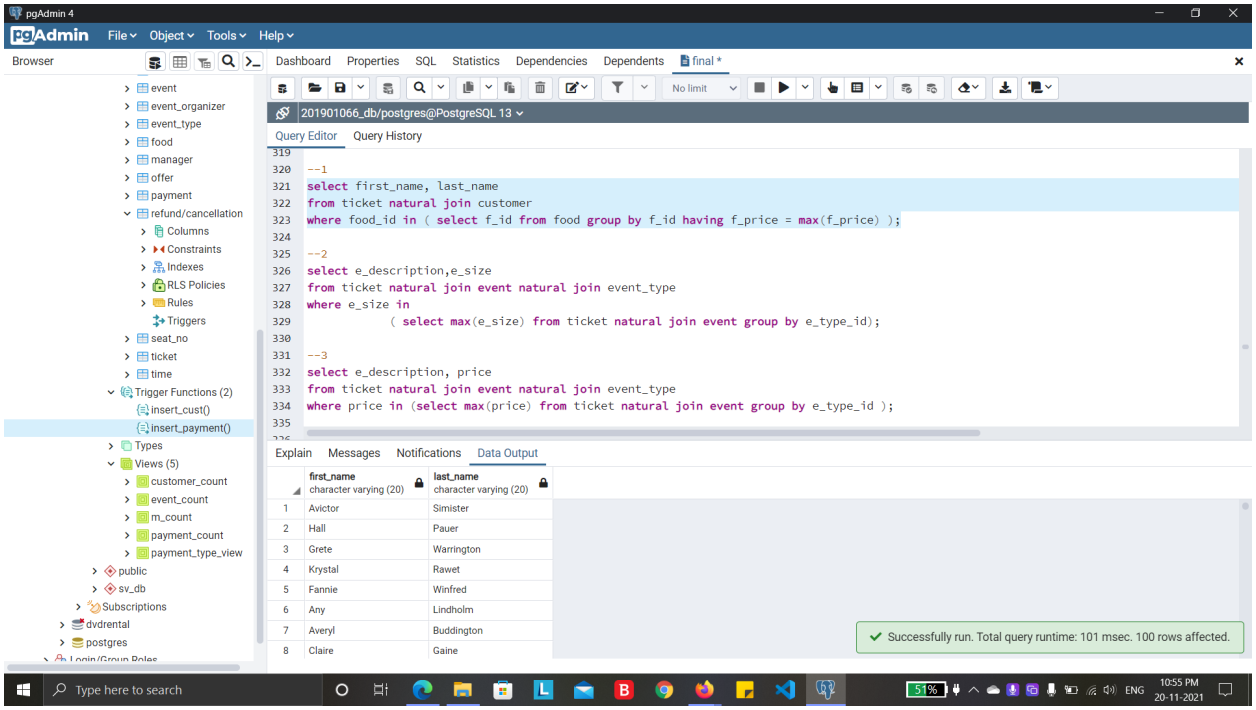
```
select first_name, last_name
from customer
where age>18;
```



Complex queries: -

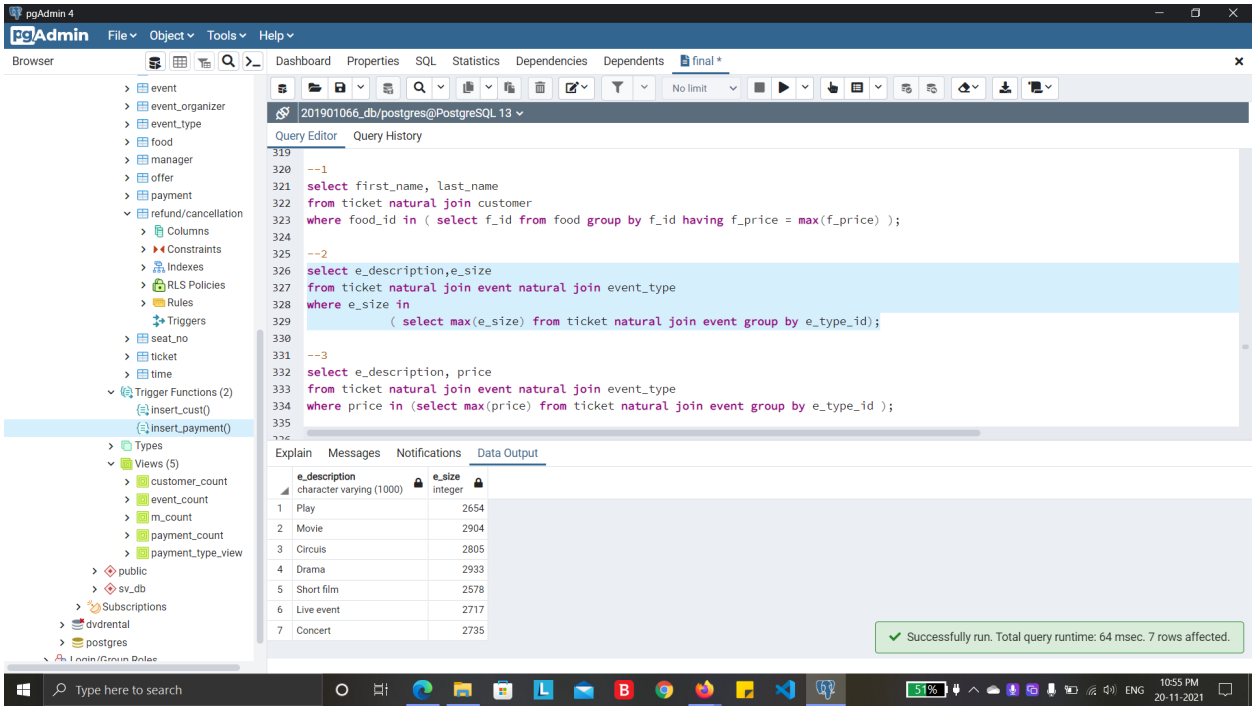
1. Name of customer who pays the maximum amount for food.

```
select first_name, last_name
from ticket natural join customer
where food_id in ( select f_id from food group by f_id having f_price = max(f_price) );
```



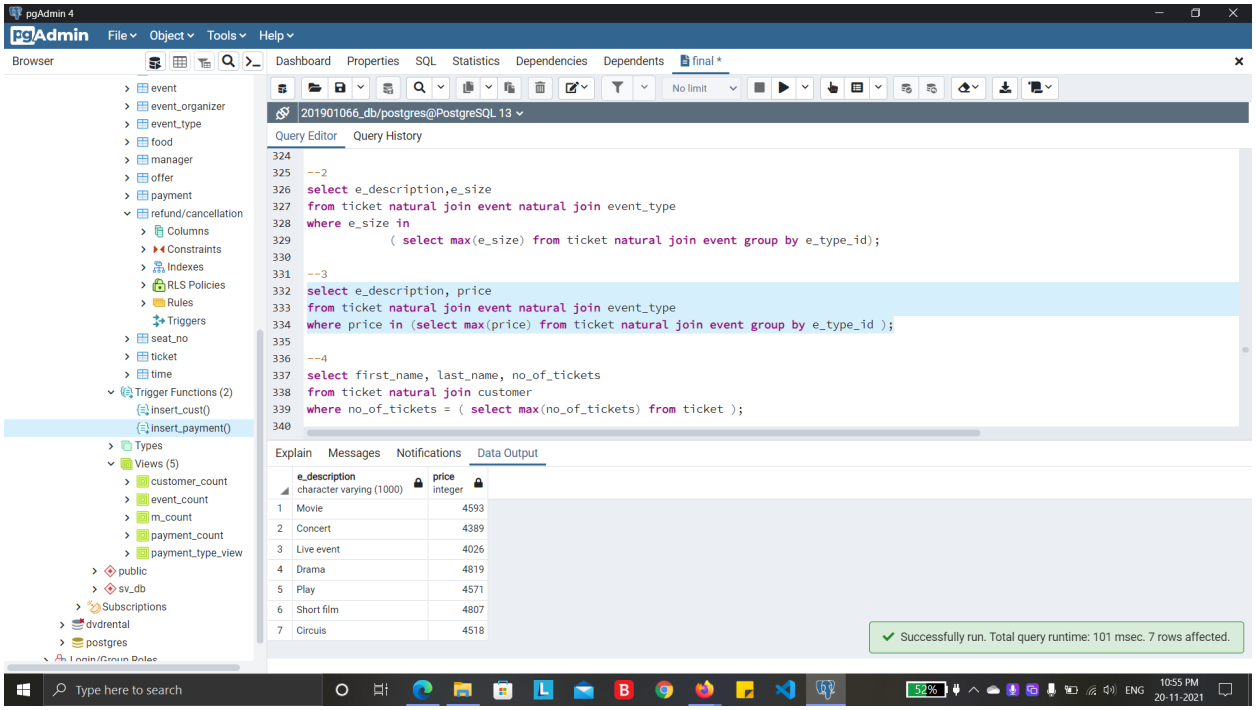
2. For event type, what is the maximum event size.

```
select e_description,e_size
from ticket natural join event natural join event_type
where e_size in
                ( select max(e_size) from ticket natural join event group by e_type_id);
```



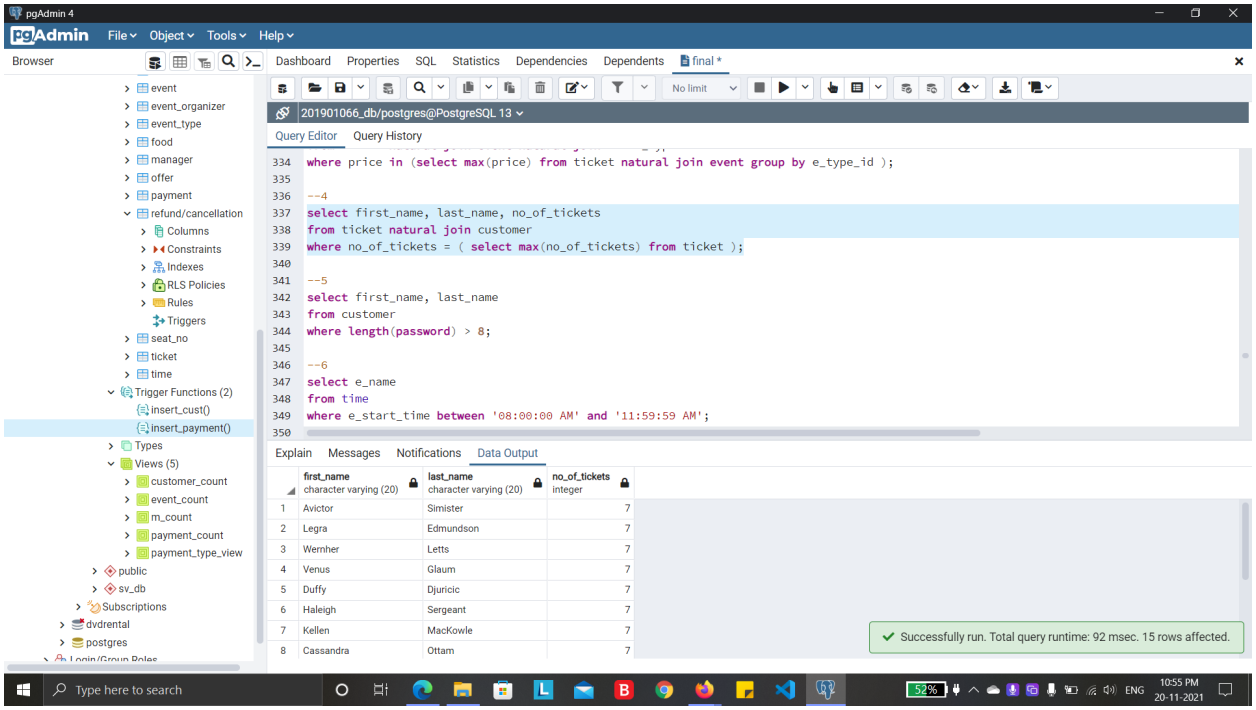
3. Maximum price of a ticket for every event type.

select e_description, price
from ticket natural join event natural join event_type
where price in (select max(price) from ticket natural join event group by e_type_id);



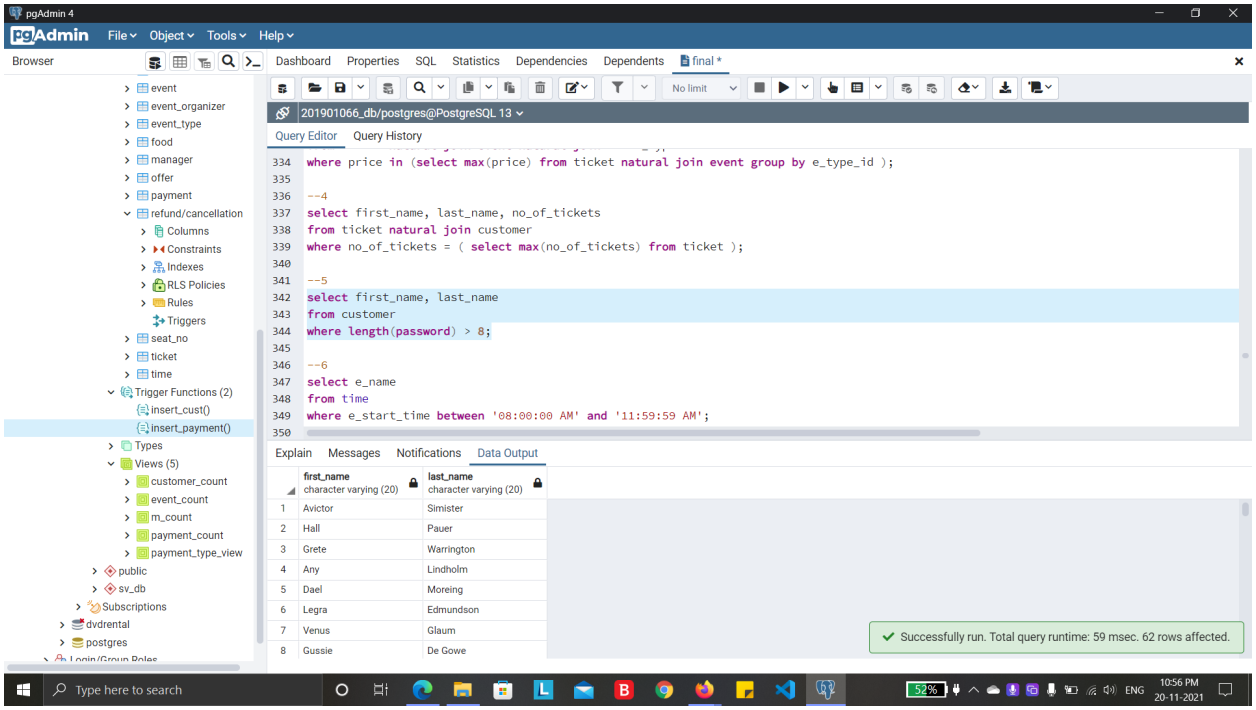
4. Name of customer who booked maximum number of tickets.

```
select first_name, last_name, no_of_tickets
from ticket natural join customer
where no_of_tickets = ( select max(no_of_tickets) from ticket );
```



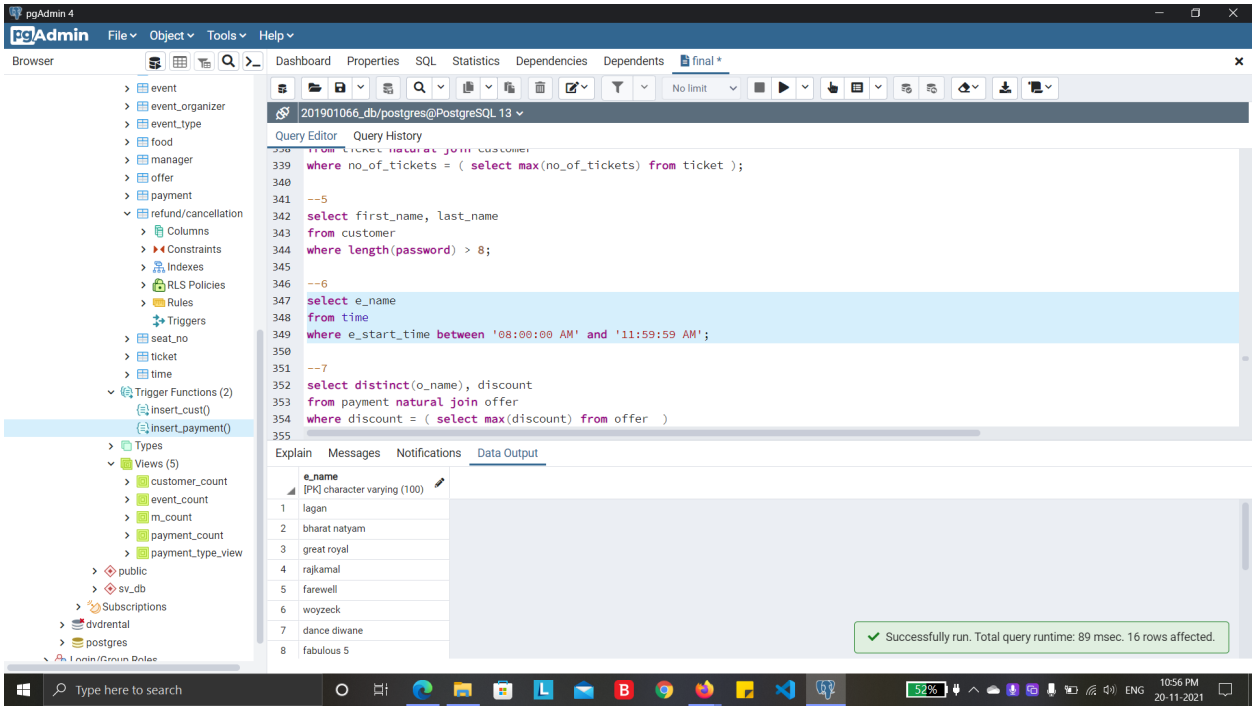
5. name of the customer whose password length is greater than 8.

```
select first_name, last_name
from customer
where length(password) > 8;
```



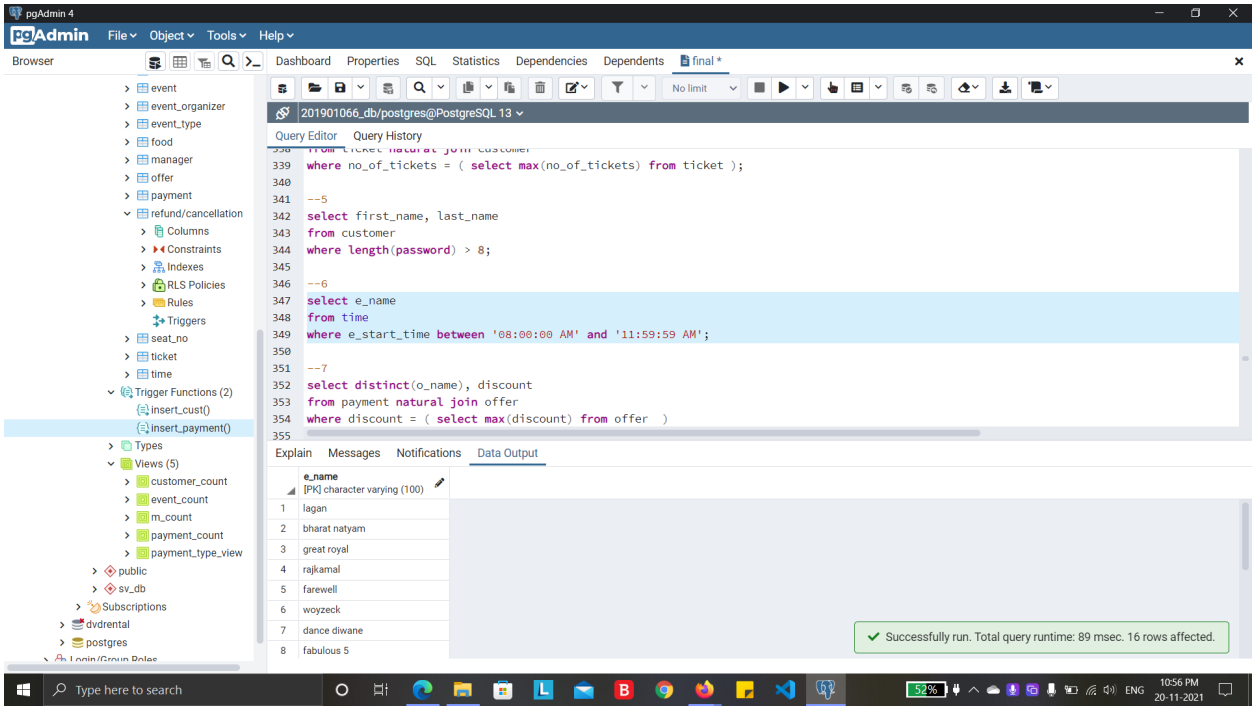
6. Name of event happens between 8:00 AM to 11:59 AM.

select e_name
from time
where e_start_time between '08:00:00 AM' and '11:59:59 AM';



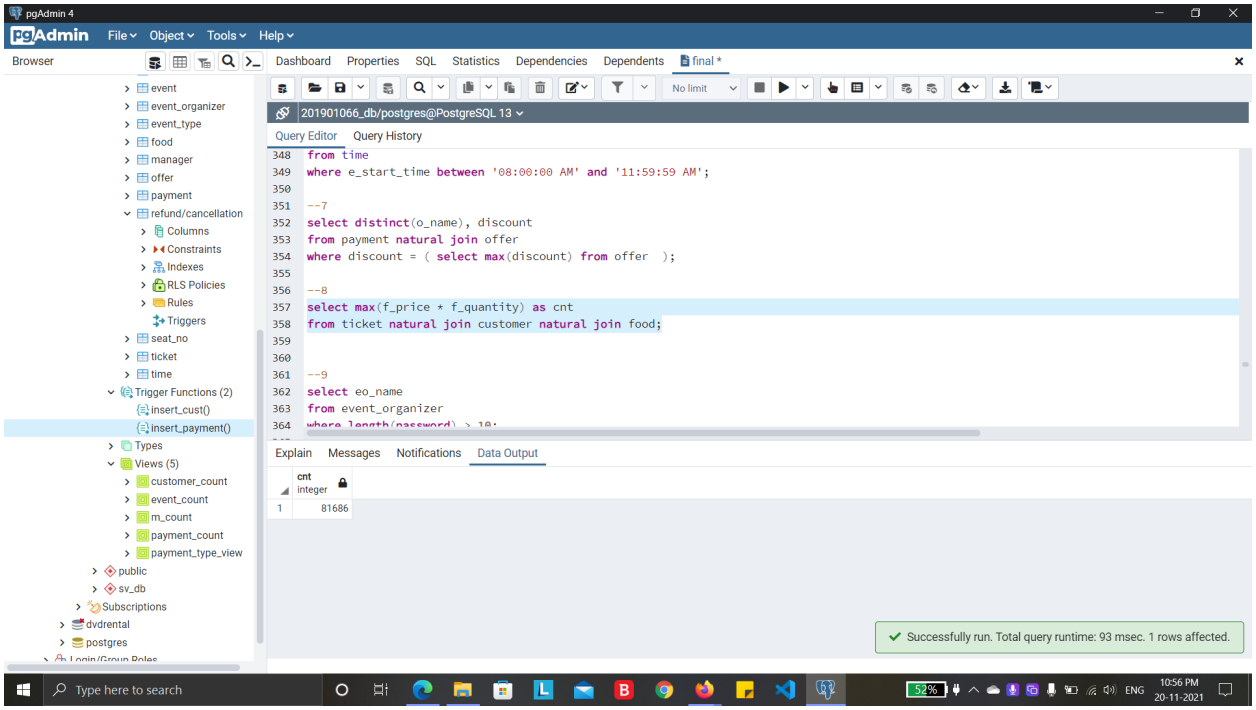
7. Name of offer which gives the maximum discount.

```
select distinct(o_name), discount
from payment natural join offer
where discount = ( select max(discount) from offer  );
```



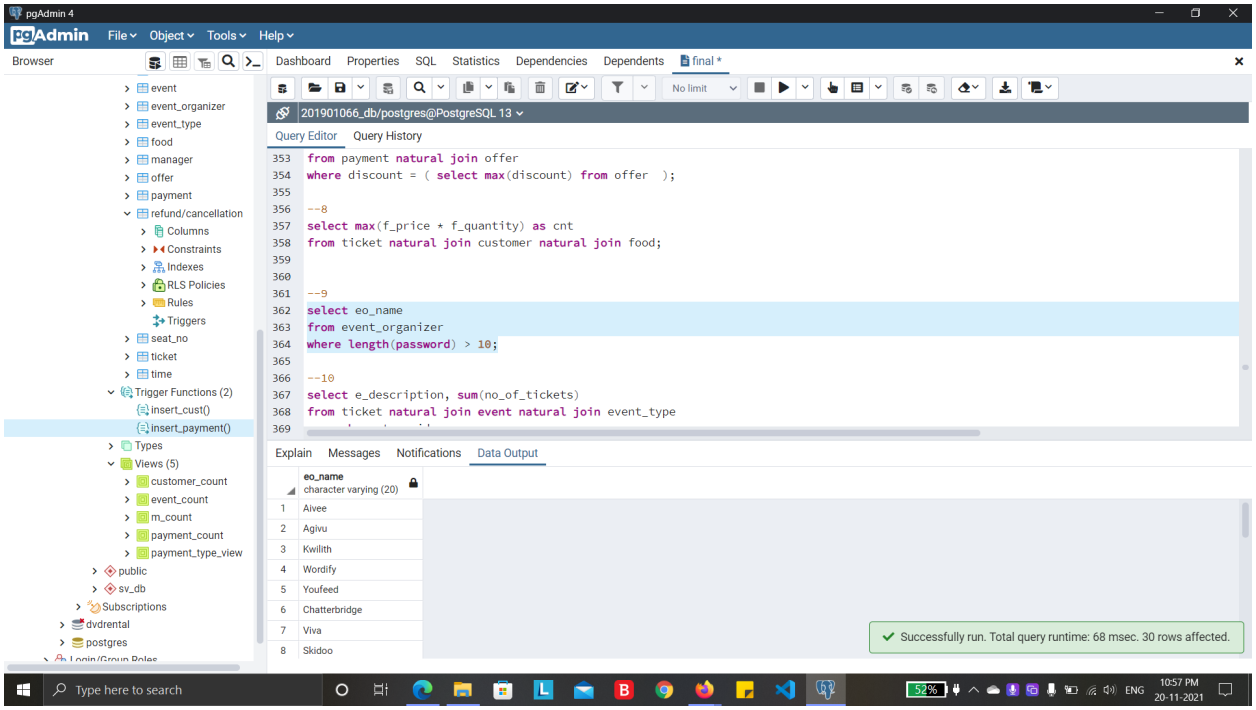
8. maximum amount for food by customer.

```
select max(f_price * f_quantity) as cnt
from ticket natural join customer natural join food;
```



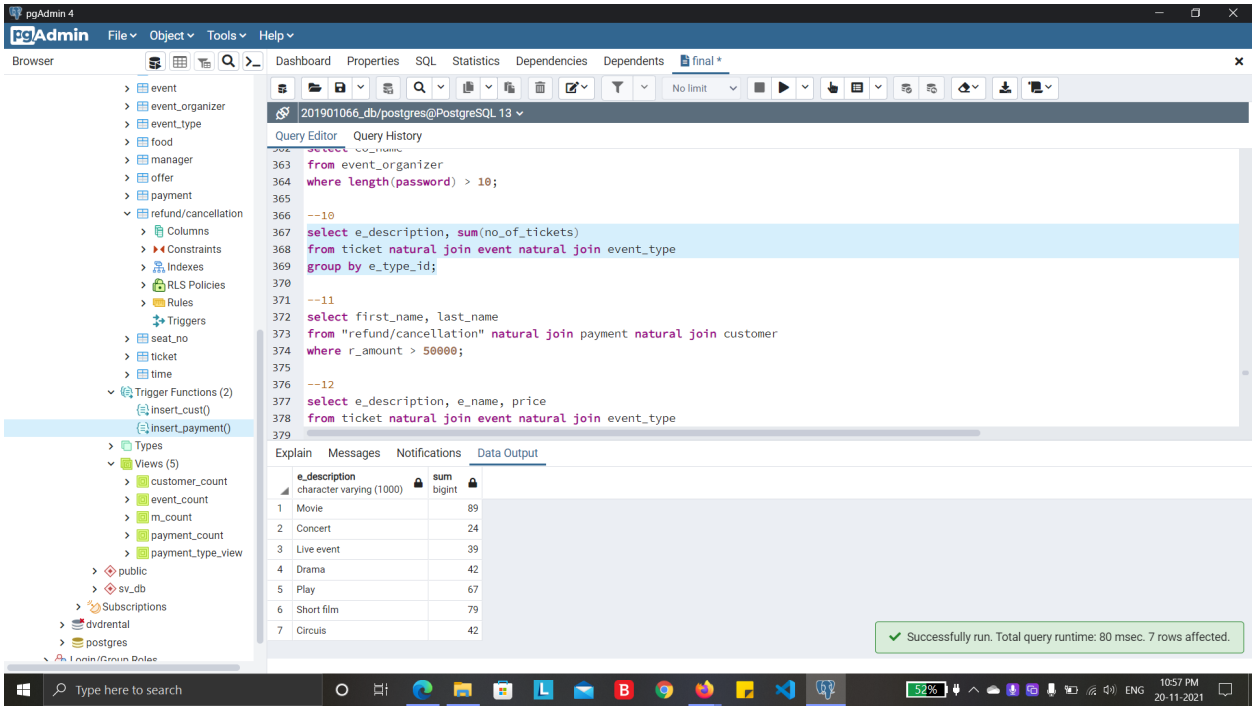
9. Name of event organize whose password length is greater than 10.

```
select eo_name
from event_organizer
where length(password) > 10;
```



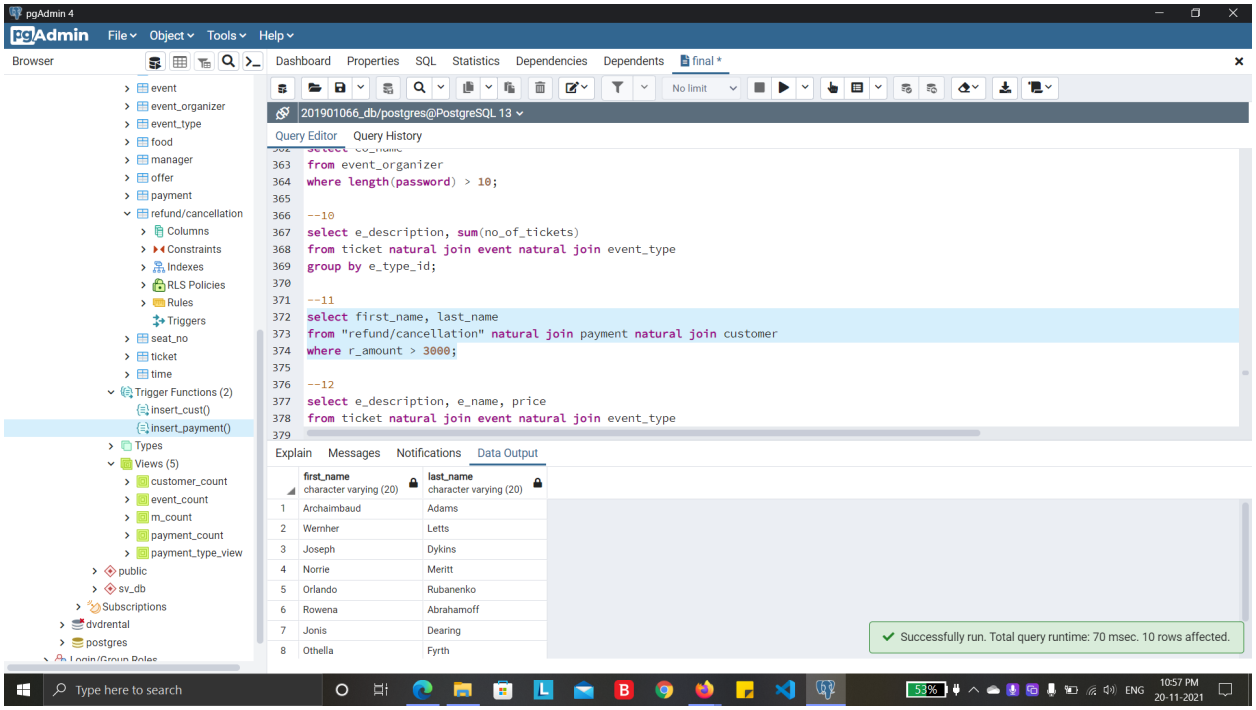
10. Total number of tickets for every event type.

```
select e_description, sum(no_of_tickets)
from ticket natural join event natural join event_type
group by e_type_id;
```



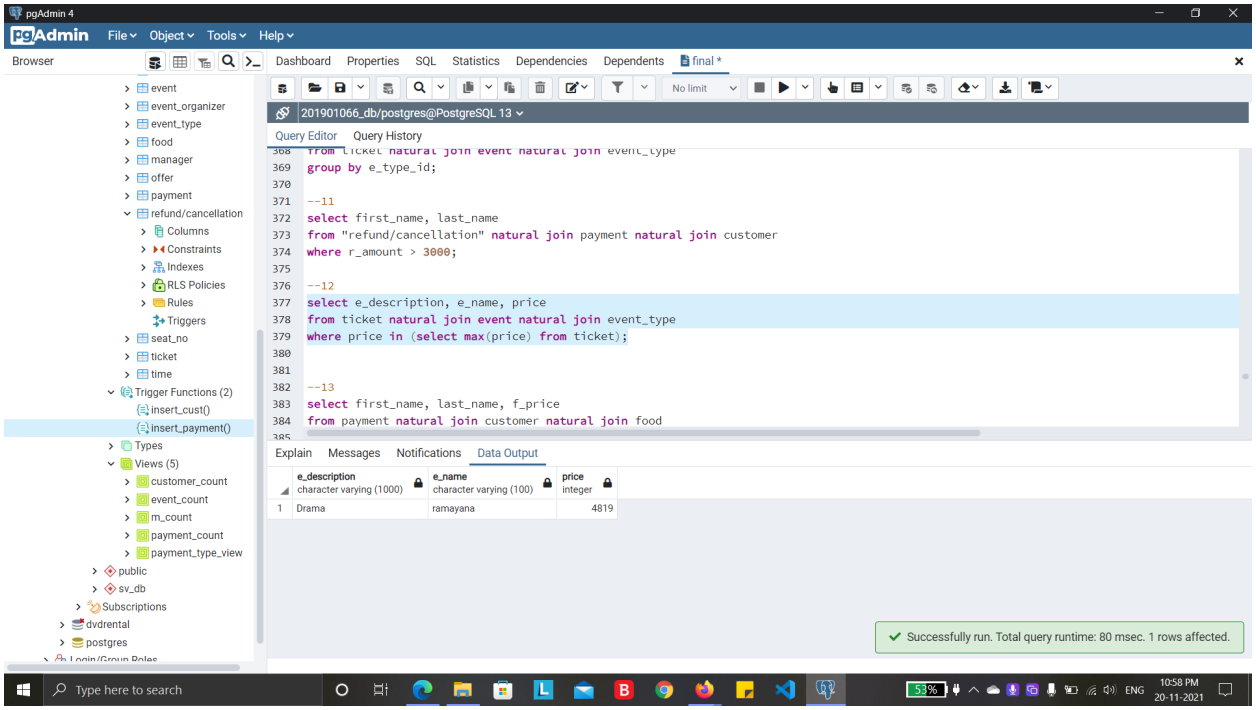
11. Name of customer who gets a refund.

```
select first_name, last_name
from "refund/cancellation" natural join payment natural join customer
where r_amount > 3000;
```



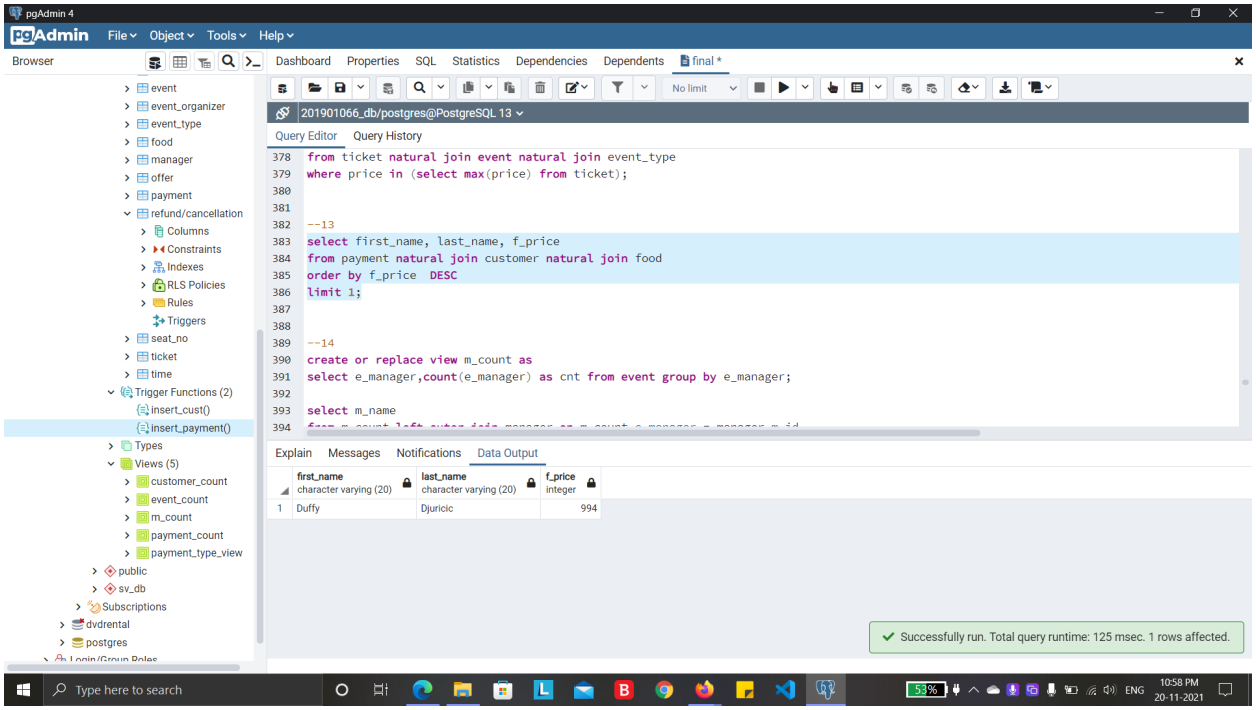
12. Event type event name with the highest ticket price.

select e_description, e_name, price
from ticket natural join event natural join event_type
where price in (select max(price) from ticket);



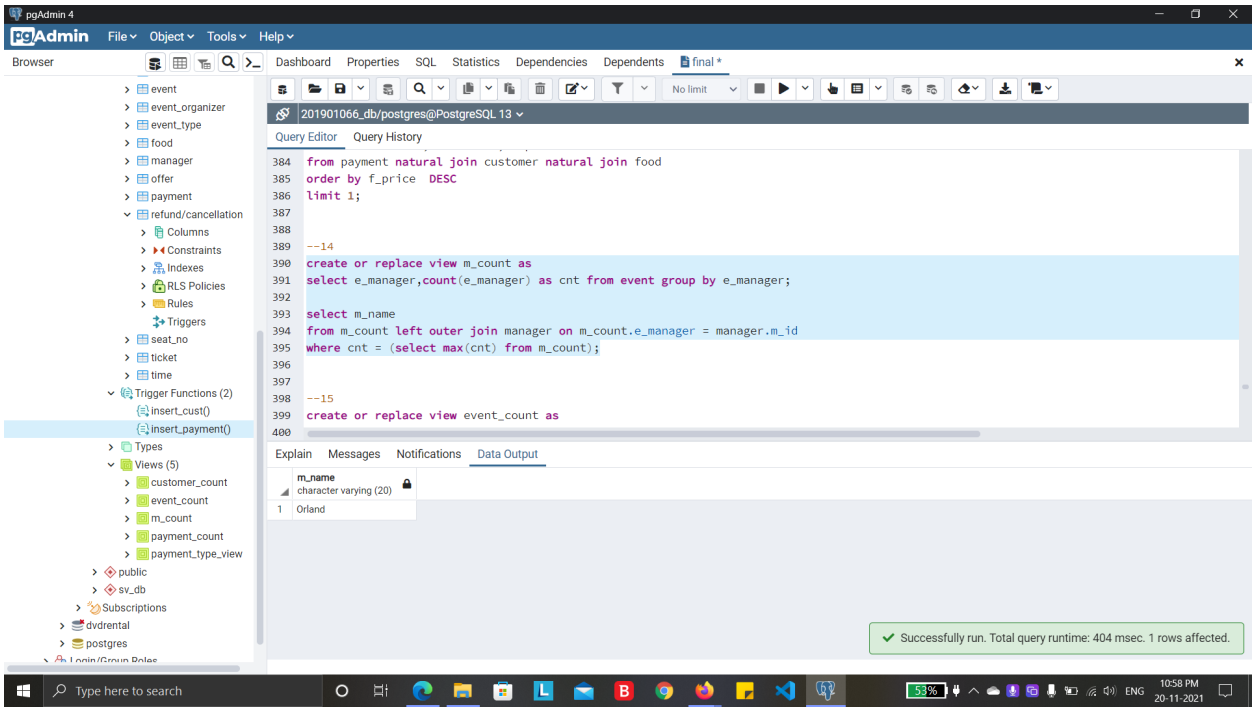
13. customer who pay highest for food.

```
select first_name, last_name, f_price
from payment natural join customer natural join food
order by f_price  DESC
limit 1;
```



14. name of the manager who handles a most number of events.

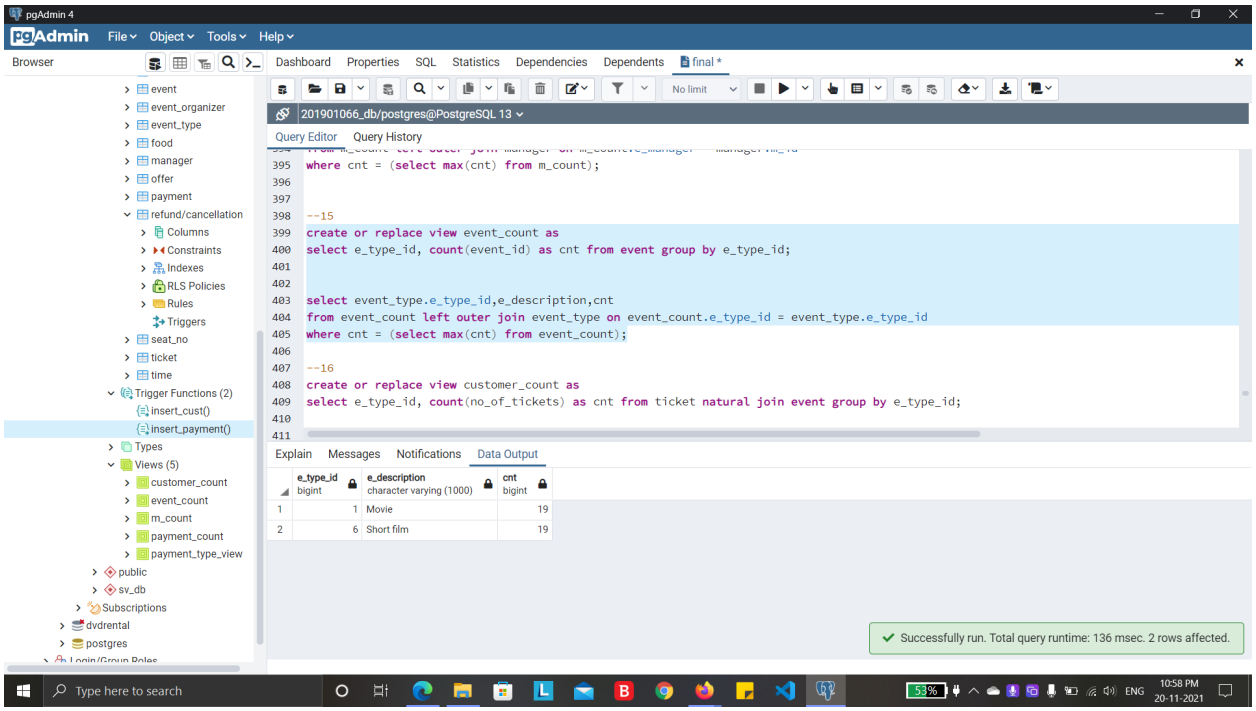
```
create or replace view m_count as
select e_manager,count(e_manager) as cnt from event group by e_manager;

select m_name
from m_count left outer join manager on m_count.e_manager = manager.m_id
where cnt = (select max(cnt) from m_count);
```



15. Total number of events for every event type.

create or replace view event_count as
select e_type_id, count(event_id) as cnt from event group by e_type_id;

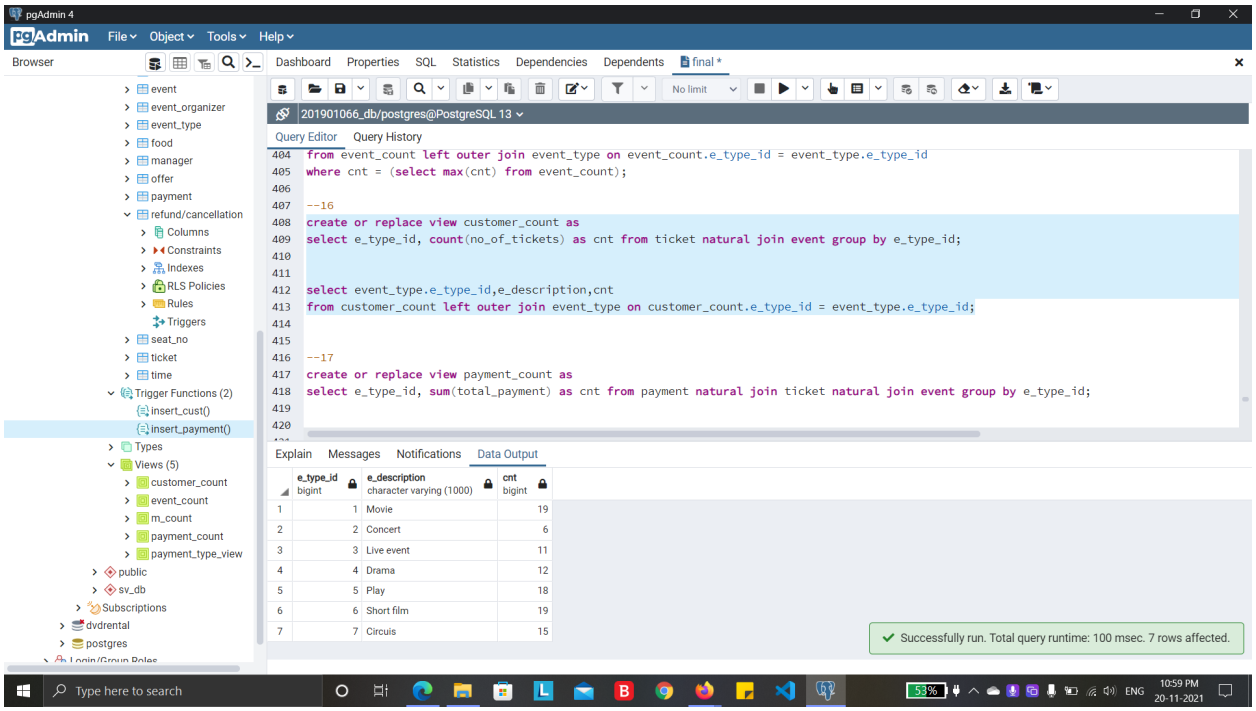
select event_type.e_type_id,e_description,cnt
from event_count left outer join event_type on event_count.e_type_id = event_type.e_type_id
where cnt = (select max(cnt) from event_count);



16. Total number of tickets for every event type.

create or replace view customer_count as
select e_type_id, count(no_of_tickets) as cnt from ticket natural join event group by e_type_id;

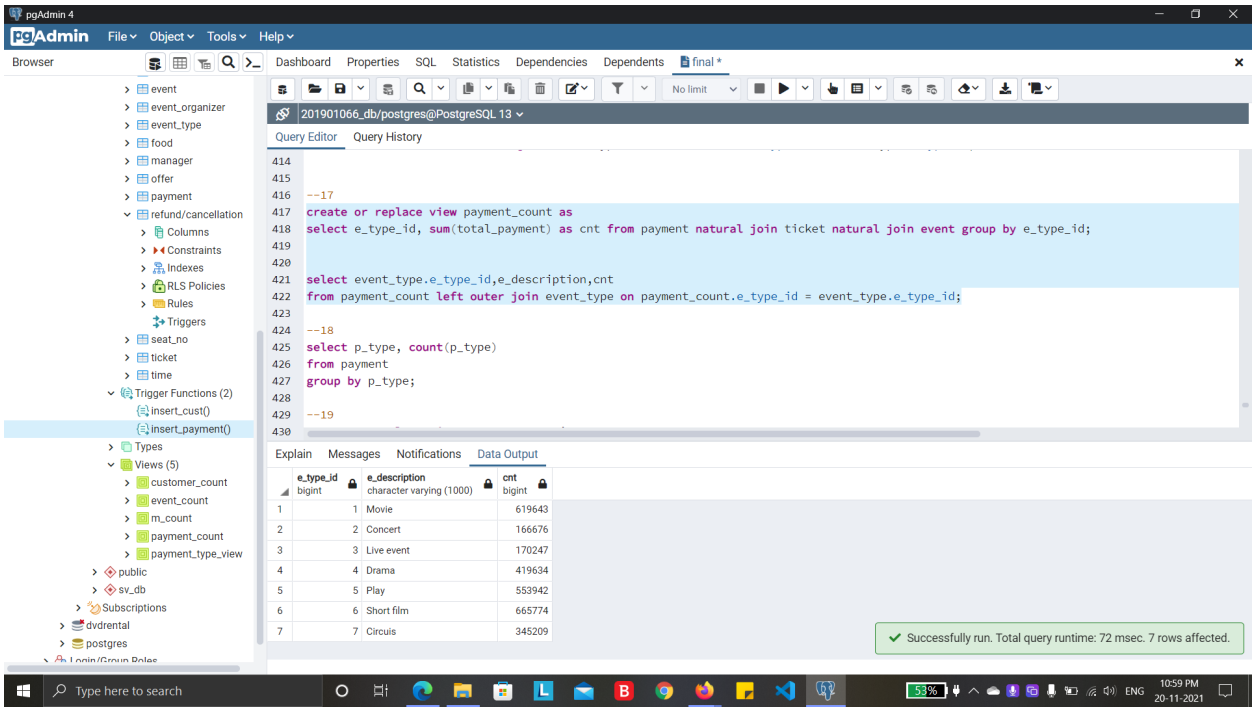
select event_type.e_type_id,e_description,cnt
from customer_count left outer join event_type on customer_count.e_type_id = event_type.e_type_id;



17. total amount received for every event type.

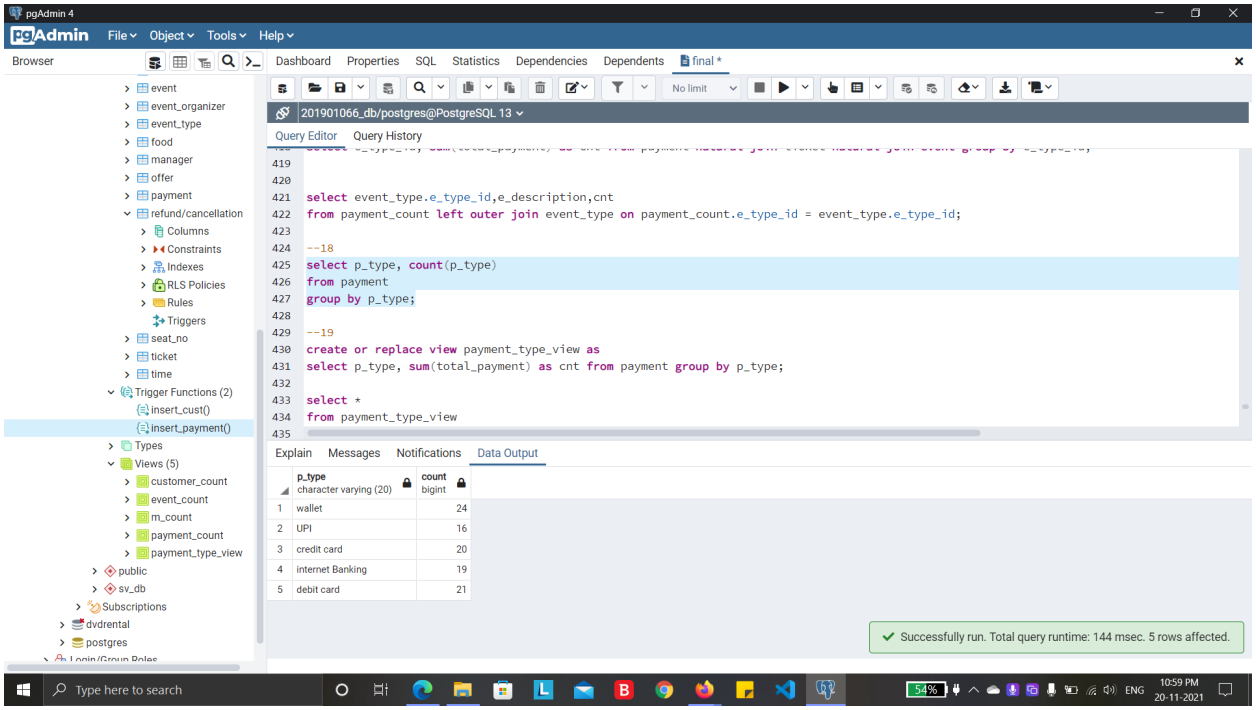
create or replace view payment_count as
select e_type_id, sum(total_payment) as cnt from payment natural join ticket natural join event group by e_type_id;

select event_type.e_type_id,e_description,cnt
from payment_count left outer join event_type on payment_count.e_type_id = event_type.e_type_id;



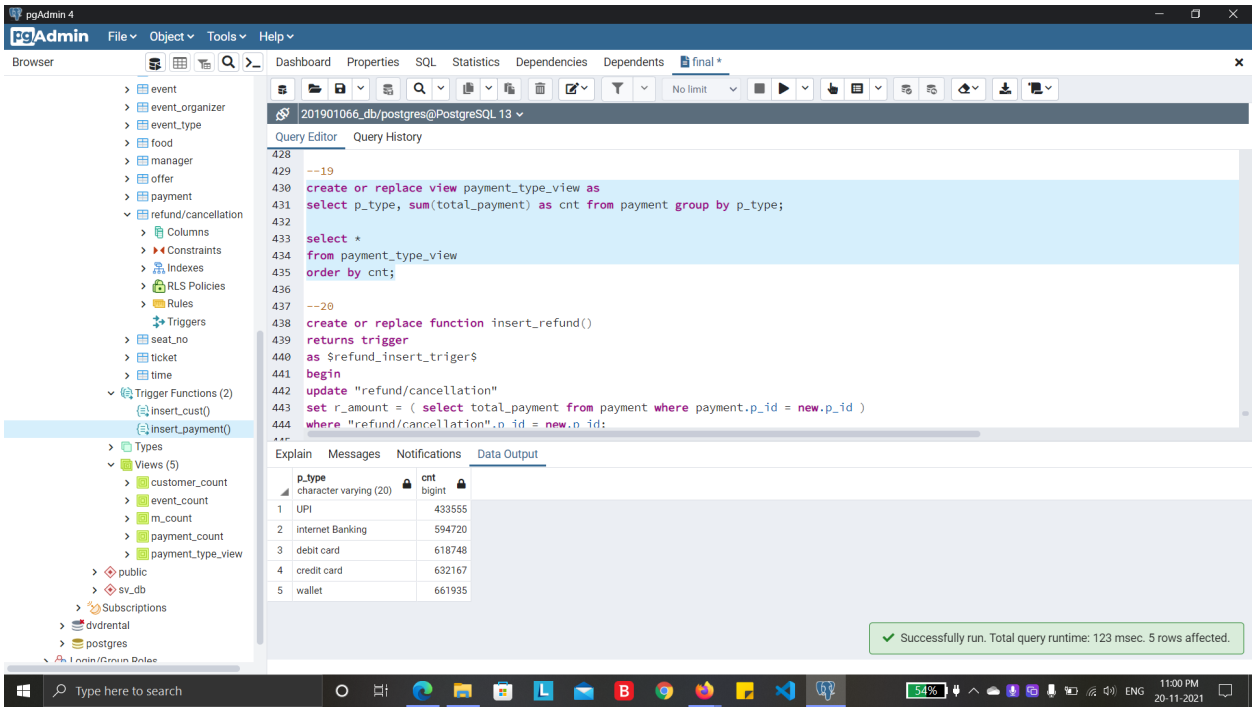
18. different payment types useed by customers.

```
select p_type, count(p_type)
from payment
group by p_type;
```



19. total amount received by every payment method.

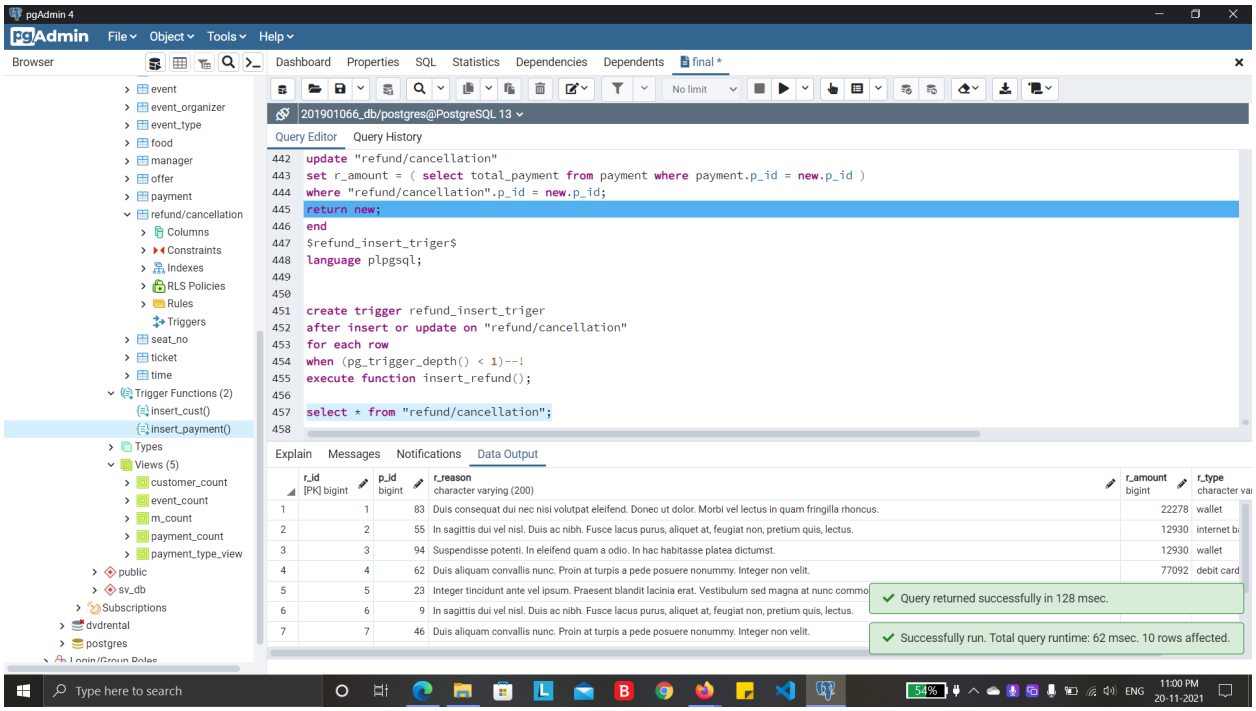
```
create or replace view payment_type_view as
select p_type, sum(total_payment) as cnt from payment group by p_type;

select *
from payment_type_view
order by cnt;
```



20. create a trigger for getting refund amount from customer payment details.

```
create or replace function insert_refund()
returns trigger
as $refund_insert_triger$
begin
update "refund/cancellation"
set r_amount = ( select total_payment from payment where payment.p_id = new.p_id )
where "refund/cancellation".p_id = new.p_id;
return new;
end
$refund_insert_triger$
language plpgsql;
```

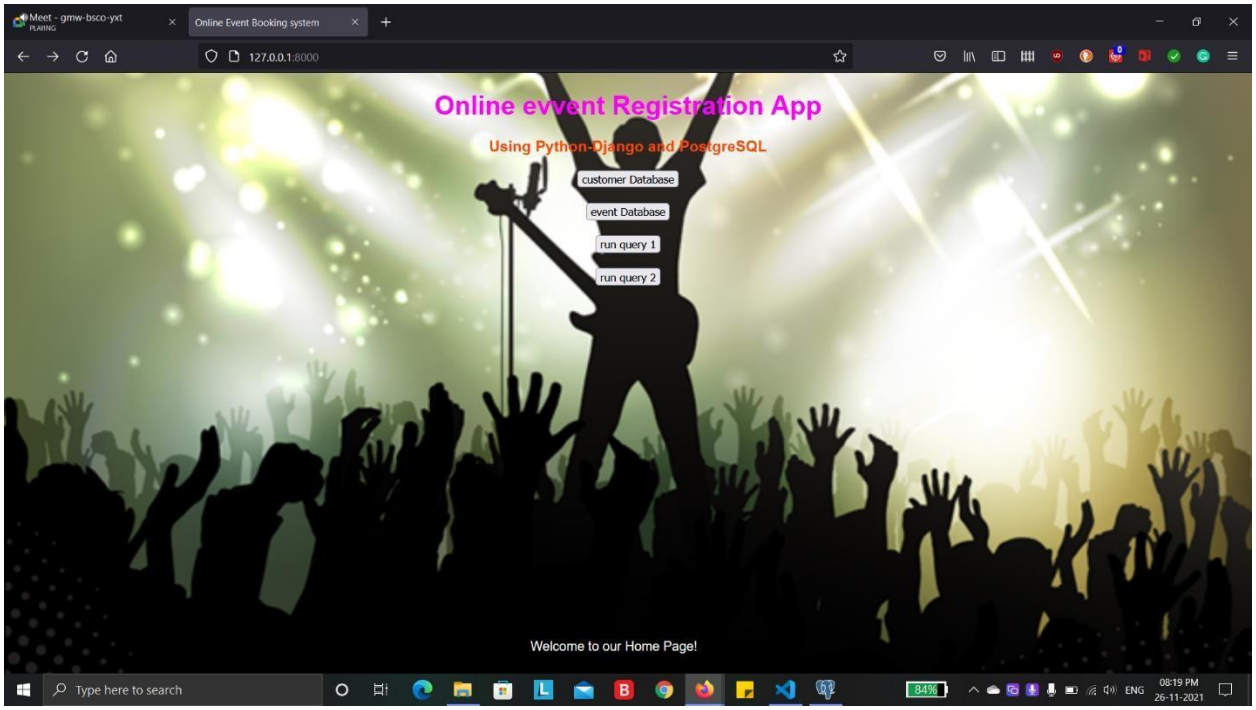
```
create trigger refund_insert_triger
after insert or update on "refund/cancellation"
for each row
when (pg_trigger_depth() < 1)--!
execute function insert_refund();

select * from "refund/cancellation";
```



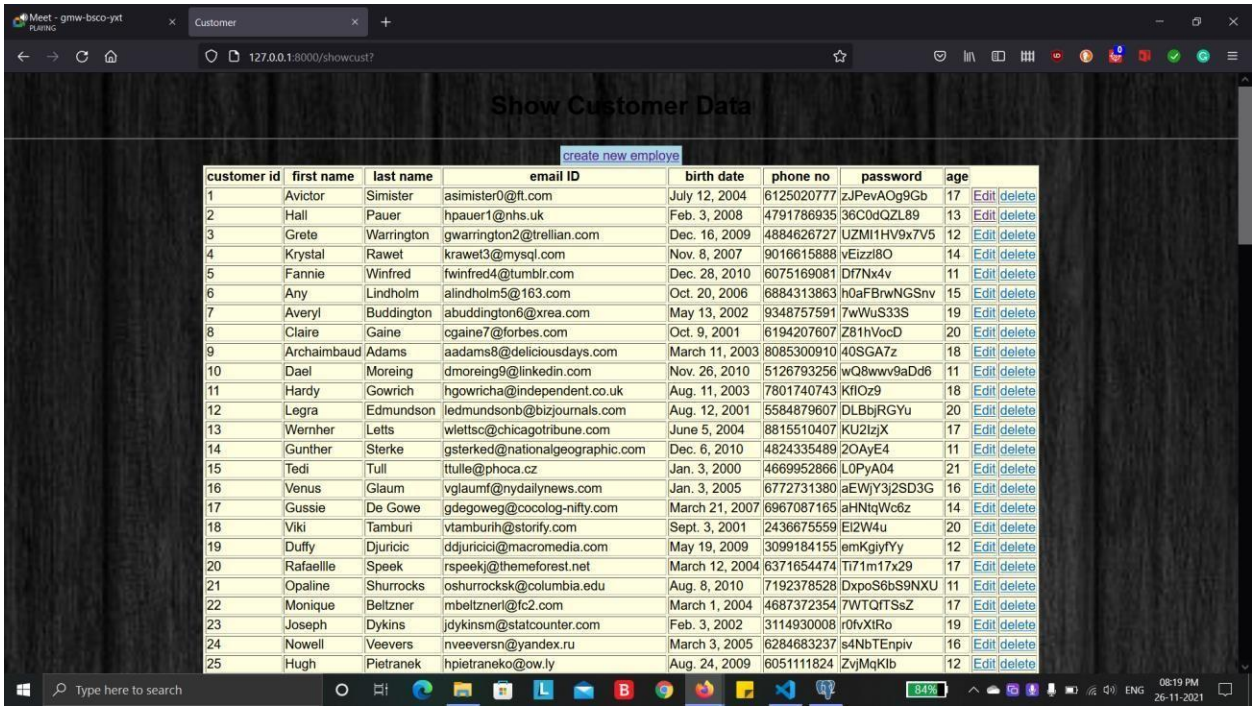
Section 7: Project Code with output screenshots

Screenshots of Website for online event registration:

(1) Homepage

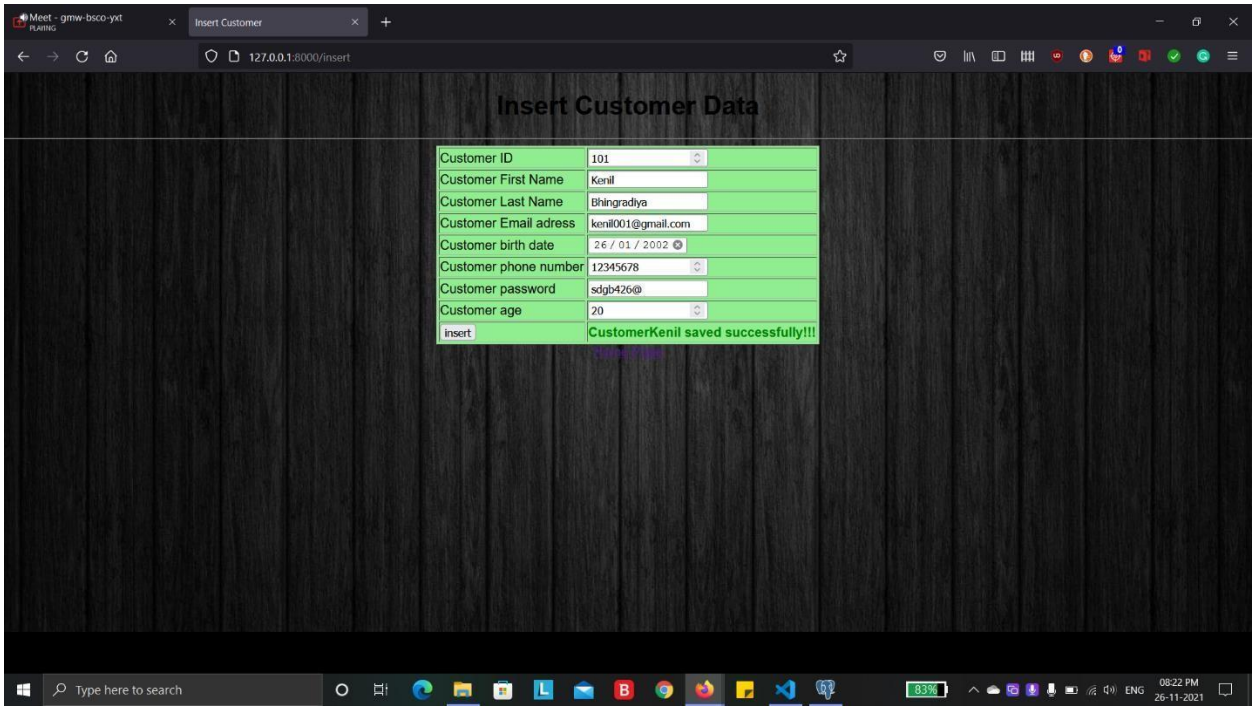


(2) Customer Database



customer id	first name	last name	email ID	birth date	phone no	password	age	
1	Avictor	Simister	asimister0@ft.com	July 12, 2004	6125020777	zJPevAOg9Gb	17	Edit delete
2	Hall	Pauer	hpauer1@nhs.uk	Feb. 3, 2008	4791786935	36C0dQZL89	13	Edit delete
3	Grete	Warrington	gwarrington2@trellian.com	Dec. 16, 2009	4884626727	UZMI1HV9x7V5	12	Edit delete
4	Krystal	Rawet	krawet3@mysql.com	Nov. 8, 2007	9016615888	vEizzl8O	14	Edit delete
5	Fannie	Winfred	fwinfred4@tumblr.com	Dec. 28, 2010	6075169081	Df7Nx4v	11	Edit delete
6	Any	Lindholm	alindholm5@163.com	Oct. 20, 2006	6884313863	h0aFBrwNGSnv	15	Edit delete
7	Averyl	Buddington	abuddington6@xrea.com	May 13, 2002	9348757591	7wWuS33S	19	Edit delete
8	Claire	Gaine	cgaine7@forbes.com	Oct. 9, 2001	6194207607	Z81hVocD	20	Edit delete
9	Archaimbaud	Adams	aadams8@deliciousdays.com	March 11, 2003	8085300910	40SGA7z	18	Edit delete
10	Dael	Moreing	dmoreing9@linkedin.com	Nov. 26, 2010	5126793256	wQ8wwv9aDd6	11	Edit delete
11	Hardy	Gowrich	hgowricha@independent.co.uk	Aug. 11, 2003	7801740743	KfIOz9	18	Edit delete
12	Legra	Edmundson	ledmundsonb@bizjournals.com	Aug. 12, 2001	5584879607	DLBbjRGYu	20	Edit delete
13	Wernher	Letts	wlettsc@chicagotribune.com	June 5, 2004	8815510407	KU2IzjX	17	Edit delete
14	Gunther	Sterke	gsterked@nationalgeographic.com	Dec. 6, 2010	4824335489	2OAyE4	11	Edit delete
15	Tedi	Tull	ttulle@phoca.cz	Jan. 3, 2000	4669952866	L0PyA04	21	Edit delete
16	Venus	Glaum	vglaumf@nydailynews.com	Jan. 3, 2005	6772731380	aEWjY3j2SD3G	16	Edit delete
17	Gussie	De Gowe	gdegoweg@cocolog-nifty.com	March 21, 2007	6967087165	aHNtqWc6z	14	Edit delete
18	Viki	Tamburi	vtamburih@storify.com	Sept. 3, 2001	2436675559	EI2W4u	20	Edit delete
19	Duffy	Djuricic	ddjuricici@macromedia.com	May 19, 2009	3099184155	emKgiyfYy	12	Edit delete
20	Rafaellle	Speek	rspeekj@themeforest.net	March 12, 2004	6371654474	Ti71m17x29	17	Edit delete
21	Opaline	Shurrocks	oshurrocksk@columbia.edu	Aug. 8, 2010	7192378528	DxpoS6bS9NXU	11	Edit delete
22	Monique	Beltzner	mbeltznerl@fc2.com	March 1, 2004	4687372354	7WTQfTSsZ	17	Edit delete
23	Joseph	Dykins	jdykinsm@statcounter.com	Feb. 3, 2002	3114930008	r0fvXtRo	19	Edit delete
24	Nowell	Veevers	nveeversn@yandex.ru	March 3, 2005	6284683237	s4NbTEnpiv	16	Edit delete
25	Hugh	Pietranek	hpietraneko@ow.ly	Aug. 24, 2009	6051111824	ZvjMqKlb	12	Edit delete

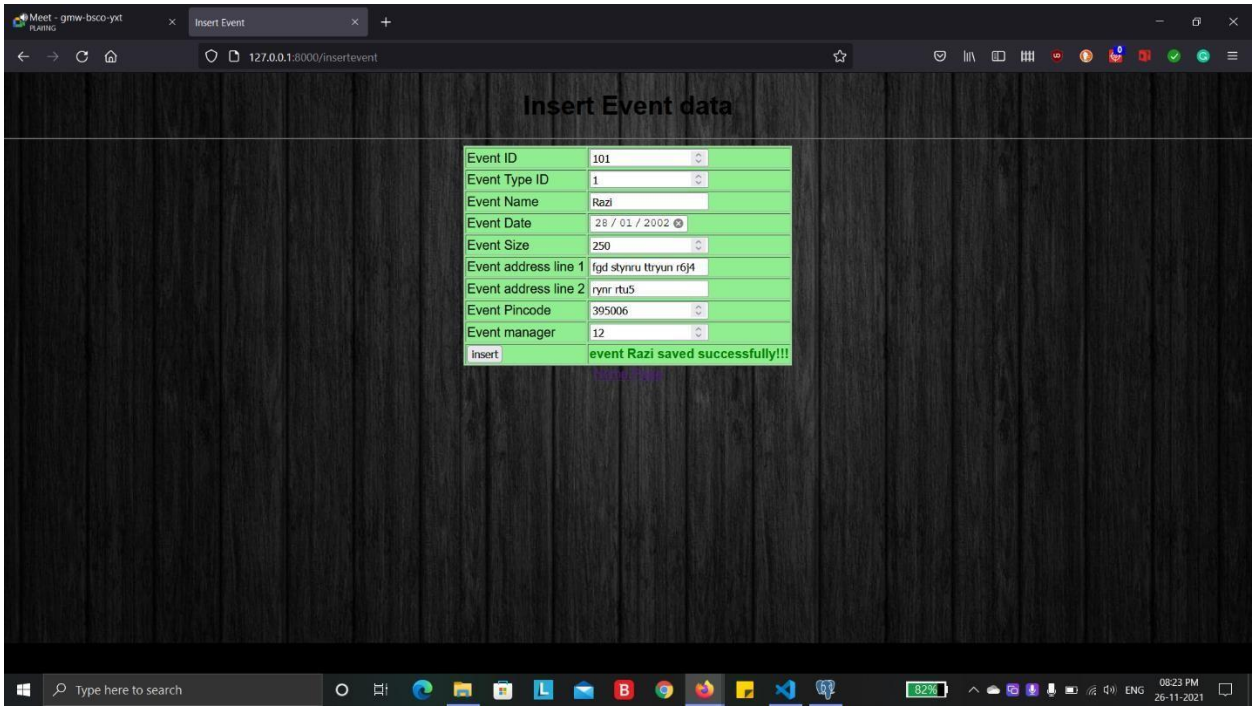
(3) Insert in the customer database



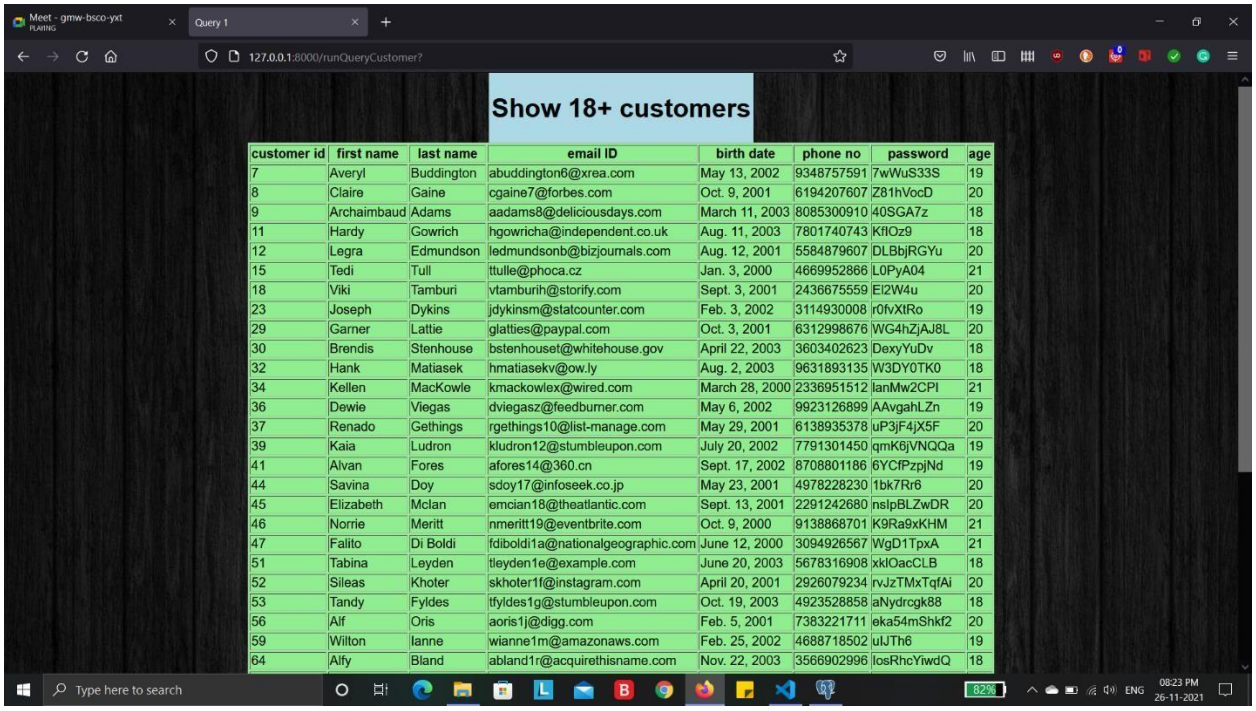
(4) Event Database

event id	event type id	event name	event date	event size	address line 1	address line 2	pincode	event manager	
1	1	baby	Feb. 28, 2021	1370	63869 Fulton Drive	599 Hovde Circle	31397912	1	Edit delete
2	3	standup comady	Aug. 3, 2021	1533	375 Mendota Street	36 Grover Drive	8807227	2	Edit delete
3	6	dhindhora	Sept. 26, 2020	928	5728 Mayer Terrace	6 Beilfuss Court	15428104	3	Edit delete
4	1	murder	May 18, 2021	1182	4623 Sloan Junction	011 Prairie Rose Junction	35102784	4	Edit delete
5	5	romio and juliat	Feb. 24, 2021	2108	93 Shopko Drive	9472 Lunder Circle	13859714	5	Edit delete
6	5	hamlet	Oct. 1, 2021	1921	6206 Maywood Way	1 Columbus Avenue	16070442	6	Edit delete
7	6	aheliya	Feb. 4, 2021	301	698 Farmco Place	37 Manitowish Crossing	30724527	7	Edit delete
8	5	angles in america	Jan. 15, 2020	2654	4 Waubesa Trail	10 Pankratz Junction	34661776	8	Edit delete
9	1	veer	Feb. 20, 2020	193	737 Dakota Plaza	24 Arizona Circle	29122232	9	Edit delete
10	1	lagan	Aug. 19, 2021	256	85 Corry Parkway	54 Johnson Pass	13293165	10	Edit delete
11	3	bharat natyam	Sept. 29, 2021	1174	8 Hoard Junction	5858 Old Shore Park	7250389	11	Edit delete
12	5	look back in anger	Nov. 10, 2021	652	50 Shopko Plaza	91629 Kinsman Avenue	10081357	12	Edit delete
13	6	toxic	Jan. 3, 2020	1871	0 Harper Parkway	5084 Elgar Pass	25294112	13	Edit delete
14	4	idea unlimited	Jan. 2, 2021	2305	52257 Hintze Crossing	4 Jackson Way	5470250	7	Edit delete
15	7	great royal	July 5, 2020	228	8430 Basil Point	326 Di Loreto Court	8578740	15	Edit delete
16	6	kriti	April 13, 2021	45	7979 Goodland Parkway	88 Pond Place	37476100	16	Edit delete
17	7	rajkamal	July 13, 2020	1463	4 Sunfield Plaza	4610 Pierstorff Plaza	6640296	17	Edit delete
18	6	the maid	May 19, 2020	2130	88 Hovde Avenue	8518 Canary Trail	30718080	18	Edit delete
19	6	positive	Jan. 15, 2021	548	87 Luster Center	61995 Moland Trail	24888722	19	Edit delete
20	4	farewell	Feb. 28, 2021	2198	290 Scoville Crossing	301 Melvin Circle	13448348	20	Edit delete
21	6	ouch	May 5, 2021	199	6827 Mallory Avenue	5 Gale Center	35660459	21	Edit delete
22	7	great raimen	June 1, 2020	716	90189 Fuller Terrace	11 Clyde Gallagher Terrace	16997343	22	Edit delete
23	5	woyzeck	May 22, 2021	1945	86333 Vermont Road	2838 Hanson Street	26331080	7	Edit delete
24	5	waiting for godot	Aug. 6, 2021	321	8 Ohio Drive	904 Myrtle Drive	4707237	24	Edit delete
25	1	satyamave jayte	June 7, 2020	883	71 Bayside Pass	9 5th Drive	9372696	25	Edit delete

(5) Insert in Event Database

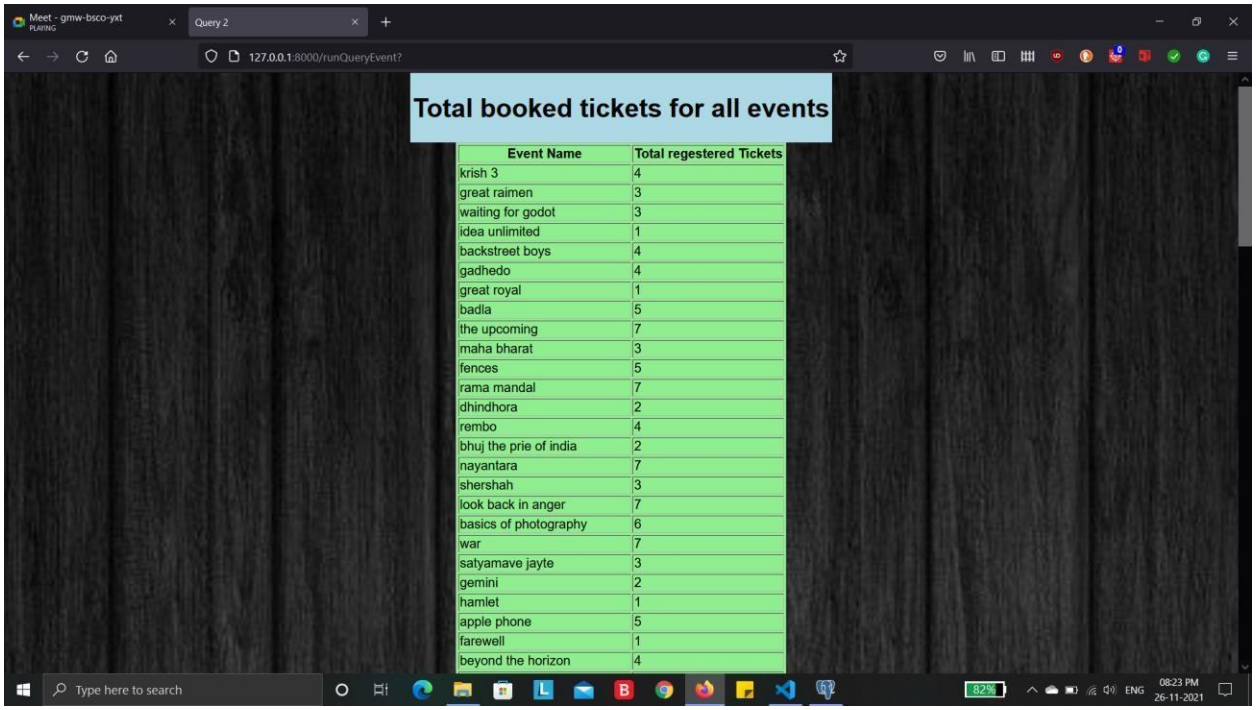


(6) Query 1: Display all the customers whose age is above 18.



customer id	first name	last name	email ID	birth date	phone no	password	age
7	Averyl	Buddington	abuddington6@xrea.com	May 13, 2002	9348757591	7wWuS33S	19
8	Claire	Gaine	cgaine7@forbes.com	Oct. 9, 2001	6194207607	Z81hVocD	20
9	Archaimbaud	Adams	aadams8@deliciousdays.com	March 11, 2003	8085300910	40SGA7z	18
11	Hardy	Gowrich	hgowricha@independent.co.uk	Aug. 11, 2003	7801740743	KfIOz9	18
12	Legra	Edmundson	ledmundsonb@bizjournals.com	Aug. 12, 2001	5584879607	DLBbjRGYu	20
15	Tedi	Tull	ttulle@phoca.cz	Jan. 3, 2000	4669952866	L0PyA04	21
18	Viki	Tamburi	vtamburih@storify.com	Sept. 3, 2001	2436675559	El2W4u	20
23	Joseph	Dykins	jdykinsm@statcounter.com	Feb. 3, 2002	3114930008	r0fvXtRo	19
29	Garner	Lattie	glatties@paypal.com	Oct. 3, 2001	6312998676	WG4hZjAJ8L	20
30	Brendis	Stenhouse	bstenhouset@whitehouse.gov	April 22, 2003	3603402623	DexyYuDv	18
32	Hank	Matiasek	hmatiasekv@ow.ly	Aug. 2, 2003	9631893135	W3DY0TK0	18
34	Kellen	MacKowle	kmackowlex@wired.com	March 28, 2000	2336951512	lanMw2CPI	21
36	Dewie	Viegas	dviegasz@feedburner.com	May 6, 2002	9923126899	AAvgahLZn	19
37	Renado	Gethings	rgethings10@list-manage.com	May 29, 2001	6138935378	uP3jF4jX5F	20
39	Kaia	Ludron	kludron12@stumbleupon.com	July 20, 2002	7791301450	qmK6jVNQQa	19
41	Alvan	Fores	afores14@360.cn	Sept. 17, 2002	8708801186	6YCfPzpjNd	19
44	Savina	Doy	sdoy17@infoseek.co.jp	May 23, 2001	4978228230	1bk7Rr6	20
45	Elizabeth	Mclan	emcian18@theatlantic.com	Sept. 13, 2001	2291242680	nsIpBLZwDR	20
46	Norrie	Meritt	nmeritt19@eventbrite.com	Oct. 9, 2000	9138868701	K9Ra9xKHM	21
47	Falito	Di Boldi	fdiboldi1a@nationalgeographic.com	June 12, 2000	3094926567	WgD1TpxA	21
51	Tabina	Leyden	tleyden1e@example.com	June 20, 2003	5678316908	xkIOacCLB	18
52	Sileas	Khoter	skhoter1f@instagram.com	April 20, 2001	2926079234	rvJzTMxTqfAi	20
53	Tandy	Fyldes	tfyldes1g@stumbleupon.com	Oct. 19, 2003	4923528858	aNydrcgk88	18
56	Alf	Oris	aoris1j@digg.com	Feb. 5, 2001	7383221711	eka54mShkf2	20
59	Wilton	Ianne	wianne1m@amazonaws.com	Feb. 25, 2002	4688718502	uIJTh6	19
64	Alfy	Bland	abland1r@acquirethisname.com	Nov. 22, 2003	3566902996	losRhcYiwdQ	18

(7) Query 2: Display the total number of booked tickets event-wise.



Event Name	Total regestered Tickets
krish 3	4
great raimen	3
waiting for godot	3
idea unlimited	1
backstreet boys	4
gadhedo	4
great royal	1
badla	5
the upcoming	7
maha bharat	3
fences	5
rama mandal	7
dhindhora	2
rembo	4
bhuj the prie of india	2
nayantara	7
shershah	3
look back in anger	7
basics of photography	6
war	7
satyamave jayte	3
gemini	2
hamlet	1
apple phone	5
farewell	1
beyond the horizon	4

CODES FOR WEBPAGE USING PYTHON AND DJANGO: -

1) Insert, Delete, Customer And query

```
from django.shortcuts import render
from Event_Registration.models import CustModel,EventModel
from django.contrib import messages
from Event_Registration.forms import Custforms,Eventforms
from django.db import connection

def homepage(request):
    return render(request, 'Homepage.html')

def showcust(request):
    showall=CustModel.objects.all()
    return render(request, 'Index.html', {"data":showall})

def InsertCust(request):
    if request.method == "POST":
        if request.POST.get('c_id') and request.POST.get('first_name') and request.POST.get('last_name') and
request.POST.get('c_email') and request.POST.get('c_Bdate') and request.POST.get('c_phno') and
request.POST.get('password') and request.POST.get('age'):
            saverecord=CustModel()
            saverecord.c_id = request.POST.get('c_id')
            saverecord.first_name = request.POST.get('first_name')
            saverecord.last_name = request.POST.get('last_name')
            saverecord.c_email = request.POST.get('c_email')
            saverecord.c_Bdate = request.POST.get('c_Bdate')
            saverecord.c_phno = request.POST.get('c_phno')
            saverecord.password = request.POST.get('password')
            saverecord.age = request.POST.get('age')
            saverecord.save()
            messages.success(request, 'Customer' + saverecord.first_name + ' saved successfully!!!')
            return render(request, 'insert.html')
    else:
            return render(request, 'insert.html')

def EditCust(request, c_id):
```

```
        edittmpobject = CustModel.objects.get(c_id=c_id)
        return render(request, 'Edit.html', {"CustModel":edittmpobject})

    def UpdateCust(request, c_id):
        updatecustmor=CustModel.objects.get(c_id=c_id)
        form=Custforms(request.POST, instance=updatecustmor)
        if form.is_valid():
            form.save()
            messages.success(request, 'record updated Successfully!!')
            return render(request, 'Edit.html', {"CustModel":updatecustmor})

    def DeleteCust(request, c_id):
        deletecust=CustModel.objects.get(c_id=c_id)
        deletecust.delete()
        showdata=CustModel.objects.all()
        return render(request, "Index.html", {"data": showdata})


def showevent(request):
        showall=EventModel.objects.all()
        return render(request, 'IndexEvent.html', {"data":showall})

    def Insertevent(request):
        if request.method == "POST":
            if request.POST.get('event_id') and request.POST.get('e_type_id') and request.POST.get('e_name')
    and request.POST.get('e_date') and request.POST.get('e_size') and request.POST.get('address_line_1') and
    request.POST.get('address_line_2') and request.POST.get('pincode') and request.POST.get('e_manager'):
                saverecord=EventModel()
                saverecord.event_id = request.POST.get('event_id')
                saverecord.e_type_id = request.POST.get('e_type_id')
                saverecord.e_name = request.POST.get('e_name')
                saverecord.e_date = request.POST.get('e_date')
                saverecord.e_size = request.POST.get('e_size')
                saverecord.address_line_1 = request.POST.get('address_line_1')
                saverecord.address_line_1 = request.POST.get('address_line_2')
                saverecord.pincode = request.POST.get('pincode')
                saverecord.e_manager = request.POST.get('e_manager')
                saverecord.save()
```

```
            messages.success(request, 'event ' + saverecord.e_name + ' saved successfully!!!')
            return render(request, 'InsertEvent.html')
    else:
            return render(request, 'InsertEvent.html')

def EditEvent(request, event_id):
    edittmpobject = EventModel.objects.get(event_id=event_id)
    return render(request, 'EditEvent.html', {"EventModel":edittmpobject})

def UpdateEvent(request, event_id):
    updateEvent=EventModel.objects.get(event_id=event_id)
    form=Eventforms(request.POST, instance=updateEvent)
    if form.is_valid():
        form.save()
        messages.success(request, 'record updated Successfully!!')
        return render(request, 'EditEvent.html', {"EventModel":updateEvent})

def DeleteEvent(request, event_id):
    deleteEvent=EventModel.objects.get(event_id=event_id)
    deleteEvent.delete()
    showdata=EventModel.objects.all()
    return render(request, "IndexEvent.html", {"data": showdata})




def runQueryCustomer(request):
    raw_query = "select * from customer where  age>=18 order by c_id;"

    cursor = connection.cursor()
    cursor.execute(raw_query)
    alldata=cursor.fetchall()

    return render(request,'runQueryCustomer.html',{"data":alldata})


def runQueryEvent(request):
    raw_query = "select e_name, sum(no_of_tickets) from ticket natural join event group by e_name;"
```

```
    cursor = connection.cursor()
    cursor.execute(raw_query)
    alldata=cursor.fetchall()

    return render(request,'runQueryEvent.html',{"data":alldata})
```

URLs

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('showcust', views.showcust, name = "showcust"),
    path('insert', views.InsertCust, name="InsertTmp"),
    path('Edit/<int:c_id>', views.EditCust, name="EditCust"),
    path('Update/<int:c_id>', views.UpdateCust, name="UpdateCust"),
    path('Delete/<int:c_id>', views.DeleteCust, name="DeleteCust"),
    path('runQueryCustomer',views.runQueryCustomer,name="runQueryCustomer"),

    path('',views.homepage,name="homepage"),

    path('showevent', views.showevent, name = "showevent"),
    path('insertevent', views.Insertevent, name="Insertevent"),
    path('Editevent/<int:event_id>', views.EditEvent, name="EditEvent"),
    path('Updateevent/<int:event_id>', views.UpdateEvent, name="UpdateEvent"),
    path('Deleteevent/<int:event_id>', views.DeleteEvent, name="DeleteEvent"),
    path('runQueryEvent',views.runQueryEvent,name="runQueryEvent"),
]
```

Setting modulo for frontend

```
from django.db import models

class CustModel(models.Model):
    c_id=models.BigIntegerField(primary_key=True)
    first_name=models.CharField(max_length=20)
    last_name=models.CharField(max_length=20)
    c_email=models.CharField(max_length=40)
    c_Bdate=models.DateField()
    c_phno=models.BigIntegerField()
    password=models.CharField(max_length=64)
    age=models.IntegerField()
    class Meta:
        db_table = "customer"


class EventModel(models.Model):
    event_id=models.BigIntegerField(primary_key=True)
    e_type_id=models.BigIntegerField()
    e_name=models.CharField(max_length=100)
    e_date=models.DateField()
    e_size=models.IntegerField()
    address_line_1=models.CharField(max_length=50)
    address_line_2=models.CharField(max_length=50)
    pincode=models.IntegerField()
    e_manager=models.IntegerField()
    class Meta:
   db_table = "event"
```

Connect Frontend with Backend

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'LabProject',
        'USER': 'postgres',
        'PASSWORD': 'admin',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Link for entire code:-
https://drive.google.com/drive/folders/1jBXk1oACNrgbyBCPIa5U-gsK5e47_SQS?usp=sharing