Dhirubhai Ambani
Institute of Information and Communication Technology

Traveler Management System

S4_T2

Course: DBMS

Course Id: IT214

ID	Name
201901001	Ridham Sonani
201901037	Bhargav Patel
201901055	Dhruvin Moradiya
201901096	Darshan Gohil

Year : 2021

Mentor TA: Vaishnavi

Instructor: Minal Bhise

Section 1: Software Requirement Specification

Introduction

Purpose

According to the WTTC, the global travel and tourism sector is thriving very well, and it has outstripped the growth of worldwide GDP in the past year. The report from the WTTC also says that travel and tourism increased by 3.9% in recent years, which is more than the worldwide GDP increase of 3.2%. This growth contributed a record $8.8 trillion to the world economy. Furthermore, it is also stated in the report that this industry generated 10.4% of all worldwide economic activity.

But as we know that during this pandemic time most affected industries are traveling agencies. And as traveling has decreased, tourism has also decreased because many other industries are also affected like hoteling, and local businesses are affected very much. So we are trying to boost our traveling agency and safe traveling during this pandemic situation so that the traveling industry can grow again. For example, many companies like Taj, Mahindra provide different 5-10years packages to keep people traveling.

Intended Audience and Reading Suggestions

The document is intended mainly for developers and documentation writers but can be tailored for project managers, users, and marketing staff. With the elimination of jargon, this report aims to document the requirements in detail in combination with adopting an easy-to-read and simplistic approach to writing the documentation. The report will be split into sections like introduction, overall description, background reading, system features, observation, and other requirements, user classes and characteristics, operating environment, product functions, assumptions and dependencies, business constraints,. It will have sub-sections corresponding to the sections within the document

A sequence that is suggested for developers is to look through the contents page and read through the Design and implementation constraints section. A developer will need to know the description, requirements, and features of the system however the sub-sections can be looked through if the developer needs additional information.

For project managers, it is suggested they read and fully understand the full report by reading each section as it is ideal for them to know and understand all aspects of the document.
Marketing staff will need to know the system features and the description/purpose of the product so they will need to look through the contents page and are suggested to read through any subsection that relates to the description of the product (e.g. they will need to know who to market the app to) and the overall description and the features of the software. Users will need to know the overall description and the features within the system

Product Scope

The Tourism Management System project can prove to be immensely useful for travelers and travel agents with no or little management facilities. There are 1652 million domestic travelers in our country according to the statistics provided by the ministry of tourism. This huge number of travelers who need an affordable and comfortable journey can use this software to find their suitable option. The Tourism Management System project is an implementation of a managing Tourism website that helps the customers to search the availability of various tourist places and prices of various hotel rooms in particular places, along with the different packages available. This project also covers various features like online registration of the users, modifying the details of the website by the management staff or administrator of the website, by adding, deleting, or modifying the customer details or packages information. In general, this website would be designed to perform like any other Tourist management website available online. The benefit to a business would be that this program can be revised to add more efficient methods to improve the backend of the business.

Description

- Customers: All the people who want to travel through the agency. I.e., Normal people, people who want to go on a family vacation, people going on business trips, etc.
- Customers would be able to check available tour packages, their current discount, and also their timing.
- Customers can select and sort Travel Reservation the way they like.
- Customers would be able to check whether the location is safe for COVID-19 or not.
- Customers will add reviews from the customers to the database to provide them with better future services.
- The database will have information about travel agents (i.e., contact information, their address) so that customers can contact them in case of any problem or information needed about a tour or destination.
- Customers can buy various memberships. They can use membership details to get extra discounts on tour packages and extra services depending on type of membership.
- Agency also provides membership details like membership id for different types of membership, membership type, membership duration, price.
- IT staff will manage customer data and try to secure that data.
- IT staff will set a unique employee password for security purposes
- We will get information about the employees (i.e., name, address, contact details). Each employee will have their unique id(Employee ID). The employee will have their password to access the database.
- IT staff are another important user class. They will be maintaining the system, making sure that the system is fully functional, and dealing with any issues that occur with the system. They will frequently be maintaining aspects of the systems, so their engagement with the system will be frequent, but usage will be lower.
- Managing staff use data from the system to produce revenue reports. Managing staff will be using the data produced by the system to analyze and create reports so that progress is tracked within the business. Storing customer details is a vital component of the system as it allows for managing staff to evaluate and collect data. Agencies can tailor promotional and special offers to individuals as their tourism places type is stored, which means emails with special offers on specific tourism places can allow customer retention.
- Agency staff: Staff associated with booking, tour guidance, managing tours, confirmation, and cancellation of tours. Travel agents, Booking executives, etc.
- After this pandemic situation, all the people(customers) are aware of the safety and spreading of the COVID 19 virus, so we have to track the condition of COVID 19 on all the destinations accurately(Place is safe to visit?: Yes or No), update databases accurately, and also provide customers with a safe search feature.
- The major role of the IT department comes when we talk of the security level of databases because it's essential to keep all the information private and safe from malicious activities, which can be altered with the original information, and then the whole system will fail.
- All the people should get accurate information through the website. So It's the responsibility of the IT department that they update all the information regarding the tour.
- Design must be able to handle large data uploading loads and keep data updating in real-time so that accurate information reaches the travel agency to make efficient decisions.
- Agencies need data from users and arrange tours according to their desire or give discount packages to attract people toward tourism.
- Agency needs the user's name (First name, middle name, last name), their address, their contact details, and his current tour so we can track their progress. Agency will provide customer ID to users to uniquely identify them.
- By creating a relationship history between a customer and his tour packages, we can get information about the customer’s past tours. And by using that information, we can know their interests so that we can give them better tour packages.

- A critical aspect of monitoring activity is transportation. That means we maintain each customer's information regarding medium of transportation like Bus and Airplane.
- Efficient databases lead to accurate information on the travel agency site so that customers can access the latest information about tour packages and related stuff.
- The notion of a centralized system is that it handles situations efficiently with the help of an accurate database and quick action.
- Updating database information comes from service-providing companies like hotels and vehicles providers, which access the database and keep updating the number of available vehicles and rooms.
- The database has been updated in real-time so that all the information reaching the customers would be accurate. While implementing this feature, we have to connect all the vehicle and hotel databases. For that, we need information about the vehicle, hotel, and other service provider companies(i.e., company name and contact details and their addresses).
- To keep customers attracted toward traveling, we need to keep track of customers' past tours and interests, and by analyzing them, we can make more reliable tours.
- On festivals, seasons, or summer vacation, travel agencies can give discounts or offer different tour packages(i.e., percentage of discount and when this discount is starting and ending). Discount will also have its unique id so that customers can use that discount using discount id to get discounts on tours.
- The role of the tour agency in the database is as a superuser that accesses all the information regarding tour packages and customers and updates it. It is crucial to keep data up to date, and if data is altered by someone accidentally or intentionally, the history of updates needs to be saved in the database.
- We classified tours into different types. By doing so, we maintain the simplicity of the database. Every tour will have a unique tour id, and using that, we can access information about tours.
- The agency should have a proper database of each person regarding its vaccination status, tour, etc.
- For better tour management, we need to keep track of tourist's data like names, no. of tourist's who are traveling, source, destination, etc.
- While fetching records of particular users, we have partial names like surnames or first names, and then such a query needs to be executed in our database to decompose our name into first name, last name, and middle name.
- Similarly, for Address, we can decompose it into streets, area-wise pin code numbers, state so this kind of database efficiently runs the complex query.
- We need to collect boarding points for all the tourists of the current package to make the shortest route for all the boarding points to pick up every customer.
- We need to make one temporary booking table so that customers can know how many seats are available on the bus or plane, and also they can see the information regarding whether their seats are confirmed or not. And travel agencies can track the booking information related to the booked seat, waiting list of customers, the availability of the seats, and vehicle details.
- We need to make sure that the database updates information regarding booking and cancellation of tours in real-time so that the integrity of the database remains maintained.
- For cancellation, we need to keep a record of cancellation time, date, and total refundable amount. Also, the cancellation agency will have a unique cancellation id so that if any problem occurs regarding the cancellation or refund of the amount paid, the agency can solve that problem.
- We also need to keep billing records like how many seats are booked by particular customers and the total amount paid by that customer. With the bill, the customer will get a booking ID and Billing ID to search for their booking details.
- Once the customer booked the tour, billing information will be generated automatically. Total price will be calculated using total booked seats, membership details and discount ID related to that tour.
- Information gathered from vehicle and hotel companies, and we will be assigning Vehicle id and Hotel ID to vehicles and hotels, respectively. We need to keep track of the availability of the hotel rooms, which type of rooms are available for the tour, and the price and the destination of the hotels.

- We have to collect all the information about the destination. Destination id will be provided to the destination to classify destinations and make one organized with the tour.
- Hotel staff can provide information about the availability of hotel rooms and resorts and provide facilities to tourists.
- Traveling management staff, Travel manager who manages vehicle allocation for different tours, maintenance, or services.
- List of famous places at the destination which are satisfactory to visitors. Also, how much these destination places are safe to visit in terms of COVID situations. Availability of safe places to rest.
- Management staff can classify the destination based on historical, natural places, holy places, etc.
- Details of the hotel and their rooms. For any given time, we have to know whether a hotel room is available for the tourist or not, types of the room like mid-range, high range. For a better user experience, we also have to know hotel reviews—whether the hotel is four stars or five stars.
- For better customer service and safe traveling, we need to check whether a vehicle can provide a good service. If not, then we have to schedule maintenance for the vehicle. (i.e., next maintenance date).
- Users need to know which places are covered in the particular tour packages, the cost of tour packages, the current discount for the packages, the schedule of the tour, and how long it takes.
- Users need to know what vehicles and what hotels are for the particular tour.
- Agencies need to provide information about the tour(i.e., description) and the total duration of the tour so that customers can access data about the tour, like which places are contained in any particular tour and how long that tour is, and when the Agency will organize that tour.
- For better convenience, we will add destination image links and destination descriptions so that customers can get a rough idea about the destination.

Background Readings

Description of reading

- In this section we collected data from various sites and Journals, below are the main points.
- Because of the vast amount of population, India needs to adopt an efficient strategy and planning to ensure safe traveling during the pandemic.
- For making any decision, the travel agency needs accurate and real-time data.
- Some of those are stated below:
    - Types of travel packages
    - Availability of travel vehicle
    - Customer details and their interest (i.e., favorite places or destinations)
    - Availability of hotel and hotel reviews (On a scale of 5 stars).
    - Different types of destinations
    - Condition of vehicles (i.e., maintenance)
    - Overall review and user satisfaction.

- Some better plans to boost tourism...
    - Taking some extra precautions and checking the COVID situation at the destination.
    - Allowing users to search for safe places according to their desire.
    - Arranging some good value for money packages to attract people.
    - Give suggestions to the customers according to their previous tour packages.

- TripIt App
  - You can set your reservations to be automatically sent to TripIt, which will let you view travel confirmations, flight itineraries, tickets, hotel and Airbnb booking info, rental car reservations, ferry tickets and driving directions without ever leaving the app. TripIt also makes it simple to share your trip plans with whoever is picking you up from the airport or train station, or anyone else who may need to coordinate with you or know what your travel plans are.

- Makemytrip
  - Makemytrip is an Indian company of online travel.The company provides services related to online travel booking and travel services such as domestic and international holiday packages and booking of buses, rails and flight tickets as well as hotel reservations. It also provides the facility of foreign currency notes, forex travel cards and demand drafts.

- References

  - https://www.dcsplus.net/blog/3-ways-travel-agents-benefit-from-database-mapping
  - https://www.dcsplus.net/blog/3-ways-travel-agents-benefit-from-database-mapping
  - https://www.tsijournals.com/articles/analysis-on-application-of-tourism-information-service-system-under-the-j2ee-environment.pdf
  - http://eprints.bournemouth.ac.uk/15750/1/Database_Marketing_In_Travel_And_Tourism.pdf
  - https://www.softwaresuggest.com/blog/must-have-features-travel-agency-software/
  - https://www.researchgate.net/publication/310122350_Tourism_Management_System
  - https://hrmars.com/papers_submitted/9206/development-of-tourism-database-management-system-creating-er-model.pdf
  - https://core.ac.uk/download/pdf/235049881.pdf
  - https://www.pilotplans.com/blog/makemytrip-review

- Combine requirement
  - Collecting information about different types of places to visit (i.eNatural places, Historical places, etc…)
  - Sorting out places with less review and improving them by giving more itineraries.
  - By taking care of maintenance, making sure that tourists get proper satisfaction.

Interview

Interview 1

System: Tourist Agencies Management Database

Project Reference: SF/SJ/2021/10

Interviewee: 1) Tanmay Raina(Role Play)
Designation: Travel Agent

Interviewer: 1) Ridham Sonani **Designation**: student at DA-IICT
2) Bhargav Patel **Designation**: student at DA-IICT
3) Dhruvin Moradiya **Designation**: student at DA-IICT
4) Darshan Gohil **Designation**: student at DA-IICT

Date: 5/10/2021 **Time:** 10:00 AM
Duration: 120 minutes **Place:** Google Meet

Purpose of Interview:

- Preliminary meeting to discuss different kinds of vacation packages for clients.

Agenda:

- Initial ideas
- What type of destinations people like to visit most
- Problems with the lack of management
- Follow-up actions

Documents to be brought to the interview:

- Documents related to a rough plan of the tourist management system.

Interview Summary 1

System: Tourist Agencies Management Database

Project Reference: SF/SJ/2021/12

Interviewee: 1) Tanmay Raina(Role Play)
Designation: Travel agent

Interviewer: 1) Ridham Sonani **Designation**: student at DA-IICT
2) Bhargav Patel **Designation**: student at DA-IICT
3) Dhruvin Moradiya **Designation**: student at DA-IICT
4) Darshan Gohil **Designation**: student at DA-IICT

Date: 5/10/2021 **Time:** 10:00 AM
Duration: 120 minutes **Place:** Google Meet

Summary of Interview:

1. Mainly to discuss usual problems noticed during trips by travel agents so that problems can be figured out.
2. Discussed instant complaints raised by travelers during the tour and how to solve them.
3. Discussed how to create tour packages for corporate trips and events.
4. What is the process for researching new destinations for clients?

Interview 2

System: Tourist Agencies Management Database

Project Reference: SF/SJ/2021/12

Interviewee: 1) Sachin Kumar(Role Play)
Designation: Manager at TravelWale

Interviewer: 1) Ridham Sonani **Designation**: student at DA-IICT
2) Bhargav Patel **Designation**: student at DA-IICT
3) Dhruvin Moradiya **Designation**: student at DA-IICT
4) Darshan Gohil **Designation**: student at DA-IICT

Date: 5/10/2021 **Time:** 13:00 PM
Duration: 150 minutes **Place:** Google Meet

Purpose of Interview:

- To identify what kind of current problem they are facing during this pandemic situation and how they overcome that issue by management.

Agenda:

- Problems with the lack of management
- Strategies for boosting tourism after pandemic
- Brief ideas about tour management.
- Follow-up actions.

Documents to be brought to the interview:

- Documents related to giving information related to the pandemic situation of tourism places.

Interview Summary 2

System: Tourist Agencies Management Database

Project Reference: SF/SJ/2021/12

Interviewee: 1)Sachin Kumar(Role Play)
Designation: Manager at TravelWale

Interviewer: 1) Ridham Sonani **Designation**: student at DA-IICT
2) Bhargav Patel **Designation**: student at DA-IICT
3) Dhruvin Moradiya **Designation**: student at DA-IICT

4) Darshan Gohil **Designation**: student at DA-IICT

Date: 5/10/2021 **Time:** 17:00 PM
Duration: 150 minutes **Place:** Google Meet

Summary of Interview:

Preliminary meeting to identify problems and requirements regarding improvement of the agency's site and regarding information related management.

1. Mainly to discuss problems related to the current pandemic situation and how they are dealing with them. I.e. Some strategies to handle tourism in the current pandemic.
2. What kind of steps are taken by the management to avoid spreading covid.
3. What kind of future challenges do they face and how much they are prepared for it.
4. What kind of changes do they want to specify for the website so that they can show the customer that agencies are very much aware of their health.

Interview 3

System: Tourist Agencies Management Database

Project Reference: SF/SJ/2021/12

Interviewee: 1) Mahesh Kumarthe (**Role Play**)
Designation: Booking executive

Interviewer: 1) Ridham Sonani **Designation**: student at DA-IICT
2) Bhargav Patel **Designation**: student at DA-IICT
3) Dhruvin Moradiya **Designation**: student at DA-IICT
4) Darshan Gohil **Designation**: student at DA-IICT

Date: 5/10/2021 **Time:** 15:00 PM
Duration: 120 minutes **Place:** Google Meet

Purpose of Interview:

- Preliminary meeting to identify people's interest in different places so that tours can be organized biasedly.

Agenda:

- Problems with the lack of management.
- Initial ideas about creating new tours to attract people toward tourism.

Documents to be brought to the interview:

- Documents related to previous booking details.

Interview Summary 3

System: Tourist Agencies Management Database

Project Reference: SF/SJ/2021/12

Interviewee: 1) Mahesh Kumarthe(**Role Play**)
Designation: Booking executive

Interviewer:	1) Ridham Sonani	**Designation**: student at DA-IICT
	2) Bhargav Patel	**Designation**: student at DA-IICT
	3) Dhruvin Moradiya	**Designation**: student at DA-IICT
	4) Darshan Gohil	**Designation**: student at DA-IICT

Date:5/10/2021 **Time:** 18:00 PM
Duration: 180 minutes **Place:** Google Meet

Summary of Interview:

Preliminary meeting to identify people's interest in different places so that tours can be organized biasedly.

1. Mainly to discuss which type of places people like to visit or which types of tour are more booked or requested by companies.
2. According to that travel agency can organize more tours and can add an extra itinerary to existing tours.
3. The tour in which people are less interested can give discounts on those tours.

Requirements

- The travel agencies must have information about the COVID situation of different tourist places and according to that, they have to manage tours.
- Need to maintain regular maintenance of database and vehicle and other facilities.
- Records of people's problems, complaints during the tour and their needs and also which places people like most.
- Records of places where people visit more or requests received by companies to arrange special tours.
- Information about the availability of vehicles and hotels.
- Information about different types of destinations people like to visit.

Questionnaires

Travel Agency Management System

This is only for study purpose. This is normal survey to get to know about how do you think about traveling or going on vacation in this pandemic situation.

201901037@daiict.ac.in (not shared) Switch account

* Required

How do you plan your trip? *

- ◯ By yourself
- ◯ Through a Travel Agent

Have you ever used traveling site? *

- ◯ Yes
- ◯ No

Are you ready to go on tour during this pandemic? *

- ◯ yes
- ◯ no

What are the most challenging problems that you face when choosing a destination? *

- ☐ Quality of services
- ☐ Reliability
- ☐ Language difficulties
- ☐ Price

Where do you go most often? *

- ☐ hill-station
- ☐ beaches
- ☐ deserts
- ☐ historical significant places
- ☐ commercially busy city

What kind of tour package price will reasonable for you? *

- ☐ 3000-5000
- ☐ 5000-8000
- ☐ 10000-15000
- ☐ 20000-40000
- ☐ 40000+

Please note any comments you have on current travel management system.

Your answer

Please note any suggestions for improving travelling experience.

Your answer

Submit

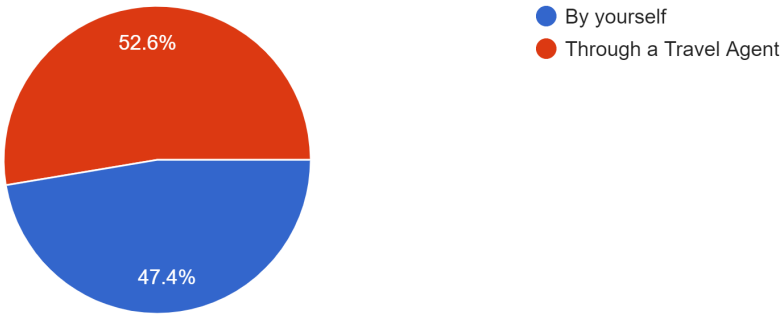
Page 1 of 1

Clear form

Responses :

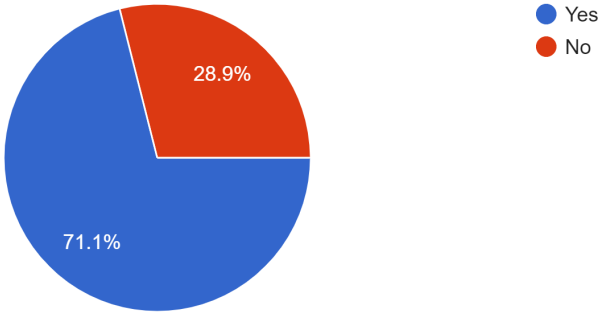
How do you plan your trip?

76 responses



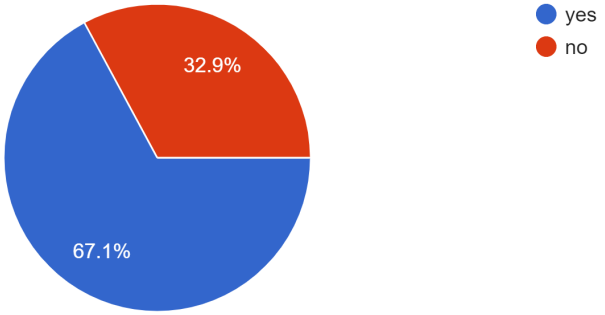
Have you ever used traveling site?

76 responses



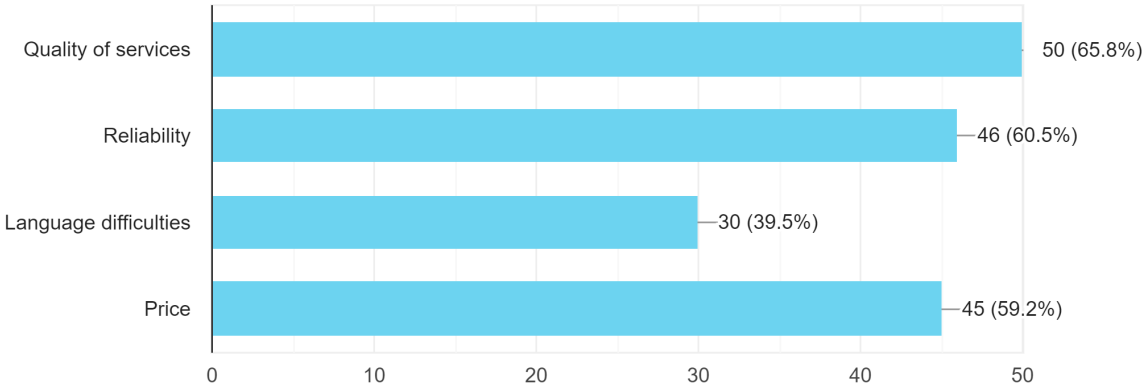
Are you ready to go on tour during this pandemic?

76 responses

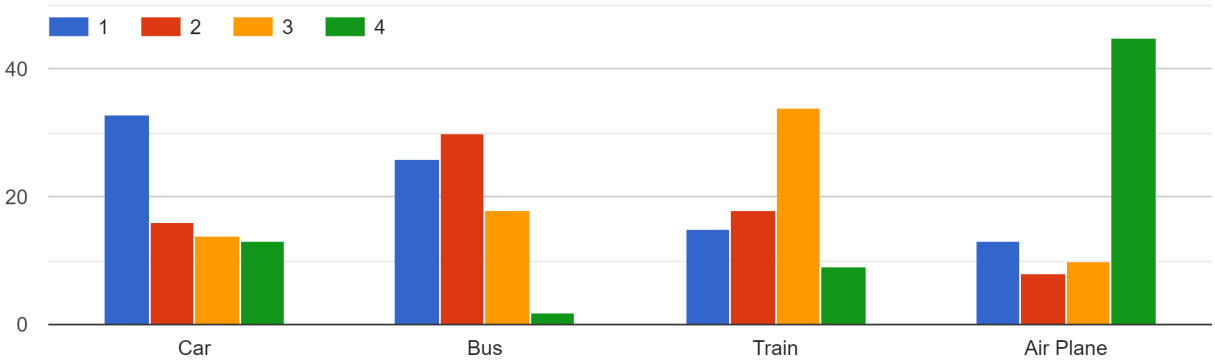


What are the most challenging problems that you face when choosing a destination?

76 responses

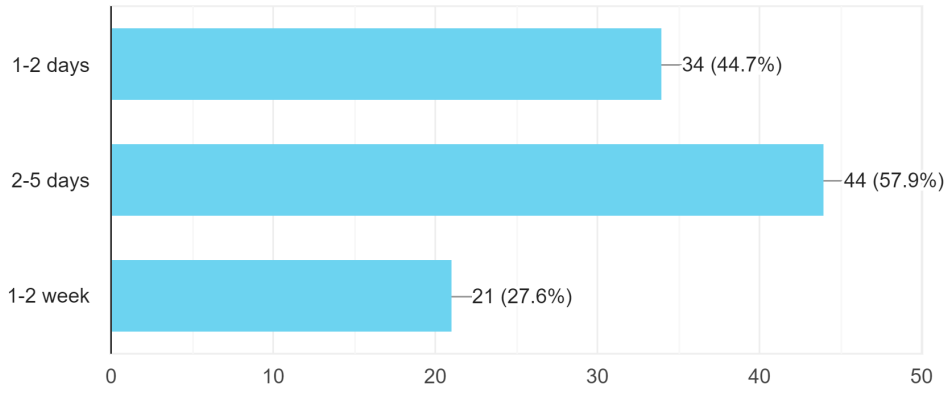


What mode of transportation do you opt? write your preferences



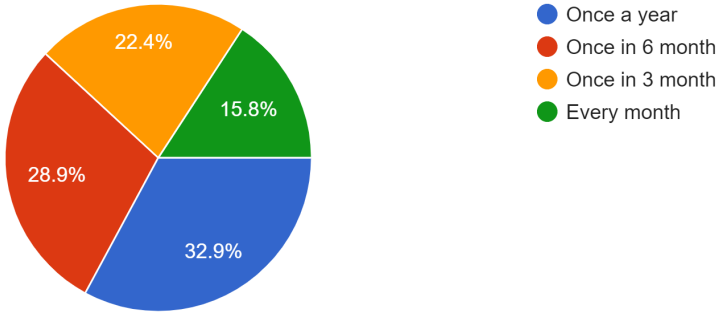
what is generally length of the tour?

76 responses



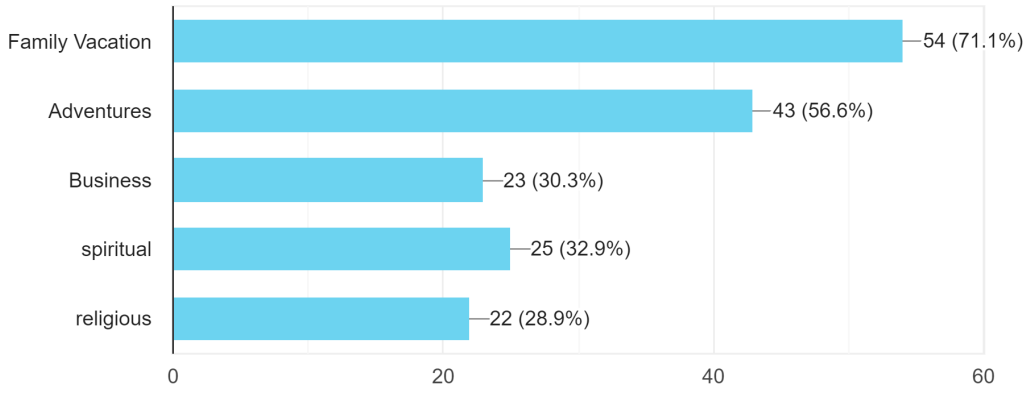
How often do you travel outstation?

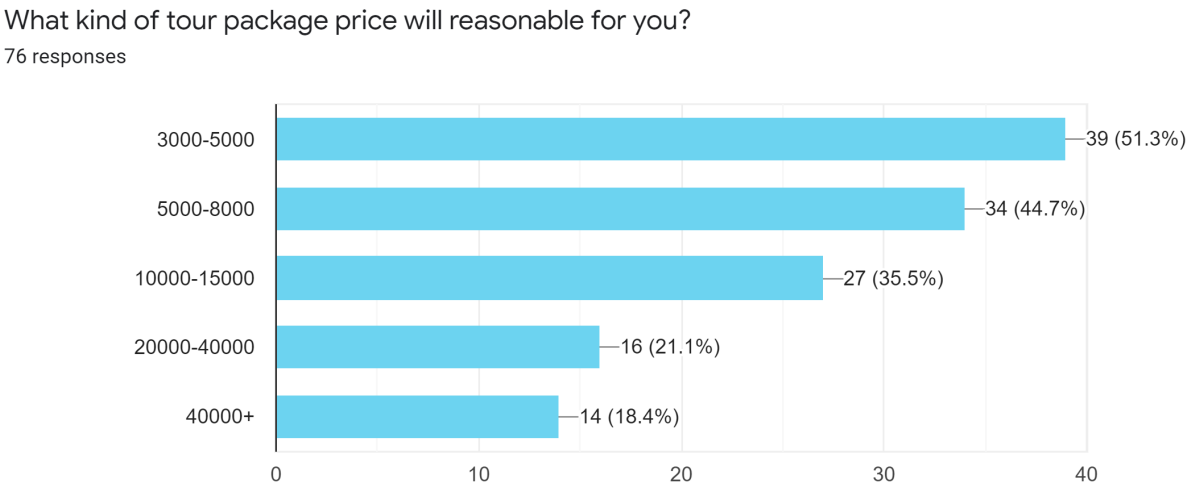
76 responses



What is most common type purpose of your travel?

76 responses





Requirements:

- Websites need to support different languages.
- Wide range of different types of tours
- According to the people's requirements, we need to provide more tour packages for middle-class people.

Observations

System: Tourist Agencies Management Database

Project Reference: SF/SJ/2021/10

Observations by: 1) Ridham Sonani **Designation**: student at DA-IICT
2) Bhargav Patel **Designation**: student at DA-IICT
3) Dhruvin Moradiya **Designation**: student at DA-IICT
4) Darshan Gohil **Designation**: student at DA-IICT

Date: 5/10/2021 **Time:** 17:30
Duration: 60 minutes **Place:** Google meet

Observations:

- Lack of appropriate decisions during a pandemic.
- Improper planning and data management.
- Lack of collaboration with hotels and travel agents.
- Bad quality of transportation vehicles.
- Fail to attract new tourists during the pandemic.
- The problems of tourists were not solved immediately.
- The schedule was not managed properly

Requirements:

- Information about the pandemic situation to ensure the safety of tourists.

- Gathering information about hotels at places safe to travel to so that travel agencies can collaborate with them.
- Maintenance report of vehicles.
- Maintaining databases frequently so that Schedules don't get messed up.

Fact-Finding Chart

Objective	Techniques	Subject(s)	Time commitment
To get background on the company and advertising industry	Background reading	Company reports and reading	2 day
To discuss problems related to management	Interview	Manager	120 min
To get information about trending tourism places where customers usually want to go.	Interview	Booking executive Travel agent	150 min 180 min
To get a public response about a current traveling system problem	Questionnaire	Google Form	5 Days
To get problems that exist in the current system	Observation	All above	4 hour

List requirements

- **Details of Destination**:
    - List of famous places at the destination which are satisfactory to visitors. Also how much these destination places are safe to visit in terms of COVID situations. Availability of safe places to rest.
    - We need to classify the destination based on its types like historical, natural places, holy places, etc.

- **Details of the hotel and their rooms:**
    - For any given time we have to know whether a hotel room is available for the tourist or not.
    - Types of the room like mid-range, high range.
    - For a better user experience, we also have to know hotel reviews.
    - Types of the Hotel whether it is 4 star or 5 stars.

- **Details of vehicle:**
    - We need to know which type of vehicle, whether a vehicle is available or not for that particular tour. If yes then how many seats are available for that tour.
    - For better customer service and safe traveling, we need to check whether a vehicle can provide a good service. If not then we have to schedule maintenance for the vehicle.

- **Details of tour packages:**
    - We need to know which places are covered in the particular tour packages, what is the cost of tour packages, and what is the current discount for the packages and also how long it takes.
    - We need to know what vehicles and what are hotels for the particular tour.
    - Also need data from users and arrange tours according to their desire or give discount packages to attract people toward tourism.

- **Details of users:**
    - We need to know the user's name, his/her address, his/her contact details, and his current tour so we can track his/her progress.
    - We need to know the users' history of the tour so that we can analyze it. And by using that information we can know his/her interest so that we can give him/her better tour packages.

User Classes and Characteristics

The user classes are listed below (high-low importance)

- **Customer**
    - All the people who want to travel through the agency. I.eNormal people, people who want to go on a family vacation, people going on business trips, etc...
    - The customer is the main stakeholder of the software-intensive system.
    - Customers would be able to check available tour packages, their current discount, and also their timing.
    - Customers can select and sort Travel Reservation the way they like.
    - Customers would be able to check whether the location is safe for COVID-19 or not.

- **IT staff**
    - IT staff will manage customer data and try to secure that data.
    - IT staff are another important user class as they will be maintaining the system and making sure that the system is fully functional and dealing with any issues that occur with the system, they will be frequently maintaining aspects of the systems so their engagement with the system will be frequent but usage will be lower.

- **Managing staff**
    - They use data from the system to produce revenue reports.
    - Managing staff will be using the data produced by the system to analyze and produce reports so that progress is tracked within the business, storing customer details is a vital component to the system as it allows for managing staff to evaluate and collect data.
    - Promotional and special offers can be tailored to individuals as their tourism places type is stored which means emails with special offers on specific tourism places can allow for customer retention.

- **Agency staff**
    - This group of users is associated with booking, tour guidance, managing tours, confirmation and cancellation of tours.

    - Booking executive, Travel agents, etc

- **Hotel**
    - Hotel staff, who can provide information about the availability of hotel rooms and resorts and provide facilities to tourists.
    - Normally hotel manager, restaurants management staff, resort management staff.

- **Traveling management staff**
    - Travel manager who manages vehicle allocation for different tours, maintenance, or services.

Operating Environment

- Hardware:- Any normal computer, mobile, tablet device can operate.
- Software requirement: Any kind of operating system can use this software. The software that creates the database is MySQL or PostgreSQL and will be connected to the company's database so data can be accessed.
- Connectivity requirement: Needs dissent internet speed to use tourism websites.
- External Interfaces requirement:
    - The hotel database provides the availability of hotel rooms, hotel reviews, and its services.
    - Database for the availability of vehicles.
    - Government Covid 19 details(Database).

Product Functions

- Booking and cancellation of the tour.
    - Update/insert/delete into booking and cancellation entity set. Mainly booking and cancellation are done by customers.
    - Billing information will be auto generated using discount and membership details.

- Insertion and deletion of the destination and tour packages
    - Insert/delete/Update: Tourist agency management staff can make changes in the tour and destination relations.

- Giving discounts on tour packages.
    - Insert/delete/Update: Offers and discounts can be added/changed in offer relation by tourist management staff.

- Searching and Sorting destinations based on price, safe places or not, and review.
    - Select: Users can select(or search) destination palaces or tour packages according to their need.

- availability of vehicles and check their condition.
    - Insert/delete/Update: Information from vehicle database about availability of vehicle, mainly accessible by tourist agency.

- Availability of hotels..
    - Insert/delete/Update: Information from hotel database about availability of hotel rooms, mainly accessible by tourist agency.

- Reviews of hotels, destinations, vehicles, and services.
    - Insert/delete/Update: Customers can make changes(or add) review of hotel, vehicle, destination or tour packages in the database.
    - View: Tourist agencies can see reviews given by customers to update databases.

Privileges

- Booking and cancellation of the tour.
- Insertion and deletion of the destination and tour packages
- Giving discounts on tour packages.
- Searching and Sorting destinations based on price, safe places or not, and review.
- availability of vehicles and check their condition.
- Availability of hotels.
- Reviews of hotels, destinations, vehicles, and services.

Assumptions and Dependencies

- All required information is in accurate form.
- Quick implementation in new policies.
- Information provided by the hotel database and vehicle database is accurate.
- We made our database on the basis of the indian destinations only.
- All the information provided by the customers are correct and accurate.
- All the reviews provided by the customers regarding tours are correct and with proper narration.
- We assume that the GUI developed by the software engineers for this online booking system is easy to learn and that all the users (customers and organizers) are already familiar with the functionalities of the system.

Business Constraints

- Agency is in touch with hotel management and vehicle management to provide a more reliable traveling experience.
- Creation of digital infrastructure, and fast accessible database server that can handle large amounts of traffic at a time.
- Gathering enough tour guides for all different types of tours.
- Making of tourism policies. Like, pandemic policies regarding guidelines, about refund amount policies while cancellation of the tour, discount applicable policies, and accident policies.
- Past tour information will be saved in the agency database for future reference and suggestion.

Section 2: Noun Analysis

Noun and verb Analysis:

Noun	Verb
Companies	Update
Customers	canceled
Hotels	access
Vehicles	keep
Destination	upload
Tour Packages	handle
Agency	reach
Vacation	implement
Review	alter
Season	aware
Superuser	spread
Hotel room	track
Booking	attract
Website	classify
COVID 19	maintain
Source	include
Transportation management	fetch
Tours	collect
Discounts	make
Tourists	add
History	come
Bus	manage
Railways	Provide
Airplane	read

Surnames	write
Firstname	boarding
Middle name	Analyze
Last name	Visit
Address	delete
Pincode	Booked
Travel Agent	Sort
Database	select
information	expect
Data	centralized
Schedule of tour	accurate
System	quick
Route	decompose
Contact information	classify
Seat	Confirmed
Service	Associated
Rooms	Available
Problems	
Department	
Waiting list	
Feature	
Vaccination status	
Users	
Price	
billing	

Security	
Employee	
Destination type	
visitors	
Boarding time	
Arrival time	
Maintenance	
Departure time	
Booking executive	
Revenue reports	
Stakeholder	
Reservation	
Progress	
IT staff	
Hotel Staff	
Company ID	
Contact details	
Customer ID	
Hotel ID	
Travel	
People	
Business Trip	
Location	
Next service Date	
Employee ID	

Password	
engagement	
Managing Staff	
Business	
evaluate	
Special offer	
individuals	
Tour guidance	
Confirmation	
Pandemic	
Private	
Malicious Activity	
Real-time	
Decision	
Efficient	
Desire	
Vehicle provider	
Summer Vacation	
Discount ID	
Discount name	
No. of tourists	
Booked seat	
Refund amount	
Booking ID	
Billing ID	

Hotel ID	
Hotel room	
Types of rooms	
Types of services	
Hotel Types	
No. of available rooms.centralized	
Maintenance	
Vehicle allocation	
satisfactory	
Destination image link	
Hotel image link	
Start date	
End date	
History	
Tour destinations	
Vehicle providers	
Hotel Providers	
Current tour vehicle	
Membership	
Membership details	
Membership type	
Pin NO.	

Accepted Noun & Verbs list

Candidate entity set	**Candidate Attribute set**	**Candidate relationship set**
Customer	Customer ID	Has
Companies	name	Current Tour Vehicle
Hotels	Contact Details	Tour Destinations
Vehicles	E-mail Address	Review
Destination	Pin	Canceled
Tour Packages	Vaccination status	Confirmed
Booking	Tour ID	History
History	Tour name	Booked
Billing	Tour duration	Associated with
Employee	Tour-start date	Vehicle provider
cancellation	Destination ID	Hotel provider
Discount	Description	Available
Membership	Tour type	Membership Details
Address	Tour Price	
	Destination Name	
	Destination Address	
	Is safe from Covid	
	Destination Type	
	Destination Description	
	Destination Image link	
	Employee name	
	Employee ID	
	Employee password	
	Employee Email Address	
	Employee address	

	Post	
	Hotel ID	
	Hotel name	
	Hotel Address	
	Hotel Description	
	Hotel type	
	Hotel Price	
	Billing ID	
	Booking ID	
	Booking Date	
	Total Price	
	Total booked seat	
	Vehicle ID	
	Booking Type	
	Is confirmed	
	Discount ID	
	Discount Name	
	Start Date	
	End date	
	Percentage	
	Cancellation ID	
	Cancellation Date	
	Refund Amount	
	Vehicle Type	
	Is available	
	Company ID	
	Capacity	
	Next service date	

	Company Name	
	Company service	
	Company contacts No.	
	Company address	
	No. of available Rooms	
	Membership type	
	Membership id	
	Membership price	
	Membership duration	

Rejected nouns and Verbs list

Noun	Reject Reason	Verb	Reject reason
Agency	Irrelevant	Update	vague
Vacation	Irrelevant	cancel	vague
Review	Duplicate	access	vague
Season	Duplicate	keep	general
Superuser	Duplicate	upload	vague
Hotel room	Irrelevant	handle	general
Website	Duplicate	reach	general
COVID 19	Irrelevant	implement	Irrelevant
Source	Irrelevant	alter	vague
Transportation management	vague	aware	general
Tourists	duplicate	spread	general
Bus	vague	track	general
Railways	vague	attract	general

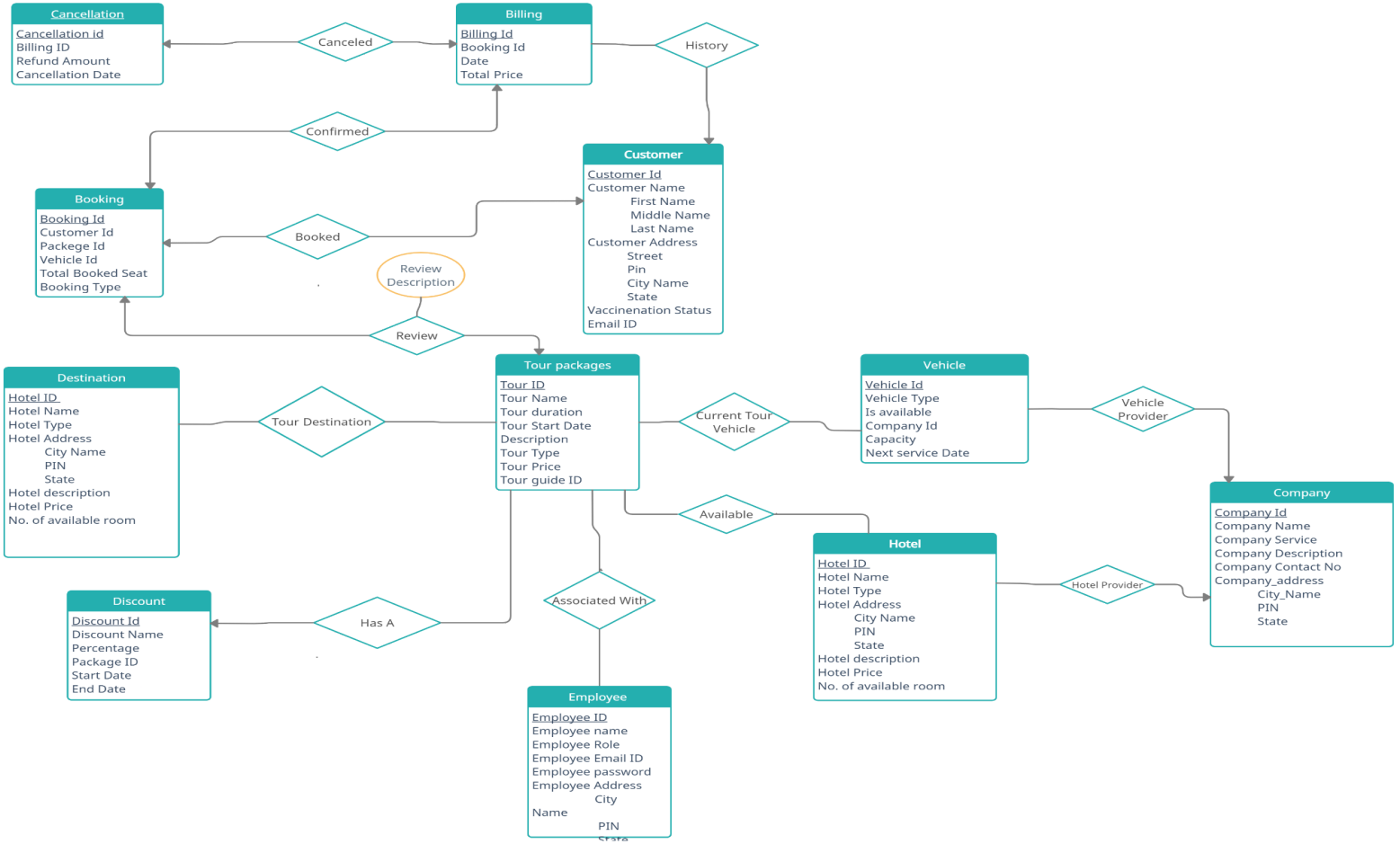
Airplane	vague	classify	vague
Surnames	duplicate	maintain	vague
Travel Agent	vague	include	general
Database	Irrelevant	fetch	vague
information	duplicate	collect	vague
Data	duplicate	make	general
Schedule of tour	general	add	Irrelevant
System	Irrelevant	come	general
Route	Irrelevant	manage	vague
Seat	duplicate	Provide	general
Department	Irrelevant	read	vague
Waiting list	vague	write	vague
Feature	vague	boarding	vague
Users	duplicate	Analyze	Irrelevant
Security	vague	Visit	Irrelevant
visitors	duplicate	delete	vague
Boarding time	vague	provide	general
Arrival time	vague	Sort	vague
Maintenance	Irrelevant	select	general
Departure time	vague	expect	general
Revenue reports	general	centralised	Irrelevant
Stakeholder	Irrelevant	accurate	general
Reservation	Irrelevant	quick	general
Progress	Irrelevant	decompose	vague
IT staff	general	classify	vague
Hotel Staff	general		
Travel	general		
People	duplicate		

Business Trip	Irrelevant		
Location	Duplicate		
engagement	general		
Managing Staff	vague		
Business	Irrelevant		
evaluate	Irrelevant		
Special offer	Duplicate		
individuals	Irrelevant		
Tour guidance	vague		
Confirmation	duplicate		
Pandemic	general		
Private	general		
Malicious Activity	general		
Real-time	general		
Decision	general		
Efficient	general		
Desire	general		
Vehicle provider	duplicate		
Summer Vacation	Irrelevant		
No. of tourists	general		
Vehicle allocation	general		
satisfactory	Irrelevant		

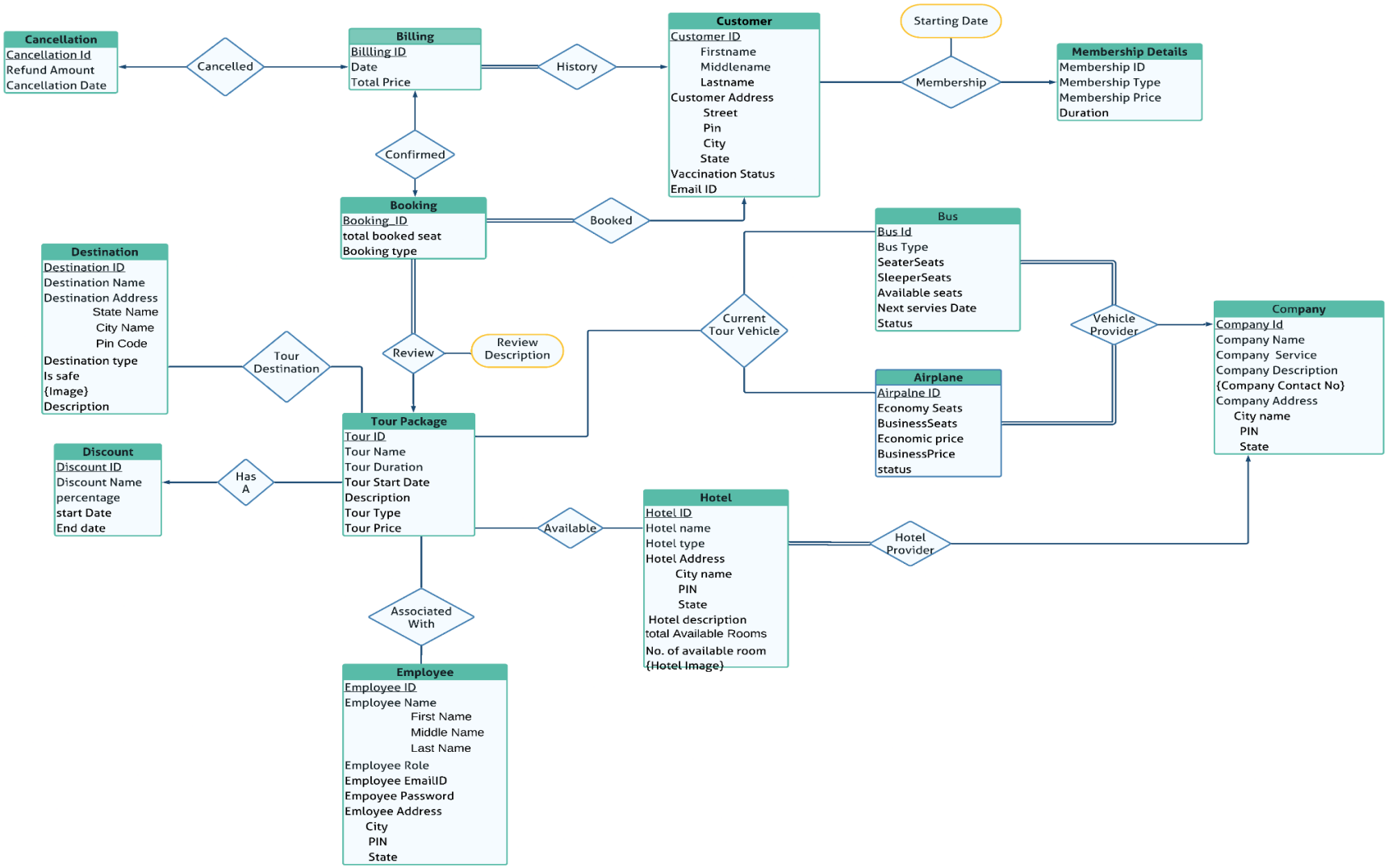
Section 3: ER-Diagram

ER Diagram version 1

S4_T2
ER_DIAGRAM_V1



ER_DIAGRAM V2 AND FINAL VERSION



Section4: Conversion of Final ER-Diagram to Relational Model

Entity type:

- Strong Entity: cancellation, billing, customer, membership details, booking, bus, destination, discount, Tour- package, Bus, Airplane, Company, Hotel, employee.

Types of Relationship:

- All relations are in a simple association link.
    1. Cancellation - Billing: This is one to one relationship. For every billing, there can be at most one cancellation possible. Also, cancellations can't be possible without billing. Every cancellation is associated with a unique billing ID, so it is total participation.

    2. Billing - Customer: This is many to one relationship. One customer can have multiple bills, but one bill can only belong to a single customer. Every bill is associated with at least one customer so that it is total participation.

    3. Customer - Membership details: This is many to one relationship because one customer can have only one type of membership, but one membership can be associated with many customers.

    4. Billing - Booking: This is one to one relationship. For one booking, there will be a single bill showing details about that particular booking. In this case, there will be one booking detail for every billing, which means this is total participation.

    5. Booking - Customer: This is a many-to-one relationship because one customer can book many seats, but one booking id is associated with a unique customer. Every booking is associated with at least one customer, so it is total participation.

    6. Booking - Tour Package: This is a many-to-one relationship because many bookings are associated with a single tour package but with a single booking at most one tour package. Since every booking is associated with at least one tour package, booking is total participation.

    7. Destination-Tour Package: This is a many-to-many relationship. One destination can be associated with multiple tours. One tour may have various destinations.

    8. Discount -Tour Package: This is a one-to-many relationship. One discount can be associated with many tour packages, but one can only be associated with one discount.

    9. Employee -Tour Package: This is a many-to-many relationship. one tour package can have many employees. One employee might be associated with multiple tours.

    10. Tour Package - Hotel: This is a many-to-many relationship. Many hotels are associated with one tour package, and one hotel can be associated with many tour packages.

    11. Hotel-Company: This is a many-to-one relationship. Multiple hotels may belong to a single company. Every hotel must belong to some company so that there is total participation.

    12. Tour Package-Bus: This is a many-to-many relationship because, in one tour package, there can be many buses associated with it, and one bus can be associated with many tour packages.

13. Tour Package-AirPlane: This is a many-to-many relationship because many airplanes can be associated with one tour package, and one airplane can be associated with many tour packages.

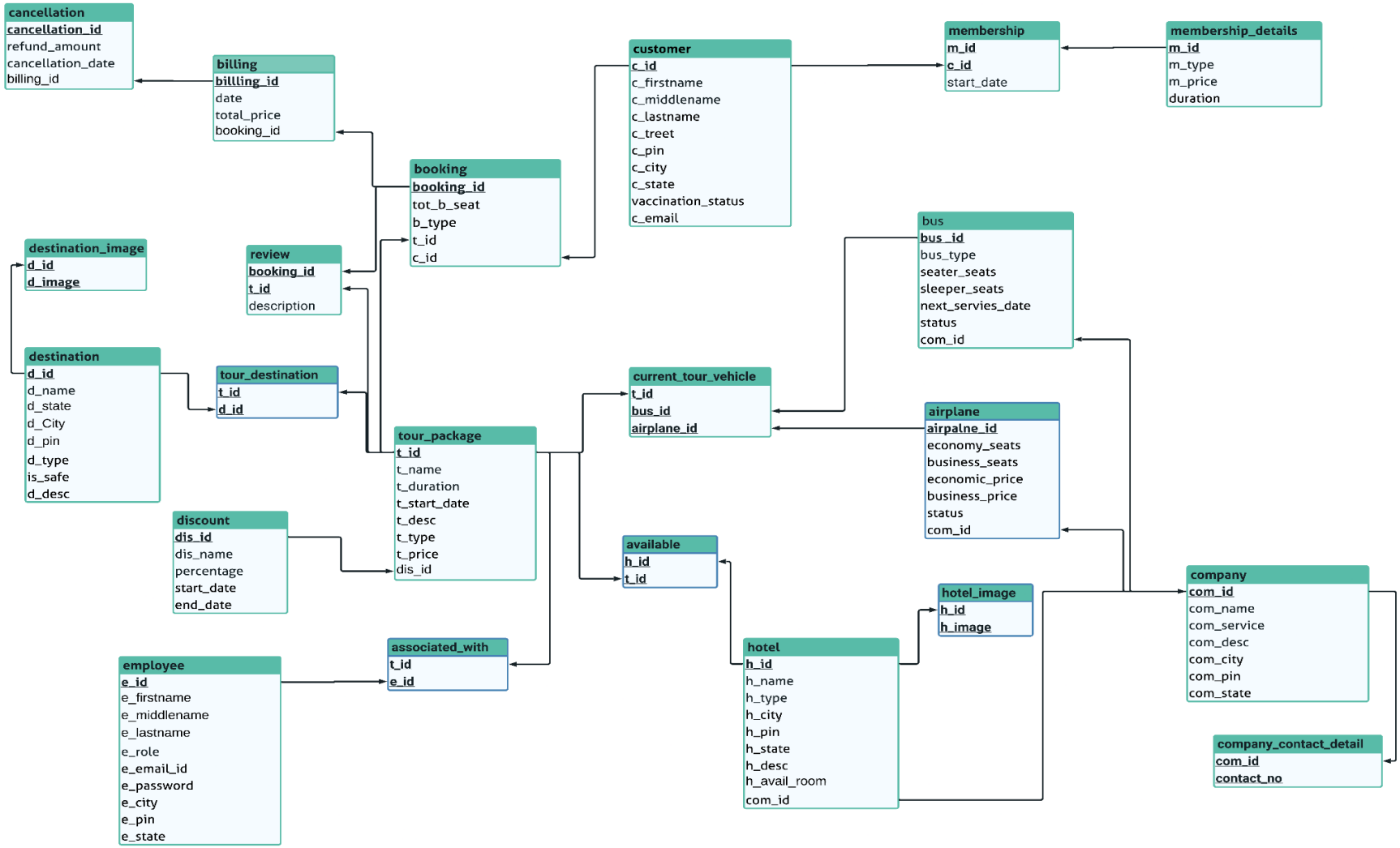
14. Bus-Company: This is a many-to-one relationship because many buses can be associated with one company. But a single bus is associated with a single company. Every Bus is connected with the Company, so it is total participation.

15. AirPlane-Company: This is a many-to-one relationship. Multiple airplanes may belong to a single company. But a single airplane is associated with a single company. Every Airplane must belong to some company so that there is total participation.

Types of Attributes:

Composite	**Multivalued**	**Single Valued**
Name Address	Contact No. Images	ID Date Price Refund Amount Status Type Seats Servies Description Percentage Is Safe Duration Email ID Password Available room Employee Role Duration

Relational Model:



ER to Relational Mapping

- cancellation(**cancellation_id**, billing_id, refund_amount, cancellation_date)
  FK(billing_id) reference billing

- customer(**c_id**, c_firstname, c_middlename, c_lastname, c_street, c_city, c_pin, c_state, vaccination_status, c_email)

- membership(**c_id,** m_id, start_date)
  FK(c_id) reference customer
  FK(m_id) reference membership_details

- billing(**billing_id**, date, price, booking_id)
  FK(booking_Id) reference booking

- membership_details(**m_id**, m_type, m_price, duration)

- booking(**booking_id**, tot_b_seat, b_type, c_id, t_id)
  FK(c_id) reference customer,
  FK(t_id) reference tour_packages

- bus(**bus_id**, bus_type, seater_seats, sleeper_seats, avail_seats, next_service_date, status, com_id)
  FK(com_id) reference company

- destination(**d_id**, d_name, d_state, d_city, d_pin, d_type, is_safe, d_desc)

- destination_image(**d_id**, **d_image**)
  FK(d_id) reference destination

- tour_destination(**t_id , d_id**)
  FK(t_id) reference tour_packages,
  FK(d_id) reference destination

- company(**com_id**, com_name, com_service, com_desc, com_city, com_pin, com_ state)

- company_contact_detail(**com_id**, **contact_no**)
  FK(com_id) reference company

- tour_package(**t_id**, t_name, t_duration, t_start_date, t_price , t_type,t_desc, dis_id)
  FK(dis_id) reference discount

- discount(**dis_id**, dis_name, percentage, start_date, end_date)

- hotel(**h_id**, h_name, h_type, h_city, h_pin, h_state , h_desc, h_avail_room, com_id)
  FK(com_id) reference company

- hotel_image(**h_id**, **h_image**)
  FK(h_id) reference hotel

- airplane(**airplane_id**, com_id, economy_seats, business_seats, economic_price, business_ price, status)
  FK(com_id) reference company

- employee(**e_id**, e_firstname, e_middlename, e_lastname, e_role, e_email_id, e_ password, e_city, e_pin, e_state)

- associated _with(**e_id, t_id**)
  FK(t_id) reference tour_package,
  FK(e_id) reference employee

- review(**t_id, booking_id**,description)
  FK(t_id,booking_id) reference tour_package,booking.
  FK(booking_id) reference booking.

- current_tour_vehicle**(t_id, bus_id, airplane_id,** v_type)
  FK(t_id) reference tour_package
  FK(bus_id) reference bus
  FK(airplane_id) reference airplane

- available(**t_id, h_id)**
  FK(t_id) reference tour_package
  FK(h_id) reference hotel

Section5: Normalization and Schema Refinement.

Normalization

- cancellation(**cancellation_id**, billing_id, cancellation_date, refund_amount)
    - Primary Key: (cancellation_id) → refund_amount, cancellation_date, billing_id
    - Foriegn Key: billing_id
    - Partial dependency: There is no partial dependency because there is only one primary key.
    - Function Dependency: cancellation _id → billing_id, refund_amount, cancellation_date, billing_id
    - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
    - Redundancy: there is no redundancy in this table.
    - Our relation is already in 1NF, and there are no composite and multivalued attributes.
    - 2NF: There is not any partial dependency so that this schema is 2NF.
    - 3NF: There is no transitive dependency, so this is 3NF.
    - BCNF: This relation is in BCNF.
        - cancellation _id → billing_id, refund_amount, cancellation_date.
        - Hence this table is in BCNF.

- customer(**c_id**, c_firstname, c_middlename, c_lastname, c_street, c_city, c_pin, c_state, vaccination_status, c_email)

    - Primary Key:(c_id) → c_firstname, c_middlename, c_lastname, c_street, c_pin, vaccination_status, c_email.
    - Functional dependencies:
        - c_pin → {c_city,c_state}
        - c_id → {c_firstname, c_middlename, c_lastname, c_street, c_pin, vaccination_status, c_email }
    - Partial dependency: There is no partial dependency because there is only one primary key.
    - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
    - Redundancy: there is no redundancy in this table.
    - Our relation is already in 1NF, and there are no composite and multivalued attributes.
    - 2NF: There is not any partial dependency so that this schema is 2NF.
    - 3NF: c_id → c_pin, c_pin → {c_city, c_state} so c_id → {c_city, c_state}
    - Since there is only one CK, BCNF and 3NF are equivalent.

    - Final schema:
        - customer(**c_id**, c_firstname, c_middlename, c_lastname, c_street, vaccination_status, c_email)
        - address(**pin,** city, state)

- membership(**c_id,** m_id, start_date)

    - Primary Key: (c_id) → start_date, m_id, c_id
    - Foriegn Key: m_id,c_id
    - Functional dependencies : c_id → m_id, c_id → start_date
    - Partial dependency: There is no partial dependency because there is only one primary key.
    - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why

there is no update, insert or delete anomaly.
    - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
    - 2NF: There is not any partial dependency so that this schema is 2NF.
    - 3NF: There is no transitive dependency, so this is 3NF.
    - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- billing(**billing_id**, date, total_price, booking_id)
    - Primary Key: (billling_id) → date, total_price, booking_id
    - Foreign Key: booking_id
    - Function dependency: billing_id → {date, total_price, booking_id, booking_id}
    - Partial dependency: There is no partial dependency because there is only one primary key.
    - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
    - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
    - 2NF: There is not any partial dependency so that this schema is 2NF.
    - 3NF: There is no transitive dependency, so this is 3NF.
    - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- membership_details(**m_id**, m_type, m_price, duration)
    - Primary Key: (m_id) → m_type, m_price, duration
    - Function dependency: m_id → {m_type, m_price, duration}
    - Partial dependency: There is no partial dependency because there is only one primary key.
    - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
    - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
    - 2NF: There is not any partial dependency so that this schema is 2NF.
    - 3NF: There is no transitive dependency, so this is 3NF.
    - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- booking(**booking_id**, tot_b_seat, b_type, c_id, t_id)
    - Primary Key: (booking_id) → tot_b_seat,b_type, c_id, t_id
    - Foriegn Key: c_id,t_id
    - Function dependency: booking_id → {tot_b_seat,b_type, t_id, c_id}
    - Partial dependency: There is no partial dependency because there is only one primary key.
    - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
    - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
    - 2NF: There is not any partial dependency so that this schema is 2NF.
    - 3NF: There is no transitive dependency, so this is 3NF.
    - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- bus(**bus_id**, bus_type, seater_seats, sleeper_seats, avail_seats, next_service_date, status, com_id)
    - Primary Key: bus_id → bus_type, seater_seats, sleeper_seats, avail_seats, next_service_date,

status, com_id
- Foreign Key: com_id
- Function dependency: bus_id → { bus_type, seater_seats, sleeper_seats, avail_seats, next_service_date, status, com_id }
- Partial dependency: There is no partial dependency because there is only one primary key.
- Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
- 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
- 2NF: There is not any partial dependency so that this schema is 2NF.
- 3NF: There is no transitive dependency, so this is 3NF.
- Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- destination(**d_id**, d_name, d_state, d_city, d_pin, d_type, is_safe, d_desc)
  - Primary Key: d_id → d_name, d_pin, d_type, is_safe, d_desc
  - Function dependency:
    - d_pin→ {d_pin, d_state}
    - d_id → {d_name, d_pin, d_type, is_safe, d_desc}
  - Partial dependency: There is no partial dependency because there is only one primary key.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - Redundancy: there is no redundancy in this table.
  - Our relation is already in 1NF, and there are no composite and multivalued attributes.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: d_id → d_pin, d_pin → {d_city, d_state} so d_id → {d_city, d_state}
  - Since there is only one CK, BCNF and 3NF are equivalent.
  - Final schema:
    - destination(**d_id**, d_name, d_pin, d_type, is_safe, d_desc)
    - address(**pin,** city, state)

- destination_image(**d_id, d_image**)
  - Primary Key: d_id, d_image
  - Foriegn Key: d_id, d_image
  - Function dependency: There is no function dependency.
  - Partial dependency: There is no partial dependency because there is not any non-prime attribute.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: There is no transitive dependency, so this is 3NF.
  - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- tour_destination(**t_id , d_id**)
  - Primary Key: d_id, t_id
  - Foriegn Key: d_id, t_id
  - Function dependency: There is no functional dependency.

- Partial dependency: There is no partial dependency because there is not any non-prime attribute.
- Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
- 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
- 2NF: There is not any partial dependency so that this schema is 2NF.
- 3NF: There is no transitive dependency, so this is 3NF.
- Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- company(**com_id**, com_name, com_service, com_desc, com_city, com_pin, com_ state)
  - Primary Key: com_id → com_name, com_service, com_desc, com_pin
  - Function dependency:
    - com_id → {com_name, com_service, com_desc, com_pin}
    - com_pin → {com_city, com_state}
  - Partial dependency: There is no partial dependency because there is only one primary key.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - Redundancy: there is no redundancy in this table.
  - Our relation is already in 1NF, and there are no composite and multivalued attributes.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: com_id → com_pin, com_pin → {com_city, com_state} so com_id → {com_city, com_state}
  - Since there is only one CK, BCNF and 3NF are equivalent.
  - Final schema:
    - company (**com_id**, com_service, com_desc, com_pin)
    - address(**pin,** city, state)

- company_contact_detail(**com_id**, **contact_no**)
  - Primary Key: com_id,contact_no
  - Foriegn Key: com_id,contact_no
  - Function dependency: There is no functional dependency.
  - Partial dependency: There is no partial dependency because there is not any non-prime attribute.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: There is no transitive dependency, so this is 3NF.
  - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- tour_package(**t_id**, t_name, t_duration, t_start_date, t_price , t_type,t_desc, dis_id)
  - Primary Key: t_id → t_name, t_duration, t_start_date, t_price , t_type,t_desc, dis_id
  - Foreign Key: dis_id
  - Function dependency: t_id → {t_name, t_duration, t_start_date, t_price , t_type,t_desc, dis_id}

- Partial dependency: There is no partial dependency because there is only one CK.
- Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
- 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
- 2NF: There is not any partial dependency so that this schema is 2NF.
- 3NF: There is no transitive dependency, so this is 3NF.
- Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- discount(**dis_id**, dis_name, percentage, start_date, end_date)
  - Primary Key: dis_id → dis_name, percentage, start_date, end_date
  - Partial dependency: There is no partial dependency because there is only one CK.
  - Functional dependency:
    - dis_id → {dis_name, percentage, start_date, end_date}
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: There is no transitive dependency, so this is 3NF.
  - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- hotel(**h_id**, h_name,h_type, h_city, h_pin, h_state ,h_desc, h_avail_room, com_id)
  - Primary Key: h_id → h_name,h_type, h_pin,h_desc, h_avail_room, com_id
  - Foreign Key: com_id
  - Function dependency:
    - h_id → h_name,h_type, h_pin,h_desc, h_avail_room, com_id
    - h_pin → { h_city, h_state}
  - Partial dependency: There is no partial dependency because there is only one CK.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: h_id → h_pin and h_pin → { h_city, h_state} so that h_id → { h_city, h_state}
  - Final schema:
    - hotel(**h_id**, h_name,h_type, h_pin, h_desc, h_avail_room, com_id)
    - address(**pin,** city, state)

- hotel_image(**h_id, h_image**)
  - Primary Key: h_id,h_image
  - Foreign Key: h_id
  - Functional dependency: There is no functional dependency.
  - Partial dependency: There is no partial dependency because there is not any non-prime attribute.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.

- 3NF: There is no transitive dependency, so this is 3NF.
- Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- airplane(**airplane_id**, com_id, economy_seats, business_seats, economic_price, business_ price, status)
  - Primary Key: airplane_id → economy_seats, business_seats, economic_price, business_ price, status, com_id
  - Foreign Key: com_id
  - Functional dependency:
    - Airplane_id→{com_id,economy_seats,business_seats, economic_price, business_ price, status}
  - Partial dependency: There is no partial dependency because there is only one CK.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: There is no transitive dependency, so this is 3NF.
  - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- employee(**e_id**, e_firstname, e_middlename, e_lastname, e_role, e_email_id, e_ password, e_city, e_pin, e_state)
  - Primary Key: e_id → e_firstname, e_middlename, e_lastname, e_role, e_email_id, e_ password, e_pin
  - Functional dependency:
    - e_id→{e_firstname,e_middlename,e_lastname, e_role, e_email_id, e_ password, e_pin}
    - e_pin→{ e_city,e_state}
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: e_id → e_pin and e_pin → {e_city, e_state} so that e_id → {e_city, e_state}
    - There is transitive dependency.
  - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.
  - Final schema:
    - employee(**e_id**, e_firstname, e_middlename, e_lastname, e_role, e_email_id, e_ password, e_pin)
    - address(**pin,** city, state)

- associated _with(**e_id, t_id**)
  - Primary Key: e_id, t_id
  - Foriegn Key: e_id, t_id
  - Functional dependency: There is no functional dependency.
  - Partial dependency: There is no partial dependency because there is not any non-prime attribute.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.

- 2NF: There is not any partial dependency so that this schema is 2NF.
- 3NF: There is no transitive dependency, so this is 3NF.
- Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

- review(**t_id, booking_id**,description)
  - Primary Key: (booking_id, t_id) → description
  - Foreign Key: booking_id, t_id
  - Functional dependency:
    - {t_id, booking_id}→ description
  - Partial dependency: There is no partial dependency because there is not any non-prime attribute.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: There is no transitive dependency, so this is 3NF.
  - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

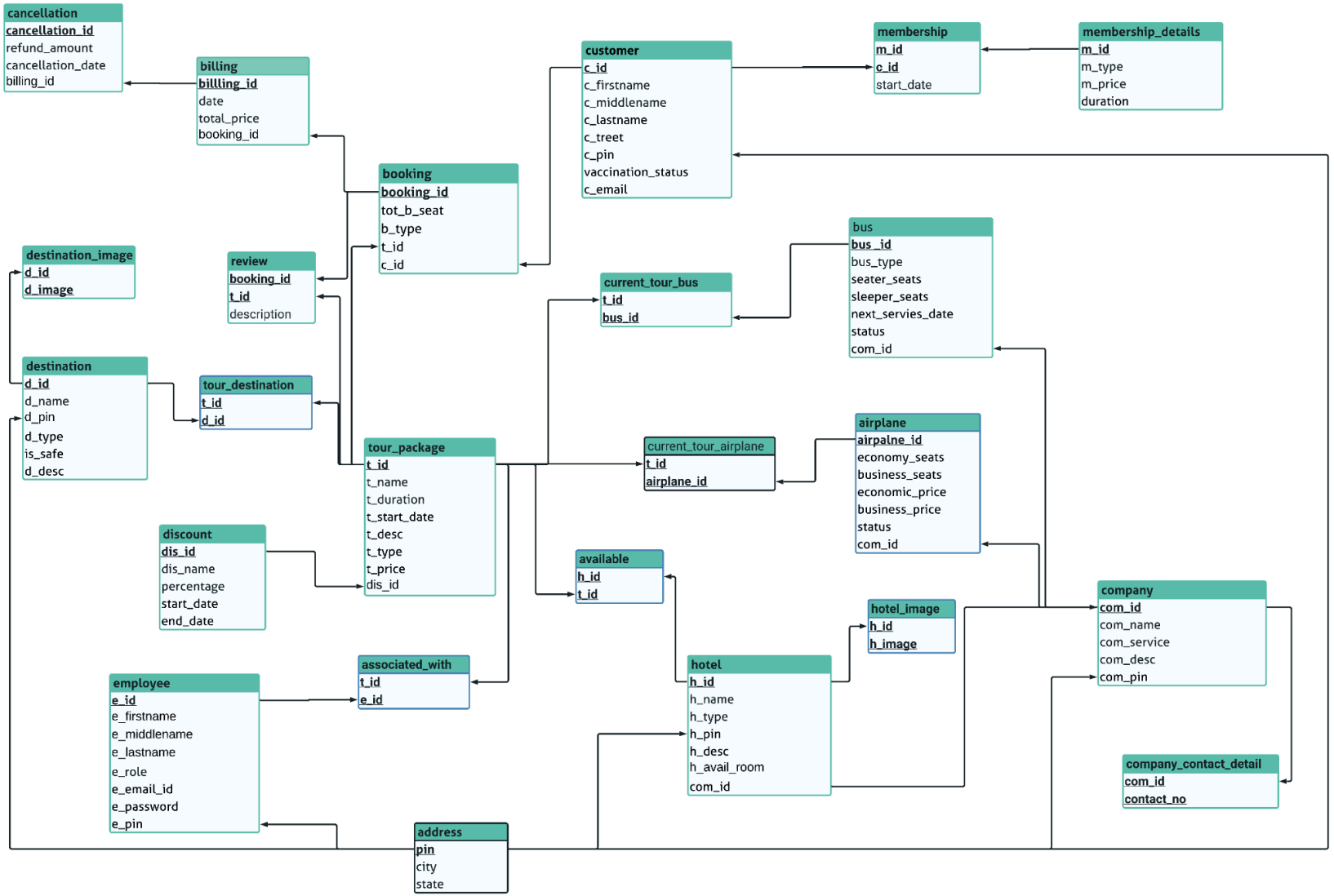
- current_tour_vehicle**(t_id, bus_id, airplane_id**)
  - Primary Key: (t_id, bus_id, airpalane_id)
  - Foriegn key: t_id, bus_id, airpalane_id
  - Functional dependency:There is no functional dependency.
  - Partial dependency: There is no partial dependency because there is not any non-prime attribute.
  - Anomalies: We can't insert bus and airplane together because some tours may not have a bus or airplane as a travel option. So there is insert anomaly.
    - current_tour_bus(**t_id, bus_id)**
    - current_tour_airplane(**t_id, airplane_id)**
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: There is no transitive dependency, so this is 3NF.
  - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.
  - Final schemas:
    - current_tour_bus(**t_id, bus_id)**
    - current_tour_airplane(**t_id, airplane_id)**

- available(**t_id, h_id)**
  - Primary Key: t_id, h_id
  - Foriegn Key: t_id, h_id
  - Functional dependency: There is no functional dependency.
  - Partial dependency: There is no partial dependency because there is only one CK.
  - Anomalies: we can update, insert, or delete without changing unnecessary information. That's why there is no update, insert or delete anomaly.
  - 1NF: There are no multivalued or composite attributes in this relation, so it is already 1NF.
  - 2NF: There is not any partial dependency so that this schema is 2NF.
  - 3NF: There is no transitive dependency, so this is 3NF.
  - Since there is only one CK, BCNF and 3NF are equivalent. For BCNF, there should be two or more CK.

Final Schemas:

- cancellation(**cancellation_id**, billing_id, cancellation_date, refund_amount)
- customer(**c_id**, c_firstname, c_middlename, c_lastname, c_street, c_pin, vaccination_status, c_email)
- membership(**c_id,** m_id, start_date)
- billing(**billing_id**, date, total_price, booking_id)
- membership_details(**m_id**, m_type, m_price, duration)
- booking(**booking_id**, tot_b_seat, b_type, c_id, t_id)
- bus(**bus_id**, bus_type, seater_seats, sleeper_seats, avail_seats, next_service_date, status, com_id)
- destination(**d_id**, d_name, d_pin, d_type, is_safe, d_desc)
- destination_image(**d_id**, **d_image**)
- tour_destination(**t_id , d_id**)
- company (**com_id**, com_name, com_service, com_desc, com_pin)
- company_contact_detail(**com_id**, **contact_no**)
- tour_package(**t_id**, t_name, t_duration, t_start_date, t_price , t_type,t_desc, dis_id)
- discount(**dis_id**, dis_name, percentage, start_date, end_date)
- hotel(**h_id**, h_name,h_type, h_pin, h_desc, h_avail_room, com_id)
- hotel_image(**h_id**, **h_image**)
- airplane(**airplane_id**, com_id, economy_seats, business_seats, economic_price, business_ price, status)
- employee(**e_id**, e_firstname, e_middlename, e_lastname, e_role, e_email_id, e_ password, e_pin)
- address(**pin,** city, state)
- associated _with(**e_id, t_id**)
- review(**t_id, booking_id**,description)
- current_tour_bus(**t_id, bus_id)**
- current_tour_airplane(**t_id, airplane_id)**
- available(**t_id, h_id)**

Final Relational diagram after Normalization



Section 6: SQL

DDL scripts:

```
CREATE TABLE address
(
    pin integer NOT NULL,
    city character varying(50) NOT NULL,
    state character varying(50) NOT NULL,
    PRIMARY KEY (pin)
);
```

```
CREATE TABLE customer
(
    c_id integer NOT NULL,
    c_firstname character varying(50) NOT NULL,
    c_middlename character varying(50),
    c_lastname character varying(50) NOT NULL,
    c_email character varying(150) NOT NULL,
    c_street character varying(100),
    vaccination_status boolean DEFAULT false,
    c_pin integer NOT NULL,
    PRIMARY KEY (c_id),
    FOREIGN KEY (c_pin)
        REFERENCES address(pin)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

```
CREATE TABLE discount
(
    dis_id integer NOT NULL,
    dis_name character varying(30) NOT NULL,
    percentage real DEFAULT 0,
    start_date date NOT NULL,
    end_date date,
    PRIMARY KEY (dis_id)
);
```

```
CREATE TABLE destination
(
    d_id integer,
    d_name character varying(100) NOT NULL,
    d_type character varying(20),
    is_safe boolean DEFAULT true,
```

```
    d_pin integer NOT NULL,
    d_desc text,
    PRIMARY KEY (d_id),
    FOREIGN KEY (d_pin)
        REFERENCES address(pin)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
            CHECK (d_type in ('Historical', 'Adventure', 'Hill station', 'Beach', 'Religious Place','Amusement park',
'Mountain', 'Forests'))
);
```

```
CREATE TABLE membership_details
(
    m_id integer NOT NULL,
    m_type character varying(30) NOT NULL,
    m_price integer NOT NULL,
    duration integer NOT NULL,
    PRIMARY KEY (m_id)
);
```

```
CREATE TABLE company
(
    com_id integer NOT NULL,
    com_name character varying(100) NOT NULL,
    com_service character varying(50) NOT NULL,
    com_desc text,
    com_pin integer NOT NULL,
    FOREIGN KEY (com_pin)
        REFERENCES address(pin)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    PRIMARY KEY (com_id)
);
```

```
CREATE TABLE employee
(
    e_id integer NOT NULL,
    e_firstname character varying(20) NOT NULL,
    e_middlename character varying(20),
    e_lastname character varying(20) NOT NULL,
    e_pin integer NOT NULL,
    e_password character varying(30) NOT NULL,
    e_email character varying(50) NOT NULL,
    e_role character varying(30) NOT NULL,
            FOREIGN KEY (e_pin)
```

```
        REFERENCES address(pin)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    PRIMARY KEY (e_id)
);
```

```
CREATE TABLE hotel
(
    h_id integer NOT NULL,
    h_name character varying(50) NOT NULL,
    h_type character varying(20) NOT NULL,
    h_pin integer NOT NULL,
    h_desc text,
    avail_room integer NOT NULL,
    com_id integer,
    PRIMARY KEY (h_id),
    FOREIGN KEY (com_id)
        REFERENCES company (com_id)
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    FOREIGN KEY (h_pin)
        REFERENCES address(pin)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
     CHECK(h_type in ('1 STAR', '2 STAR', '3 STAR', '4 STAR', '5 STAR'))
);
```

```
CREATE TABLE tour_package
(
    t_id integer NOT NULL,
    t_name character varying(50) NOT NULL,
    t_duration character varying(30) NOT NULL,
    t_start_date date NOT NULL,
    t_price real NOT NULL,
    t_type character varying(30),
    t_desc text,
    dis_id integer,
    PRIMARY KEY (t_id),
    FOREIGN KEY (dis_id)
        REFERENCES discount (dis_id)
        ON UPDATE SET NULL
        ON DELETE SET NULL
);
```

```
CREATE TABLE booking
(
   booking_id integer NOT NULL,
   c_id integer NOT NULL,
   t_id integer NOT NULL,
   tot_b_seat integer DEFAULT 1,
   b_type character varying(10) DEFAULT 'offline',
   PRIMARY KEY (booking_id),
   FOREIGN KEY (t_id)
      REFERENCES tour_package (t_id)
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
   FOREIGN KEY (c_id)
      REFERENCES customer(c_id)
      ON UPDATE NO ACTION
      ON DELETE CASCADE,
   check(b_type in ('online', 'offline'))
);
```

```
CREATE TABLE billing
(
   billing_id integer NOT NULL,
   booking_id integer,
   date date NOT NULL,
   price real,
   PRIMARY KEY (billing_id),
   FOREIGN KEY (booking_id)
      REFERENCES booking (booking_id)
      ON UPDATE CASCADE
      ON DELETE CASCADE
);
```

```
CREATE TABLE cancellation
(
   cancellation_id integer NOT NULL,
   billing_id integer,
   cancellation_date date NOT NULL,
   refund_amount real NOT NULL,
   PRIMARY KEY (cancellation_id),
   FOREIGN KEY (billing_id)
      REFERENCES billing (billing_id)
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
);
```

```
CREATE TABLE bus
(
   bus_id integer NOT NULL,
   com_id integer NOT NULL,
   seater_seats integer NOT NULL,
   sleeper_seats integer NOT NULL,
   bus_type character varying(100) NOT NULL,
   next_service_date date NOT NULL,
   avail_seats integer DEFAULT 0,
   status boolean DEFAULT FALSE,
   PRIMARY KEY (bus_id),
   FOREIGN KEY (com_id)
      REFERENCES company (com_id)
      ON UPDATE NO ACTION
      ON DELETE CASCADE

);
```

```
CREATE TABLE airplane
(
   airplane_id integer NOT NULL,
   com_id integer NOT NULL,
   economy_seats integer NOT NULL,
   business_seats integer NOT NULL,
   economy_price real NOT NULL,
   business_price real NOT NULL,
   status boolean DEFAULT true,
   PRIMARY KEY (airplane_id),
   FOREIGN KEY (com_id)
      REFERENCES company (com_id)
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
);
```

```
CREATE TABLE destination_image
(
   d_id integer NOT NULL,
   d_image character varying(300) NOT NULL,
   PRIMARY KEY (d_id, d_image),
   FOREIGN KEY (d_id)
      REFERENCES destination (d_id)
      ON UPDATE CASCADE
      ON DELETE CASCADE
 );
```

```
CREATE TABLE hotel_image
(
    h_id integer NOT NULL,
    h_image character varying(300) NOT NULL,
    PRIMARY KEY (h_id, h_image),
    FOREIGN KEY (h_id)
        REFERENCES hotel (h_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

```
CREATE TABLE company_contact_details
(
    com_id integer NOT NULL,
    contact_no decimal(10,0) NOT NULL,
    PRIMARY KEY (com_id, contact_no),
    FOREIGN KEY (com_id)
        REFERENCES company (com_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
);
```

```
CREATE TABLE available
(
    t_id integer NOT NULL,
    h_id integer NOT NULL,
    PRIMARY KEY (t_id, h_id),
    FOREIGN KEY (t_id)
        REFERENCES tour_package (t_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (h_id)
        REFERENCES hotel (h_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

```
CREATE TABLE review
(
    booking_id integer NOT NULL,
    t_id integer NOT NULL,
    description text,
    PRIMARY KEY (booking_id, t_id),
    FOREIGN KEY (booking_id)
```

```
    REFERENCES booking (booking_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  FOREIGN KEY (t_id)
    REFERENCES tour_package (t_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
```

```
CREATE TABLE associated_with
(
  e_id integer NOT NULL,
  t_id integer NOT NULL,
  PRIMARY KEY (e_id, t_id),
  FOREIGN KEY (e_id)
    REFERENCES employee (e_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  FOREIGN KEY (t_id)
    REFERENCES tour_package (t_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
```

```
CREATE TABLE tour_destination
(
  t_id integer NOT NULL,
  d_id integer NOT NULL,
  PRIMARY KEY (t_id, d_id),
  FOREIGN KEY (t_id)
    REFERENCES tour_package (t_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
  FOREIGN KEY (d_id)
    REFERENCES destination (d_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
```

```
CREATE TABLE membership
(
  m_id integer NOT NULL,
  c_id integer NOT NULL,
  start_date date NOT NULL,
  PRIMARY KEY (c_id),
  FOREIGN KEY (c_id)
```

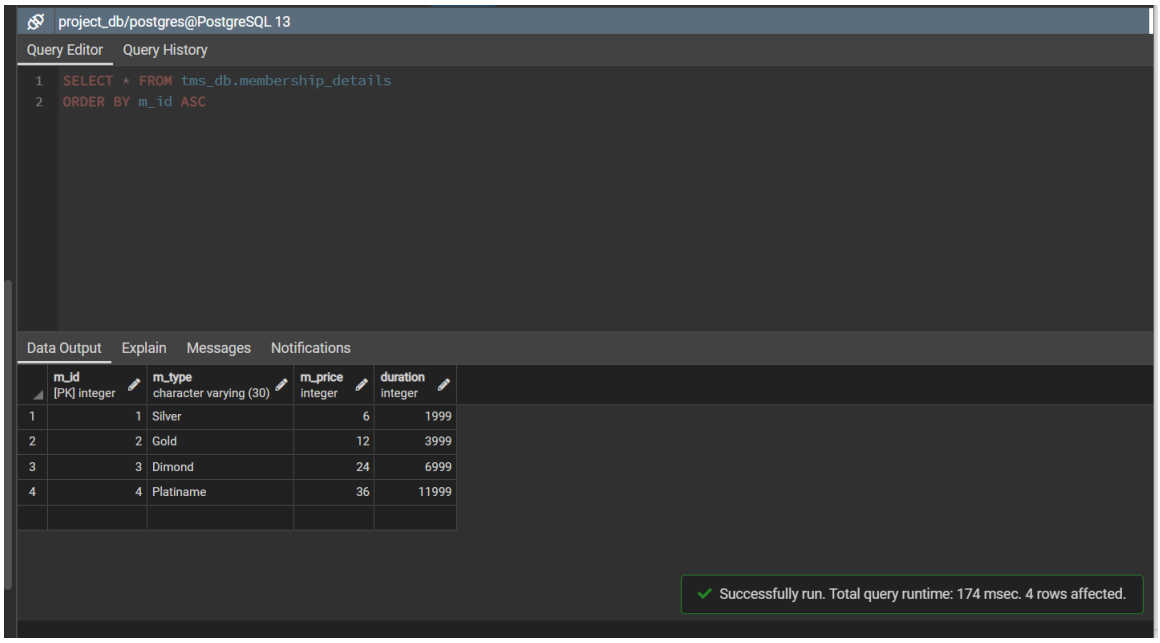
```
    REFERENCES customer (c_id)
    ON UPDATE NO ACTION
    ON DELETE CASCADE,
  FOREIGN KEY (m_id)
    REFERENCES membership_details (m_id)
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
);
```

```
CREATE TABLE current_tour_bus
(
  bus_id integer NOT NULL,
  t_id integer NOT NULL,
  PRIMARY KEY (bus_id, t_id),
  FOREIGN KEY (bus_id)
    REFERENCES bus (bus_id)
    ON UPDATE NO ACTION
    ON DELETE NO ACTION,
  FOREIGN KEY (t_id )
    REFERENCES tour_package(t_id )
    ON UPDATE NO ACTION
    ON DELETE NO ACTION

);
```

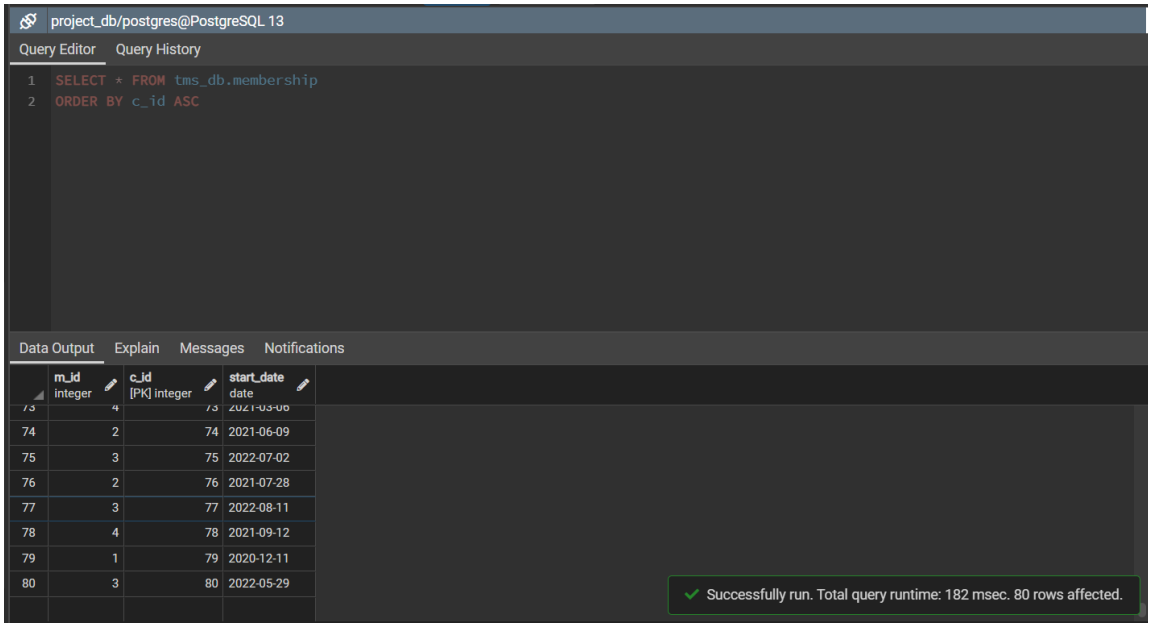
```
CREATE TABLE current_tour_airplane
(
  airplane_id integer NOT NULL,
  t_id integer NOT NULL,
  PRIMARY KEY (airplane_id, t_id),
   FOREIGN KEY (airplane_id )
    REFERENCES airplane(airplane_id )
    ON UPDATE NO ACTION
    ON DELETE NO ACTION,
  FOREIGN KEY (t_id )
    REFERENCES tour_package(t_id )
    ON UPDATE NO ACTION
    ON DELETE NO ACTION

);
```

Inserted Data

- membership_details:
  No. of tuples 4.



- membership:
  No. of tuples: 80



- employee:
  No. of tuples: 80

project_db/postgres@PostgreSQL 13

Query Editor | Query History

```
SELECT * FROM tms_db.employee
ORDER BY e_id ASC
```

Data Output | Explain | Messages | Notifications

	e_id [PK] integer	e_firstname character varying (20)	e_middlename character varying (20)	e_lastname character varying (20)	e_pin integer	e_password character varying (30)	e_email character varying (50)	e_role character varying (30)
74	74	Kim	Dhiraj	Balasubramanian	713101	KimDhiraj@3100	KimDhiraj@gmail.com	Management Staff
75	75	Bhairavi	Hemendra	Kumer	722101	BhairaviHemendra@7384	BhairaviHemendra@gmail.com	Booking Executive
76	76	Yasmin	Javed	Murty	754223	YasminJaved@2040	YasminJaved@gmail.com	Tour Guide
77	77	Rohini	Manpreet	Mogul	756056	RohiniManpreet@3481	RohiniManpreet@gmail.com	Tour Guide
78	78	Deep	Malik	Bhattacharyya	756100	DeepMalik@3136	DeepMalik@gmail.com	Tour Guide
79	79	Anees	Rupal	Shah	759021	AneesRupal@2081	AneesRupal@gmail.com	Travel Agent
80	80	Arpit	Daanish	Anthony	767001	ArpitDaanish@518	ArpitDaanish@gmail.com	Booking Executive

- destination:
  No. of tuples: 92

project_db/postgres@PostgreSQL 13

Query Editor | Query History

```
SELECT * FROM tms_db.destination
ORDER BY d_id ASC
```

Data Output | Explain | Messages | Notifications

	d_id [PK] integer	d_name character varying (100)	d_type character varying (20)	is_safe boolean	d_pin integer	d_desc text
79	79	Munnar Hills, Idukki	Mountain	true	759021	Old Goa, also called Velha Goa, is beside the Mandovi River in the
80	80	Loktak Lake, Moirang	Mountain	true	767001	Jaisalmer - The name Jaisalmer evokes a vivid picture of sheer m
81	81	Leh Ladakh – Land of the High Passes.	Hill station	true	768027	The Red Fort or Lal Qila is a historic fort in Old Delhi, Delhi in India
82	82	Nainital, Uttarakhand – The Hill Station of Lakes.	Hill station	true	781301	Kutch district is a district of Gujarat state in western India, with its
83	83	Manali, Himachal Pradesh – Valley of the Gods.	Hill station	true	781313	Pangong Tso or Pangong Lake is an endorheic lake spanning eas
84	84	Mussoorie, Uttarakhand – The Queen of the Hill Stations.	Hill station	true	796001	Valley of Flowers National Park is an Indian national park, locatec
85	85	Gulmarg, Jammu and Kashmir	Hill station	true	796015	Jaisalmer - The name Jaisalmer evokes a vivid picture of sheer m
86	86	Dash n Splash (closed)	Amusement park	true	799011	The Red Fort or Lal Qila is a historic fort in Old Delhi, Delhi in India
87	87	EVP World	Amusement park	true	804401	Kutch district is a district of Gujarat state in western India, with its
88	88	Kishkinta	Amusement park	true	811106	Pangong Tso or Pangong Lake is an endorheic lake spanning eas
89	89	MGM Dizzee World	Amusement park	true	811211	Old Goa, also called Velha Goa, is beside the Mandovi River in the
90	90	Queen's Land	Amusement park	true	813101	Jaisalmer - The name Jaisalmer evokes a vivid picture of sheer m
91	91	Snow Kingdom	Amusement park	true	824301	The Red Fort or Lal Qila is a historic fort in Old Delhi, Delhi in India
92	92	VGP Universal Kingdom	Amusement park	true	854311	Kutch district is a district of Gujarat state in western India, with its

- hotels:
  No. of tuples: 80

project_db/postgres@PostgreSQL 13

Query Editor | Query History

```
SELECT * FROM tms_db.hotel
ORDER BY h_id ASC
```

Data Output | Explain | Messages | Notifications

	h_id [PK] integer	h_name character varying (50)	h_type character varying (20)	h_pin integer	h_desc text
67	2567	The White Rock Hotel	4 STAR	583101	36 smoke-free guestrooms
68	2568	The Hot Springs Hotel	5 STAR	583155	Prices you can't beat!
69	2569	Venture Hotel	3 STAR	587101	Manage your bookings online,
70	2570	The Lakefront	3 STAR	590001	Welcoming Booking.com guests since 19 Jul 2011
71	2571	The Peninsula	3 STAR	591304	Private check-in/check-out
72	2572	The Lakefront	3 STAR	608901	Daily housekeeping
73	2573	Always Welcome	5 STAR	686102	best price for 3 star properties.
74	2574	Tower Hotel	4 STAR	713101	highly reated by guests 100% would recommended
75	2575	Sunny Canopy	4 STAR	722101	36 smoke-free guestrooms
76	2576	Royal Galaxy	4 STAR	754223	Prices you can't beat!
77	2577	The Watson Hotel	4 STAR	756056	Manage your bookings online,
78	2578	Treebones Resort	4 STAR	756100	Welcoming Booking.com guests since 19 Jul 2011

- bus:
  No. of tuples: 100

project_db/postgres@PostgreSQL 13

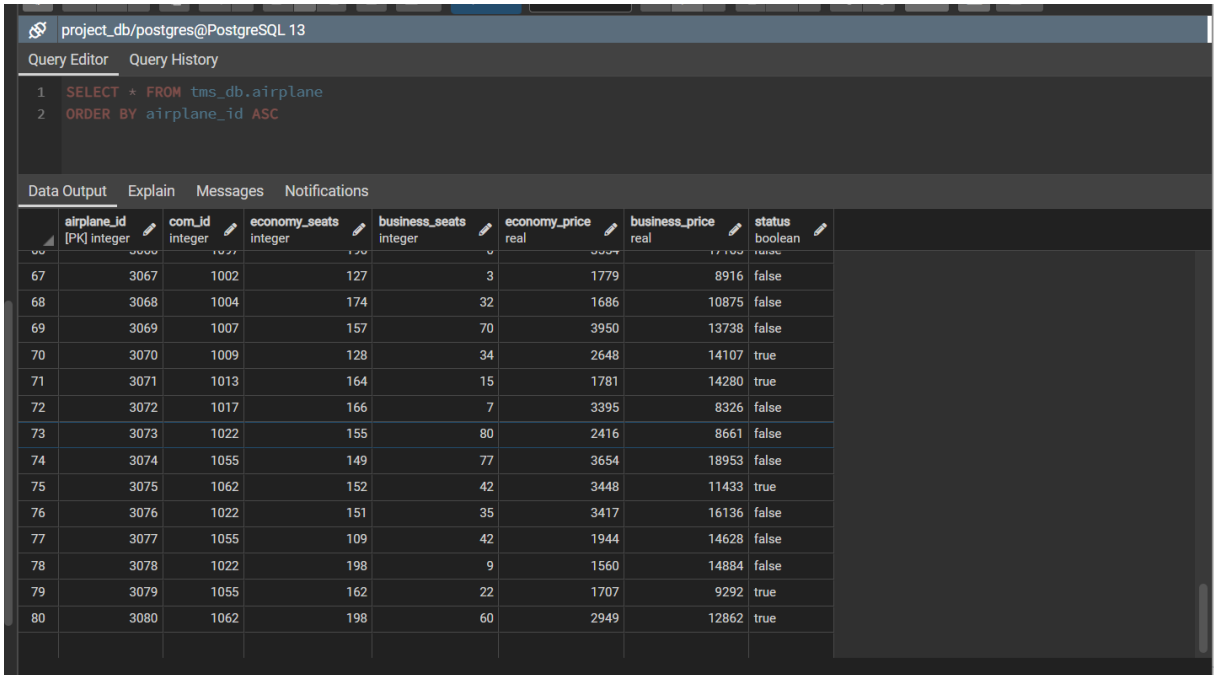
Query Editor | Query History

```
SELECT * FROM tms_db.bus
ORDER BY bus_id ASC
```

Data Output | Explain | Messages | Notifications

	bus_id [PK] integer	com_id integer	seater_seats integer	sleeper_seats integer	bus_type character varying (100)	next_service_date date	avail_seats integer	status boolean
87	22087	1012	0	30	AC	2022-03-26	30	false
88	22088	1032	0	30	AC	2021-11-21	30	true
89	22089	1033	55	0	Non AC	2022-06-28	55	false
90	22090	1034	55	0	Non AC	2021-05-24	55	false
91	22091	1035	30	19	Non AC	2020-12-09	49	false
92	22092	1036	30	19	Non AC	2020-12-04	49	true
93	22093	1037	55	0	Non AC	2022-02-07	55	false
94	22094	1033	55	0	Non AC	2022-01-10	55	false
95	22095	1034	55	0	Non AC	2021-12-24	55	false
96	22096	1033	55	0	Non AC	2022-09-16	55	true
97	22097	1034	0	30	AC	2020-12-28	30	false
98	22098	1033	0	30	AC	2021-03-23	30	false
99	22099	1034	0	30	AC	2022-01-16	30	true
100	22100	1006	55	0	Non AC	2022-03-07	55	false

- airplane:
  No. of tuples: 80



project_db/postgres@PostgreSQL 13

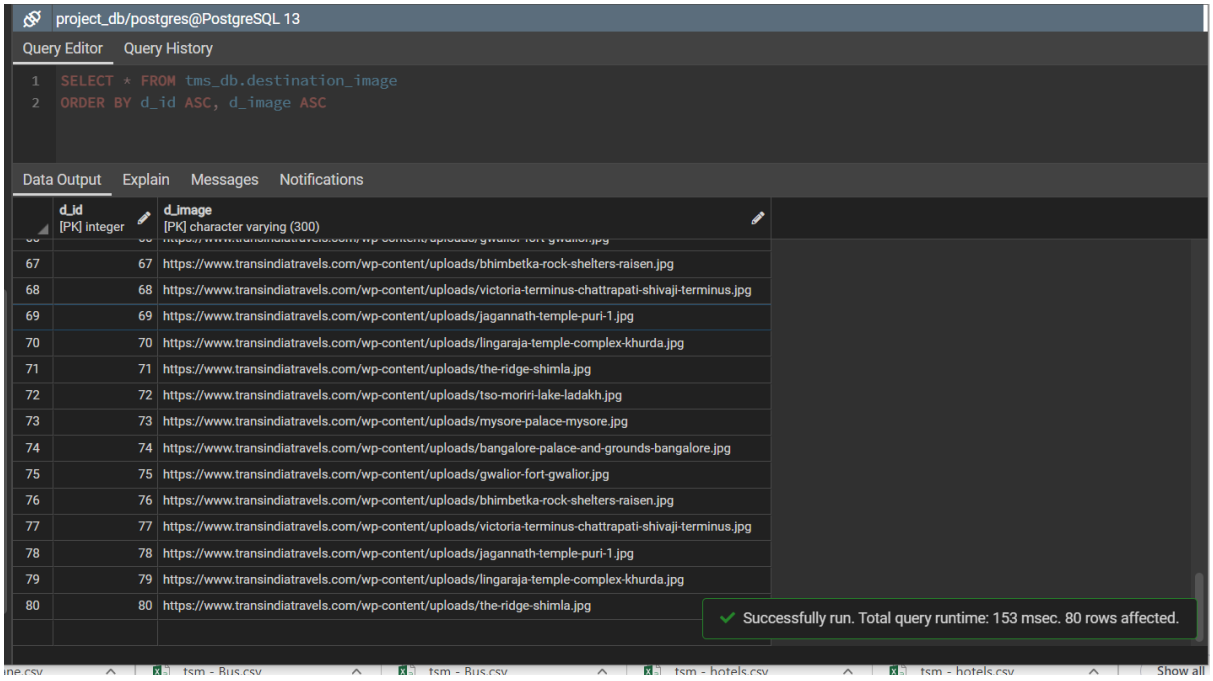
Query Editor | Query History

```
SELECT * FROM tms_db.airplane
ORDER BY airplane_id ASC
```

Data Output | Explain | Messages | Notifications

	airplane_id [PK] integer	com_id integer	economy_seats integer	business_seats integer	economy_price real	business_price real	status boolean
67	3067	1002	127	3	1779	8916	false
68	3068	1004	174	32	1686	10875	false
69	3069	1007	157	70	3950	13738	false
70	3070	1009	128	34	2648	14107	true
71	3071	1013	164	15	1781	14280	true
72	3072	1017	166	7	3395	8326	false
73	3073	1022	155	80	2416	8661	false
74	3074	1055	149	77	3654	18953	false
75	3075	1062	152	42	3448	11433	true
76	3076	1022	151	35	3417	16136	false
77	3077	1055	109	42	1944	14628	false
78	3078	1022	198	9	1560	14884	false
79	3079	1055	162	22	1707	9292	true
80	3080	1062	198	60	2949	12862	true

- Destination_images:
  No. of tuples: 92



project_db/postgres@PostgreSQL 13

Query Editor | Query History

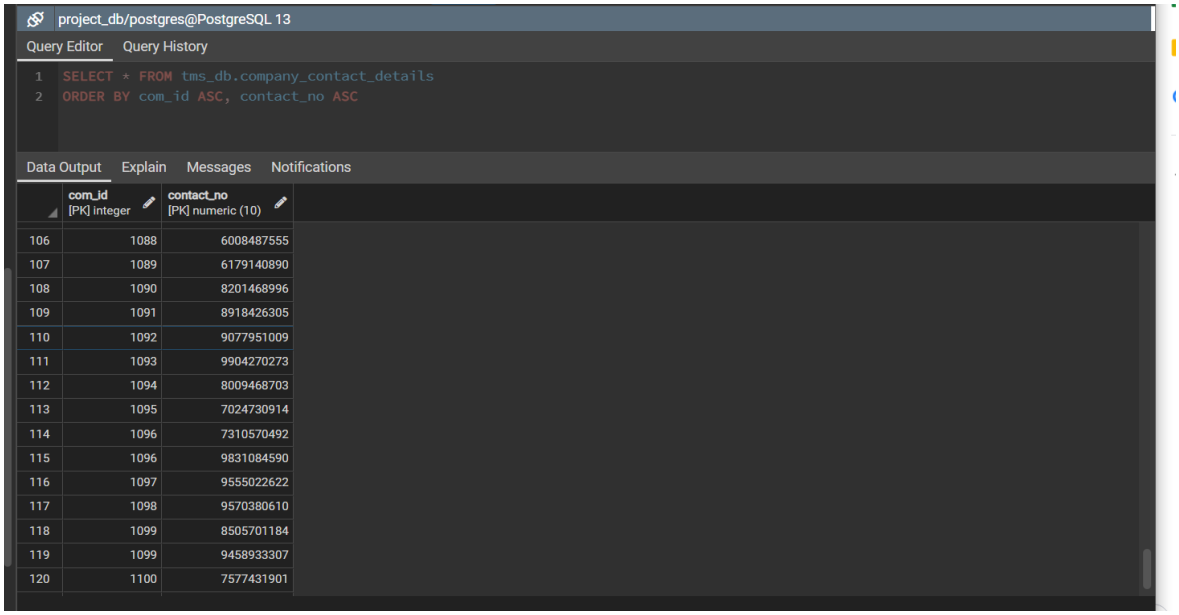
```
SELECT * FROM tms_db.destination_image
ORDER BY d_id ASC, d_image ASC
```

Data Output | Explain | Messages | Notifications

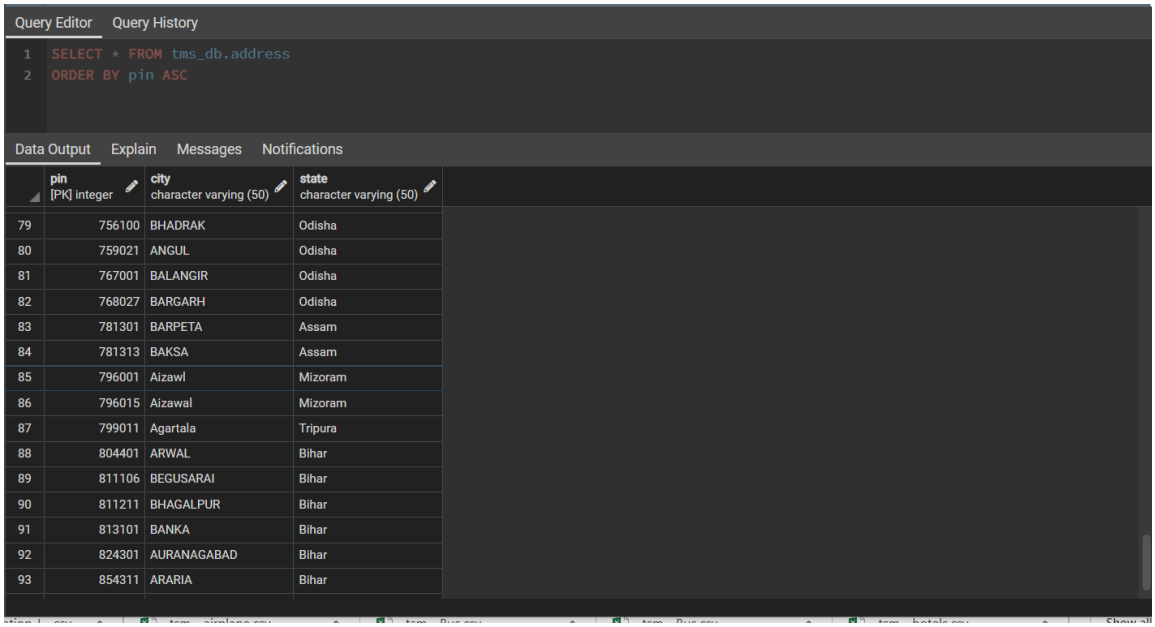
	d_id [PK] integer	d_image [PK] character varying (300)
67	67	https://www.transindiatravels.com/wp-content/uploads/bhimbetka-rock-shelters-raisen.jpg
68	68	https://www.transindiatravels.com/wp-content/uploads/victoria-terminus-chattrapati-shivaji-terminus.jpg
69	69	https://www.transindiatravels.com/wp-content/uploads/jagannath-temple-puri-1.jpg
70	70	https://www.transindiatravels.com/wp-content/uploads/lingaraja-temple-complex-khurda.jpg
71	71	https://www.transindiatravels.com/wp-content/uploads/the-ridge-shimla.jpg
72	72	https://www.transindiatravels.com/wp-content/uploads/tso-moriri-lake-ladakh.jpg
73	73	https://www.transindiatravels.com/wp-content/uploads/mysore-palace-mysore.jpg
74	74	https://www.transindiatravels.com/wp-content/uploads/bangalore-palace-and-grounds-bangalore.jpg
75	75	https://www.transindiatravels.com/wp-content/uploads/gwalior-fort-gwalior.jpg
76	76	https://www.transindiatravels.com/wp-content/uploads/bhimbetka-rock-shelters-raisen.jpg
77	77	https://www.transindiatravels.com/wp-content/uploads/victoria-terminus-chattrapati-shivaji-terminus.jpg
78	78	https://www.transindiatravels.com/wp-content/uploads/jagannath-temple-puri-1.jpg
79	79	https://www.transindiatravels.com/wp-content/uploads/lingaraja-temple-complex-khurda.jpg
80	80	https://www.transindiatravels.com/wp-content/uploads/the-ridge-shimla.jpg

Successfully run. Total query runtime: 153 msec. 80 rows affected.

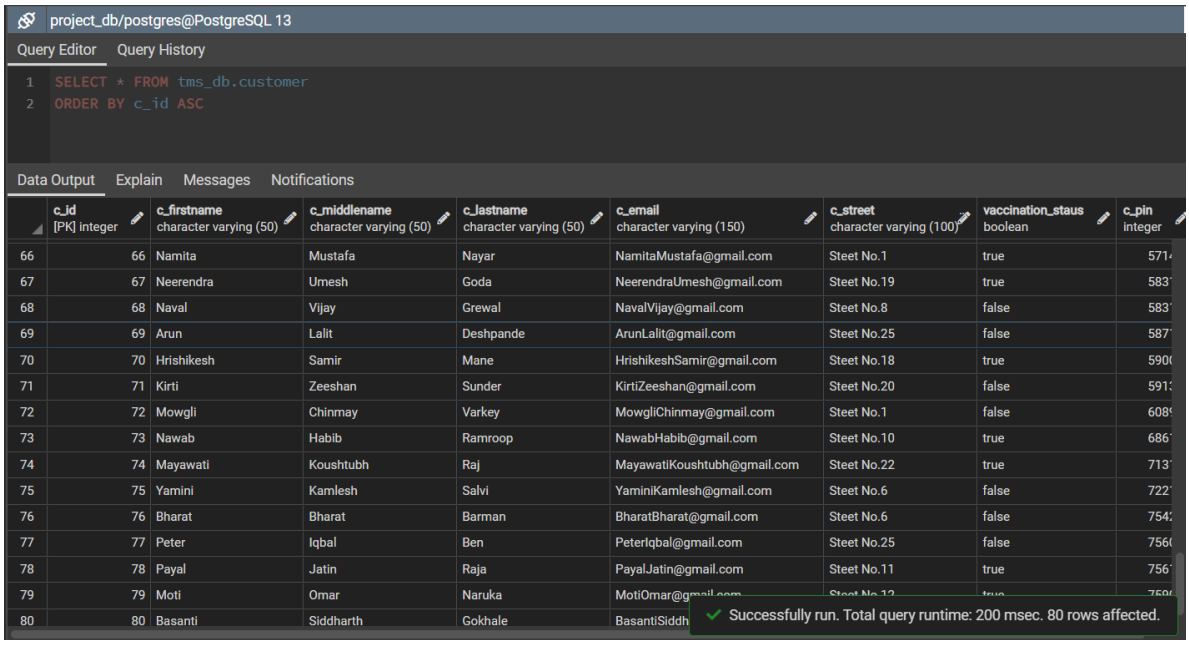
- Company_contact_details:
  No. of tuples: 120



- address:
  No. of tuples: 93



- customer:
  No. of tuples: 80



- discount:
  No. of tuples: 11

project_db/postgres@PostgreSQL 13

Query Editor | Query History

```
SELECT * FROM tms_db.discount
ORDER BY dis_id ASC
```

Data Output | Explain | Messages | Notifications

	dis_id [PK] integer	dis_name character varying (30)	percentage real	start_date date	end_date date
1	1	beat the heat	5	2022-10-01	2022-10-10
2	2	Its time to Save More	10	2022-01-20	2022-01-30
3	3	char dham yatra	15	2021-09-18	2021-10-30
4	4	Diwali offer	20	2021-11-01	2021-11-11
5	5	Summer offer	25	2022-01-04	2022-04-30
6	6	Mega offer	30	2021-01-12	2021-12-30
7	7	Today's offer	10	2021-11-14	2021-11-15
8	8	Weekend offer	15	2021-11-13	2021-11-14
9	9	The tour time	20	2021-11-20	2021-11-30
10	10	Adventure time	10	2022-10-05	2022-05-30
11	11	Our haritage	40	2022-01-02	2022-02-02

Successfully run. Total query runtime: 130 msec. 11 rows affected.

- company:
  No. of tuples: 100

project_db/postgres@PostgreSQL 13

Query Editor | Query History

```
SELECT * FROM tms_db.company
ORDER BY com_id ASC
```

Data Output | Explain | Messages | Notifications

	com_id [PK] integer	com_name character varying (100)	com_service character varying (50)	com_desc text
87	1087	North Bengal State Transport Corporation (NBSTC)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter jou
88	1088	South Bengal State Transport Corporation (SBSTC)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter jou
89	1089	West Bengal Transport Corporation (WBTC)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter jou
90	1090	Uttar Pradesh State Road Transport Corporation (UPSRTC)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter jou
91	1091	Lucknow Mahanagar Parivahan Sewa (LMPS)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter jou
92	1092	Lucknow Upnagariya Parivahan Sewa (LUPS)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter jou
93	1093	Kanpur Lucknow Roadways Service (KLRS)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter jou
94	1094	Noida Metro Rail Corporation (NMRCL)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter jou
95	1095	Atihad Airways	Airplane	Go First, founded as GoAir,[4] is an Indian ultra-low-cost airline based in Mumbai, Maharasht
96	1096	The New Yorker	Hotels	Daily housekeeping
97	1097	Air Arbia	Airplane	AirAsia India is an airline in India headquartered in Bengaluru, Karnataka. The airline is a join
98	1098	Beachwalk Resort	Hotels	best price for 3 star properties.
99	1099	Etiquette Suites	Hotels	highly reated by guests 100% would recommended
100	1100	State Express Transport Corporation	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter jou

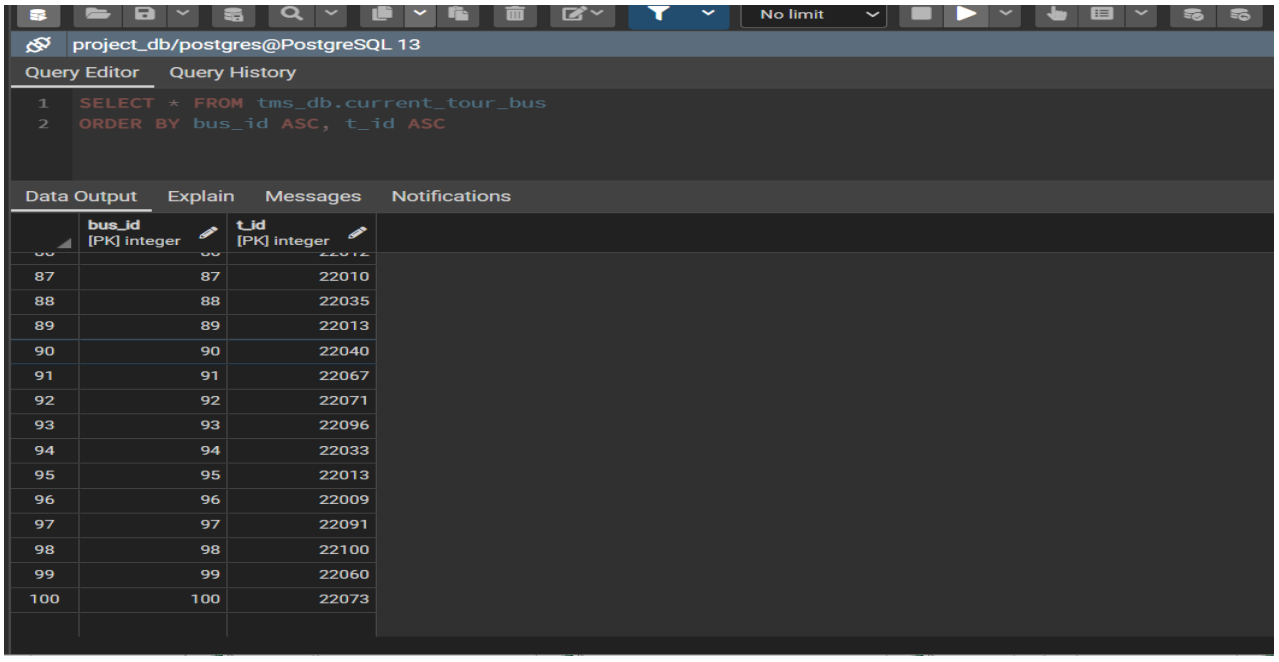
Successfully run. Total query runtime: 436 msec. 100 rows affected.

- tour_package:
  No. of tuples: 84

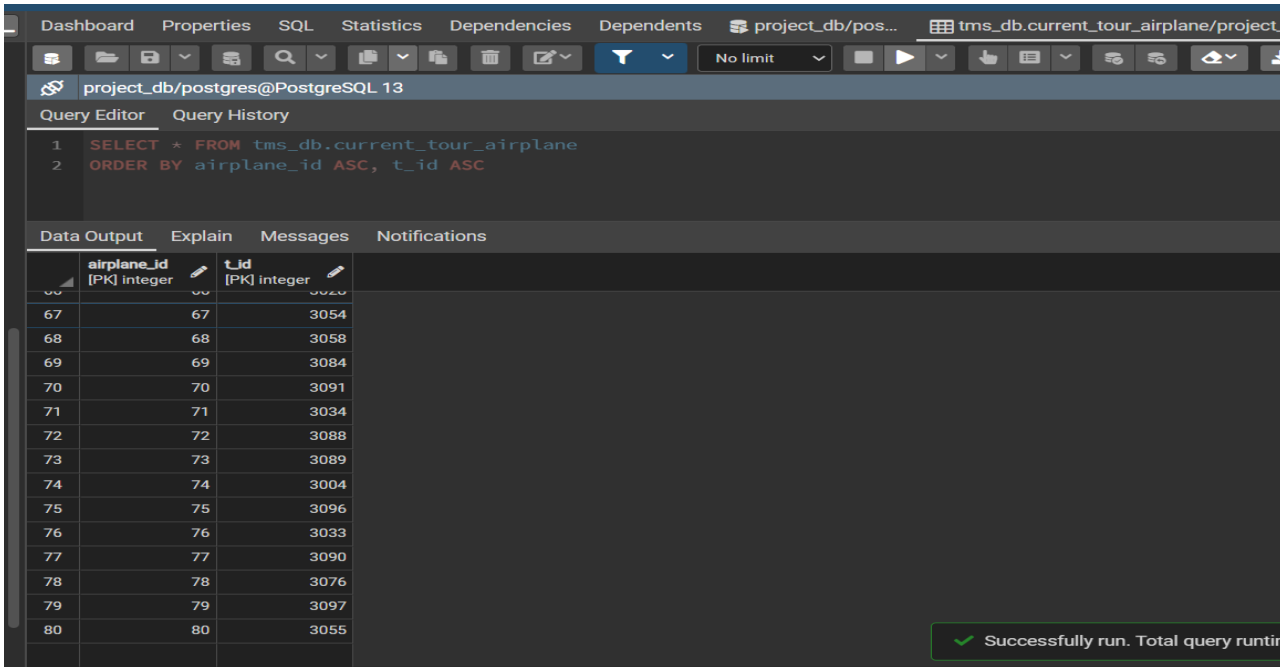
Data Output | Explain | Messages | Notifications

	t_id [PK] integer	t_name character varying (50)	t_duration character varying (30)	t_start_date date	t_price real	t_type character varying (30)	t_desc text
70	70	Weekend Getaways Chennai	5 Days and 6 Nights	2022-03-31	6999	[null]	South India is a mystic and magnificent destination tha
71	71	Weekend Getaways Bangalore	3 Days and 2 Nights	2022-03-11	25999	[null]	A tour through East India takes you through a dramatica
72	72	Weekend Getaways Nagpur	9 Days and 8 Nights	2021-12-23	20999	[null]	Western India comprises of the states of Goa, Gujarat, I
73	73	Weekend Getaways Hyderabad	5 Days and 6 Nights	2022-05-15	25999	[null]	Central India is simply the heart of the country, a soft ar
74	74	Weekend Getaways Cochin	3 Days and 2 Nights	2021-12-27	3999	[null]	North India Tour will take you to its historic, majestic an
75	75	Weekend Getaways Chandigarh	3 Days and 2 Nights	2022-08-04	25999	[null]	South India is a mystic and magnificent destination tha
76	76	Weekend Getaways Ahmedabad	9 Days and 8 Nights	2022-06-01	2999	[null]	A tour through East India takes you through a dramatica
77	77	Weekend Getaways Pune	5 Days and 6 Nights	2022-11-05	6999	[null]	Western India comprises of the states of Goa, Gujarat, I
78	78	Weekend Getaways Jaipur	3 Days and 2 Nights	2022-02-05	3999	[null]	Central India is simply the heart of the country, a soft ar
79	79	Trekking	3 Days and 2 Nights	2022-09-13	15999	[null]	A tour through East India takes you through a dramatica
80	80	Ice mountaine	7 Days and 8 Nights	2022-07-07	11999	[null]	Western India comprises of the states of Goa, Gujarat, I
81	81	Desert Safari	7 Days and 8 Nights	2022-04-21	3999	[null]	Central India is simply the heart of the country, a soft ar
82	82	Peak Climbing	5 Days and 6 Nights	2022-09-10	35999	[null]	North India Tour will take you to its historic, majestic an
83	83	Mountain Camp	7 Days and 8 Nights	2021-12-07	6999	[null]	South India is a mystic and magnificent destination tha
84	84	River Rafting	4 Days and 5 Nights	2022-02-22	15999	[null]	A tour through East India takes you through a dramatica

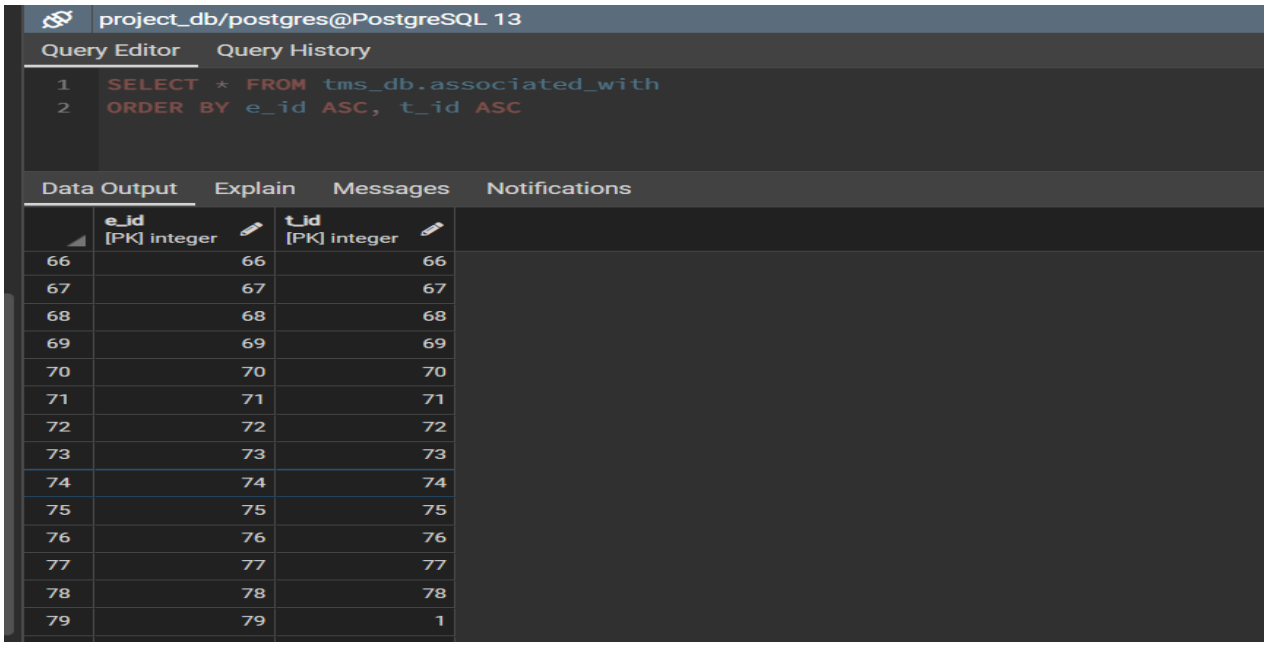
- current_tour_bus:
  No. of tuples: 100



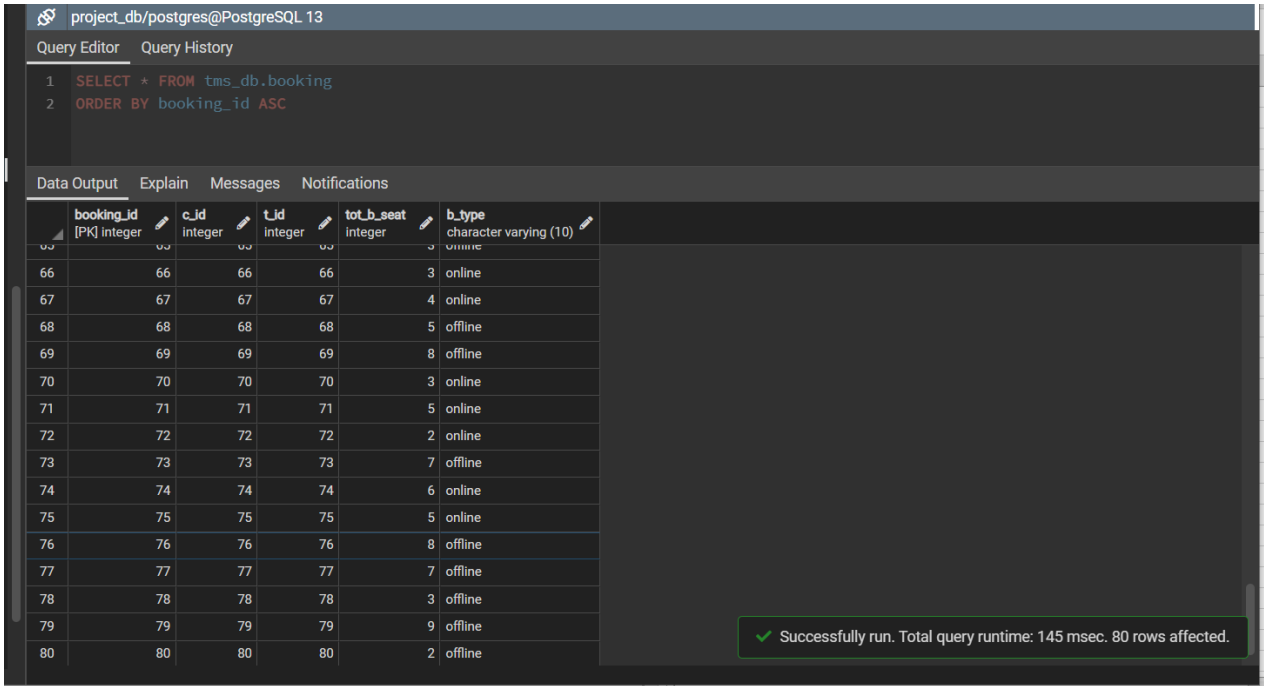
- current_tour_airplane:
  No. of tuples: 80



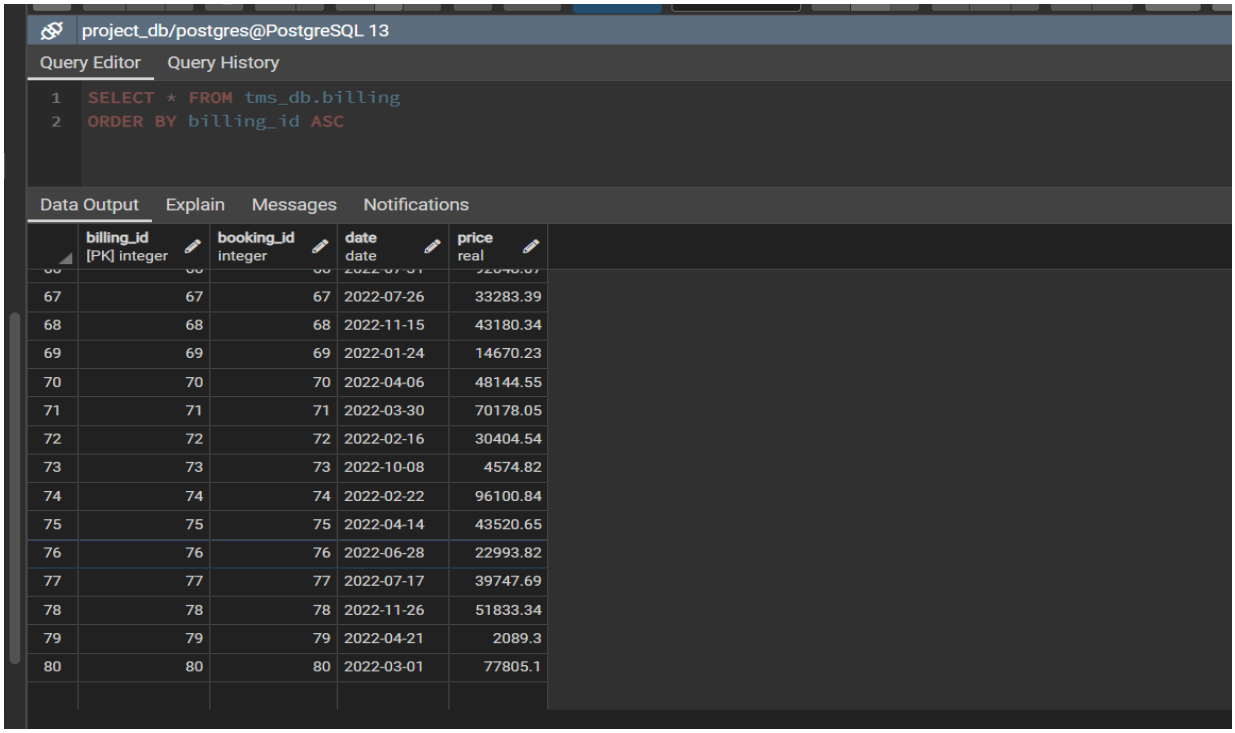
- associated_with:
  No. of tuples: 80



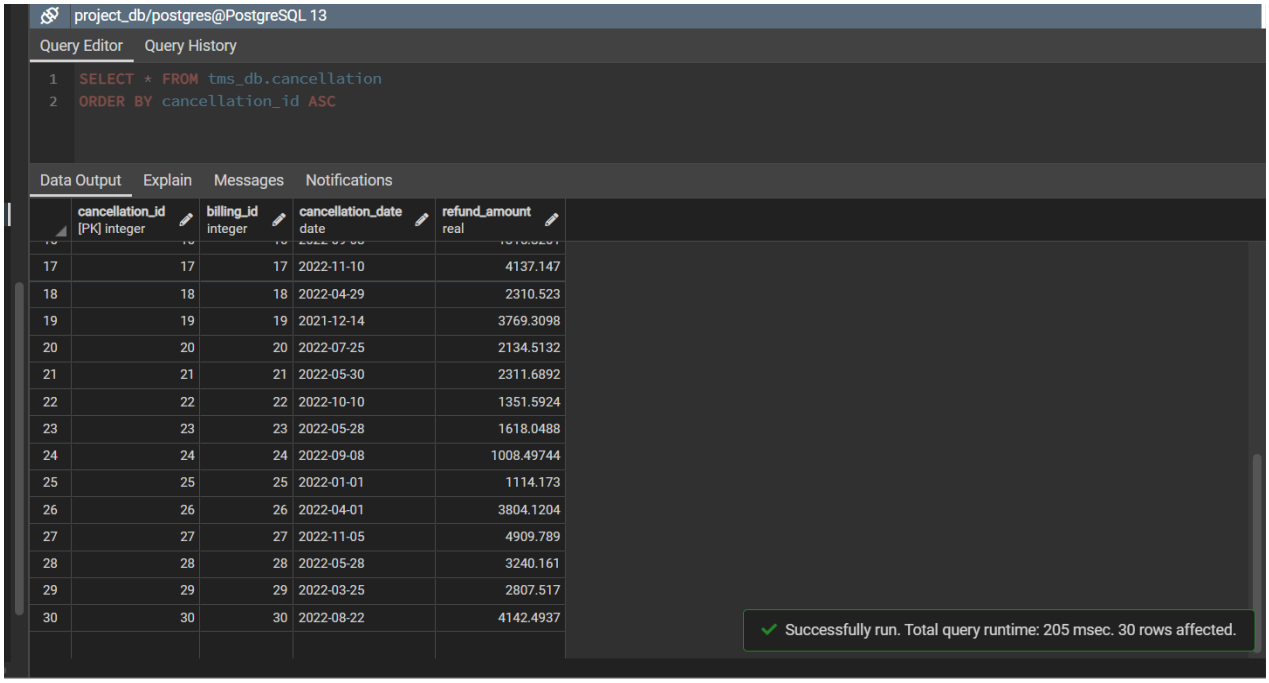
- booking:
  No. of tuples: 80



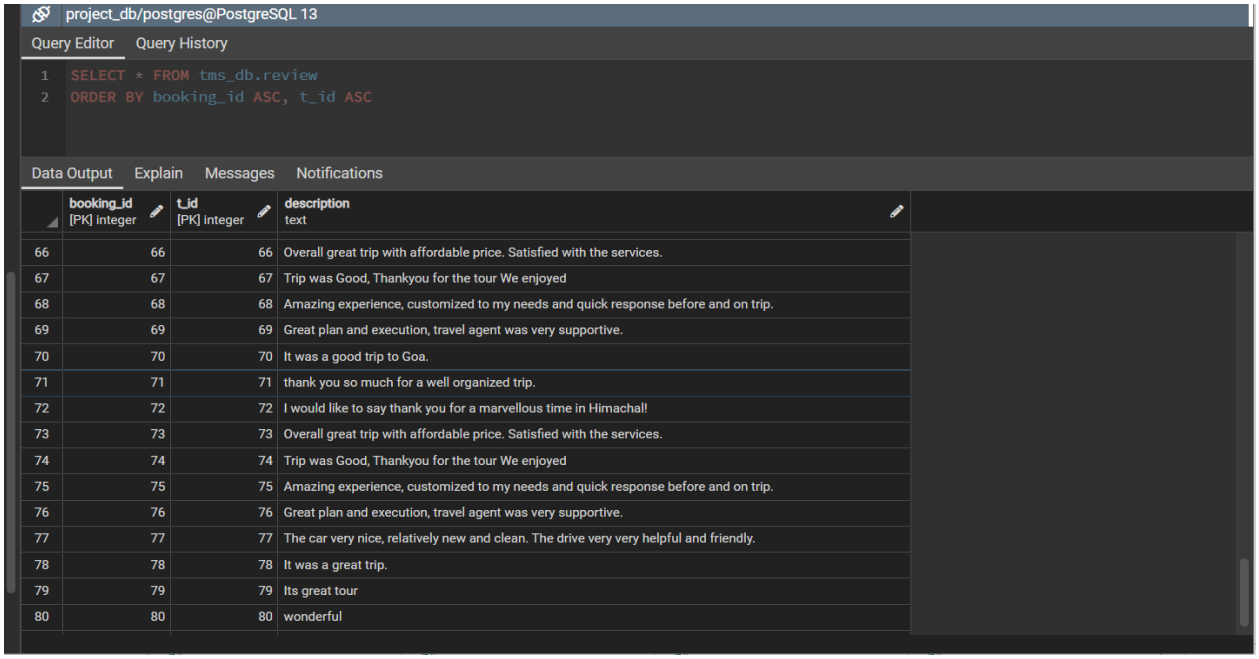
- billing:
  No. of tuples: 80



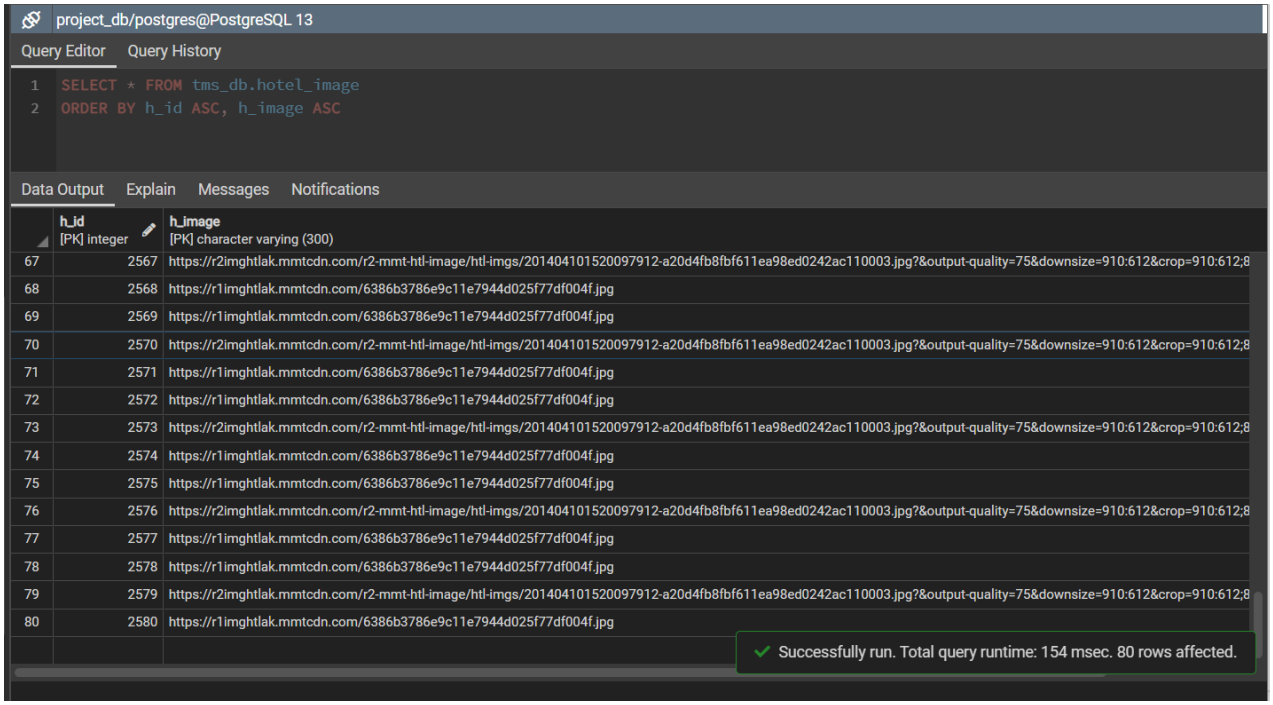
- cancellation:
  No. of tuples: 30



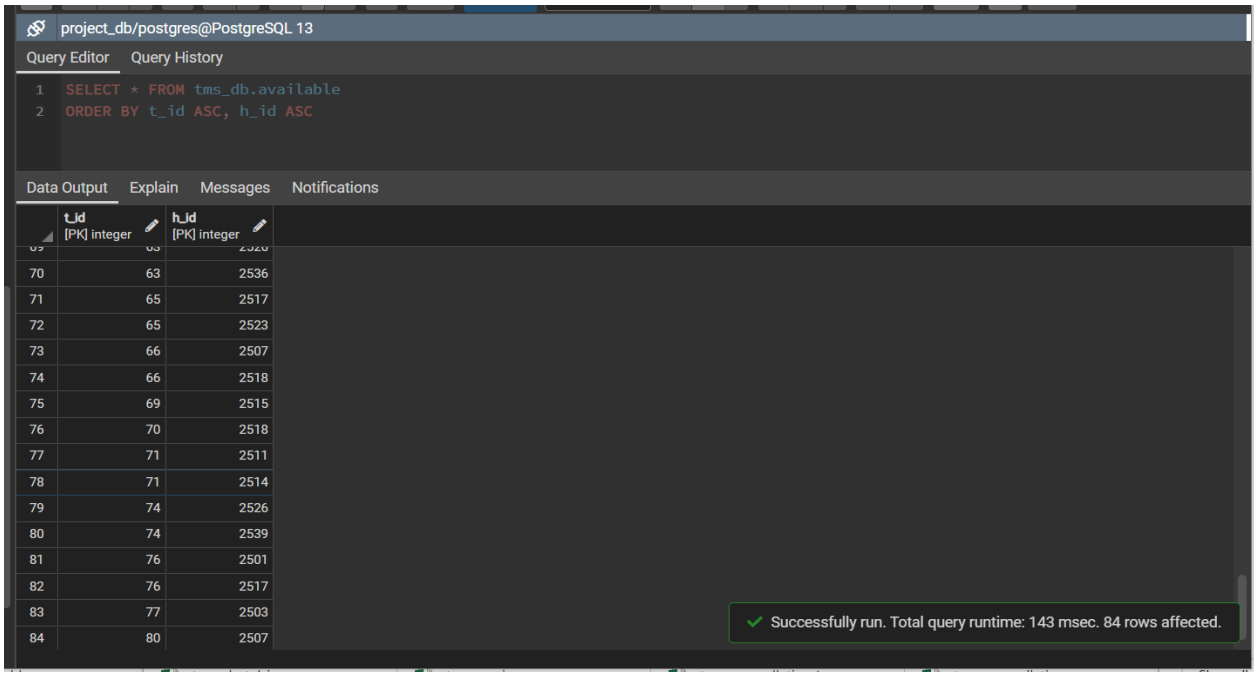
- Review:
  No. of tuples: 80



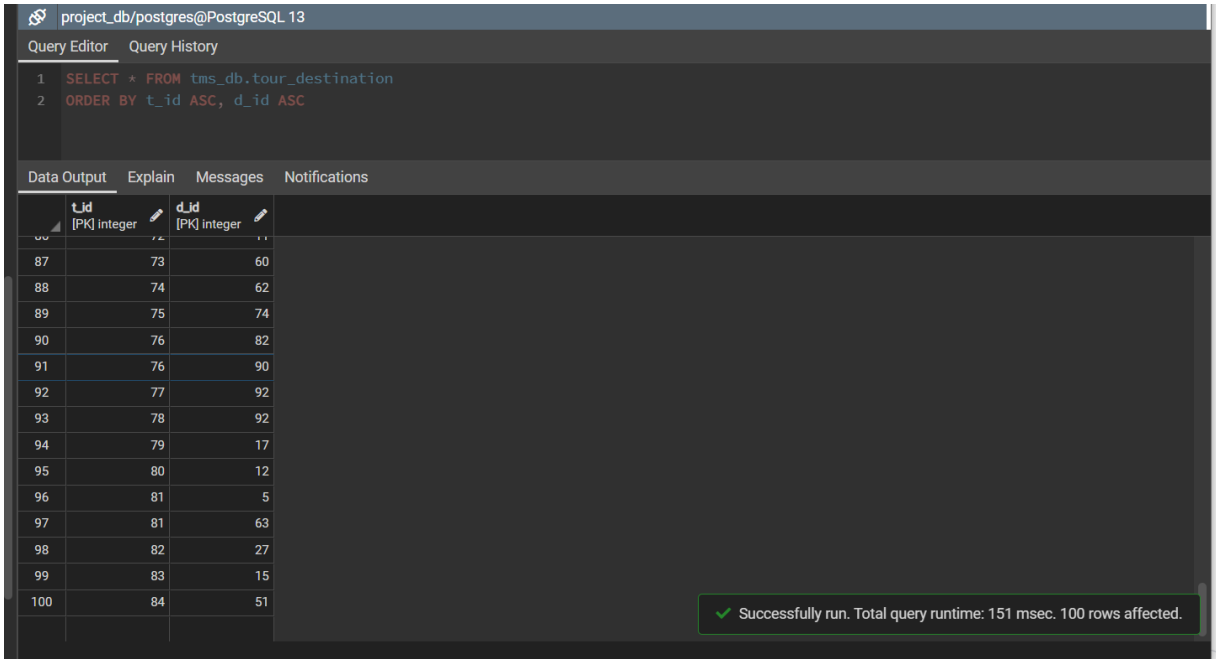
- Hotel_images:
  No. of tuples: 80



- available:
  No. of tuples: 84



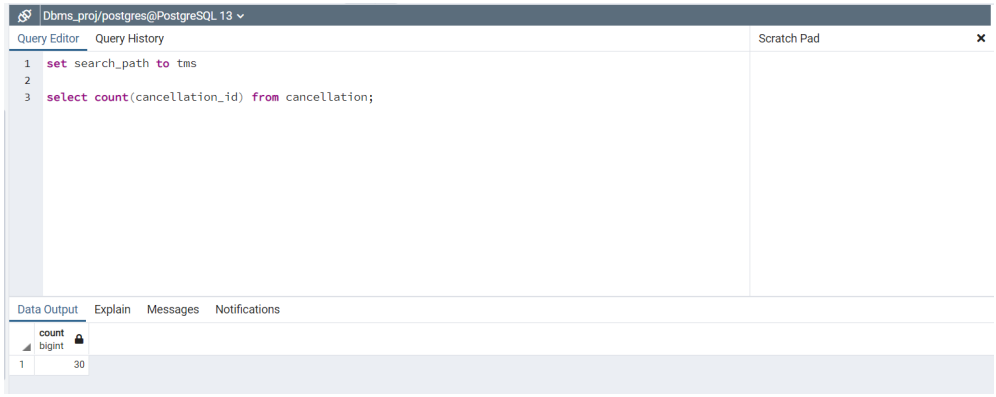
- Tour_destination
  No. of tuples: 100



SQL queries:

1. **Find Total number of customers that have cancelled the tickets.**

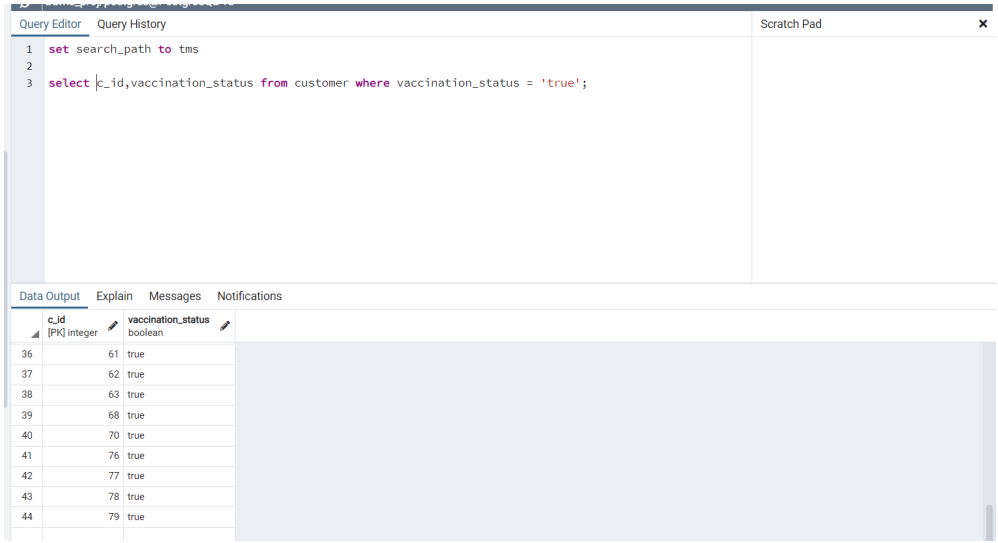
   select count(cancellation_id) from cancellation;



Number of tuples: 1

2. **Total number of customers that have taken the vaccine.**

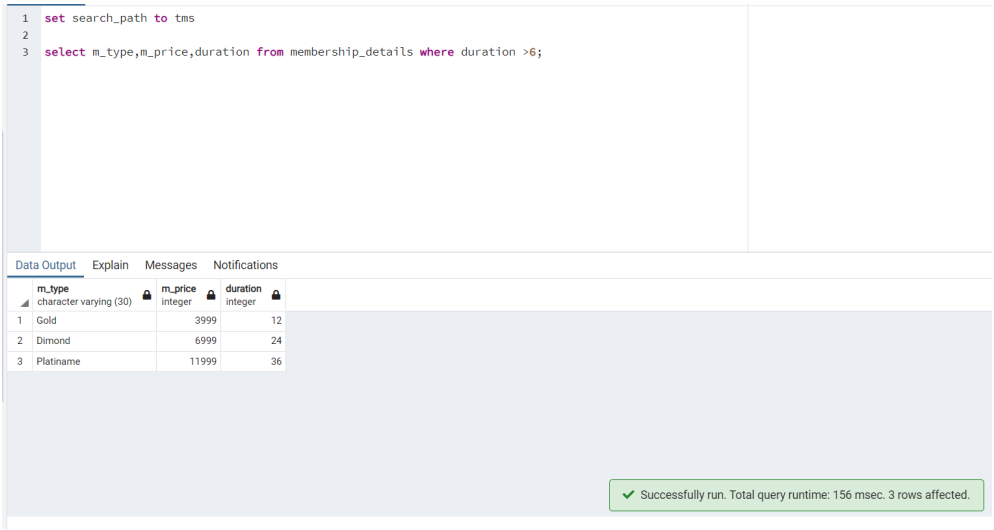
   select c_id, vaccination_status from customer where vaccination_status = ‘True’;



Number of tuples: 44

3. **find all membership types that have duration >6**

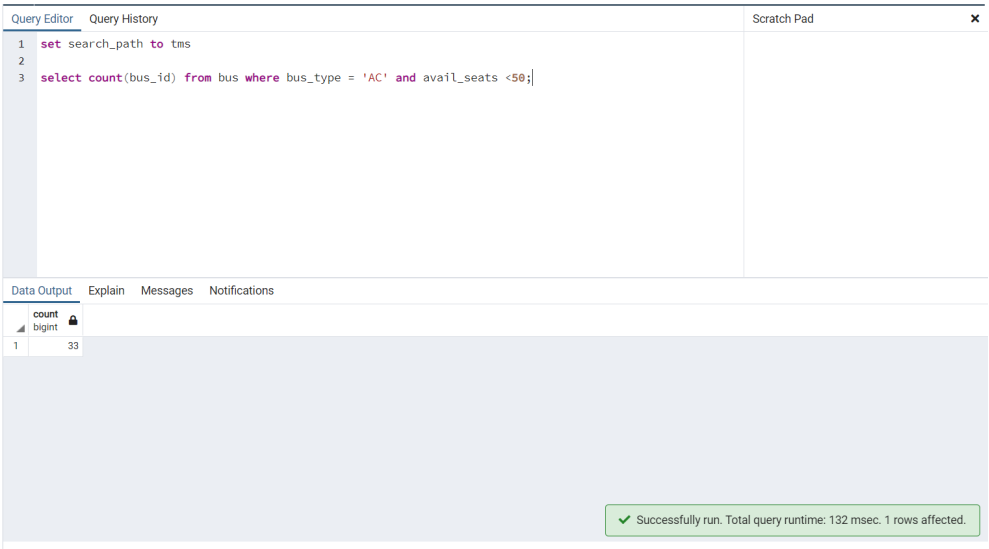
select m_type,m_price,duration from membership_details where duration >6;



Number of tuples: 3

4. **Select total number of Ac buses which have total available seats <50.**

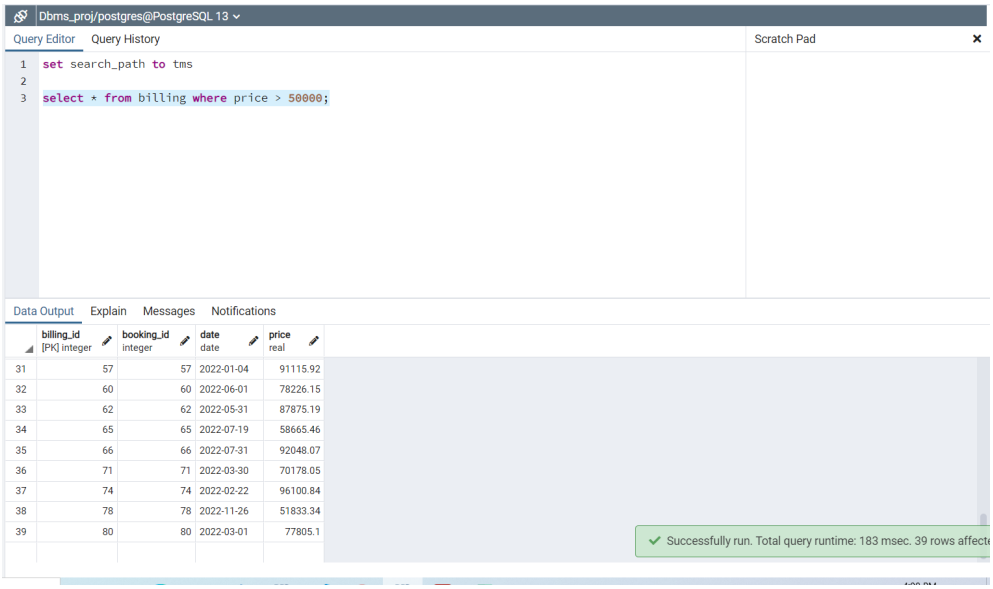
select count(bus_id) from bus where bus_type = ‘AC’ and avail_seats < 50;



Number of tuples: 1.

5. **Find customer details who have billing_price > 50000.00**

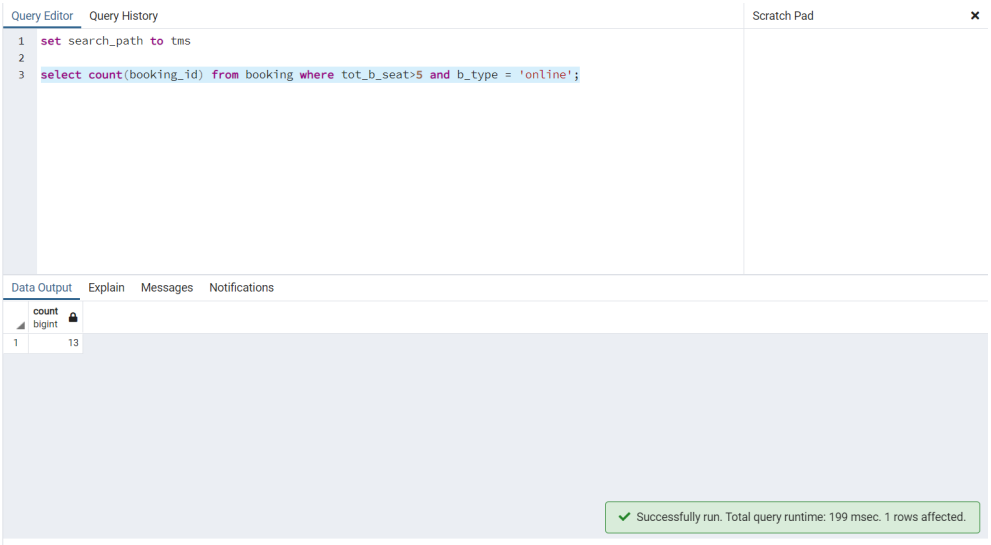
select * from billing where price > 50000;



Number of tuples: 39

6. **Find total number of online bookings which have total_ booked_seat > 5.**

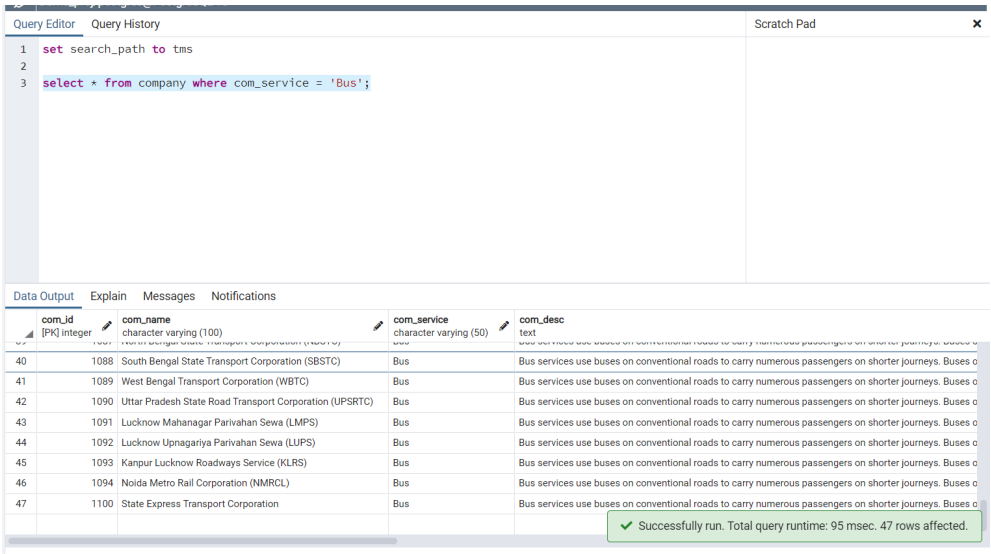
select count(booking_id) from booking where tot_b_seats > 5 and b_type = ‘online’;



Number of tuples:1.

7. **List of companies which provide the bus .**

select * from a company where com_service = ‘Bus’.



```
set search_path to tms

select * from company where com_service = 'Bus';
```

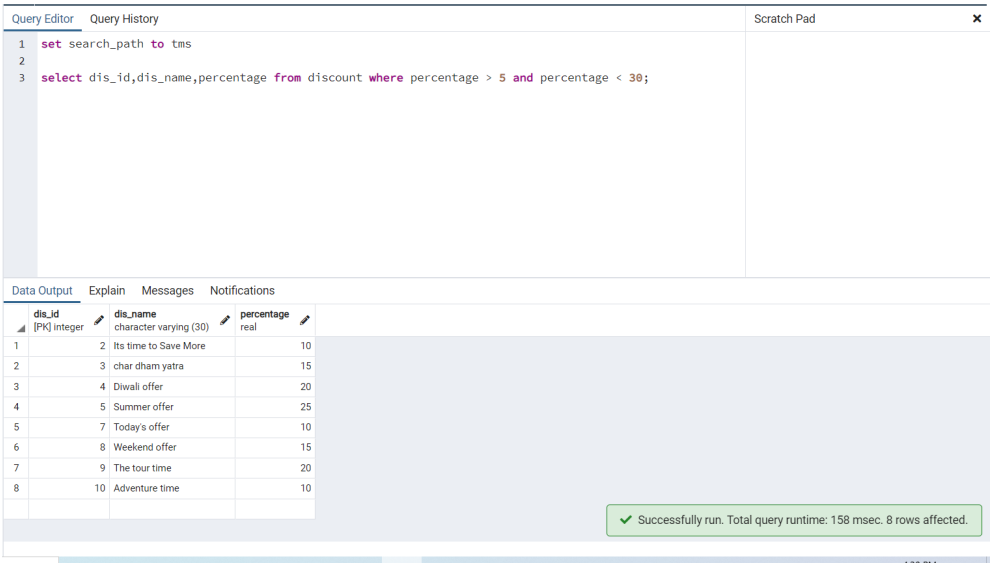
	com_id [PK] integer	com_name character varying (100)	com_service character varying (50)	com_desc text
40	1088	South Bengal State Transport Corporation (SBSTC)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter journeys. Buses o
41	1089	West Bengal Transport Corporation (WBTC)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter journeys. Buses o
42	1090	Uttar Pradesh State Road Transport Corporation (UPSRTC)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter journeys. Buses o
43	1091	Lucknow Mahanagar Parivahan Sewa (LMPS)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter journeys. Buses o
44	1092	Lucknow Upnagariya Parivahan Sewa (LUPS)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter journeys. Buses o
45	1093	Kanpur Lucknow Roadways Service (KLRS)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter journeys. Buses o
46	1094	Noida Metro Rail Corporation (NMRCL)	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter journeys. Buses o
47	1100	State Express Transport Corporation	Bus	Bus services use buses on conventional roads to carry numerous passengers on shorter journeys. Buses o

Successfully run. Total query runtime: 95 msec. 47 rows affected.

Number of tuples: 47

8. **Find the discount name which has a discount > 5 and < 30.**

select dis_id,dis_name,percentage from discount where percentage > 5 and percentage < 30.



```
set search_path to tms

select dis_id,dis_name,percentage from discount where percentage > 5 and percentage < 30;
```

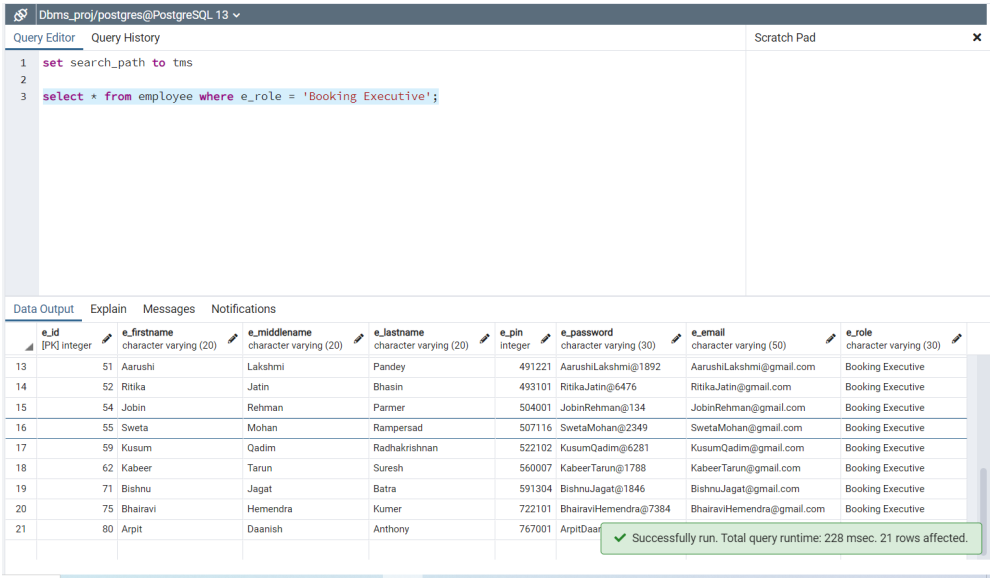
	dis_id [PK] integer	dis_name character varying (30)	percentage real
1	2	Its time to Save More	10
2	3	char dham yatra	15
3	4	Diwali offer	20
4	5	Summer offer	25
5	7	Today's offer	10
6	8	Weekend offer	15
7	9	The tour time	20
8	10	Adventure time	10

Successfully run. Total query runtime: 158 msec. 8 rows affected.

Number of tuples: 8.

9. **Find employee details whose role is booking executive.**

select * from employee where e_role = ‘Booking Executive’;

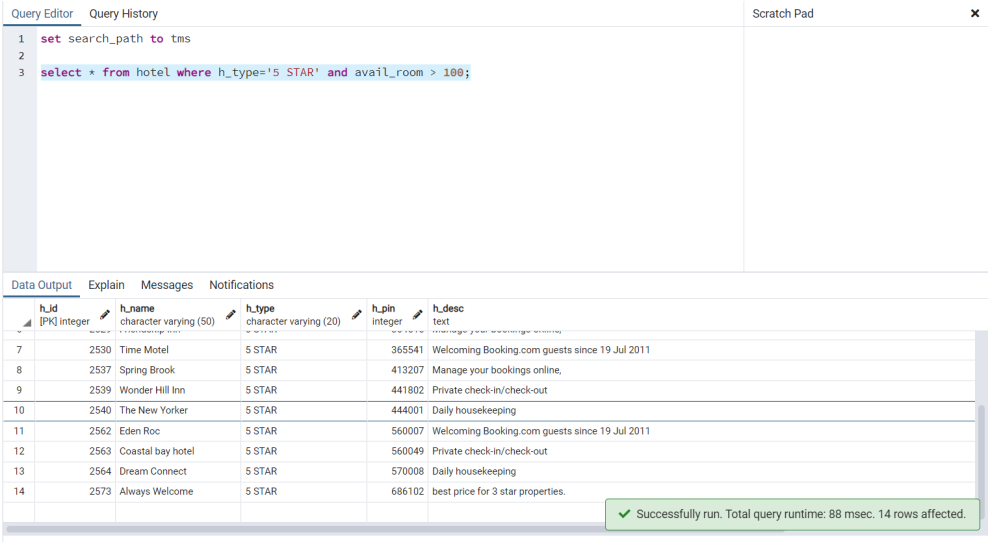


e_id [PK] integer	e_firstname character varying (20)	e_middlename character varying (20)	e_lastname character varying (20)	e_pin integer	e_password character varying (30)	e_email character varying (50)	e_role character varying (30)
13	51	Aarushi	Lakshmi	Pandey	491221	AarushiLakshmi@1892	AarushiLakshmi@gmail.com	Booking Executive
14	52	Ritika	Jatin	Bhasin	493101	RitikaJatin@6476	RitikaJatin@gmail.com	Booking Executive
15	54	Jobin	Rehman	Parmer	504001	JobinRehman@134	JobinRehman@gmail.com	Booking Executive
16	55	Sweta	Mohan	Rampersad	507116	SwetaMohan@2349	SwetaMohan@gmail.com	Booking Executive
17	59	Kusum	Qadim	Radhakrishnan	522102	KusumQadim@6281	KusumQadim@gmail.com	Booking Executive
18	62	Kabeer	Tarun	Suresh	560007	KabeerTarun@1788	KabeerTarun@gmail.com	Booking Executive
19	71	Bishnu	Jagat	Batra	591304	BishnuJagat@1846	BishnuJagat@gmail.com	Booking Executive
20	75	Bhairavi	Hemendra	Kumer	722101	BhairaviHemendra@7384	BhairaviHemendra@gmail.com	Booking Executive
21	80	Arpit	Daanish	Anthony	767001	ArpitDaa		

Number of tuples: 21.

10. **Find all the 5 star hotels which have over 100 available rooms.**

select * from a hotel where h_type ‘5 STAR’ and avail_room > 100.

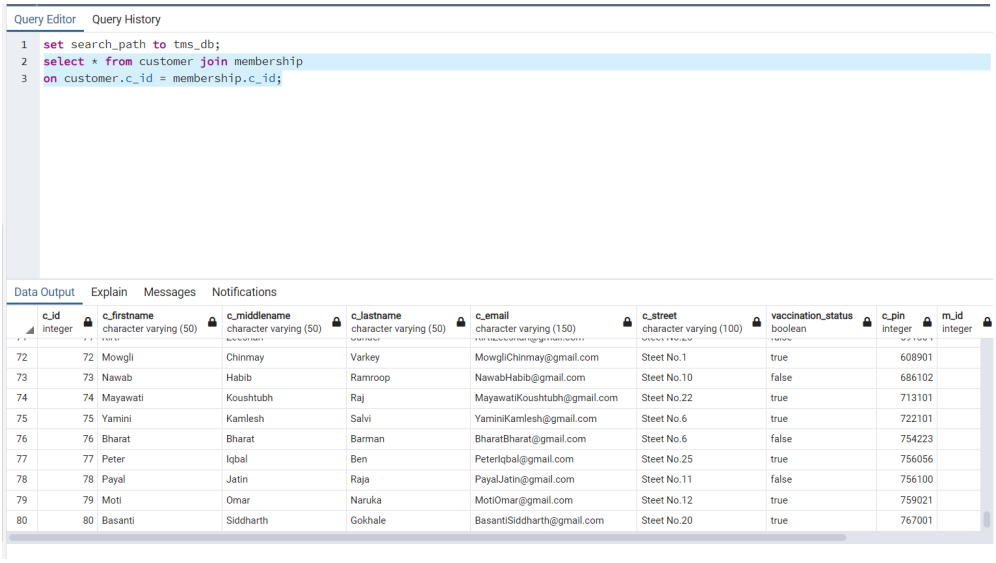


h_id [PK] integer	h_name character varying (50)	h_type character varying (20)	h_pin integer	h_desc text
7	2530	Time Motel	5 STAR	365541	Welcoming Booking.com guests since 19 Jul 2011
8	2537	Spring Brook	5 STAR	413207	Manage your bookings online,
9	2539	Wonder Hill Inn	5 STAR	441802	Private check-in/check-out
10	2540	The New Yorker	5 STAR	444001	Daily housekeeping
11	2562	Eden Roc	5 STAR	560007	Welcoming Booking.com guests since 19 Jul 2011
12	2563	Coastal bay hotel	5 STAR	560049	Private check-in/check-out
13	2564	Dream Connect	5 STAR	570008	Daily housekeeping
14	2573	Always Welcome	5 STAR	686102	best price for 3 star properties.

Number of tuples: 14

11. Find details about customers who have taken tour membership.

```
select * from customer join membership
on customer.c_id = membership.c_id;
```



Query Editor | Query History

```
1  set search_path to tms_db;
2  select * from customer join membership
3  on customer.c_id = membership.c_id;
```

Data Output | Explain | Messages | Notifications

	c_id integer	c_firstname character varying (50)	c_middlename character varying (50)	c_lastname character varying (50)	c_email character varying (150)	c_street character varying (100)	vaccination_status boolean	c_pin integer	m_id integer
72	72	Mowgli	Chinmay	Varkey	MowgliChinmay@gmail.com	Steet No.1	true	608901	
73	73	Nawab	Habib	Ramroop	NawabHabib@gmail.com	Steet No.10	false	686102	
74	74	Mayawati	Koushtubh	Raj	MayawatiKoushtubh@gmail.com	Steet No.22	true	713101	
75	75	Yamini	Kamlesh	Salvi	YaminiKamlesh@gmail.com	Steet No.6	true	722101	
76	76	Bharat	Bharat	Barman	BharatBharat@gmail.com	Steet No.6	false	754223	
77	77	Peter	Iqbal	Ben	PeterIqbal@gmail.com	Steet No.25	true	756056	
78	78	Payal	Jatin	Raja	PayalJatin@gmail.com	Steet No.11	false	756100	
79	79	Moti	Omar	Naruka	MotiOmar@gmail.com	Steet No.12	true	759021	
80	80	Basanti	Siddharth	Gokhale	BasantiSiddharth@gmail.com	Steet No.20	true	767001	

Number of tuples: 80

12. Find customers details who has booked a tour and the tour has a 5 star hotel

```
Select * from customer where c_id in
                (select distinct c.c_id from available as A
                                inner join booking as B on A.t_id=B.t_id
                                inner join customer as C on C.c_id=B.c_id
inner join hotel as h on h.h_id=A.h_id
                                Where h_type='5 STAR')
```

```
set SEARCH_PATH to tms_db;
select * from customer where c_id in
            (select distinct c.c_id from available as A
                     inner join booking as B on A.t_id=B.t_id
                     inner join customer as C on C.c_id=B.c_id
                     inner join hotel as h on h.h_id=A.h_id
                     where h_type='5 STAR')
```

Data Output | Explain | Messages | Notifications

	c_id [PK] integer	c_firstname character varying (50)	c_middlename character varying (50)	c_lastname character varying (50)	c_email character varying (150)	c_street character varying (100)	vaccination_status boolean	c_pin integer
15	37	Roma	Hassan	Bhandari	RomaHassan@gmail.com	Steet No.9	false	413207
16	43	Jagdish	Dinesh	Chahal	JagdishDinesh@gmail.com	Steet No.20	true	457882
17	45	Chirag	Varun	Sibal	ChiragVarun@gmail.com	Steet No.6	true	465230
18	51	Nupur	Kirti	Nawal	NupurKirti@gmail.com	Steet No.25	true	491221
19	56	Rajesh	Wahid	Master	RajeshWahid@gmail.com	Steet No.4	true	515001
20	59	Chitra	Mitesh	Suresh	ChitraMitesh@gmail.com	Steet No.20	true	522102
21	60	Radhika	Arpit	Deshpande	RadhikaArpit@gmail.com	Steet No.3	true	533221
22	61	Omar	Nawab	Rampersad	OmarNawab@gmail.com	Steet No.17	true	560001
23	62	Pooja	Virat	Karpe	PoojaVirat@gmail.com	Steet No.23	false	560007
24	71	Kirti	Zeeshan	Sunder	KirtiZeeshan@gmail.com	Steet No.20	true	591304
25	74	Mayawati	Koushtubh	Raj	MayawatiKoushtubh@gmail.com	Steet No.22	true	713101

No of tuples: 25

13. **Find customers details who have membership of Diamond and also visited the destination 'The Taj Mahal, Agra.'**

SQL Query: select * from customer where c_id in
(select D.c_id from tour_destination as A
Inner join destination as B on B.d_id = A.d_id
Inner join booking as C on C.t_id = A.t_id
Inner join customer as D on D.c_id = C.c_id
Inner join membership as E on E.c_id = D.c_id
Inner join membership_details as M on M.m_id = E.m_id
where B.d_name = 'The Taj Mahal, Agra' and m.m_type = 'Diamond');

```
set SEARCH_PATH to tms_db;
select * from customer where c_id in
                    (select D.c_id  from tour_destination as A
                     inner join destination as B on B.d_id=A.d_id
                     inner join booking as C on C.t_id=A.t_id
                     inner join customer as D on D.c_id=C.c_id
                     inner join membership as E on E.c_id=D.c_id
                     inner join membership_details as M on M.m_id=E.m_id
                     where B.d_name='The Taj Mahal, Agra.' and m.m_type='Diamond')
```

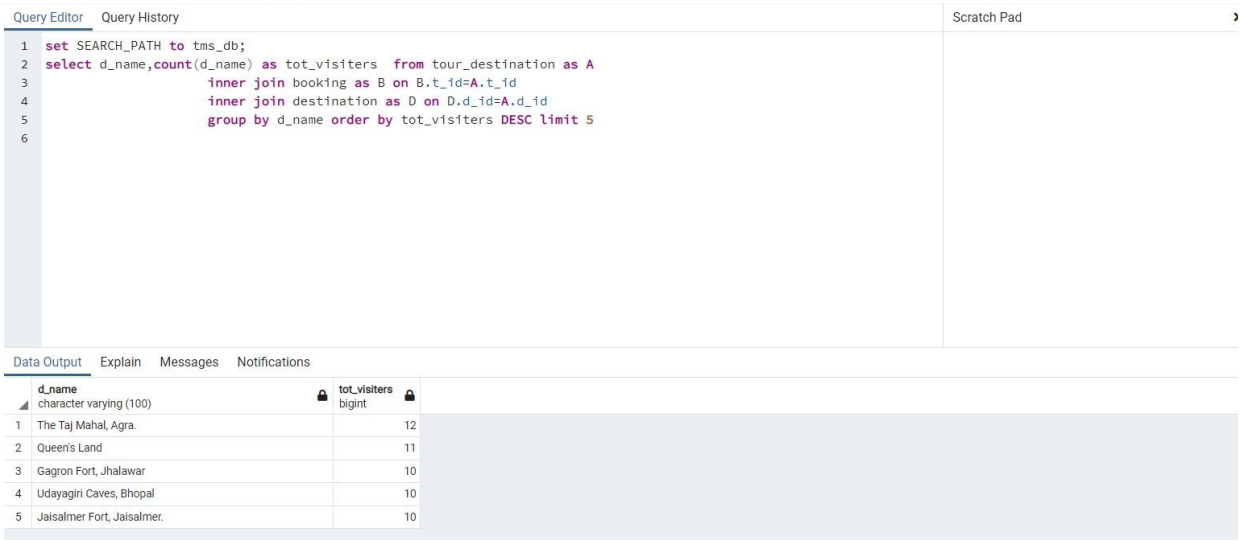
Data Output | Explain | Messages | Notifications

	c_id [PK] integer	c_firstname character varying (50)	c_middlename character varying (50)	c_lastname character varying (50)	c_email character varying (150)	c_street character varying (100)	vaccination_status boolean	c_pin integer
1	1	[...] Isha	Surya	Bhatnagar	[...] IshaSurya@gmail.com	Street No.3	true	193101
2	3	Yasmin	David	Chandra	YasminDavid@gmail.com	Steet No.1	false	202001
3	45	Chirag	Varun	Sibal	ChiragVarun@gmail.com	Steet No.6	true	465230

Number of tuples: 3.

14. Find the five most visited destinations by travel agency customers.

select d_name, count(d_name) as tot_visiters from tour_destination as A
Inner join as B on B.t_id = A.t_id
Inner join destination as D on d.d_id=A.d_id
group by d_name order by tot_visiters DESC limit 5;



Number of tuples: 5.

15. Find all billing details of customers from Gujarat state.

Select billing_id,date,price,A.booking_id from billing as A
Inner join booking as B on a.booking_id=B.booking_id
Inner join customer as C on C.c_id = B.c_id
Inner join address as D on D.pin = C.c_pin
Where state= ‘Gujarat’.

```
set SEARCH_PATH to tms_db;
select billing_id,date,price,A.booking_id from billing as A
                    inner join booking as B  on A.booking_id=B.booking_id
                    inner join customer as C on C.c_id=B.c_id
                    inner join address as D on D.pin=C.c_pin
                    where state='Gujarat'
```

Data Output | Explain | Messages | Notifications

	billing_id [PK] integer	date date	price real	booking_id integer
1	28	2022-05-31	8147.4	28
2	29	2022-03-30	4829.02	29
3	30	2022-08-07	29306.31	30
4	31	2022-04-01	82537.38	31
5	32	2022-08-10	78373.07	32
6	33	2022-11-27	28902.68	33
7	34	2022-11-27	86730.81	34
8	35	2022-01-14	2628.04	35

Number of tuples: 8

16. **Find the number of customers whose start date of membership is between 2021-10-02 and date 2022-5-01 and is grouped by membership type.**

Select m_type, count(m_type) from
(select * from membership natural join membership_details
Where start_dete
between date ‘2021-10-02’ and date ‘2022-05-01’)
as A group by m_type

```
set SEARCH_PATH to tms_db;
select m_type,count(m_type) from
            (select * from membership natural join membership_details
             where start_date
             between date '2021-10-02' and date '2022-5-01')
             as A group by m_type
```

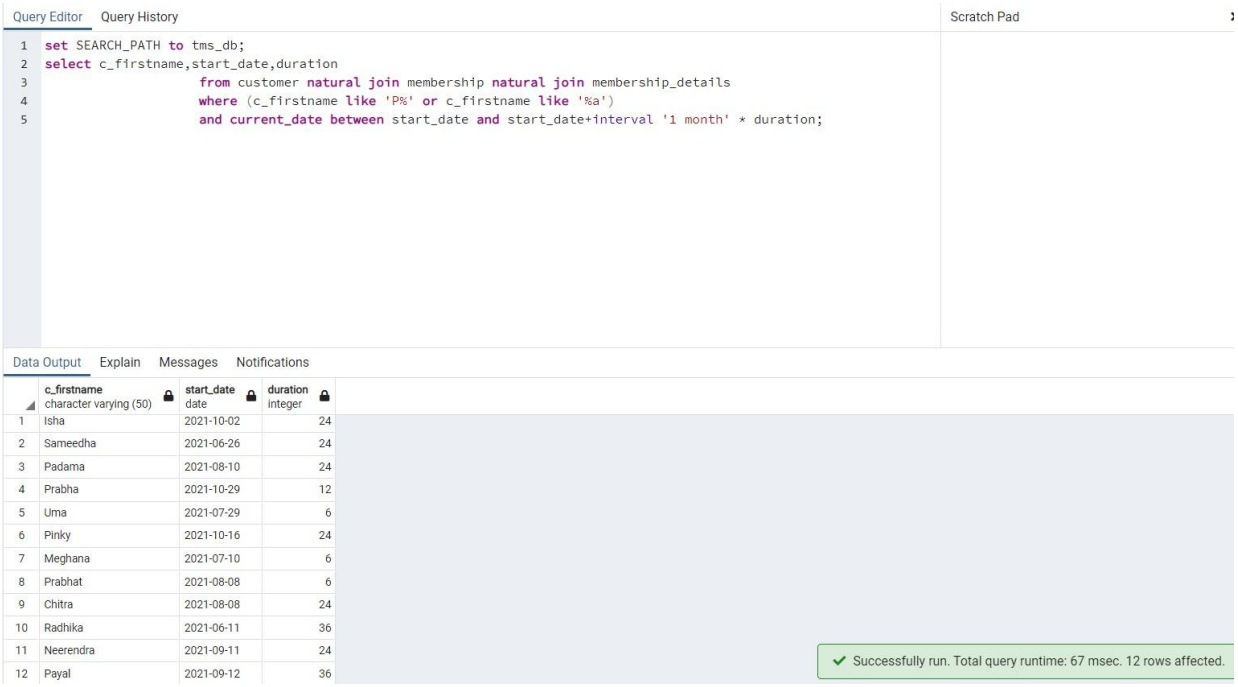
Data Output | Explain | Messages | Notifications

	m_type character varying (30)	count bigint
1	Gold	4
2	Diamond	11
3	Platinum	5
4	Silver	2

No of tuples: 4

17. **Find the first name, membership start date and duration of customers whose first name starts with 'P' or ends with 'a' and also whose membership is currently running.**

```
Select c_firstname, start_date,duration
        from customer natural join membership natural join membership_details
        where (c_firstname like 'P%' or c_firstname like '%a')
        and current_date between start_date and start_date + interval '1 month' * duration;
```



Query Editor | Query History | Scratch Pad

```
set SEARCH_PATH to tms_db;
select c_firstname,start_date,duration
                    from customer natural join membership natural join membership_details
                    where (c_firstname like 'P%' or c_firstname like '%a')
                    and current_date between start_date and start_date+interval '1 month' * duration;
```

Data Output | Explain | Messages | Notifications

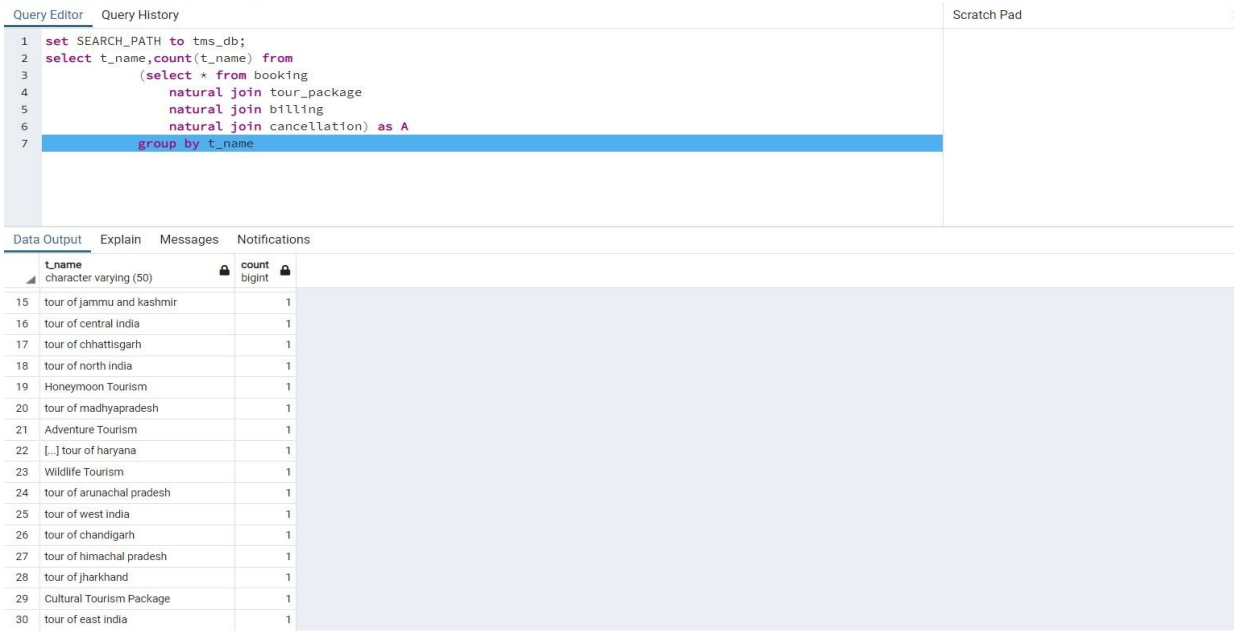
	c_firstname character varying (50)	start_date date	duration integer
1	Isha	2021-10-02	24
2	Sameedha	2021-06-26	24
3	Padama	2021-08-10	24
4	Prabha	2021-10-29	12
5	Uma	2021-07-29	6
6	Pinky	2021-10-16	24
7	Meghana	2021-07-10	6
8	Prabhat	2021-08-08	6
9	Chitra	2021-08-08	24
10	Radhika	2021-06-11	36
11	Neerendra	2021-09-11	24
12	Payal	2021-09-12	36

Successfully run. Total query runtime: 67 msec. 12 rows affected.

Number of tuples: 1

18. **Find the total number of cancellation of each and every tour package.**

```
Select t_name,count(t_name) from
                        (select * from booking
                          Natural join tour_package
                          Natural join billing
                        Natural join cancellation) as A
Group by t_name;
```



```
set SEARCH_PATH to tms_db;
select t_name,count(t_name) from
            (select * from booking
                natural join tour_package
                natural join billing
                natural join cancellation) as A
            group by t_name
```

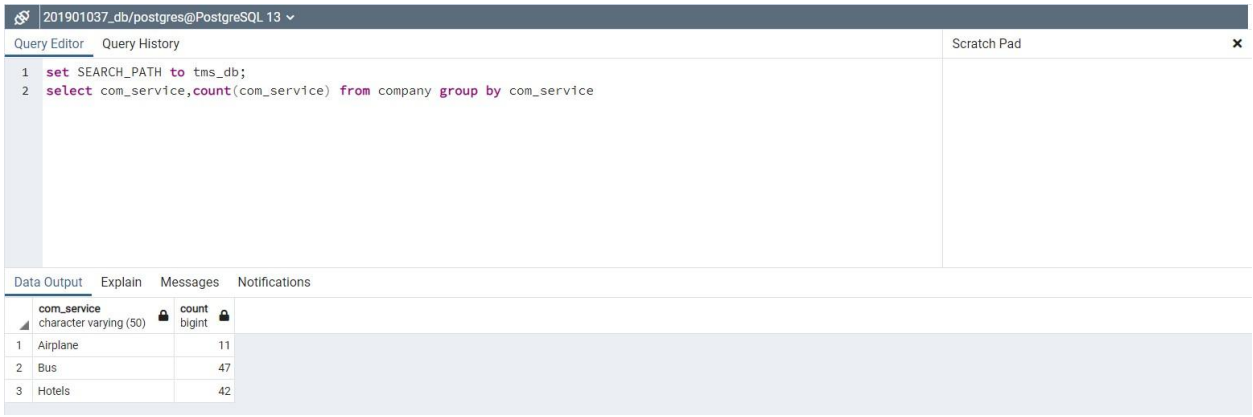
Data Output | Explain | Messages | Notifications

	t_name character varying (50)	count bigint
15	tour of jammu and kashmir	1
16	tour of central india	1
17	tour of chhattisgarh	1
18	tour of north india	1
19	Honeymoon Tourism	1
20	tour of madhyapradesh	1
21	Adventure Tourism	1
22	[...] tour of haryana	1
23	Wildlife Tourism	1
24	tour of arunachal pradesh	1
25	tour of west india	1
26	tour of chandigarh	1
27	tour of himachal pradesh	1
28	tour of jharkhand	1
29	Cultural Tourism Package	1
30	tour of east india	1

Number of tuples: 30

19. Find the total number of companies of each and every service.

select com_service, count(com_service) from company group by com_service;



201901037_db/postgres@PostgreSQL 13

```
set SEARCH_PATH to tms_db;
select com_service,count(com_service) from company group by com_service
```

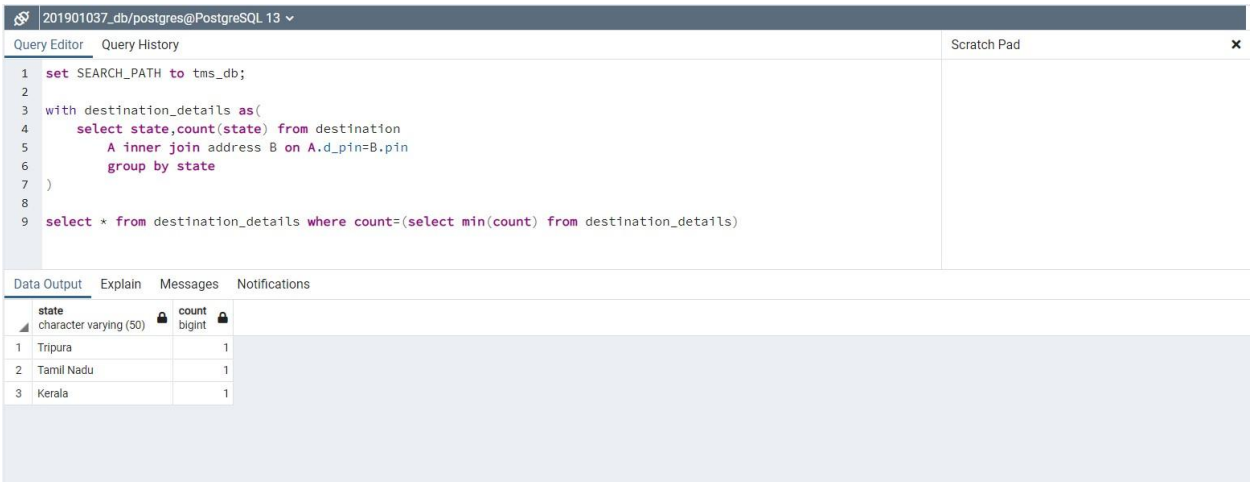
Data Output | Explain | Messages | Notifications

	com_service character varying (50)	count bigint
1	Airplane	11
2	Bus	47
3	Hotels	42

Number of tuples: 3

20. Find the states with the least number of destinations.

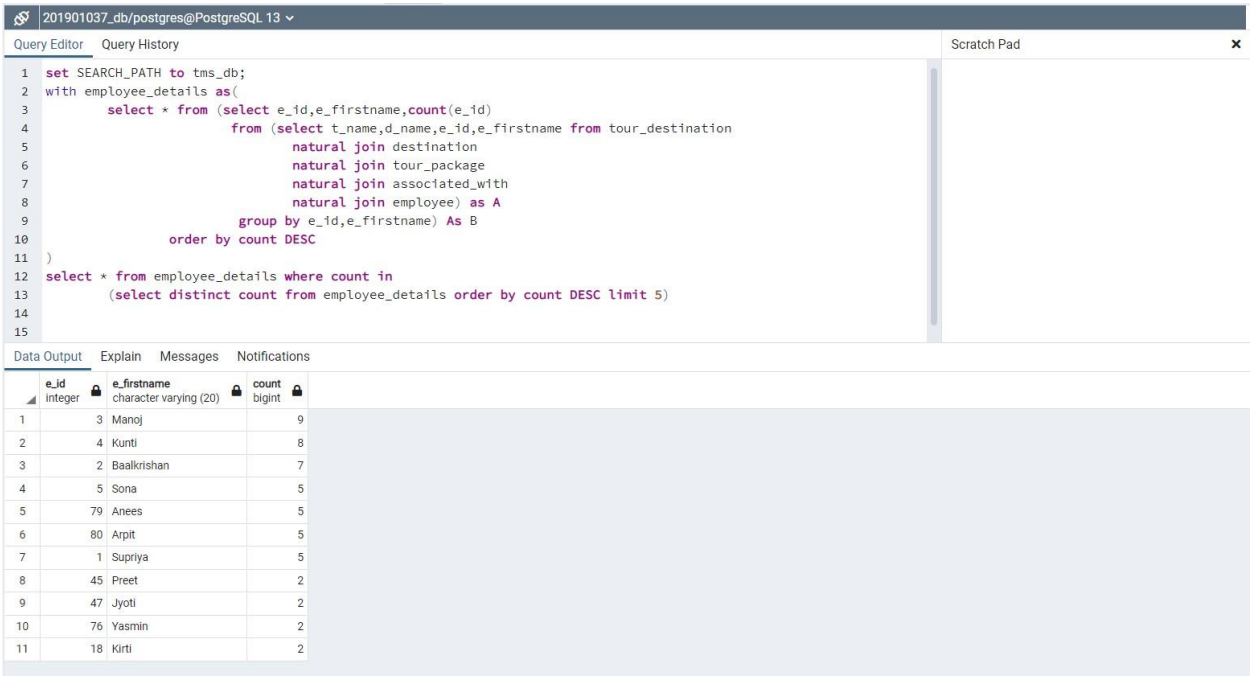
```
with destination_details as(
        Select state, count(state) from destination
        A inner join address B on A.d_pin = B.pin
        group by state)
select * from destination_details where count = (select min(count) from destination_details)
```



Number of tuples: 3

21. **List of top 5 tour guides who visited the maximum number of destinations.(consider the same number of counts as a single count.)**

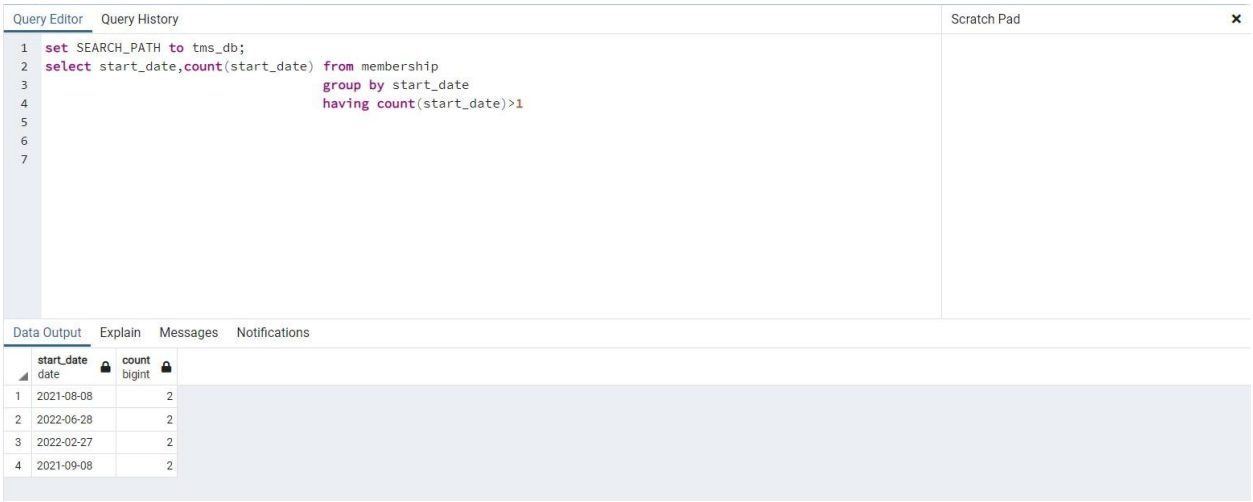
```
with emplloyee_details as(
        select  * from ( select t_name,d_name,e_id,e_firstname from tour_destination
        natural join destination
        natural join tour_package
        natural join associated_with
        natural join employee) as A
group by e_id,e_firstname) as B
Order by count DESC)

Select * from employee_details where count in
        (select distinct count from emloyee_details order by count DESC limit 5)
```



Number of tuples: 11

22. Write a query to find duplicate membership start dates.

Select start_date,count(start_date) from membership
Group by start_date
Having count(start_date)>1;



Number of tuples:4.

23. Find details of tours which have duration around 3 days and whose price is less than 10000 or after discount price is less than 15000.

Select * from tour_package A
Left join discount B on A.dis_id = B.dis_id
where t_duration like '3 Days%'
And ((A.dis_id is null and t_price<=10000) or t_price -(t_price*percentage)/100<= 15000)

Query Editor | Query History | Scratch Pad

```
set SEARCH_PATH to tms_db;
select * from tour_package A
            left join discount B on A.dis_id=B.dis_id
            where t_duration like '3 Days%'
            and ((A.dis_id is null and t_price<=10000) or t_price-(t_price*percentage)/100<=15000)
```

Data Output | Explain | Messages | Notifications

	t_id integer	t_name character varying (50)	t_duration character varying (30)	t_start_date date	t_price real	t_type character varying (30)	t_desc text
4	23	tour of chhattisgarh	3 Days and 2 Nights	2021-12-19	15999	[null]	Chhattisgarh is one such state in India where you get to cover all these aspects under one roof.
5	28	tour of kerala	3 Days and 2 Nights	2021-12-15	11999	[null]	Western India comprises of the states of Goa, Gujarat, Maharashtra, Madhya Pradesh and the u
6	29	tour of lakshdweep	3 Days and 2 Nights	2022-01-04	15999	[null]	Central India is simply the heart of the country, a soft and mystical land peppered with ancient t
7	35	tour of orissa	3 Days and 2 Nights	2022-01-21	6999	[null]	North India Tour will take you to its historic, majestic and celestial destinations such as Delhi, H
8	36	tour of pudduchery	3 Days and 2 Nights	2022-11-09	4999	[null]	South India is a mystic and magnificent destination that combines bright colours, fascinating c
9	42	tour of west bengal	3 Days and 2 Nights	2022-01-04	3999	[null]	Central India is simply the heart of the country, a soft and mystical land peppered with ancient t
10	45	Golden Triangle with Tiger Tour	3 Days and 2 Nights	2022-04-28	8999	[null]	A tour through East India takes you through a dramatically varied landscape: mangroves in the
11	58	Bandhavgarh National Park	3 Days and 2 Nights	2022-06-23	3999	[null]	A tour through East India takes you through a dramatically varied landscape: mangroves in the
12	66	Alor Holiday Resort	3 Days and 2 Nights	2021-12-02	3999	[null]	A tour through East India takes you through a dramatically varied landscape: mangroves in the
13	74	Weekend Getaways Cochin	3 Days and 2 Nights	2021-12-27	3999	[null]	North India Tour will take you to its historic, majestic and celestial destinations such as Delhi, H
14	79	Trekking	3 Days and 2 Nights	2022-09-13	15999	[null]	A tour through East India takes you through a dramatically varied landscape: mangroves in the
15	78	Weekend Getaways Jaipur	3 Days and 2 Nights	2022-02-05	3999	[null]	Central India is simply the heart of the country, a soft and mystical land peppered with ancient t

Number of tuples:15.

24.**find the name of all employees whose name starts with 'Ch'**

select e_firstname, e_middlename,e_lastname
From employee
where e_firstname like 'Ch%';

Dbms_proj/postgres@PostgreSQL 13

Query Editor | Query History | Scratch Pad

```
set search_path to tms

select e_firstname,e_middlename,e_lastname
from employee
where e_firstname like 'Ch%';
```

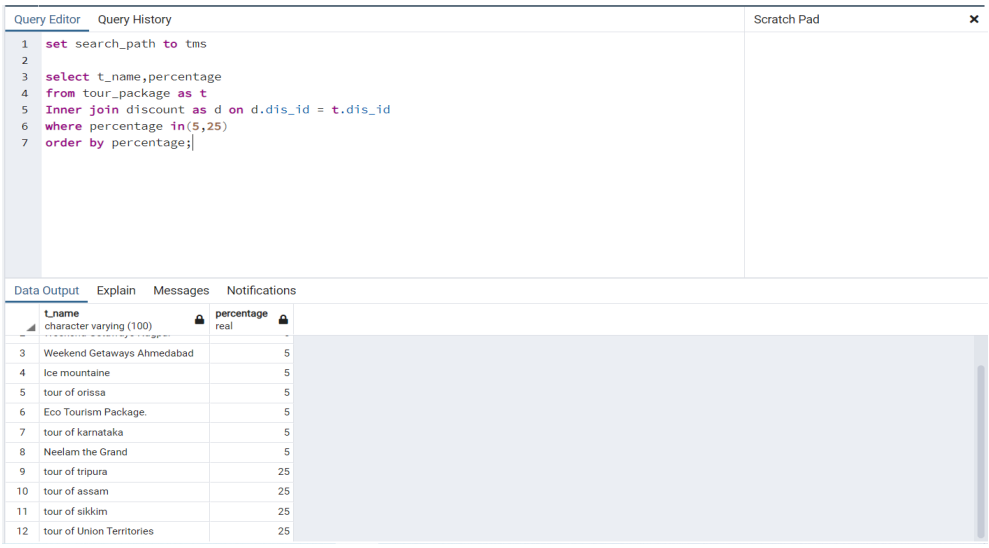
Data Output | Explain | Messages | Notifications

	e_firstname character varying (20)	e_middlename character varying (20)	e_lastname character varying (20)
1	Chirag	Mahmood	Murthy
2	Chinmay	Lakshmi	Ramachandran
3	Chhaya	Yash	Chhabra

Number of tuples:3.

25. Find the discount percentage of each tour package.

```
Select t_name,percentage
From tour_package as t
Inner join discount as d on d.dis_id = t.dis_id
Where percentage in (5,25)
Order by percentage;
```



Number of tuples: 12

26. Find reviews which contain Hotel keywords and relate it with hotels.

```
select h_name, review.description from available
                natural join tour_package
                natural join review
                natural join hotel
                where lower(review.description) like '%hotel%'
```

```
set SEARCH_PATH to tms_db;
select h_name,review.description from available
                natural join tour_package
                natural join review
                natural join hotel
                where lower(review.description) like '%hotel%'
```

	h_name character varying (50)	description text
1	The New Yorker	Hotel room is not clean. Bus is not comfortable
2	Hotel Agoura	i absolutely ridiculous I will NEVERRRRR IN MY LIFE WASTE MONEY ON A hotel ROOM LIKE THAT AGAIN
3	nandini	The hotel room was disgusting! Very dirty. You could see foot prints on the floor where other people walked! It looked as though it had not been swept in a very long time! Seemed like a good room for smoking crack.
4	sahyog	They offered clean and tidy hotel room to us. Though we requested ahead to check in early they didn't allow us to check in. Service was good and staff was polite.
5	The New View	Hotel room is not clean. Bus is not comfortable
6	Time Motel	Hotel room is not clean. Bus is not comfortable
7	The Venetian	Trip was so well organized and very perfect. But the Hotel is not value for money
8	Comfort B&B	The hotel room was disgusting! Very dirty. You could see foot prints on the floor where other people walked! It looked as though it had not been swept in a very long time! Seemed like a good room for smoking crack.

Number of tuples: 8

27. Find the hotels from each type which has a minimum number of available rooms.

```
Select h_name,h_type,avail_room from hotel
Where
Avail_room in (
        Select min(avail_room)
        From hotel
        Group by(h_type)
)
Order by avail_room DESC;
```

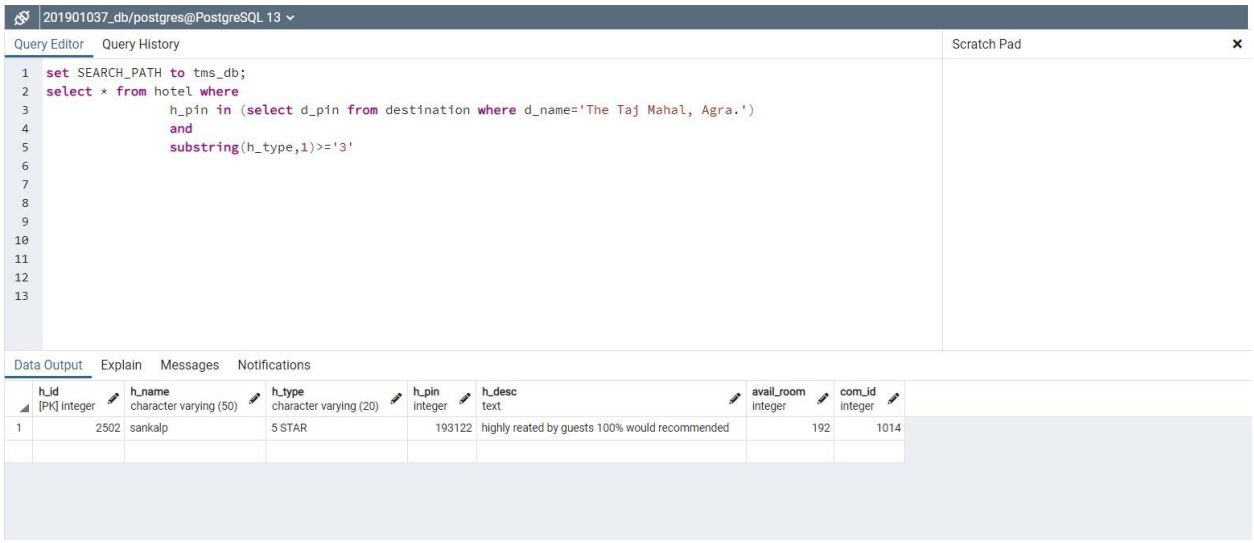
```
set search_path to tms

select h_name,h_type,avail_room from hotel
where
avail_room in (
    select min(avail_room)
    from hotel
    group by(h_type)
)
order by avail_room DESC;
```

	h_name character varying (50)	h_type character varying (20)	avail_room integer
1	Fountain Fun	4 STAR	18
2	Motel On Main	3 STAR	15
3	Hotel Agoura	5 STAR	11

Number of tuples: 3

28. List of hotels near Taj Mahal and has more than 2 STAR.

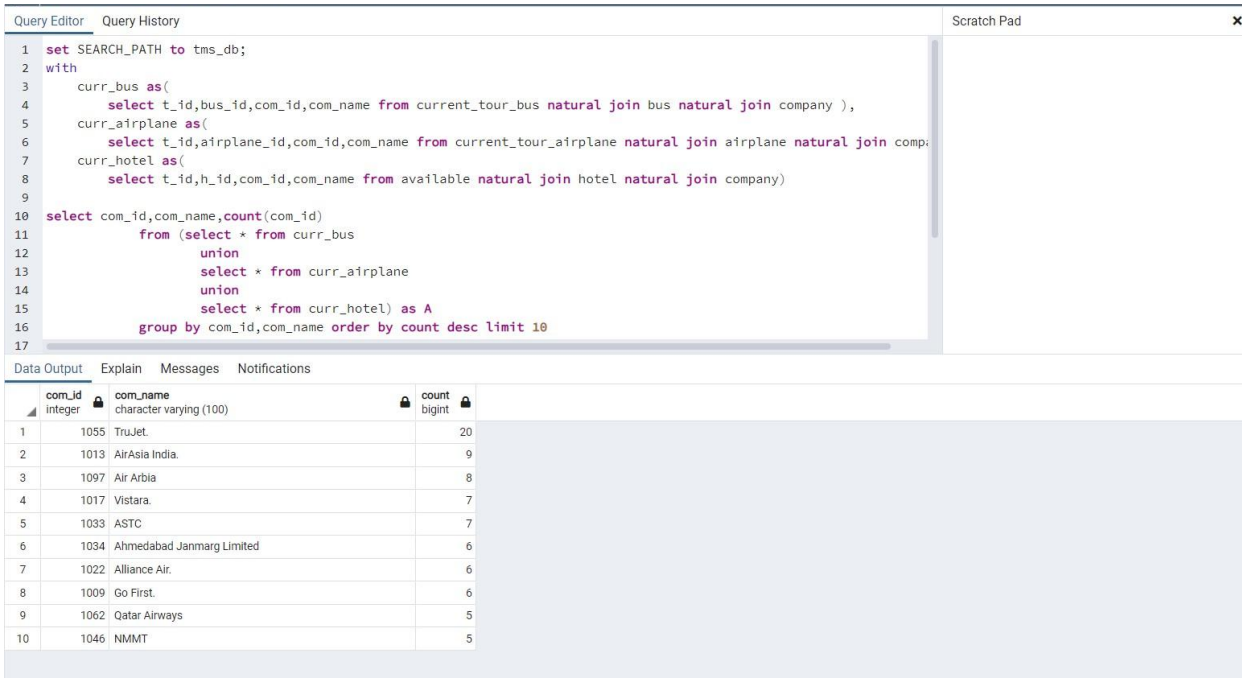
select * from hotel where h_pin in (select d_pin from destination where d_name='The Taj Mahal, Agra.') and substring(h_type,1)>='3'



Number of tuples: 1

29. Top 10 most service providing companies.

```
with
        curr_bus as(
        select t_id,bus_id,com_id,com_name from current_tour_bus natural join bus natural join company
),
        curr_airplane as(
        select t_id,airplane_id,com_id,com_name from current_tour_airplane natural join airplane natural
join company),
        curr_hotel as(
        select t_id,h_id,com_id,com_name from available natural join hotel natural join company)

select com_id,com_name,count(com_id)
                        from (select * from curr_bus
                                        union
                                        select * from curr_airplane
                                        union
                                        select * from curr_hotel) as A
                        group by com_id,com_name order by count desc limit 10
```



Number of tuples: 10

30. Most purchased membership's customer's most visited destinations.

```
with detination_member as
        (select d_id,d_name,count(d_id) from
                (select booking_id,A.t_id,d_id,d_name from booking  A
                        natural join tour_package B
                        join tour_destination C  on C.t_id=B.t_id
                        natural join destination
                        where c_id in
                                (select c_id from membership
                                        where m_id in
                                        (select m_id from
                                                (select  m_id,count(m_id)
                                                        from membership
                                                        group by m_id
                                                        order by count desc
                                                        limit 1
                                                ) as A
                                        )
                                )
                ) as B
                group by d_id,d_name order by count Desc
        )
select * from detination_member where count =(select max(count) from detination_member )
```

```
set SEARCH_PATH to tms_db;
with detination_member as
    (select d_id,d_name,count(d_id) from
        (select booking_id,A.t_id,d_id,d_name from booking  A
            natural join tour_package B
            join tour_destination C  on C.t_id=B.t_id
            natural join destination
            where c_id in
                (select c_id from membership
                    where m_id in
                    (select m_id from
                        (select  m_id,count(m_id)
                            from membership
                            group by m_id
                            order by count desc
                            limit 1
                        ) as A
                    )
                )
         ) as B
        group by d_id,d_name order by count Desc
    )
select * from detination_member where count =(select max(count) from detination_member )
```

d_id integer	d_name character varying (100)	count bigint
2	The Taj Mahal, Agra.	11

Number of tuples: 1.

31. Find names of the second most visited tour package by customers.

```
with ans as( select t_id,t_name,count(t_id) from booking
                                    natural join tour_package
                                    group by t_id,t_name
                                    order by count desc)
select t_name from ans where count=(select distinct count from ans order by count desc offset 1 limit 1 )
```

```
set SEARCH_PATH to tms_db;

with ans as( select t_id,t_name,count(t_id) from booking
                natural join tour_package
                group by t_id,t_name
                order by count desc)
select t_name from ans where count=(select distinct count from ans order by count desc offset 1 limit 1 )
```

t_name character varying (50)
Weekend Getaways Pune
tour of Union Territories
Corbett National Park
tour of kerala
tour of arunachal pradesh
tour of maharashtra
Weekend Getaways Ahmedabad
tour of chhattisgarh
Weekend Getaways Chennai
Div and Daman
tour of west india
Weekend Getaways Mumbai
tour of nagaland
tour of manipur
tour of mizoram
summer trip

Number of tuples: 79.

32. Top 5 hotel provider companies name and contact details.

```
with hotel_com as(
        select com_id,count(com_id) from hotel
                natural join company
                group by com_id
                order by com_id desc)
select com_name,contact_no  from company natural join company_contact_details where com_id in
                        (select com_id from hotel_com
                                order by count desc limit 5)
```

```
set SEARCH_PATH to tms_db;
with hotel_com as(
    select com_id,count(com_id) from hotel
        natural join company
        group by com_id
        order by com_id desc)

select com_name,contact_no  from company natural join company_contact_details
               where com_id in
               (select com_id from hotel_com
                   order by count desc limit 5)
```

Data Output | Explain | Messages | Notifications

	com_name character varying (100)	contact_no numeric (10)
1	Lake Place Inn	6147463602
2	Lake Place Inn	9105681105
3	Beacon Motel	9015380231
4	Comfort B&B	6155634628
5	The Fresco Hotel	7037419247
6	Motel On Main	8292489556

Number of tuples: 6.

33. Find the number of online transactions and number of offline transactions.

```
select 'online' as booking_type,count(*) from booking where b_type='online'
            union
select 'offline' as booking_type,count(*) from booking where b_type='offline'
```

Query Editor | Query History | Scratch Pad

```
set SEARCH_PATH to tms_db;

select 'online' as booking_type,count(*) from booking where b_type='online'
union
select 'offline' as booking_type,count(*) from booking where b_type='offline'
```

Data Output | Explain | Messages | Notifications

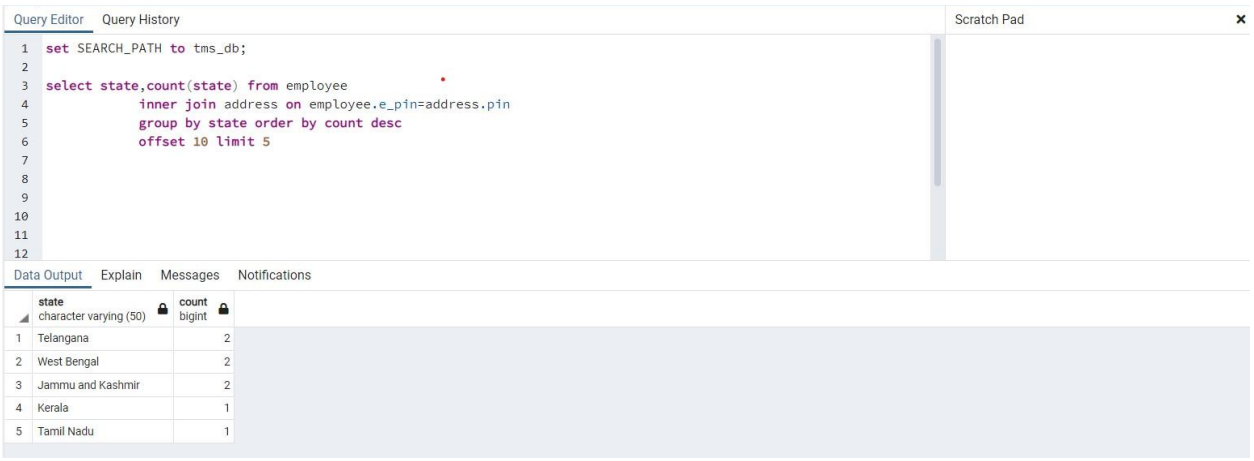
	booking_type text	count bigint
1	offline	45
2	online	44

No of tuples : 2

34. Find a state which has a maximum number of employees with offset 10 and limit 5.

set SEARCH_PATH to tms_db;

```
select state,count(state) from employee
                inner join address on employee.e_pin=address.pin
                group by state order by count desc
                offset 10 limit 5
```

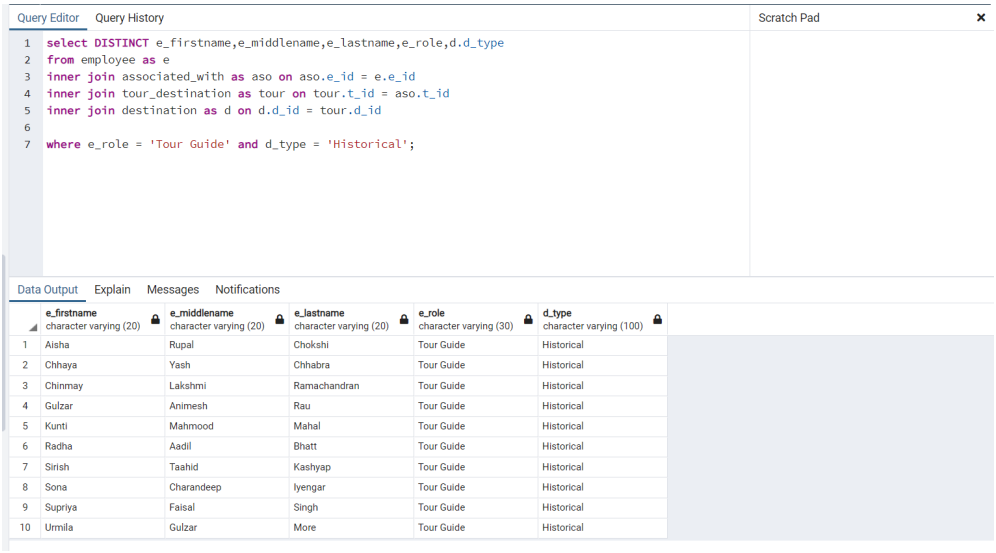


	state character varying (50)	count bigint
1	Telangana	2
2	West Bengal	2
3	Jammu and Kashmir	2
4	Kerala	1
5	Tamil Nadu	1

No of tuples: 5

35. Find the employee details who guided in historical destinations.

```
select e_firstname,e_middlename from employee as e
                inner join associated_with as aso on aso.e_id = e.e_id
                inner join tour_destination as tour on tour.t_id = aso.t_id
                inner join destination as d on d.d_id = tour.d_id
                where e_role = 'Tour Guide' and d_type = 'Historical';
```

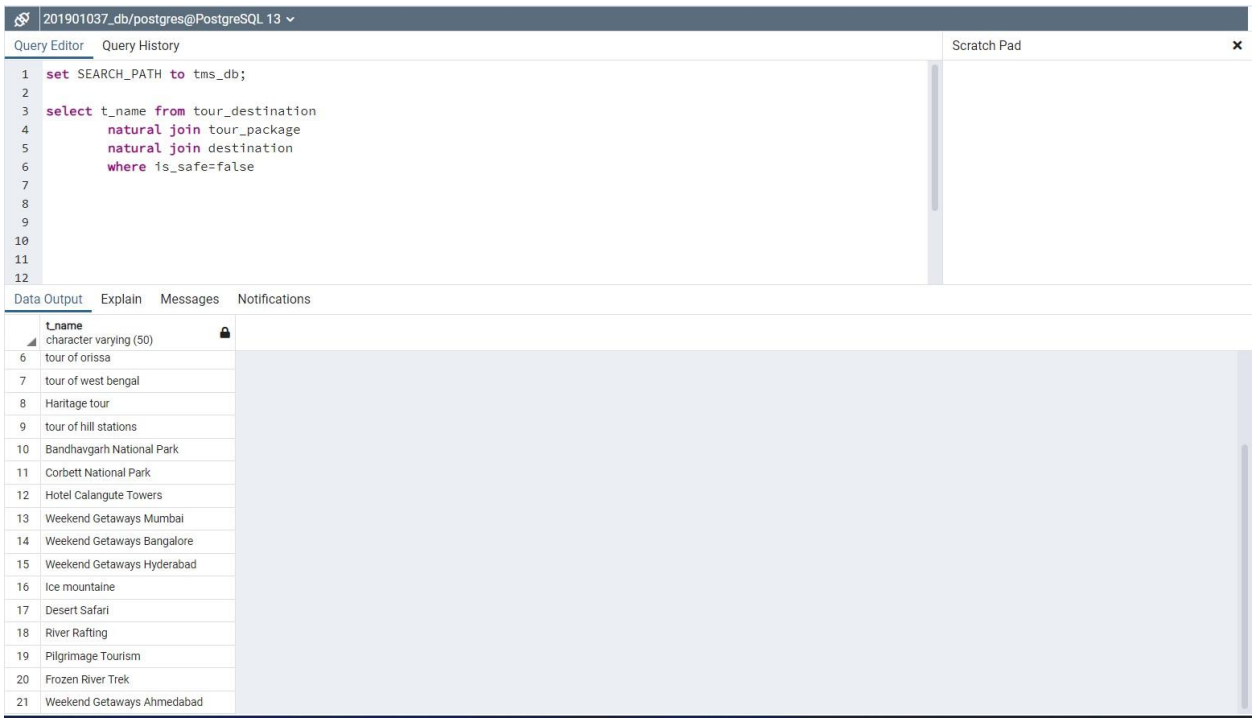


	e_firstname character varying (20)	e_middlename character varying (20)	e_lastname character varying (20)	e_role character varying (30)	d_type character varying (100)
1	Aisha	Rupal	Chokshi	Tour Guide	Historical
2	Chhaya	Yash	Chhabra	Tour Guide	Historical
3	Chinmay	Lakshmi	Ramachandran	Tour Guide	Historical
4	Gulzar	Animesh	Rau	Tour Guide	Historical
5	Kunti	Mahmood	Mahal	Tour Guide	Historical
6	Radha	Aadil	Bhatt	Tour Guide	Historical
7	Sirish	Taahid	Kashyap	Tour Guide	Historical
8	Sona	Charandeep	Iyengar	Tour Guide	Historical
9	Supriya	Faisal	Singh	Tour Guide	Historical
10	Urmila	Gulzar	More	Tour Guide	Historical

No of tuples: 10

36. Find tour packages which contain tour destinations which are not safe from Covid19.

```
select t_name from tour_destination
                natural join tour_package
                natural join destination
                where is_safe=false
```



No of tuples: 21

37. Write a query to find the total number of the customer in the city with a city name of at least 2 characters and having a second character 'A'.

```
select city,count(city) from customer
                A inner join address B on B.pin=A.c_pin
                group by city
                having city like '_A%'
```

Query Editor | Query History | Scratch Pad

```
1  set SEARCH_PATH to tms_db;
2
3  select city,count(city) from customer
4          A inner join address B on B.pin=A.c_pin
5          group by city
6          having city like '_A%'
7
8
9
10
11
12
```

Data Output | Explain | Messages | Notifications

	city character varying (50)	count bigint
10	BALLARI	1
11	BAGESHWAR	1
12	BAREILLY	1
13	BARABANKI	1
14	BALOD BAZER	1
15	BALLIA	1
16	BANDIPUR	1
17	BANDA	1
18	BAGALKOT	1
19	BASTI	1
20	BALESWAR	1
21	BARAN	1
22	BALAGHAT	1
23	BANASKANTHA	1
24	BARWANI	1
25	BALOD	1

No of tuples: 25

38. Count the total number of images for each tour package (destination images + Hotel images) in the database and sort them in decreasing order of image count.

```
with images as (select * from tour_destination natural join destination_image
                                    union
                                select * from available natural join hotel_image)

select t_id,count(t_id) from images group by t_id order by count desc
```

```
set SEARCH_PATH to tms_db;

with images as(select * from tour_destination natural join destination_image
                union
               select * from available natural join hotel_image)

select t_id,count(t_id) from images group by t_id order by count desc
```

Data Output | Explain | Messages | Notifications

	t_id integer	count bigint
57	29	1
58	34	1
59	83	1
60	35	1
61	6	1
62	50	1
63	75	1
64	46	1
65	53	1
66	12	1
67	24	1
68	49	1
69	64	1
70	27	1
71	23	1
72	58	1

No of tuples: 72

39. Find the number of destinations having the same destination type. Destinations should be located in Gujarat, Maharashtra, and Rajasthan.

select d_type, count(*) from
destination inner join address
on destination.d_pin = address.pin
where address.state in ('Gujarat','Maharashtra','Rajasthan')
group by d_type;

Query Editor | Query History

```
select d_type, count(*) from
destination inner join address
on destination.d_pin = address.pin
where address.state in ('Gujarat','Maharashtra','Rajasthan')
group by d_type;
```

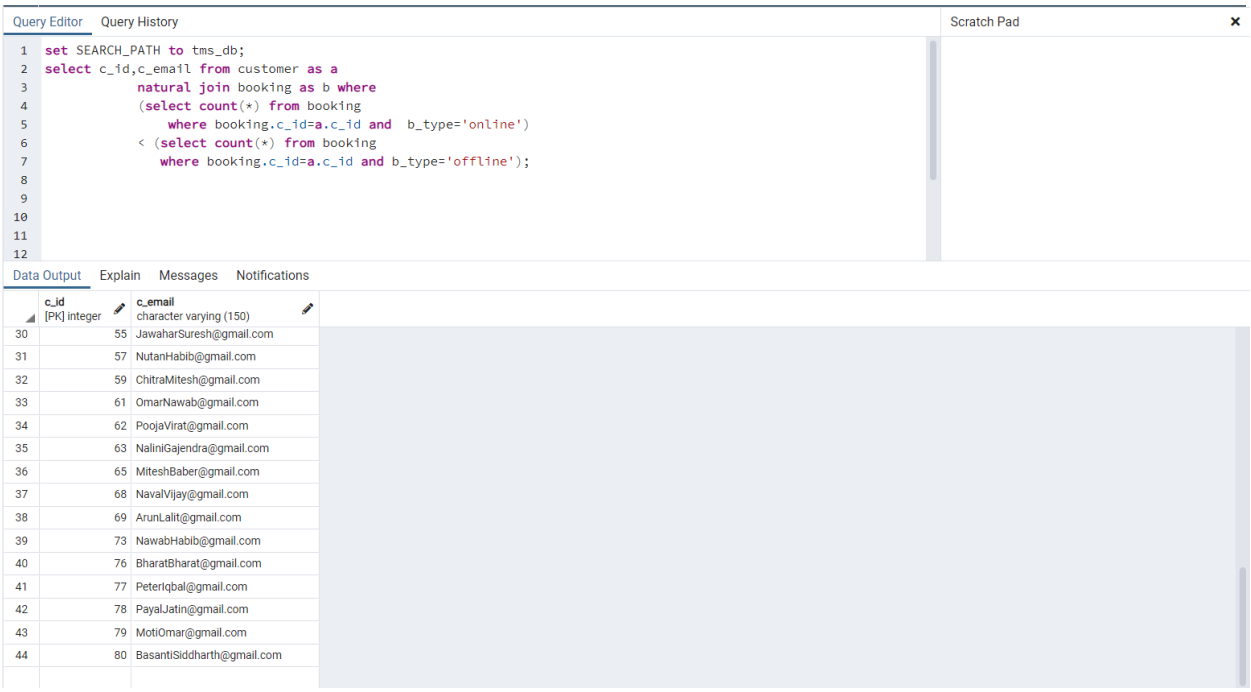
Data Output | Explain | Messages | Notifications

	d_type character varying (20)	count bigint
1	Religious Place	6
2	Mountain	1
3	Beach	1
4	Historical	11
5	Forests	1
6	Adventure	1

No of tuples : 6

40. **Find customer ID and email address of customers who have booked online more than offline.**

```
select c_id,c_email from customer as a
                natural join booking as b where
                (select count(*) from booking
                        where booking.c_id=a.c_id and  b_type='online')
                < (select count(*) from booking
                  where booking.c_id=a.c_id and b_type='offline');
```



Query Editor | Query History | Scratch Pad

```
set SEARCH_PATH to tms_db;
select c_id,c_email from customer as a
            natural join booking as b where
            (select count(*) from booking
                where booking.c_id=a.c_id and  b_type='online')
            < (select count(*) from booking
               where booking.c_id=a.c_id and b_type='offline');
```

Data Output | Explain | Messages | Notifications

	c_id [PK] integer	c_email character varying (150)
30	55	JawaharSuresh@gmail.com
31	57	NutanHabib@gmail.com
32	59	ChitraMitesh@gmail.com
33	61	OmarNawab@gmail.com
34	62	PoojaVirat@gmail.com
35	63	NaliniGajendra@gmail.com
36	65	MiteshBaber@gmail.com
37	68	NavalVijay@gmail.com
38	69	ArunLalit@gmail.com
39	73	NawabHabib@gmail.com
40	76	BharatBharat@gmail.com
41	77	PeterIqbal@gmail.com
42	78	PayalJatin@gmail.com
43	79	MotiOmar@gmail.com
44	80	BasantiSiddharth@gmail.com

No of tuples: 44

Trigger Functions

1. bill

```
CREATE FUNCTION tms_db.bill()
RETURNS trigger
LANGUAGE 'plpgsql'
NOT LEAKPROOF
AS $BODY$
DECLARE bill_id integer;
DECLARE tour_price real;
DECLARE dis real;
DECLARE ds_id integer;
DECLARE mem_id integer;
BEGIN
```

```
select max(billing_id)+1 into bill_id from billing;
select t_price,dis_id into tour_price,ds_id from tour_package where t_id=NEW.t_id;
select m_id into mem_id from membership where c_id=NEW.c_id;

if(mem_id is not null)
        then select (2*m_price)/1000 into dis from membership_details where m_id=mem_id;
else
        dis=0;
end if;

if (ds_id is not null)
   then select dis+percentage into dis from discount where dis_id=ds_id;
end if;

insert into billing values
(bill_id,new.booking_id,current_date,tour_price*NEW.tot_b_seat-tour_price*NEW.tot_b_seat*dis/100);
return null;
end
$BODY$;


CREATE TRIGGER call_bill
  AFTER INSERT
   ON tms_db.booking
   FOR EACH ROW
   EXECUTE FUNCTION tms_db.bill();
```

2. curr_bus

```
CREATE FUNCTION tms_db.curr_bus()
   RETURNS trigger
   LANGUAGE 'plpgsql'
    NOT LEAKPROOF
AS $BODY$
BEGIN
        update bus set status=false where bus_id = new.bus_id;
        return null;
END
$BODY$;

ALTER FUNCTION tms_db.curr_bus()
   OWNER TO postgres;

 CREATE TRIGGER call_curr_bus
   AFTER INSERT
   ON tms_db.current_tour_bus
   FOR EACH ROW
   EXECUTE FUNCTION tms_db.curr_bus();
```

3. curr_airplane

```
CREATE FUNCTION tms_db.curr_airplane()
    RETURNS trigger
    LANGUAGE 'plpgsql'
     NOT LEAKPROOF
AS $BODY$
BEGIN
            update airplane set status=false where airplane_id = new.airplane_id;
            return null;
END
$BODY$;

ALTER FUNCTION tms_db.curr_airplane()
    OWNER TO postgres;

CREATE TRIGGER call_curr_plane
    AFTER INSERT
    ON tms_db.current_tour_airplane
    FOR EACH ROW
    EXECUTE FUNCTION tms_db.curr_airplane();
```

4. delete_curr_bus

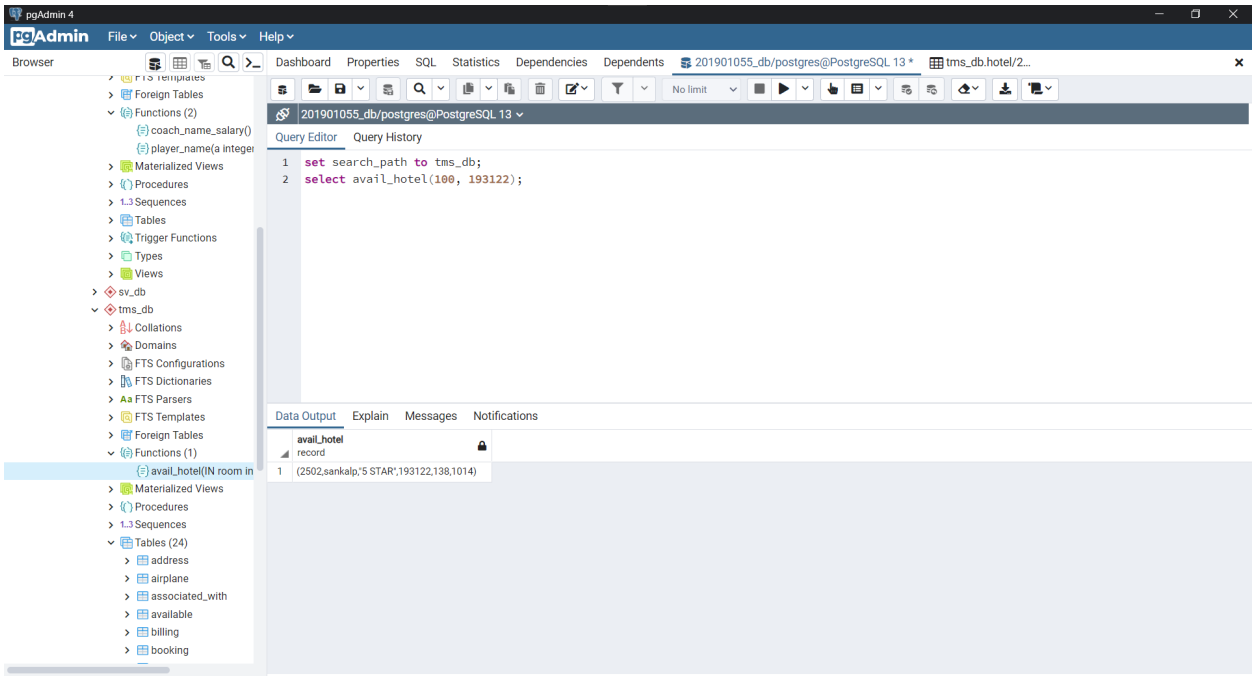
```
CREATE OR REPLACE FUNCTION tms_db.delete_curr_bus()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    VOLATILE
    COST 100
AS $BODY$
BEGIN
            update bus set status=true,avail_seats=seater_seats+sleeper_seats where bus_id = old.bus_id;
            return null;
END
$BODY$;

ALTER FUNCTION tms_db.delete_curr_bus()
    OWNER TO postgres;

CREATE TRIGGER call_delete_curr_bus
    AFTER DELETE
    ON tms_db.current_tour_bus
    FOR EACH ROW
    EXECUTE FUNCTION tms_db.delete_curr_bus();
```

Functions

1. avail_hotel

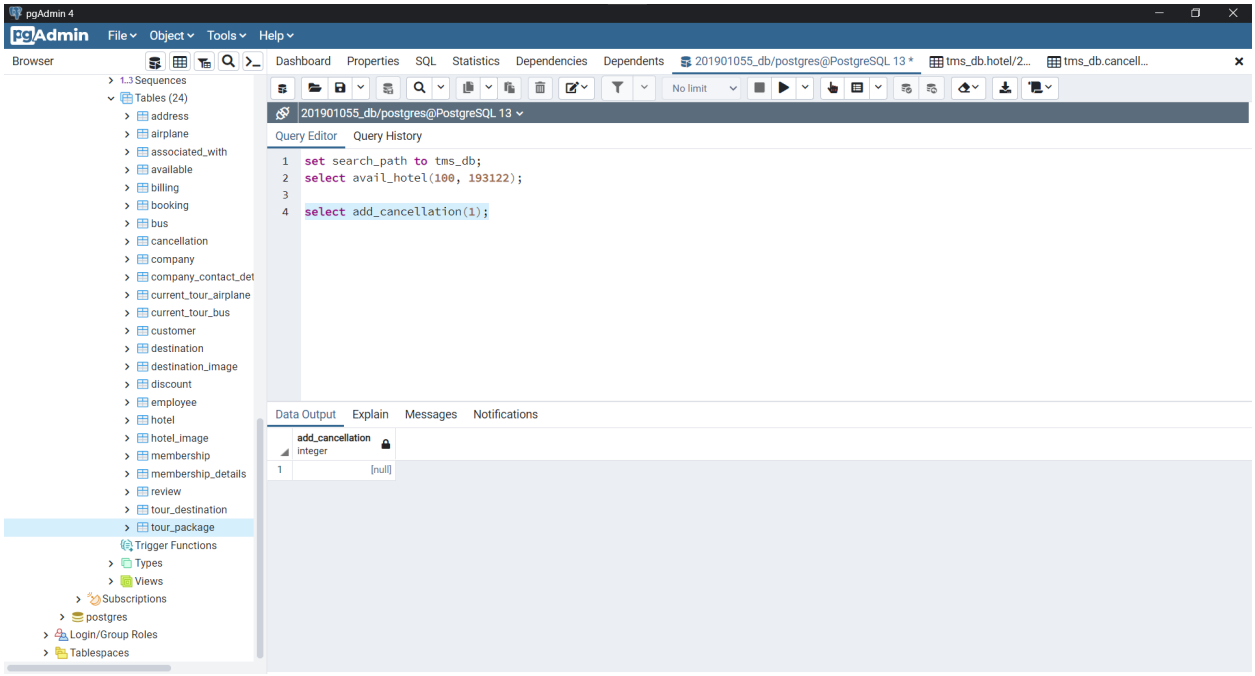
```
CREATE OR REPLACE FUNCTION tms_db.avail_hotel (
	IN room INTEGER
	,IN pin INTEGER)
RETURNS TABLE
	(
	  h_id INTEGER
	  ,h_name CHARACTER VARYING
	  ,h_type CHARACTER VARYING
	  ,h_pin INTEGER
	  ,avail_room INTEGER
	  ,com_id INTEGER
	) LANGUAGE 'plpgsql' VOLATILE PARALLEL UNSAFE COST 100 ROWS 1000 AS $BODY$
DECLARE R_LIST2 record;
BEGIN CREATE
   TEMP TABLE
	test1 (
	  h_id INTEGER,
	  h_name CHARACTER VARYING (50),
	  h_type CHARACTER VARYING (20),
	  h_pin INTEGER,
	  avail_room INTEGER,
	  com_id INTEGER,
	) ON COMMIT DROP;
 FOR R_LIST2 IN (
   SELECT
	  hotel.h_id
	  ,hotel.h_name
	  ,hotel.h_type
	  ,hotel.h_pin
	  ,hotel.avail_room
	  ,hotel.com_id
	FROM
	  hotel
	WHERE
	  hotel.avail_room >= room
	  AND hotel.h_pin = pin
)
LOOP INSERT
```

```
    INTO
        test1 (
            h_id
            ,h_name
            ,h_type
            ,h_pin
            ,avail_room
            ,com_id
        )
    VALUES (
        R_LIST2.h_id
        ,R_LIST2.h_name
        ,R_LIST2.h_type
        ,R_LIST2.h_pin
        ,R_LIST2.avail_room
        ,R_LIST2.com_id
    );
END LOOP;
RETURN QUERY TABLE test1;
END $BODY$;
```



2. add_cancellation

```
CREATE OR REPLACE FUNCTION tms_db.add_cancellation(IN bill_id integer)
    RETURNS integer
    LANGUAGE 'plpgsql'
    VOLATILE
    PARALLEL UNSAFE
    COST 100

AS $BODY$
DECLARE canc_id integer;
DECLARE diff integer;
DECLARE s_date date;
DECLARE book_id integer;
DECLARE per real;
DECLARE refund real;

BEGIN
select booking_id into book_id from billing where billing_id = bill_id;
select t_start_date into s_date from tour_package join booking on booking.t_id = tour_package.t_id where
booking.booking_id = book_id;
select s_date - current_date into diff;
if(diff>=5) then
        per = 0.9;
end if;
if(diff>=3 and diff<5) then
        per = 0.8;
end if;
if(diff>=1 and diff<3) then
        per = 0.6;
end if;
if(diff<1) then
        per = 0;
end if;
select count(*)+1 into canc_id from cancellation;
select price*per into refund from billing where billing_id = bill_id;
insert into cancellation values(canc_id, bill_id, current_date, refund);
return NULL;
END
$BODY$;
```


pgAdmin 4
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
201901055_db/postgres@PostgreSQL 13 *
tms_db.hotel/2...
tms_db.cancell...
No limit
Sequences
Tables (24)
address
airplane
associated_with
available
billing
booking
bus
cancellation
company
company_contact_det
current_tour_airplane
current_tour_bus
customer
destination
destination_image
discount
employee
hotel
hotel_image
membership
membership_details
review
tour_destination
tour_package
Trigger Functions
Types
Views
Subscriptions
postgres
Login/Group Roles
Tablespaces
201901055_db/postgres@PostgreSQL 13
Query Editor
Query History
1 set search_path to tms_db;
2 select avail_hotel(100, 193122);
3
4 select add_cancellation(1);
Data Output
Explain
Messages
Notifications
add_cancellation
integer
1
[null]

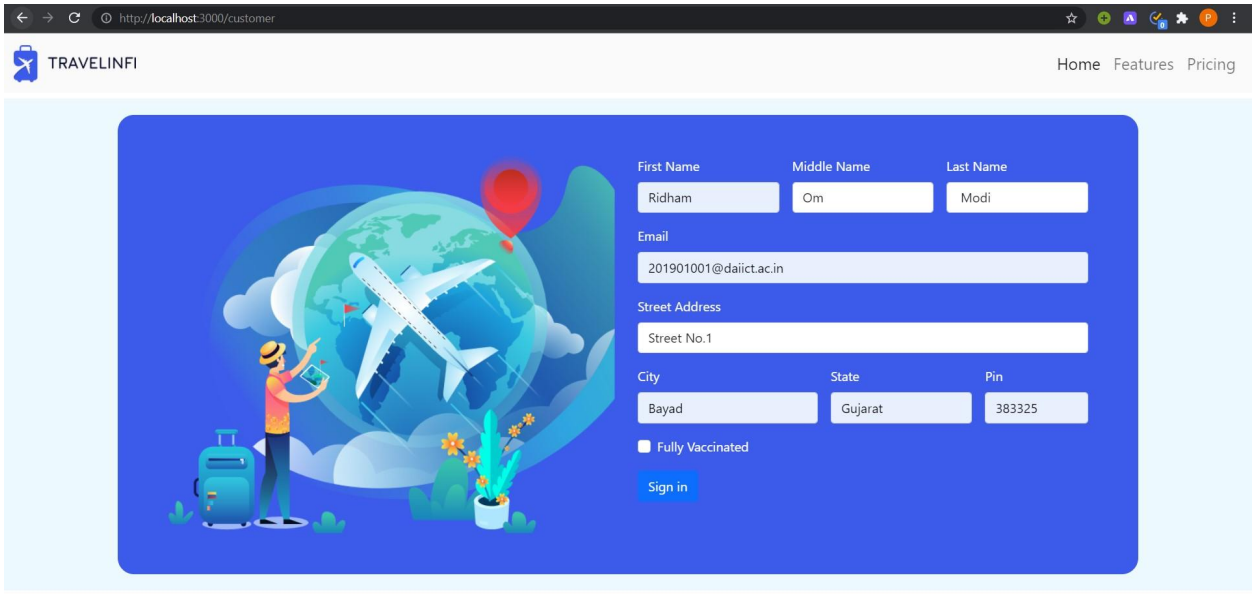
Section7: Project Code with output screenshots.

- **Details of implementation**
  - Frontend: React.js
  - Backend: Node.js Express.js
  - Database: PostgreSQL

- **Project Github Link:**
  https://github.com/bhargav-01/Travel-Management-System

- **Screenshots**

- **Customer**



Before insertion

Query Editor | Query History

```
SELECT * FROM tms_db.customer
ORDER BY c_id ASC
```

Data Output | Explain | Messages | Notifications

c_id [PK] integer	c_firstname character varying (50)	c_middlename character varying (50)	c_lastname character varying (50)	c_email character varying (150)	c_street character varying (100)	vaccination_status boolean	c_pin integer
72	72	Mowgli	Chinmay	Varkey	MowgliChinmay@gmail.com	Steet No.1	true	608901
73	73	Nawab	Habib	Ramroop	NawabHabib@gmail.com	Steet No.10	false	686102
74	74	Mayawati	Koushtubh	Raj	MayawatiKoushtubh@gmail.com	Steet No.22	true	713101
75	75	Yamini	Kamlesh	Salvi	YaminiKamlesh@gmail.com	Steet No.6	true	722101
76	76	Bharat	Bharat	Barman	BharatBharat@gmail.com	Steet No.6	false	754223
77	77	Peter	Iqbal	Ben	PeterIqbal@gmail.com	Steet No.25	true	756056
78	78	Payal	Jatin	Raja	PayalJatin@gmail.com	Steet No.11	false	756100
79	79	Moti	Omar	Naruka	MotiOmar@gmail.com	Steet No.12	true	759021
80	80	Basanti	Siddharth	Gokhale	BasantiSiddharth@gmail.com	Steet No.20	true	767001

After insertion :

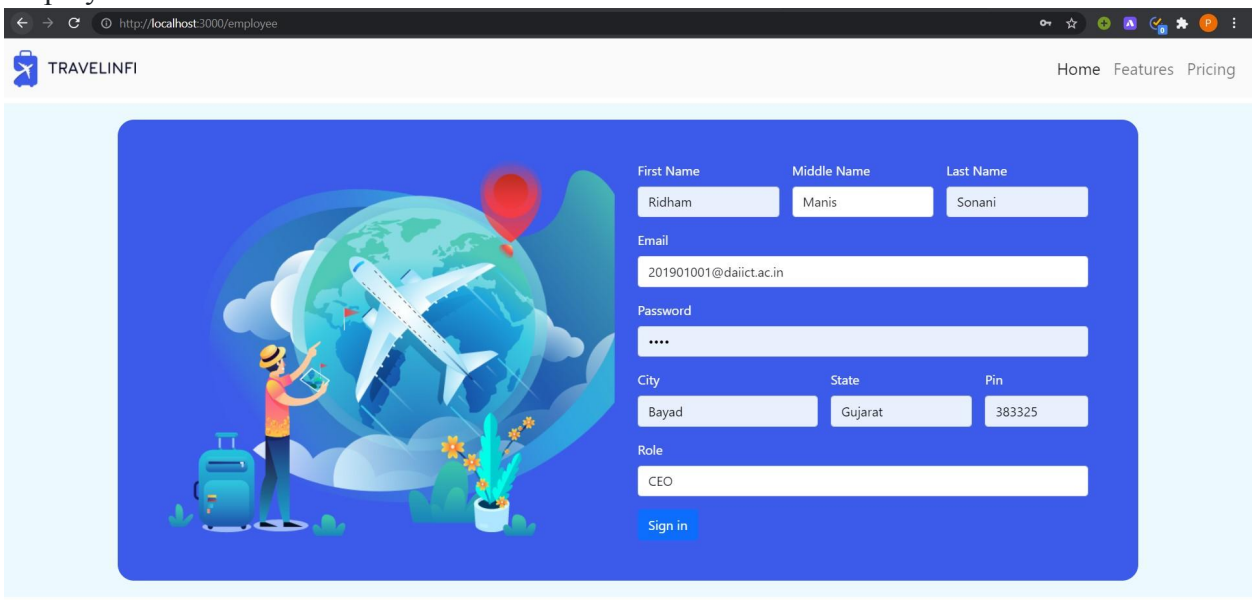
Query Editor | Query History

```
SELECT * FROM tms_db.customer
ORDER BY c_id ASC
```

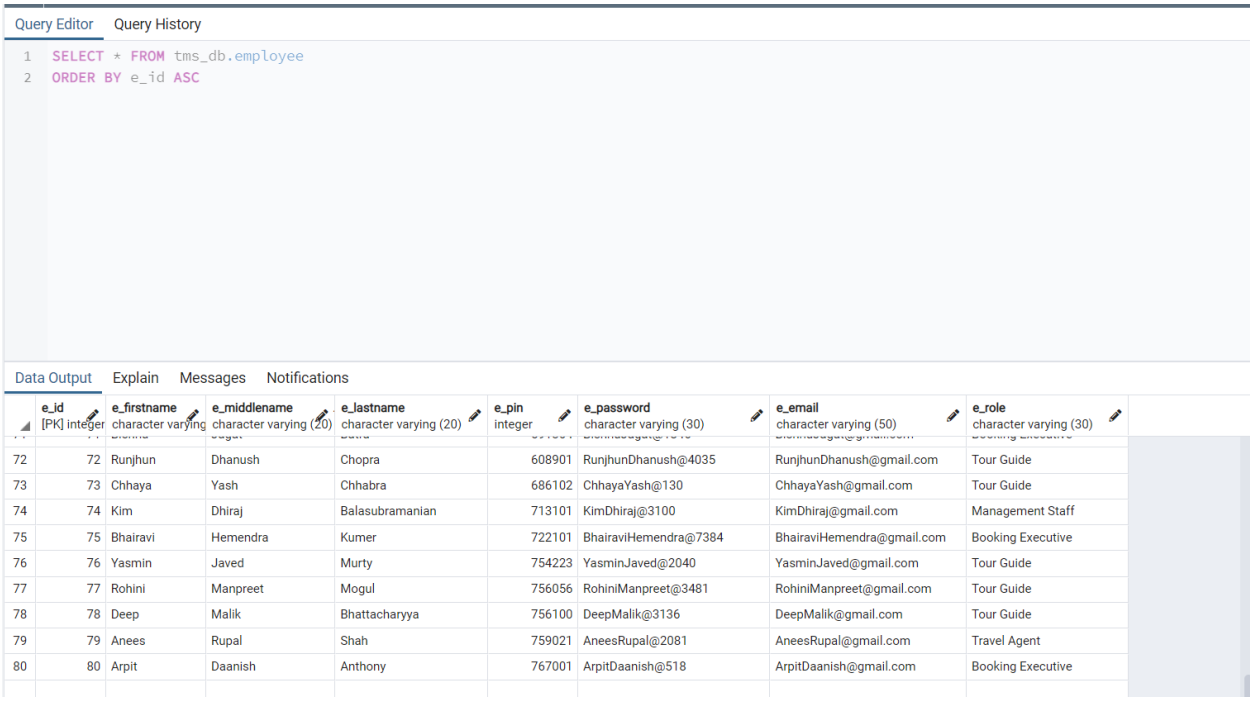
Data Output | Explain | Messages | Notifications

c_id [PK] integer	c_firstname character varying (50)	c_middlename character varying (50)	c_lastname character varying (50)	c_email character varying (150)	c_street character varying (100)	vaccination_status boolean	c_pin integer
73	73	Nawab	Habib	Ramroop	NawabHabib@gmail.com	Steet No.10	false	686102
74	74	Mayawati	Koushtubh	Raj	MayawatiKoushtubh@gmail.com	Steet No.22	true	713101
75	75	Yamini	Kamlesh	Salvi	YaminiKamlesh@gmail.com	Steet No.6	true	722101
76	76	Bharat	Bharat	Barman	BharatBharat@gmail.com	Steet No.6	false	754223
77	77	Peter	Iqbal	Ben	PeterIqbal@gmail.com	Steet No.25	true	756056
78	78	Payal	Jatin	Raja	PayalJatin@gmail.com	Steet No.11	false	756100
79	79	Moti	Omar	Naruka	MotiOmar@gmail.com	Steet No.12	true	759021
80	80	Basanti	Siddharth	Gokhale	BasantiSiddharth@gmail.com	Steet No.20	true	767001
81	86	Ridham	Om	Modi	201901001@daiict.ac.in	Street No.1	false	383325

- Employee



Before insertion :



	e_id [PK] integer	e_firstname character varying	e_middlename character varying (20)	e_lastname character varying (20)	e_pin integer	e_password character varying (30)	e_email character varying (50)	e_role character varying (30)
72	72	Runjhun	Dhanush	Chopra	608901	RunjhunDhanush@4035	RunjhunDhanush@gmail.com	Tour Guide
73	73	Chhaya	Yash	Chhabra	686102	ChhayaYash@130	ChhayaYash@gmail.com	Tour Guide
74	74	Kim	Dhiraj	Balasubramanian	713101	KimDhiraj@3100	KimDhiraj@gmail.com	Management Staff
75	75	Bhairavi	Hemendra	Kumer	722101	BhairaviHemendra@7384	BhairaviHemendra@gmail.com	Booking Executive
76	76	Yasmin	Javed	Murty	754223	YasminJaved@2040	YasminJaved@gmail.com	Tour Guide
77	77	Rohini	Manpreet	Mogul	756056	RohiniManpreet@3481	RohiniManpreet@gmail.com	Tour Guide
78	78	Deep	Malik	Bhattacharyya	756100	DeepMalik@3136	DeepMalik@gmail.com	Tour Guide
79	79	Anees	Rupal	Shah	759021	AneesRupal@2081	AneesRupal@gmail.com	Travel Agent
80	80	Arpit	Daanish	Anthony	767001	ArpitDaanish@518	ArpitDaanish@gmail.com	Booking Executive

After insertion :

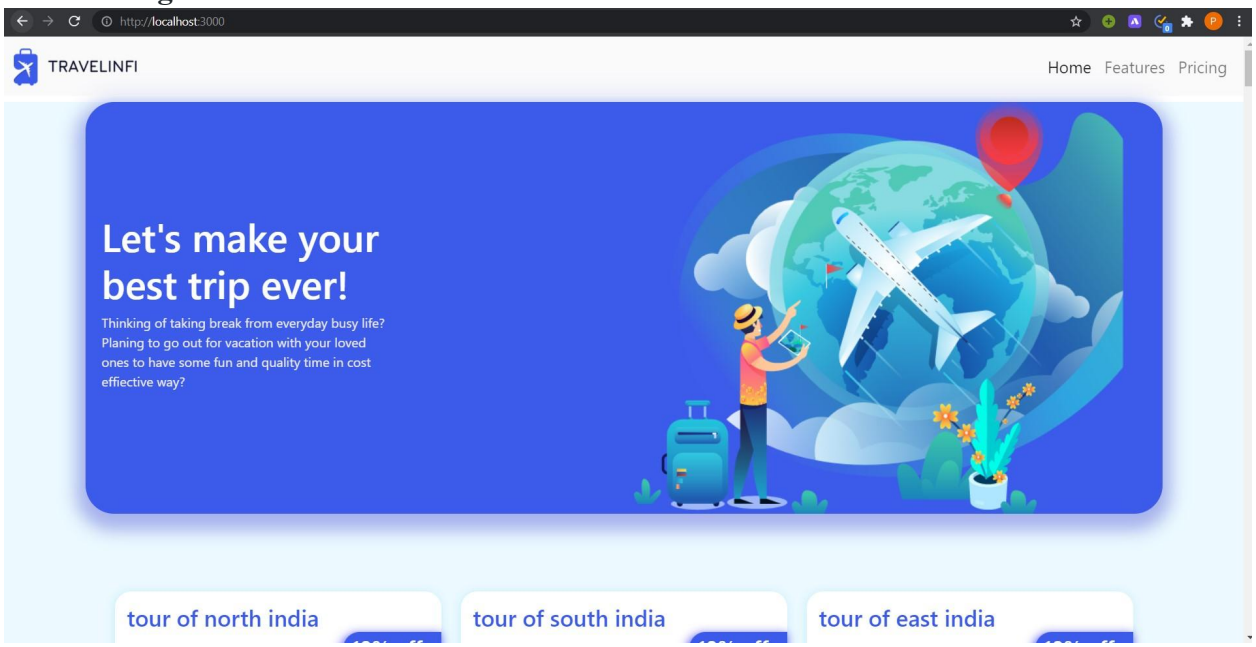
Query Editor | Query History

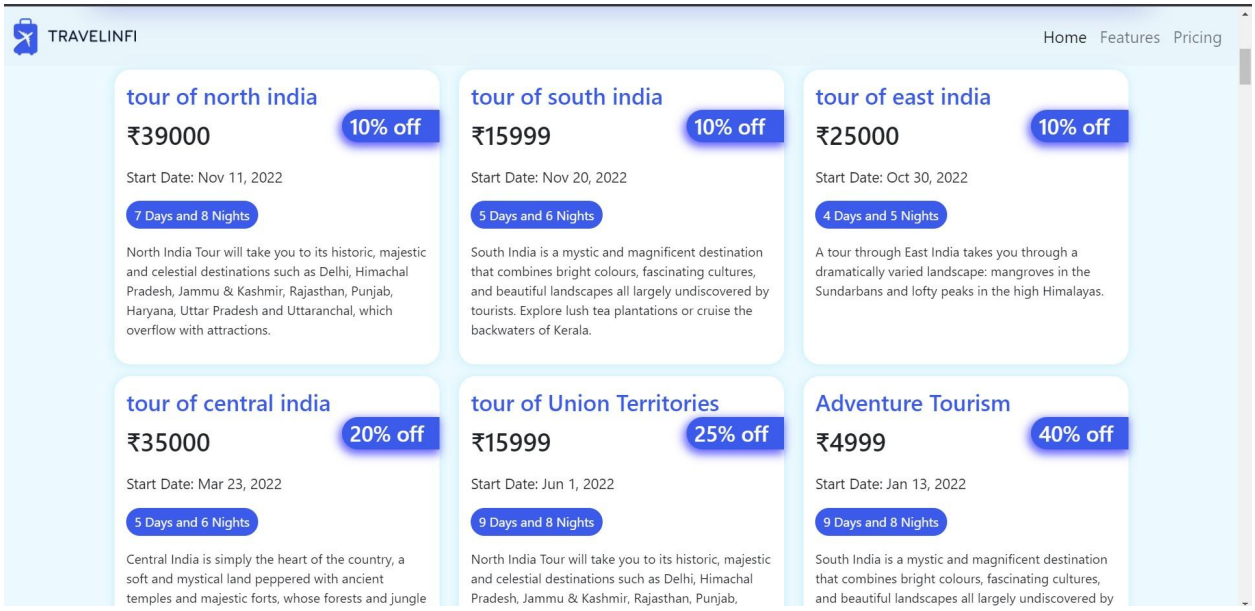
```
1  SELECT * FROM tms_db.employee
2  ORDER BY e_id ASC
```

Data Output | Explain | Messages | Notifications

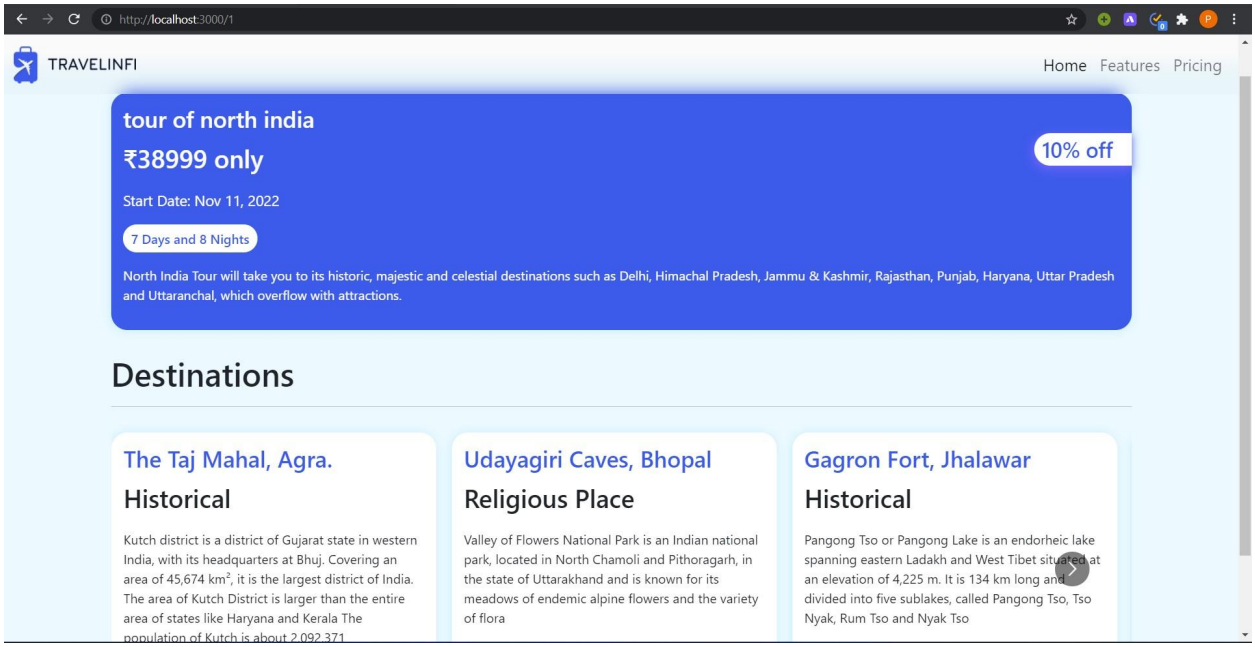
e_id [PK] integer	e_firstname character varying	e_middlename character varying (20)	e_lastname character varying (20)	e_pin integer	e_password character varying (30)	e_email character varying (50)	e_role character varying (30)
73	Chhaya	Yash	Chhabra	686102	ChhayaYash@130	ChhayaYash@gmail.com	Tour Guide
74	Kim	Dhiraj	Balasubramanian	713101	KimDhiraj@3100	KimDhiraj@gmail.com	Management Staff
75	Bhairavi	Hemendra	Kumer	722101	BhairaviHemendra@7384	BhairaviHemendra@gmail.com	Booking Executive
76	Yasmin	Javed	Murty	754223	YasminJaved@2040	YasminJaved@gmail.com	Tour Guide
77	Rohini	Manpreet	Mogul	756056	RohiniManpreet@3481	RohiniManpreet@gmail.com	Tour Guide
78	Deep	Malik	Bhattacharyya	756100	DeepMalik@3136	DeepMalik@gmail.com	Tour Guide
79	Anees	Rupal	Shah	759021	AneesRupal@2081	AneesRupal@gmail.com	Travel Agent
80	Arpit	Daanish	Anthony	767001	ArpitDaanish@518	ArpitDaanish@gmail.com	Booking Executive
81	Ridham	Manis	Sonani	383325	user	201901001@daiict.ac.in	CEO

- **Tour Package**

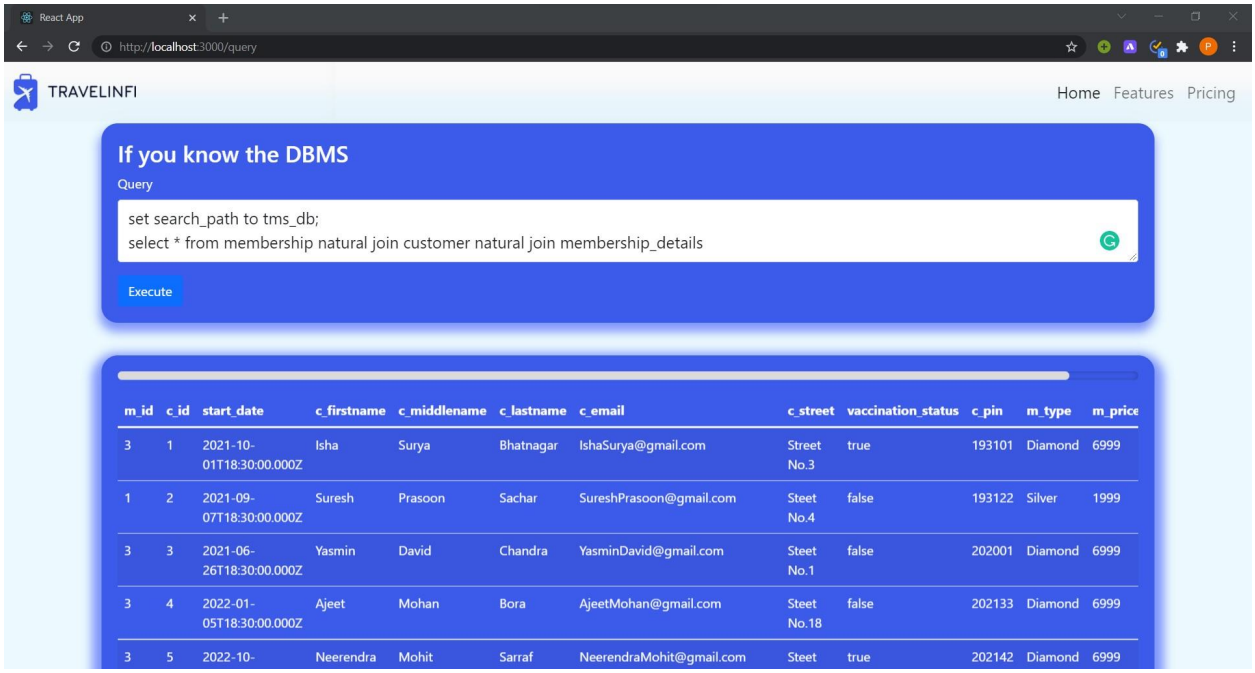




- **Tour Details**



- **QueryTool**



Code

- We have not added all the code because it is too long, so we only added some parts of the code.

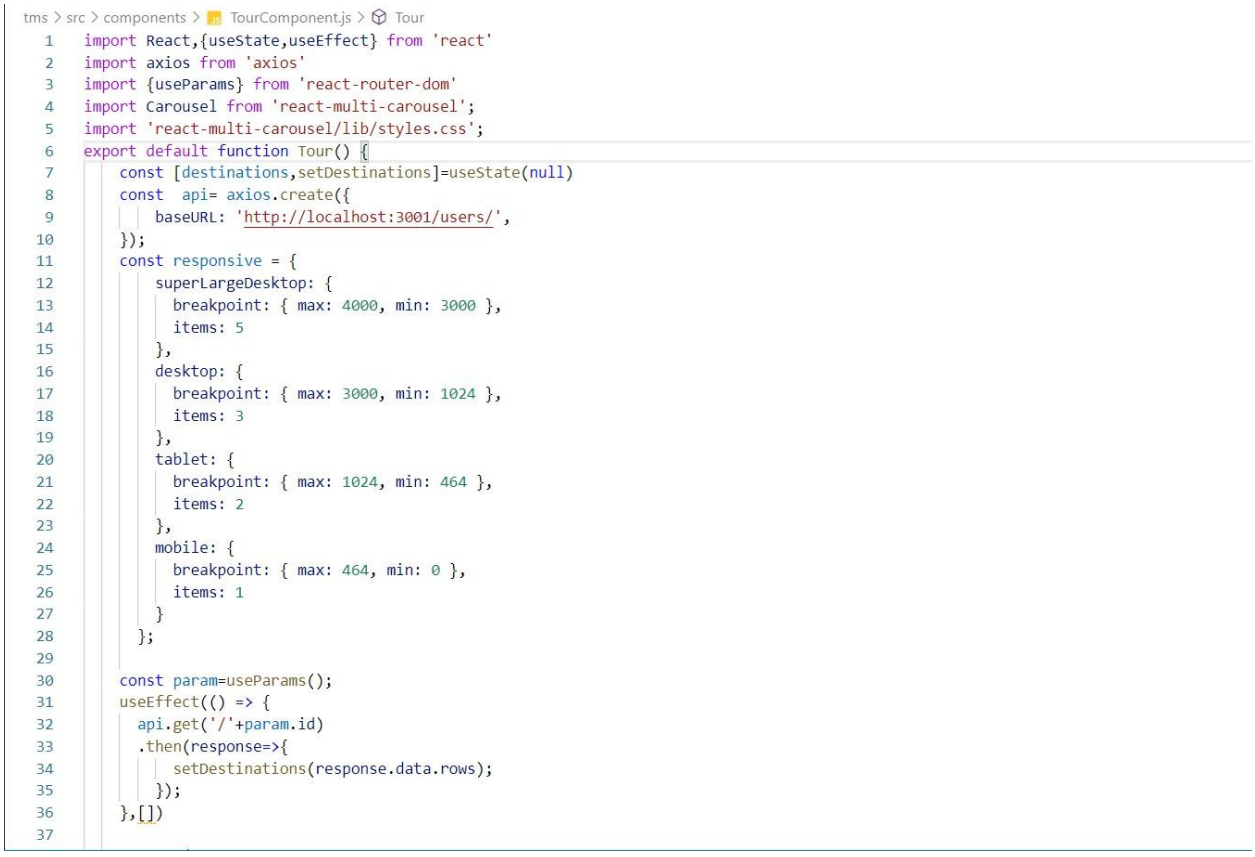
Backend in node.js

```
var express = require('express');
var router = express.Router();
const { Pool, Client } = require('pg')
const pool = new Pool({
  user: 'postgres',
  host: 'localhost',
  database: '201901037_db',
  password: 'admin',
  port: 5432,
})

router.get('/', function(req, res, next) {

  pool.query('SELECT * from tms_db.tour_package  natural left join tms_db.discount ', (err, ans) => {
    res.send(ans);
```

```
  })

});

router.get('/:id', function(req, res, next) {

    pool.query('SELECT * from tms_db.tour_destination as a  natural join  tms_db.destination natural join tms_db.tour_package natural join  tms_db.discount where a.t_id='+req.params.id, (err, ans) => {
   res.send(ans);
  })

});

router.post('/customer', function(req, res, next) {
                                                                                                                                                const value=[req.body.fname,req.body.mname,req.body.lname,req.body.email,req.body.street,req.body.safe,req.body.c_pin,req.body.city,req.body.c_state];
  pool.query('SELECT tms_db.insert_customer($1,$2,$3,$4,$5,$6,$7,$8,$9)',value, (err, ans) => {
    console.log(err, ans)
    res.send(ans);
  })

});


router.post('/employee', function(req, res, next) {
                                                                                                                                                const value=[req.body.fname,req.body.mname,req.body.lname,req.body.pin,req.body.password,req.body.email,req.body.role,req.body.city,req.body.state];
  pool.query('SELECT tms_db.insert_employee($1,$2,$3,$4,$5,$6,$7,$8,$9)',value, (err, ans) => {
    console.log(err, ans)
    res.send(ans);
  })

});

router.post('/query', function(req, res, next) {
  pool.query(req.body.query, (err, ans) => {
    console.log(err, ans)
    if(err!== undefined)
    {
```

```
    console.log(err.error);
    res.send(err);
  }
  else
  {
    if(Array.isArray(ans)==false)
      return res.send(ans);
    else
      res.send(ans[ans.length-1]);
  }

 })

});

module.exports = router;
```

- **Tour Component**



```
tms > src > components > TourComponent.js > Tour
import React,{useState,useEffect} from 'react'
import axios from 'axios'
import {useParams} from 'react-router-dom'
import Carousel from 'react-multi-carousel';
import 'react-multi-carousel/lib/styles.css';
export default function Tour() {
    const [destinations,setDestinations]=useState(null)
    const  api= axios.create({
        baseURL: 'http://localhost:3001/users/',
    });
    const responsive = {
        superLargeDesktop: {
          breakpoint: { max: 4000, min: 3000 },
          items: 5
        },
        desktop: {
          breakpoint: { max: 3000, min: 1024 },
          items: 3
        },
        tablet: {
          breakpoint: { max: 1024, min: 464 },
          items: 2
        },
        mobile: {
          breakpoint: { max: 464, min: 0 },
          items: 1
        }
      };

    const param=useParams();
    useEffect(() => {
      api.get('/'+param.id)
      .then(response=>{
          setDestinations(response.data.rows);
          });
    },[])

```

```
return (
    <div style={{padding:"20px"}}>
        <div>
            {destinations!=null &&
                <div className="tp_card" style={{background:"#3d5bea",color:"white",width:"100%"}}>
                    <h3 className="tp_name" style={{color:"white"}}>{destinations[0].t_name}</h3>
                    <p className="price">₹{destinations[0].t_price-1} only </p>
                    <p className="date">{"Start Date: "+new Date(destinations[0].t_start_date).toLocaleString('en-us',{day:"numeric",month:'short', year:'numeric'})}<
                    <p className="tp_duration" style={{background:"white",color:"#3d5bea",fontWeight:600}}>{destinations[0].t_duration}</p>
                    <p style={{"text-align": "initial"}}>{destinations[0].t_desc}</p>
                    {destinations[0].dis_id!=null && <div className="discount" style={{background:"white",color:"#3d5bea"}}>{ destinations[0].percentage}% off</div>}
                </div>
            }
        </div>
        <h1 style={{"text-align": "left",marginTop:"30px"}}>Destinations</h1>
        <hr></hr>
        <Carousel responsive={responsive} >
            {destinations!=null &&
                destinations.map((tp)=>{
                    return(
                        <div className="tp_card" style={{height:"350px"}}>
                            <h3 className="tp_name">{tp.d_name}</h3>
                            <p className="price">{tp.d_type}</p>
                            <p style={{"text-align": "initial"}}>{tp.d_desc}</p>
                        </div>
                    )
                })
            }
        </Carousel>
    </div>
)
}
```

- **QueryComponent**

tms > src > components > QueryComponent.js > Query > submit > then() callback

```
import React from 'react'
import axios from 'axios'

function Query() {
    const [query,setQuery]=React.useState('');
    const [ans,setans]=React.useState(null);
    const [err,seterr]=React.useState(null);
    const [header,setHeader]=React.useState(null);

    const  api= axios.create({
        baseURL: 'http://localhost:3001/users/query',
    });

    const submit=(e)=>{
        e.preventDefault();
        api.post('/',{
            query:query,
        })
        .then(response=>{
            console.log('ans'+response.data)
            setans(response.data.rows)
            setHeader(response.data.fields)
        });
    }
```

```
}
return (
    <div style={{paddingTop:"20px"}}>
        <div
            style={{
            marginTop:"20px",
            "padding": "20px",
            "border-radius": "20px",
            "background": "#3d5bea",
            color:"white",
            "box-shadow": "-1px 9px 12px 2px #8096fd"}}>
            <h3>If you know the DBMS</h3>
            <div class="mb-3" >
                <label for="exampleFormControlInput1" class="form-label">Query</label>
                <textarea type="text" class="form-control" id="exampleFormControlInput1" placeholder="select * from "
                    label="query"
                    value={query}
                    onChange={(e)=>setQuery(e.target.value)}
                    style={{"font-size": "20px",
                        "height": "50px",
                        "background": "white"}}
                />
            </div>
            <div class="col-12">
                <button type="submit" class="btn btn-primary" onClick={(e)=>submit(e)}>Execute</button>
            </div>
        </div>
        {
            ans!=null &&
            <div
                style={{
                marginTop:"40px",
                "padding": "20px",
                "border-radius": "20px",
                "background": "#3d5bea",
                "box-shadow": "rgb(128 150 253) 0px 9px 12px 12px"}}>
```

```
            <div
                style={{
                    transform: "rotateX(180deg)",
                    "overflow-x": "scroll",
                }}>
                <table class="table table-hover table-striped"
                    style={{marginTop:"20px",color:"white",
                    transform: "rotateX(180deg)",
                }}
                >
                    <thead>
                        <tr>
                            {
                                header!=null &&
                                header.map((h)=>{
                                    return(
                                        <th>{Object.values(h)[0]}</th>
                                    )
                                })
                            }
                        </tr>
                    </thead>
                    <tbody>
                    {
                        ans!=null &&
                        ans.map((ans)=>{
                            return(
                                <tr>
                                {
                                    Object.values(ans).map((q)=>{
                                        return(
                                            <td style={{color:"white"}}>{String(q)} </td>
                                        )
                                    })
                                }
                                </tr>
                            )
```

```
                                    })
                                }
                                </tbody>
                            </table>
                        </div>
                    </div>
                }
            </div>
        )
}

export default Query

```

- **EmployeeComponent**

tms > src > components > EmployeeComponent.js > Employee

```
import React, { useState } from 'react'
import './one.css'
import traveling from '../travel.png'
import axios from 'axios'

function Employee() {
    const [fname,setFname]=useState('')
    const [lname,setLname]=useState('')
    const [mname,setMname]=useState('')
    const [email,setEmail]=useState('')
    const [password,setPassword]=useState('')
    const [city,setCity]=useState('')
    const [pin,setPin]=useState('')
    const [state,setState]=useState('')
    const [role,setRole]=useState('')
    const  api= axios.create({
        baseURL: 'http://localhost:3001/users/employee',
    });

    const submit=(e)=>{
        e.preventDefault();
        api.post('/',{
            fname:fname,
            lname:lname,
            mname:mname,
            email:email,
            password:password,
            city:city,
            pin:pin,
            state:state,
            role:role,
        })
        .then(response=>{

          });
    }
```

```
return (
    <div className="c_container">
        <div className="row c_form" >
            <div className="col-6">
                <img src={traveling} alt="traveling" style={{width:"550px"}}/>
            </div>
            <div className="col-6">
                <form class="row g-3 " onSubmit={(e) => submit(e)}>
                    <div class="col-md-4">
                        <label for="fname" class="form-label">First Name</label>
                        <input type="text" class="form-control" id="fname"
                            label="fname"
                            value={fname}
                            onChange={(e)=>setFname(e.target.value)}
                        />
                    </div>
                    <div class="col-md-4">
                        <label for="inputfirstd4" class="form-label">Middle Name</label>
                        <input type="text" class="form-control" id="inputfirstd4"
                            label="mname"
                            value={mname}
                            onChange={(e)=>setMname(e.target.value)}
                        />
                    </div>
                    <div class="col-md-4">
                        <label for="inputlast4" class="form-label">Last Name</label>
                        <input type="text" class="form-control" id="inputlast4"
                            label="lname"
                            value={lname}
                            onChange={(e)=>setLname(e.target.value)}
                        />
                    </div>
                    <div class="col-12">
                        <label for="inputemail" class="form-label">Email</label>
                        <input type="email" class="form-control" id="inputAddress"
                            label="email"
                            value={email}
                            onChange={(e)=>setEmail(e.target.value)}
                        />
                    </div>
                    <div class="col-12">
                        <label for="inputemail" class="form-label">Password</label>
                        <input type="password" class="form-control" id="inputAddress"
                            label="password"
                            value={password}
                            onChange={(e)=>setPassword(e.target.value)}
                        />
                    </div>
                    <div class="col-md-5">
                        <label for="inputCity" class="form-label">City</label>
                        <input type="text" class="form-control" id="inputCity"
                            label="city"
                            value={city}
                            onChange={(e)=>setCity(e.target.value)}
                        />
                    </div>
                    <div class="col-md-4">
                        <label for="inputState" class="form-label">State</label>
                        <input type="text" class="form-control" id="inputState"
                            label="state"
                            value={state}
                            onChange={(e)=>setState(e.target.value)}
                        />
                    </div>
                    <div class="col-md-3">
                        <label for="inputZip" class="form-label">Pin</label>
                        <input type="number" class="form-control" id="inputZip"
                            label="pin"
                            value={pin}
                            onChange={(e)=>setPin(e.target.value)}
                        />
                    </div>
                    <div class="col-12">
                        <label for="inputZip" class="form-label">Role</label>
```

```
                    <div class="col-12">
                        <label for="inputZip" class="form-label">Role</label>
                        <input type="text" class="form-control" id="inputZip"
                            label="role"
                            value={role}
                            onChange={(e)=>setRole(e.target.value)}
                        />
                    </div>

                    <div class="col-12">
                        <button type="submit" class="btn btn-primary">Sign in</button>
                    </div>
                </form>
            </div>

        </div>

    </div>
  )
}

export default Employee
```

- **Tour Packages Component**

```
import React,{useState,useEffect} from 'react'
import axios from 'axios'
import {useParams} from 'react-router-dom'
import Carousel from 'react-multi-carousel';
import 'react-multi-carousel/lib/styles.css';
export default function Tour() {
    const [destinations,setDestinations]=useState(null)
    const  api= axios.create({
        baseURL: 'http://localhost:3001/users/',
    });
    const responsive = {
        superLargeDesktop: {
          breakpoint: { max: 4000, min: 3000 },
          items: 5
        },
        desktop: {
          breakpoint: { max: 3000, min: 1024 },
          items: 3
        },
        tablet: {
          breakpoint: { max: 1024, min: 464 },
          items: 2
        },
        mobile: {
          breakpoint: { max: 464, min: 0 },
          items: 1
        }
      };

    const param=useParams();
    useEffect(() => {
      api.get('/'+param.id)
      .then(response=>{
          setDestinations(response.data.rows);
        });
    },[])

```

```
    return (
        <div style={{padding:"20px"}}>
            <div>
                {destinations!=null &&
                    <div className="tp_card" style={{background:"#3d5bea",color:"white",width:"100%"}}>
                        <h3 className="tp_name" style={{color:"white"}}>{destinations[0].t_name}</h3>
                        <p className="price">₹{destinations[0].t_price-1} only </p>
                        <p className="date">{"Start Date: "+new Date(destinations[0].t_start_date).toLocaleString('en-us',{day:"numeric",month:'short', year:'numeric'})}<
                        <p className="tp_duration" style={{background:"white",color:"#3d5bea",fontWeight:600}}>{destinations[0].t_duration}</p>
                        <p style={{"text-align": "initial"}}>{destinations[0].t_desc}</p>
                        {destinations[0].dis_id!=null && <div className="discount" style={{background:"white",color:"#3d5bea"}}>{ destinations[0].percentage}% off</div>}
                    </div>
                }
            </div>
            <h1 style={{"text-align": "left",marginTop:"30px"}}>Destinations</h1>
            <hr></hr>
            <Carousel responsive={responsive} >
                {destinations!=null &&
                    destinations.map((tp)=>{
                        return(
                            <div className="tp_card" style={{height:"350px"}}>
                                <h3 className="tp_name">{tp.d_name}</h3>
                                <p className="price">{tp.d_type}</p>
                                <p style={{"text-align": "initial"}}>{tp.d_desc}</p>
                            </div>
                        )
                    })
                }
            </Carousel>
        </div>
    )
}

```

- **CustomerComponent**

```
tms > src > components > CustomerComponent.js > Customer > api
import React, { useState } from 'react'
import './one.css'
import traveling from '../travel.png'
import axios from 'axios'


function Customer() {
    const [fname,setFname]=useState('')
    const [lname,setLname]=useState('')
    const [mname,setMname]=useState('')
    const [email,setEmail]=useState('')
    const [street,setstreet]=useState('')
    const [city,setCity]=useState('')
    const [pin,setPin]=useState('')
    const [state,setState]=useState('')
    const [safe,setSafe]=useState(false)
    const  api= axios.create({
        baseURL: 'http://localhost:3001/users/customer',
    });

    const submit=(e)=>{
        e.preventDefault();
        api.post('/',{
            fname:fname,
            lname:lname,
            mname:mname,
            email:email,
            street:street,
            city:city,
            c_pin:pin,
            c_state:state,
            safe:safe,
        })
        .then(response=>{

          });
    }
```

```
    return (
        <div className="c_container">
            <div className="row c_form" >
                <div className="col-6">
                    <img src={traveling} alt="traveling" style={{width:"550px"}}/>
                </div>
                <div className="col-6">
                    <form class="row g-3 " onSubmit={(e) => submit(e)}>
                        <div class="col-md-4">
                            <label for="fname" class="form-label">First Name</label>
                            <input type="text" class="form-control" id="fname"
                                label="fname"
                                value={fname}
                                onChange={(e)=>setFname(e.target.value)}
                            />
                        </div>
                        <div class="col-md-4">
                            <label for="inputfirstd4" class="form-label">Middle Name</label>
                            <input type="text" class="form-control" id="inputfirstd4"
                                label="mname"
                                value={mname}
                                onChange={(e)=>setMname(e.target.value)}
                            />
                        </div>
                        <div class="col-md-4">
                            <label for="inputlast4" class="form-label">Last Name</label>
                            <input type="text" class="form-control" id="inputlast4"
                                label="lname"
                                value={lname}
                                onChange={(e)=>setLname(e.target.value)}
                            />
                        </div>
                        <div class="col-12">
                            <label for="inputemail" class="form-label">Email</label>
                            <input type="email" class="form-control" id="inputAddress"
                                label="email"
                                value={email}
                                onChange={(e)=>setEmail(e.target.value)}
                            />
                        </div>
                        <div class="col-12">
                            <label for="inputAddress" class="form-label">Street Address</label>
                            <input type="text" class="form-control" id="inputAddress"
                                label="street"
                                value={street}
                                onChange={(e)=>setstreet(e.target.value)}
                            />
                        </div>
                        <div class="col-md-5">
                            <label for="inputCity" class="form-label">City</label>
                            <input type="text" class="form-control" id="inputCity"
                                label="city"
                                value={city}
                                onChange={(e)=>setCity(e.target.value)}
                            />
                        </div>
                        <div class="col-md-4">
                            <label for="inputState" class="form-label">State</label>
                            <input type="text" class="form-control" id="inputState"
                                label="state"
                                value={state}
                                onChange={(e)=>setState(e.target.value)}
                            />
                        </div>
                        <div class="col-md-3">
                            <label for="inputZip" class="form-label">Pin</label>
                            <input type="number" class="form-control" id="inputZip"
                                label="pin"
                                value={pin}
                                onChange={(e)=>setPin(e.target.value)}
                            />
                        </div>
```

```
                    <div class="col-12">
                        <div class="form-check">
                        <input class="form-check-input" type="checkbox" id="gridCheck"
                            value={safe}
                            label="safe"
                            onChange={(e)=>setSafe(e.target.value)}
                        />
                        <label class="form-check-label" for="gridCheck">
                            Fully Vaccinated
                        </label>
                        </div>
                    </div>
                    <div class="col-12">
                        <button type="submit" class="btn btn-primary">Sign in</button>
                    </div>
                </form>
            </div>

        </div>

    </div>
  )
}

export default Customer

```