



Dhirubhai Ambani
Institute of Information and Communication Technology

Tourist Agencies Management Database

Team ID: S4_T4

**IT214: Database Management System
Assign By Prof. Minal Bhise**

Mentor TA: Vaishnavi

Prepared By:

201901075 - Shreyas Sakariya
201901102 - Ronak Parmar
201901108 - Ankit Rathod
201901116 - Kishan Gajnotar

Table of contents

Section1: The final version of SRS	5
1.1 Introduction	6
1.2 Document the Requirements Collection/ Fact-Finding Phase	8
1.3 Interview/s –	9
1.4 Questionnaire/s	11
1.5 Observation	15
1.6 Fact-Finding Chart	15
1.7 List Requirements	16
1.8 User Classes and Characteristics	17
1.9 Operating Environment	18
1.10 Product Functions	18
1.11 Privileges	19
1.12 Assumptions	20
1.13 Business Constraints	20
Section 2: Noun Analysis	21
2.1 Final Problem Description	22
2.2 Noun (& Verb) Analysis	24
2.3 Accepted Noun & Verbs list	29
2.4 Rejected Noun & Verbs list	30
Section3: ER-Diagrams all versions.	32
3.1 ER Diagram(ERD) Version 1	33
3.2 ER Diagram(ERD) Version 2	34
3.3 Final ER diagram	35
Section4: Conversion of Final ER-Diagram to Relational Model	36
4.1 Mapping E-R Model to Relational Model	36
4.2 Relational Schema	37
Section5: Normalization and Schema Refinement	39
5.1 Original Database	40
5.2 Functional Dependencies:	40
5.3 Redundancies:	43
5.4 Update, insert and delete anomalies:	43
5.5 Normalization to 1NF:	44
5.6 Normalization to 2NF:	44
5.7 Normalization to 3NF:	44

5.8 Normalization to BCNF:	45
5.9 Final relations with the schema	45
Section6 : SQL: Final DDL Scripts, Insert statements, 40 SQL Queries with Snapshots of the output of each query	46
6.1 Final DDL Scripts	47
6.2 Creation of Tables	54
6.3 Insert statements	54
6.4 40 SQL Queries with Snapshots of the output of each query.	63
Section7: Project Code with output screenshots	102
7.1 Video Link:	103
7.2 Codes File Link:	103
7.3 Screenshot of output or web pages with data.	105

Section1: The final version of SRS

1.1 Introduction

Purpose

In today's world, every person must need a break from their boring lives to spend some time traveling with family and friends. In the present system, a customer has to approach various agencies to find details of places and to book tickets which requires a lot of time and effort. But sometimes they didn't get their requirements trip, satisfactory hotels, transport facilities, and destination brochure. Thus we create a project on the Tourist agencies Management Database which helps agencies and provide a better experience to travelers and make their journey satisfactory.

The purpose of the project is to design a system that automates the processes and activities of a travel and tourism agency and also to design a system in which travelers can get all information about traveler agencies and all packages on his requirement. Travelers no need to visit every agency they will get all information from their homes. This can be utilized for both business trips and leisure trips.

Intended Audience and Reading Suggestions

This computer program is additionally valuable for the clients specifically getting to the website.

The intended audience is,

- Travel agencies and travelers
- Project managers
- Users
- Testers
- Interested developers

Product Scope

The Tourism Management System project can prove to be immensely useful for travelers and travel agents with management facilities. According to the statistics provided by the ministry of tourism, there are lots of domestic travelers agencies in our country but they have no software that can provide all information regarding their package to travelers. So by using this software program many travelers who need a comfortable and affordable trip can find their preferred option according to their budget and interest. Also, they can provide relevant information about places situations like they are safe to visit in the covid situation, how is the weather of that place, how many people are going to visit their, etc.

Description

We are designing software that keeps track of various travel places, service availability, and traveling expenses and devising a comprehensive travel strategy so, it will help give travelers the best package according to their interests and budget.

This covid pandemic has impacted lots in our life since last 2 years. Most people can't go outside from their homes due to this pandemic. But now the covid cases are decreasing and new cases are not coming. so the people started to travel outside. In fact, they started traveling all over India. So Tourist agencies Management Database comes into the

picture where people can find their travel package according to their requirement and make a trip with their family or friends.

The Tourist agencies Management Database will help all travelers to find different packages and book for the same if they're interested. and it also helps the agencies to find all data about travelers. According to the current situation, our management system includes a covid situation at that place so that people can be aware of the covid conditions before visiting that place.

Workflow

In the old manual system, all the records are not kept perfectly because all the work is done on paper, so keeping up-to-date details of the trip, timings of bookings, seat availability for reservation, vehicles, or rooms/hotels availability is not done. Which is the time taking system and needs a lot of manual calculation to keep it update which also can lead to a mistake in calculation.

In this project we create a system that provides information regarding tourist agencies and their package. first, we store all data of our agencies and all about packages in our database then we create a one google form and then give it to the user. He will fill that form and from that, according to his interest and preference we will give him all data, packages and all other information of our agencies. That is the way our system works.

Design and Implementation Constraints

The application should be portable, safe, and can not be accessible by unauthorized users. The application should be well functionality and the application interface should not be complex and it's easy to understand for every people. Time is the very essential point of designing the application. Response time of application should be very less and it should not give any error during service. It has a large amount capacity to store users' data. Also, the application should work on every operating system. The application should be well maintained or up to date.

1.2 Document the Requirements Collection/ Fact-Finding Phase

Background Readings

So we went through the internet finding articles related to the Tourist management and found some issues in which people are suffering in finding a safe place, good hotels in the current covid situation and management also need to lot's of work to maintain all this stuff.

Manual Paper Work

In the old system, all the records are not kept perfectly because all the work is done manually, so keeping up-to-date details of the trip, timings of bookings, seat availability for reservation, vehicles, or rooms/hotels availability is not done. Which is a time taking system and needs a lot of manual calculation to keep it updated which also can lead to a mistake in calculation.

Software implemented system

The solution to the manual paperwork problem is to use a software-implemented system that is more efficient than the manual system. This system is highly automated and makes the traveling agencies' work much easier and flexible. The system doesn't need to do all base tasks manually, such as the forms transactions, and reports which are added advantages. This system is a completely computer-based application that uses concepts of DBMS. Thousands of records can be searched and displayed without taking any significant time and users can use it very easily which leads to the increased trust of the customer.

References

- <https://projectsgeek.com/2016/01/tourism-management-system.html>
- <https://www.gujarattourism.com/>
- <https://ijcsmc.com/docs/papers/October2019/V8I10201903.pdf>
- <https://www.touristlink.com/>
- <https://sites.google.com/site/ignoubcaffinalyearprojects/project-report/tours-and-travel-management-system-project-report>

Requirements

FUNCTIONAL REQUIREMENTS:

1. Administrator Module: This module provides administrator-related functionality. The administrator manages all information about travelers and has all rights to add, delete, edit and view the data related to travelers, places, routes, restaurants etc.

2. Travels Module: This module provides the details of various packages based on information provided by travelers. A user can select the appropriate packages depending on convenience and accessibility.

3. Routes Module: This module provides information related to various routes connecting sources and destinations. For each route, information such as source, destination, fare, reservation details, pick-up points, etc is provided. Only administrators can add, delete, edit and manage the data. Users can only view the information.

4. Reservation Module: This module provides functionalities that allow a user to book tickets for train, plane, and bus. This module also provides cancellation of previously booked tickets. The module maintains the details of all reservations made so far and allows administrators to either confirm or reject the bookings.

5. Testimonials Module: Users of this application can post their opinions, complaints, and suggestions regarding this portal and services to the administrator. Accordingly, the administrator can take various steps to act on the complaints and suggestions.

1.3 Interview/s –

Interview 1

System: Citizen

Project Reference: SF/SJ/2003/12

Interviewee: 1)Rajesh Gajnotar

Designation: Govt.Teacher

Interviewer: 1) Ankit Rathod
2) Ronak Parmar
3) Shreyas Sakariya
4) Kishan Gajnotar

Designation: Student AT DAIICT
Designation: Student AT DAIICT
Designation: Student AT DAIICT
Designation: Student AT DAIICT

Date: 8/10/2021

Time: 21:00

Duration: 1 hour 25 minutes

Place: Virtual

Purpose of Interview:

- Gathering information about his travel experience and his interest. Also, we have discussed his future plan and places that he will be going to travel in the covid 19 situation.
- Get some idea about how we can arrange some trip(school trip) that is very beneficial to our system and users.
- How has he experienced other traveling websites and what is the suggestion for our current system?

Agenda:

- List all the requirements regarding safety concerns in this covid19 situation
- Also, list all requirements about his trip that he wants.
- Classify them according to the requirements

Interview 2

System: Citizen

Project Reference: SF/SJ/2003/12

Interviewee: 1) Jenish Rathod Designation: IT engineer

Interviewer:

1) Shreyas Sakariya	Designation: Student AT DAIICT
2) Kishan Gajnotar	Designation: Student AT DAIICT

Date: 5/10/2021 Time: 17:30

Duration: 1 hour 35 minutes Place: Virtual

Purpose of Interview:

- Gathering information about his travel experience and his interest. Also, we have discussed his future plan and places that he will be going to travel.

Agenda:

- List all the information
- Make a list of places that people generally prefer to visit.
- Classify them according to the requirements

Documents to be brought to the interview:

- A website that indicates current information of different Places.

Interview 3

System: Citizen

Project Reference: SF/SJ/2003/12

Interviewee: 1)Hiten Padaliya Designation: Student

Interviewer: 1) Ankit Rathod Designation: Student AT DAIICT
2) Ronak Parmar Designation: Student AT DAIICT

Date: 6/10/2021 Time: 17:30

Duration: 1 hour 30 minutes Place: Virtual

Purpose of Interview:

- Gathering information about his travel experience and his interest. Also, we have discussed his future plan and places that he will be going to travel in the covid 19 situation.

Agenda:

- List all the requirements regarding safety concerns in this covid19 situation
- List all the information
- Classify them according to the requirements

Documents to be brought to the interview:

- A website that indicates current information of different Places.

Combined Requirements gathered from Response/s.

We have gathered all the information from interviews we noted the following Requirements

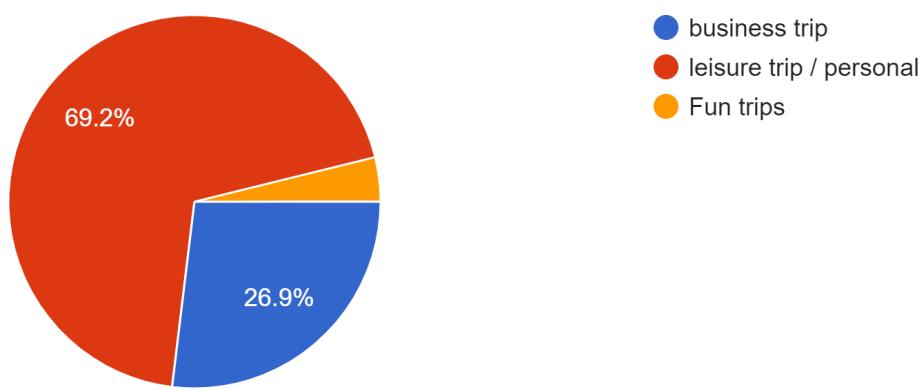
- Transportation services
- Medical emergency
- Reliable package
- Covid precaution
- Guide services
- Photographer
- Good restaurants

Also, we need to provide a good security policy so that the user data will be safe and users can easily trust our system.

1.4 Questionnaire/s

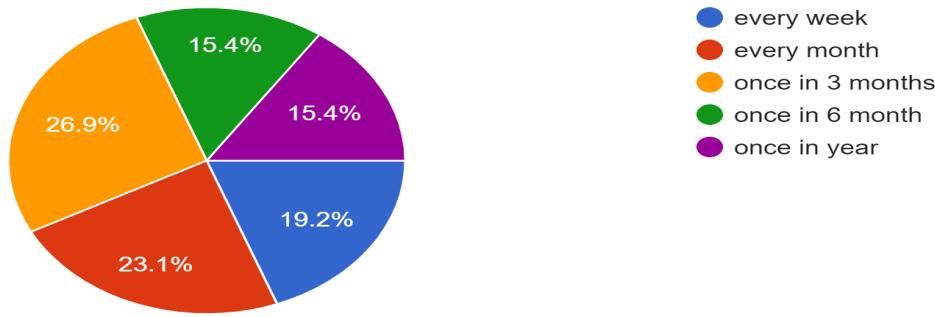
1) Which kind of trip do you have the most?

- A. business trip
- B. leisure trip / personal



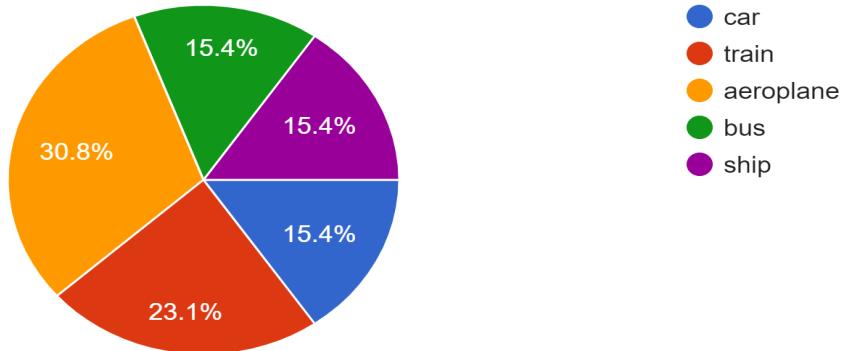
2) How Often do you travel to the outstation?

- A. every week
- B. every month
- C. once in 3 months
- D. once in 6 month
- E. once in year



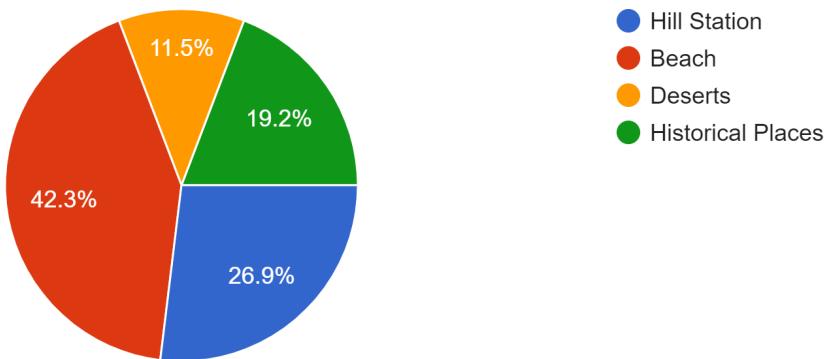
3) What mode of transportation do you prefer most of the time?

- A. Car
- B. Train
- C. Airplane
- D. Bus
- E. Ship



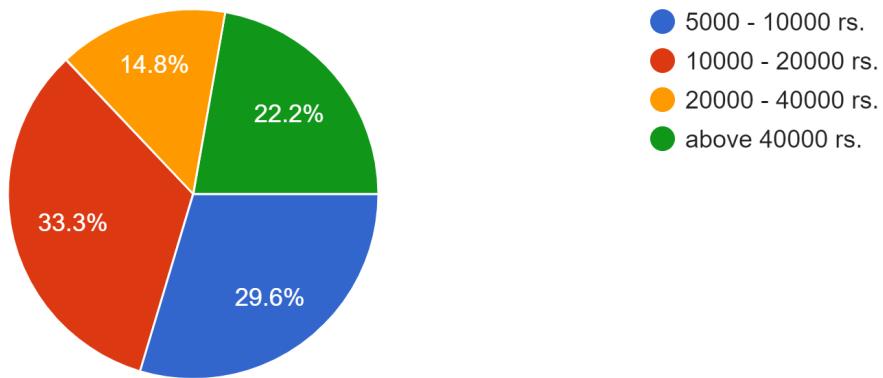
4) What kind of place do you prefer most?

- A. Hill Station
- B. Beach
- C. Deserts
- D. Historical Places



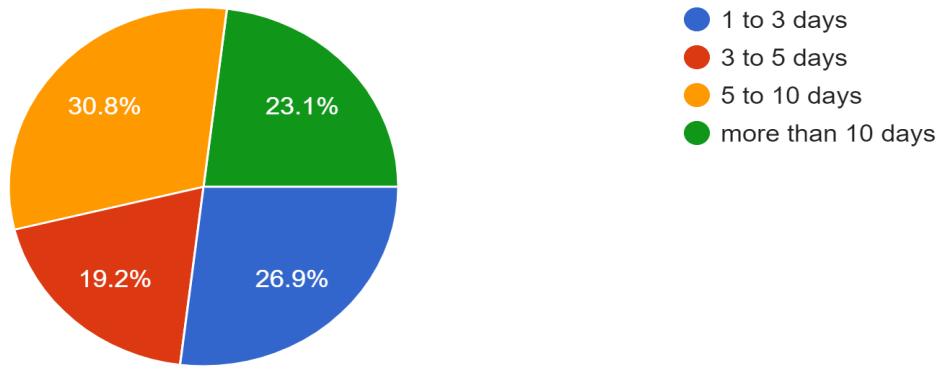
5)What is the package of your trip around?

- A. 5000 - 10000 rs.
- B. 10000 - 20000 rs.
- C. 20000 - 40000 rs.
- D. Above 40000 rs.



6)most common length of your trip

- A. 1 to 3 days
- B. 3 to 5 days
- C. 5 to 10 days
- D. more than 10 days



Requirements from the questionnaire

- In the covid-19 situation, people like to travel to a safe place which is given by the minister of government. So we have to focus on providing safe places.
- At this time people prefer airplanes to travel most so we have to ensure that we provide good service and best offer to them.
- Most people like to travel with family and friends so we provide the best offer on a family package which reduces travel costs.
- Most people like to travel at Hill station and Beach for peaceful traveling so we research more on this type of place and provide more places like this.
- Most people don't afford expensive traveling packages so we find more offers on train/bus traveling and include more low expenses packages with good services.
- Budget, destination weather, and covid-19 cases in that location are all significant factors in ensuring a safe and comfortable journey. As a result, our algorithm should filter these items and recommend the ideal trip.

1.5 Observation

- Observers can see the actual workflow during visits or personal experiences to add/suggest more requirements that might have been missed by other techniques like background reading, interviews, and questionnaires.
- We will give a good system which provides all functionalities of checking hotel details ,place description, transportation details and detailed information about package description with user understandable interface.
- We should ensure that the system gives the highest possible level of accuracy about hotel vacancy, place safety, and transportation details.
- In case of any emergency such as health deterioration, loss of any personal things, forgetting things to any place, etc. The emergency entity provides details such as Type, Contact of agency member, Address, etc.

- The system covers the details about the registration of users which can be registered by themselves by adding data like name, password, email id, and further details. After registration, they can sign in by their username and password.
- At some places the guidelines for the covid19 are not followed strictly, so we include all this type of detailed information in safety descriptions so at the tourist places no our customer suffers of any type of covid infection.
- Tickets can be issued for the user on the issue ticket page on the homepage of the user the certain booked packages only can be issued.
- The system is mainly based on admin. The system will check the admin username and password for authentication. After the verification for authorization, the admin can be able to precede the process. All works are done under his control.
- We will give a review section on our website where you share your experience with our agencies and list your suggestions and complain about our system so we look into our suggestion and make necessary steps on your complaint and provide you with the best experience .

1.6 Fact-Finding Chart

Objective	Technique	Subject(s)	Time Commitment
To get background on Tourist Agencies Management System	Background Reading	Articles, books, publications	2 day
Collect all ideas of their work and try to implement them in our system.	Interview	With Gov.Teacher / Social Worker	1 hour 25 min
Gathering information about his travel experience and his interest and Classify them according to the requirements	Interview	1 Citizen/IT engineer	1 hour 35 min
To establish additional requirements to our system	Interview	1 Citizen	1 hour 30 min
To know the opinion and interest of the people about traveling	Questionnaires/ Google forms	Shared with family and friends and others	2 days

1.7 List Requirements

- The administrator manages all information about travelers and has all rights to add, delete, edit and view the data related to travelers, places, routes, restaurants, etc.. the system will check the administrator's username and password for authentication so only the administrator has all rights.
- We will give a good system which provides all functionalities of checking hotel details ,place description, transportation details and detailed information about package description with user understandable interface.
- Systems need to provide a data security policy so that the user data will be safe and don't accessed by anyone. so users can easily trust our system.
- In case of any emergency such as health deterioration, loss of any personal things, forgetting things to any place, etc. Emergency entities provide details such as Type, Contact of agency member, Address, etc.
- In the covid-19 situation, people like to travel to a safe place which is given by the minister of government. So we have to focus on providing safe places.
- We should ensure that the system gives the highest possible level of accuracy about hotel vacancy, place safety, and transportation details.
- We have gathered all the information from interviews we noted the following Requirements
 - Transportation services
 - Covid precaution
 - Guide services
 - Photographer
 - Good restaurants
- The system provides the details of various packages based on information provided by travelers. A user can select the appropriate packages depending on convenience and accessibility.
- The system provides functionalities that allow a user to book tickets for trains, planes, and buses. This module also provides cancellation of previously booked tickets. The system maintains the details of all reservations made so far and allows administrators to either confirm or reject the bookings.
- The system provides information related to various routes connecting sources and destinations. For each route, information such as source, destination, fare, reservation details, pick-up points, etc is provided.
- Users of this application can post their opinions, complaints, and suggestions regarding this portal and services to the administrator. Accordingly, the administrator can take various steps to act on the complaints and suggestions.
- Performance Requirements
 - The database shall be able to accommodate a minimum of 10,000 records of customers.
 - The software shall support the use of multiple users at a time.

- Security Requirements
 - Some of the factors that are identified are to protect the software from accidental or malicious access.
 - Keep a specific log or history data sets
 - Check data integrity for critical variables
 - when the application crashed along with suggestions on how to prevent the error from occurring again.

1.8 User Classes and Characteristics

The system provides access to Visitors, Customers, Managers, and Administrators.

- **Visitor:** Users who are currently not logged in and have not registered to the website are visitors. Visitors can check the availability of flight reservations. In order to book the ticket, the visitor will need to log in with a valid username and password else register with the app and create an account.
- **Customer:** Customers can check the availability of flight reservations and also book tickets by making payments online. They can also view their booking details, payment details, etc.
- **Manager:** The person who manages the system, modifies/add, or delete the customer
- **Administrator:** The administrator can manage customer details, hotels, rooms, fight details, tour package details, etc. Also, the admin can view and print transaction reports, booking reports, cancellation reports, etc.

1.9 Operating Environment

Software Interfaces

- Operating System: Windows XP or Higher versions
- Front End :HTML, CSS and BOOTSTRAP
- Back End : Python Django with PostgreSQL

Hardware Interfaces

- Processor : Pentium IV
- Speed : 2.0 GHz above
- Hard Disk : 40 GB
- RAM : 512 MB
- CD Drive : 48x
- Input devices : Keyboard and mouse
- Monitor : Compatible monitor with 600 x 800 resolutions
- Internet : 100kbps above

External Interface Requirements

- The following sections will introduce the numerous requirements of the system from the point of view of different users and will introduce a number of decisions that have been made regarding implementation.
- It allows users to view quick reports.
- The user interface must be customizable by the administrator.
- All the modules provided with the software must fit into this graphical user interface and accomplish the standard defined.
- The user interface for this system will have to be simple and clear. Most importantly, the ages must be easy to read, easy to understand, and accessible.

1.10 Product Functions

- **Sightseeing Tours:** Tourists can enjoy sightseeing tours to any of the places listed in the Agencies Data File. Before that Customer has to make the booking by registering his/her name in the Data-File and informing the date of journey and No. of operations.
- **Flight ticket booking:** In this module, the customer can search source-destination, check flight timings, and book a ticket online.
- **Bus ticket reservation module:** In the bus ticket reservation module there has been a collection of buses, Customers can book the ticket by selecting the agency, date, and departure time of the bus.
- **Train ticket reservation module:** In the train reservation module customers can check train scheduled timings and they can book tickets online by entering Place, date, and time.
- **Hotel reservation:** After booking a train ticket, bus ticket, or flight ticket the system will provide the option to search hotels. Customers can reserve the hotels by selecting check-in date, check-out date, and hotel name.
- **Holiday package:** The customer can search tour packages by entering a destination, duration of travel, the month of travel, and the number of people.
- **Payment module:** This module allows customers to make payments after the reservation. The customer will receive a ticket receipt by mail after the payment.
- **Report Generation:** Details about the locations, hotels in that location, and final report on the journey fare.
- **My Account:** This module is for customers where customers can access the account module after login id and password. This module will display booked tickets, Billing receipts, ticket cancellation pages, logout, Etc.
- **Cancel booking:** Customer can cancel flight tickets, bus tickets, train tickets, or hotel bookings. The system refunds the paid amount after deducting cancellation charges.
- **Dashboard module:** This module allows the administrator of this site to manage the different aspects of ticket reservation-related activities. The administrator of the site can add or modify the records of flights, buses, trains, holiday packages, etc.

1.11 Privileges

Facilities accessible to Administrator

- Dashboard Module
- Insert, delete and update the record of any module
- Has access to the key database

Facilities accessible to the user

- Create an Account on site
- Flight ticket booking and cancellation
- Train ticket booking and cancellation
- Bus ticket booking and cancellation
- Holiday package selection
- Hotel reservation
- View generated report
- Payment method

1.12 Assumptions

- The user should operate on a computer or laptop along with an internet facility.
- Roles and responsibilities are already established.
- The administrator is already created.
- The coding should be error-free.
- The system should be user-friendly so it is easy to use for all users.
- The user must need to register himself first to use the system.
- The information of all users must be stored in a database that is accessible by the website.

1.13 Business Constraints

A business constraint is anything that interferes with the profitability of a company or business endeavor. Improving profitability requires the removal or reduction of business constraints. Common business constraints include time, financial concerns, and management.

Time Constraints

Time constraints include not only the amount of time required to complete a task but also the amount of time needed to obtain details about places, hotels, and transportation services (manage all traveling schedules). Once identified as a primary constraint, management can take steps to address time factors and improve business performance.

Financial Constraints

Financial factors are often limiting constraints for businesses. They can range from inadequate budget allocations to excessive salaries or overhead expenditures. for example: if more employees are needed, but the budget cannot accommodate additional salaries, growth is limited.

Management and Staffing

As businesses grow and change, their staffing and management need change, as well. This can constrain business growth and productivity when employees cannot adapt to new demands or when additional employees are needed but the capital to pay them is not yet available.

Section 2 : Noun Analysis

2.1 Final Problem Description

In this covid situation, most of the people are bored from staying home. But now the covid cases are decreasing and new cases are not coming. so the people started to travel outside. In fact, they started traveling all over India. So Tourist agencies Management Database comes into the picture where people can find their travel package according to their requirement and make a trip with their family or friends.

In the present system, a customer has to approach various agencies to find details of places and to book tickets which requires a lot of time and effort. But sometimes they didn't get their requirements for the trip like satisfactory hotels, transport facilities, and destination brochure. Thus we create a project on the Tourist agencies Management Database which helps agencies and provide a better experience to travelers and make their journey satisfactory.

In the old manual system, all the records are not kept perfectly because all the work is done on paper, so keeping up-to-date details of the trip, timings of bookings, seat availability for reservation, vehicles, or rooms/hotels availability is not done. Which is a time taking system and needs a lot of manual calculation to keep it updated which also can lead to a mistake in calculation. According to the statistics provided by the ministry of tourism, there are lots of travel agencies in our country but they don't have software that can provide us with all relevant information about tour packages to travelers.

That's why we created a Tourism Management System that can prove to be immense for travelers and travel agents with management facilities. So by using this software program many travelers who need a comfortable and affordable trip can find their preferred option according to their budget and interest. Also, they can provide relevant information about places that are safe to visit in the covid situation, how is the weather of that place, how many people are going to visit there, etc.

In this project, we create a system that provides information regarding tourist agencies and their packages. For that first, we store all data of our agencies and all about packages in our database then we create a one google form and then give it to the user. They will fill that form and from that, according to his interest and preference, we will give him all data, packages, and all other information about our agencies. That is the way our system works. This system should be portable, safe, and can not be accessible by unauthorized users. The system should be well functional and the system interface should not be complex and it's easy to understand for everyone. Response time of the system should be very less and it should not give any error during service. It has a large capacity to store users' data. Also, the system should work on every operating system. The system should be well maintained or up to date.

Our system provides access to Visitors, Customers, Managers, and Administrators. Users who are currently not logged in and have not registered to the website are visitors. Visitors can check the availability of flight reservations. In order to book the ticket, the visitor will need to log in with a valid username and password, else register with the app and create an account. Customers can check the availability of flight reservations and also book tickets by making payments online. They can also view their booking details, payment details, etc. The manager is the person who manages the system, modifies/adds, or deletes the customer. The administrator has access to the key database of our system and manages customer details, hotels, rooms, fight details, tour package details, etc. Also, the admin can view and print transaction reports, booking reports, cancellation reports, etc.

To make the system more efficient we created a google form to take surveys and also did some interviews. From all that we listed some requirements that can improve our system,

- The system provides the details of various packages based on information provided by travelers. A user can select the appropriate packages depending on convenience and accessibility. We should ensure that the system gives the highest possible level of accuracy about hotel vacancy, place safety, and transportation details.
- In the covid-19 situation, people like to travel to a safe place which is given by the minister of government. So we have to focus on providing safe places.

- In case of any emergency such as health deterioration, loss of any personal things, forgetting things to any place, etc. Emergency entities provide details such as Type, Contact of agency member, Address, etc.
- The system provides functionalities that allow a user to book tickets for trains, planes, and buses. This module also provides cancellation of previously booked tickets. The system maintains the details of all reservations made so far and allows administrators to either confirm or reject the bookings.
- Systems need to provide a data security policy so that the user data will be safe and don't access by anyone. so users can easily trust our system.
- We have gathered all the information from the interviews. We noted some requirements like Transportation services, Covid precaution, Guide services, Photographer, Good restaurants, Vehicle facility.
- The administrator manages all information about travelers and has all rights to add, delete, edit and view the data related to travelers, places, routes, restaurants, etc.. the system will check the administrator's username and password for authentication so only the administrator has all rights.
- The system provides information related to various routes connecting sources and destinations. For each route, information such as source, destination, fare, reservation details, pick-up points, etc is provided.
- Users of this application can post their opinions, complaints, and suggestions regarding this portal and services to the administrator. Accordingly, the administrator can take various steps to act on the complaints and suggestions.
- Performance Requirements are like the database shall be able to accommodate a minimum of 10,000 records of customers, the software shall support the use of multiple users at a time.
- Security Requirements: Some of the factors that are identified are to protect the software from accidental or malicious access. Utilize certain cryptographic techniques Keep a specific log or history data set. Assign certain functions to different modules. Check data integrity for critical variables. The software will include an error tracking log that will help the user understand what error occurred. When the application crashed along with suggestions on how to prevent the error from occurring again.

According to users requirements and by the observer our system has the following functionalities

- Sightseeing Tours: Tourists can enjoy sightseeing tours to any of the places listed in the Agencies Data File. Before that Customer has to make the booking by registering his/her name in the Data-File and informing the date of journey and No. of operations.
- Flight ticket booking: In this module, the customer can search source-destination, check flight timings, and book a ticket online.
- Bus ticket reservation module: In the bus ticket reservation module there has been a collection of buses, Customers can book the ticket by selecting the agency, date, and departure time of the bus.
- Train ticket reservation module: In the train reservation module customers can check train scheduled timings and they can book tickets online by entering Place, date, and time.
- Hotel reservation: After booking a train ticket, bus ticket, or flight ticket the system will provide the option to search hotels. Customers can reserve the hotels by selecting check-in date, check-out date, and hotel name.
- Holiday package: The customer can search tour packages by entering a destination, duration of travel, the month of travel, and the number of people.

- Payment module: This module allows customers to make payments after the reservation. The customer will receive a ticket receipt by mail after the payment.
- Report Generation: Details about the locations, hotels in that location, and final report on the journey fare.
- My Account: This module is for customers where customers can access the account module after login id and password. This module will display booked tickets, Billing receipts, ticket cancellation pages, logout, Etc.
- Cancel booking: Customers can cancel flight tickets, bus tickets, train tickets, or hotel bookings. The system refunds the paid amount after deducting cancellation charges.
- Dashboard module: This module allows the administrator of this site to manage the different aspects of ticket reservation-related activities. The administrator of the site can add or modify the records of flights, buses, trains, holiday packages, etc.

2.2 Noun (& Verb) Analysis

Table 1. Nouns & Verbs

Noun	Verb
tourist agencies	travel
covid cases	can
tourist	find
management database	make
travel package	present
present system	approach
various agencies	get
book tickets	like
satisfactory hotels	transport
transport facilities	create
check-in date	provide
old manual system	kept
maintenance	experience
seat availability	seat
hotel reservation	keep

manual calculation	lead
travel agencies	have
relevant information	prove
tour packages	allows
book tickets online	update
tourism management	need
travel agents	modify
management facilities	visit
software program	store
ticket reservation module	fight
relevant information	protect
covid situation	guide
visitors	going
google form	operating
system interface	regarding
response time	using
large capacity	sightseeing
covid situation	occurring
managers	depending
visitors	tracking
administrators	reached
customers	pick
flight reservations	registering
users	add

valid username	act
user data	utilize
administrator's username	delete
reservation details	edit
pick-up points	reject
payments online	report
payment details	fight
various packages	receive
appropriate packages	enjoy
hotel vacancy	prevent
place safety	help
transportation details	assign
covid-19 situation	include
safe places	give
personal things	fill
health deterioration	access
emergency	check
cancellation charges	print
contact	select
agency member	appropriate
address	take
data security policy	charges

transportation services	order
guide	focus
good restaurants	view
vehicle	understand
users requirements	service
sightseeing tours	well
tourists	gives
flight ticket	confirm
customer	surveys
check flight timings	booked
ticket online	bored
ticket reservation module	informing
various steps	selecting
multiple users	Billing
critical variables	maintains
data integrity	allow
certain functions	deducting
malicious access	given
certain cryptographic techniques	
utilize	
bus	
hotel name	

departure time	
bus ticket reservation module	
ticket receipt	
customers	
generation	
account	
billing	
login id	
train	
ticket cancellation	
hotel bookings	
system refunds	

2.3 Accepted Noun & Verbs list

Entity Set with Attributes

- **Admin**(admin_id, admin_name, ad_login_id, password)
- **Customer**(c_id, customername, customername, c_password, gender, contact_no, address, state, country, pincode)
- **Booking**(booking_id, c_id, contact_no, booking_type, booking_date, package_id)
- **Bus**(bus_id, bus_type, ticket_id, arrival time, departure time, boarding_point)
- **Train**(train_id, train_type, ticket_id, arrival time, departure time, boarding_point, destination_location)
- **Flight**(flight_id, flight_type, ticket_id, arrival time, departure time, boarding_point, destination_location)
- **Cancellation**(cancellation_id, booking_id, cancel_status, payment_id)
- **Payment**(payment_id, booking_id, payment_method, amount)
- **Emergency**(emergency_id, type, contact)
- **Tour_Package**(package_id, package_name, package_type, no_of_days, package_amount, package_description)
- **Tourist_Places**(place_id, place_name, place_type, place_description, place_images, safety_description, weather)
- **Hotel**(hotel_id, hotel_address, hotel_name, hotel_type, hotel_image, room_type, checking_date, checkout_date)

Candidate relationship set

- **Customer** pay **Payment**
- **Customer** selecting **Tourist_Place**
- **Admin** update **Tour_Package**
- **Admin** monitor **Customer**
- **Cancellation** access by **Admin**
- **Tour_Package** and **Tour_Places** and **hotel** gives details of tour
- **Emergency** service available at all **Tourist_Places**
- **Payment** confirm the **Bus's** ticket **Booking**
- **Payment** confirm the **Flight's** ticket **Booking**
- **Payment** confirm the **Train's** ticket **Booking**

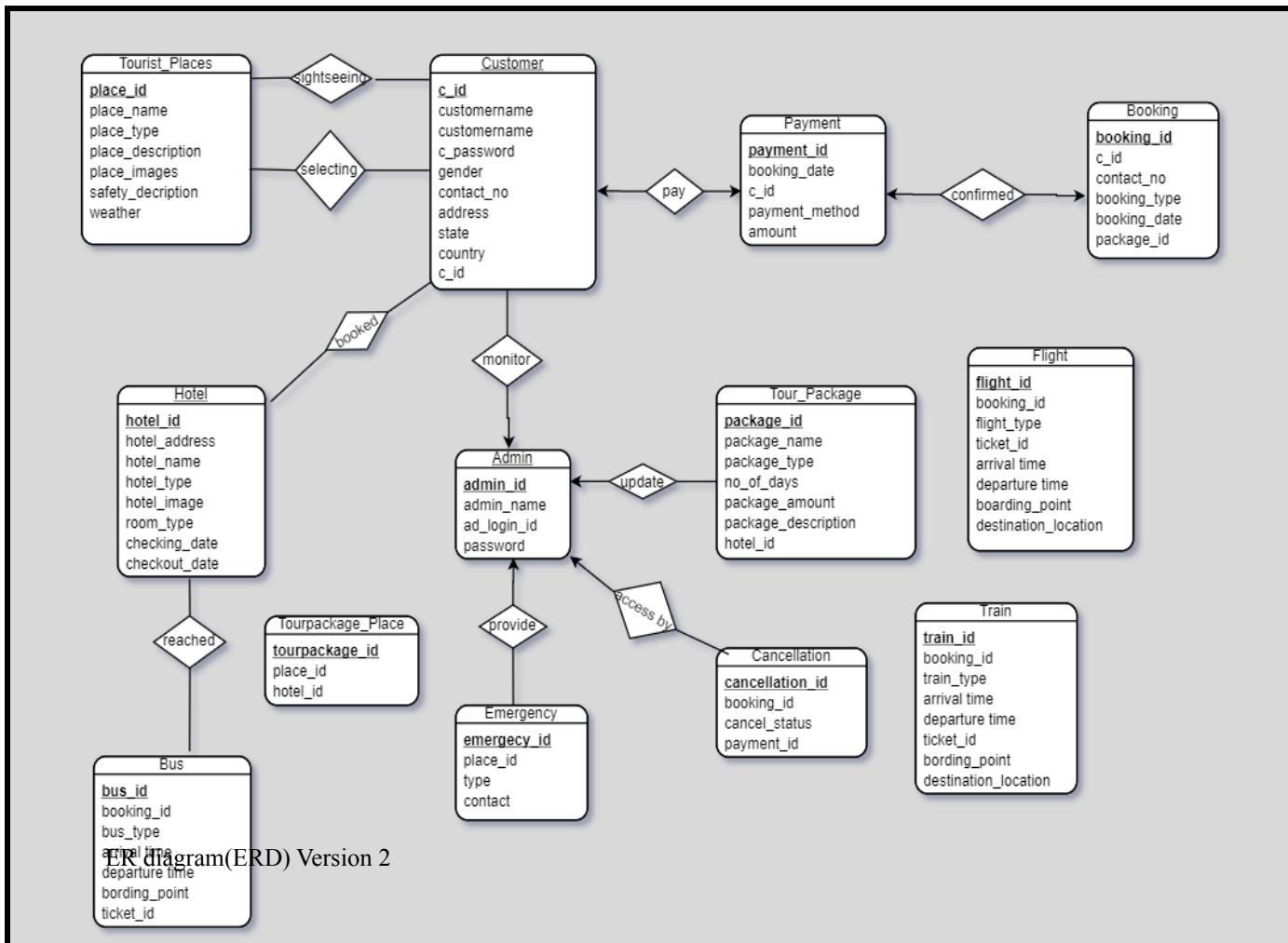
2.4 Rejected Noun & Verbs list

Noun	Reject Reason
tourist	Vague
travel package	Attribute
present system	Irrelevant
old manual system	Vague
seat availability	Attribute
manual calculation	Irrelevant
tour packages	Duplicates
that's	General
travel agents	Attributes
relevant information	Duplicates
google form	General
system interface	Vague
visitors	Duplicates
users	Duplicates
administrator's username	Attributes
pick-up points	Attributes

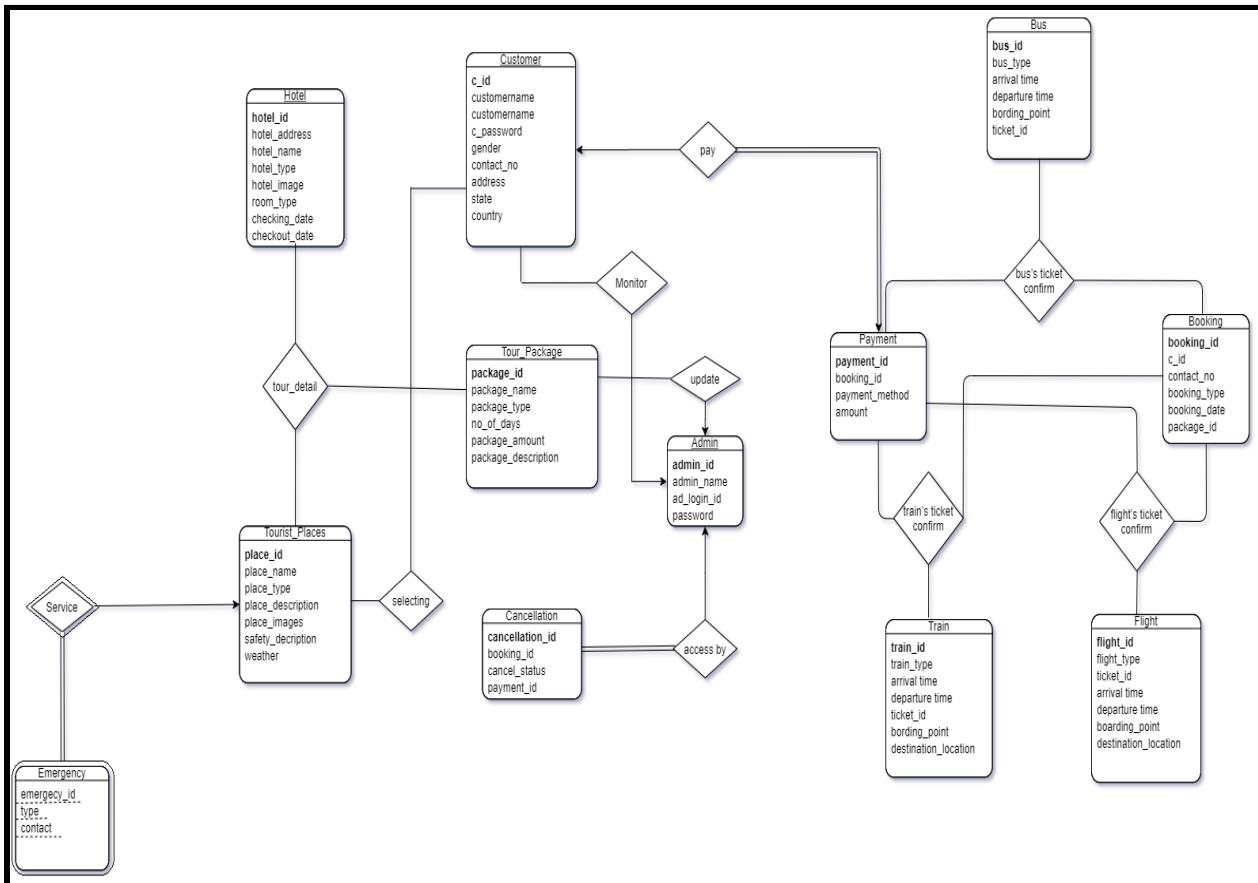
hotel vacancy	Attributes
covid-19 situation	Duplicates
agency member	Attributes
various routes	Irrelevant
users requirements	Vague
data integrity	Vague
malicious access	Vague
certain cryptographic techniques	Irrelevant
customers	General
login id	Attributes
cancellation charges	Vague
hotel name	Attributes
departure time	Attributes
check-in date	Irrelevant

Section3 : ER-Diagrams all versions

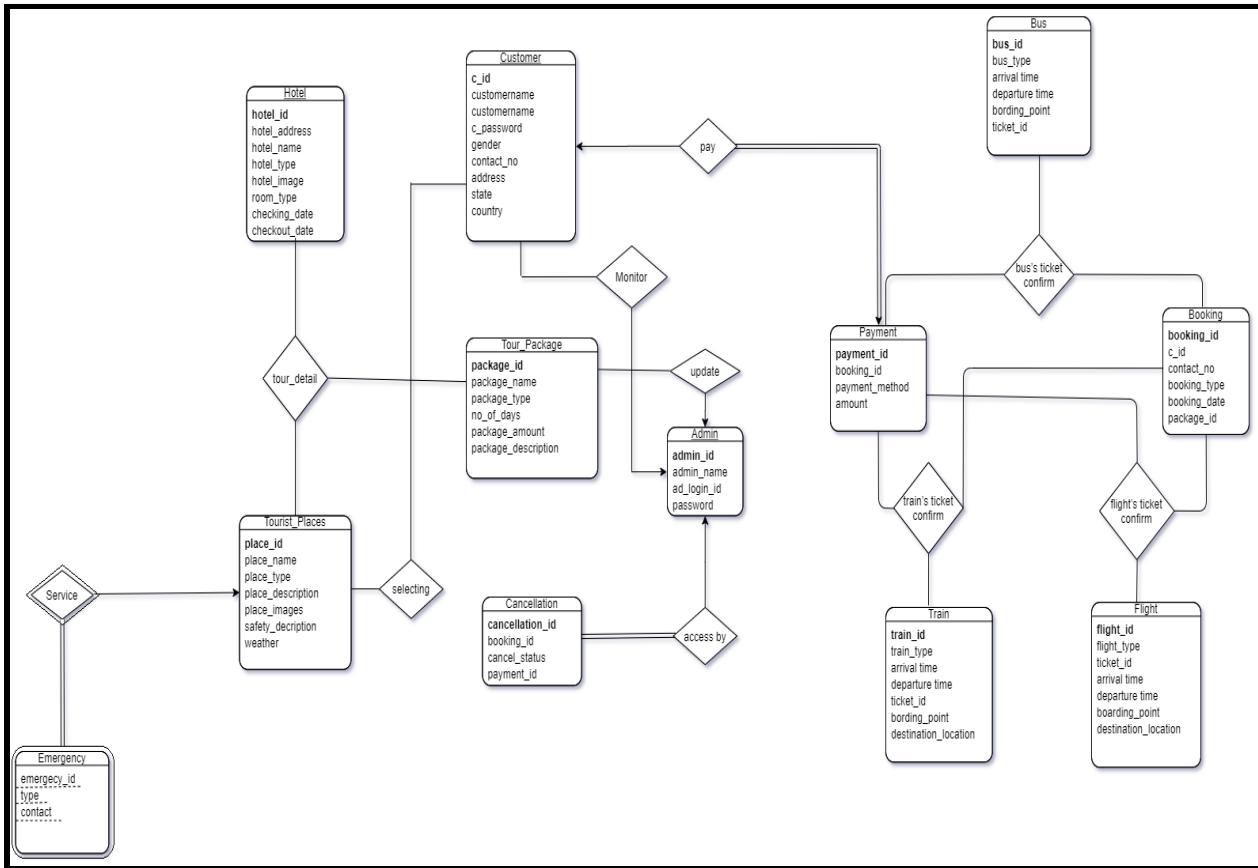
3.1 ER Diagram(ERD) Version 1



3.2 ER Diagram(ERD) Version 2



3.3 Final ER diagram

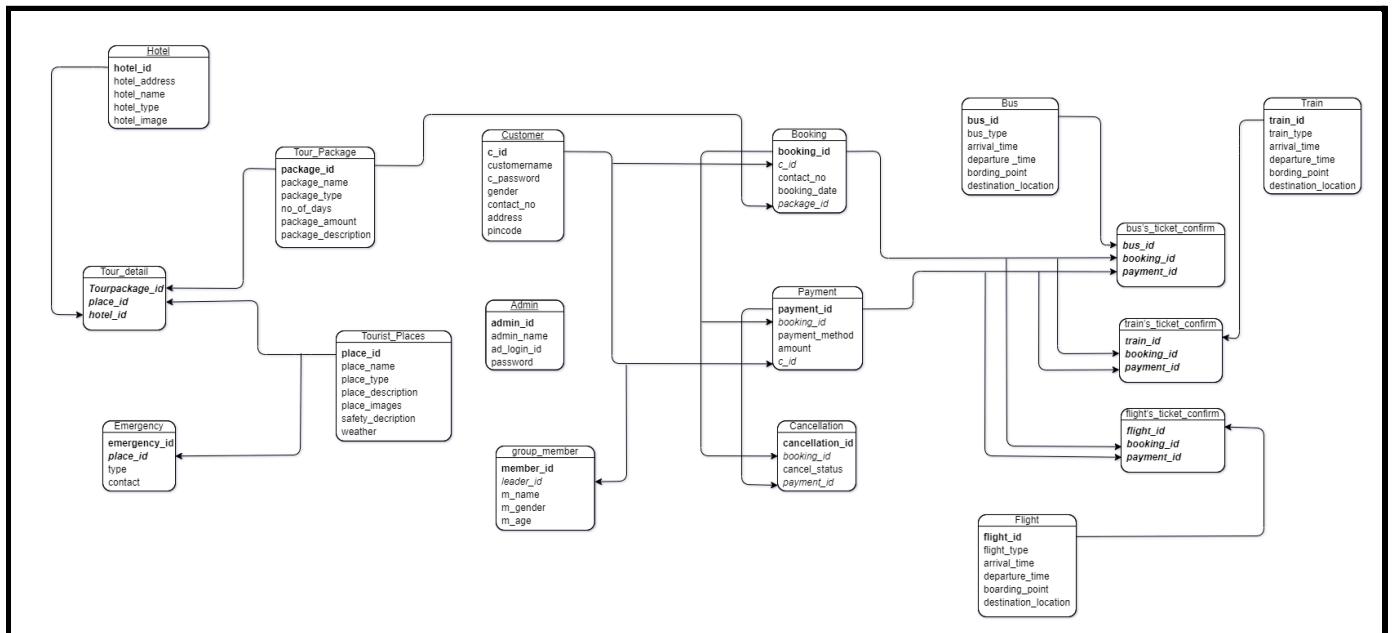


Section4 : Conversion of Final ER-Diagram to Relational Model

4.1 Mapping E-R Model to Relational Model

- Admin(admin_id, admin_name, ad_login_id, password)
- Customer(c_id, customername, c_password, gender, contact_no, address, pincode)
- group_member(member_id, leader_id, m_name, m_age, m_gender)
- Booking(booking_id, c_id, contact_no, booking_date, package_id)
- Bus(bus_id, bus_type, arrival_time, departure_time, boarding_point, destination_location)
- Train(train_id, train_type, arrival_time, departure_time, boarding_point, destination_location)
- Flight(flight_id, flight_type, arrival_time, departure_time, boarding_point, destination_location)
- Cancellation(cancellation_id, booking_id, cancel_status, payment_id)
- Payment(payment_id, booking_id, c_id, payment_method, amount)
- Emergency(emergecye_id, type, place_id, contact)
- Tour_Package(package_id, package_name, package_type, no_of_days, package_amount, package_description)
- Tour_detail(tourpackage_id, place_id, hotel_id)
- Tourist_Places(place_id, place_name, place_type, place_description, place_images, safety_decription, weather)
- Hotel(hotel_id, hotel_address, hotel_name, hotel_type, hotel_image)
- bus_ticket_confirm(bus_id, payment_id, booking_id)
- flight_ticket_confirm(flight_id, payment_id, booking_id)
- train_ticket_confirm(train_id, payment_id, booking_id)

4.2 Relational Schema



Section5 : Normalization and Schema Refinement

5.1 Original Database

- Admin(**admin_id**, admin_name, ad_login_id, password)
- Customer(**c_id**, customername, c_password, gender, contact_no, address, pincode)
- group_member(**member_id**, leader_id, m_name, m_age, m_gender)
- Booking(**booking_id**, c_id, contact_no, booking_date, package_id)
- Bus(**bus_id**, bus_type, arrival_time, departure_time, boarding_point, destination_location)
- Train(**train_id**, train_type, arrival_time, departure_time, boarding_point, destination_location)
- Flight(**flight_id**, flight_type, arrival_time, departure_time, boarding_point, destination_location)
- Cancellation(**cancellation_id**, booking_id, cancel_status, payment_id)
- Payment(**payment_id**, booking_id, c_id, payment_method, amount)
- Emergency(**emergecy_id**, type, place_id, contact)
- Tour_Package(**package_id**, package_name, package_type, no_of_days, package_amount, package_description)
- Tour_detail(**tourpackage_id**, **place_id**, **hotel_id**)
- Tourist_Places(**place_id**, place_name, place_type, place_description, place_images, safety_decription, weather)
- Hotel(**hotel_id**, hotel_name, hotel_address, hotel_type, hotel_image)
- bus_ticket_confirm(**bus_id**, **payment_id**, **booking_id**)
- flight_ticket_confirm(**flight_id**, **payment_id**, **booking_id**)
- train_ticket_confirm(**train_id**, **payment_id**, **booking_id**)

5.2 Functional Dependencies:

- Admin(**admin_id**, admin_name, ad_login_id, password)

Primary Key : admin_id

Functional dependencies :

$$\begin{aligned} \text{admin-id} &\rightarrow \text{admin_name} \\ \text{admin-id} &\rightarrow \text{ad_login_id} \\ \text{admin-id} &\rightarrow \text{password} \end{aligned}$$
- Customer(**c_id**, customername, c_password, gender, contact_no, address, pincode)

Primary Key : c_id

Functional dependencies :

$$\begin{aligned} \text{c_id} &\rightarrow \text{customername} \\ \text{c_id} &\rightarrow \text{c_password} \\ \text{c_id} &\rightarrow \text{gender} \\ \text{c_id} &\rightarrow \text{contact_no} \\ \text{c_id} &\rightarrow \text{address} \\ \text{c_id} &\rightarrow \text{pincode} \end{aligned}$$

- group_member(**member_id** , leader_id , m_name, m_age, m_gender)

Primary Key : member_id

Foreign Key : leader_id

Functional dependencies :

member_id → m_name

member_id → m_age

member_id → m_gender

- Booking(**booking_id**, c_id, contact_no, booking_date, package_id)

Primary Key : booking_id

Foreign Key : c_id, package_id

Functional dependencies :

booking_id → contact_no

booking_id → booking_date

- Bus(**bus_id**, bus_type, arrival_time, departure_time, boarding_point, destination_location)

Primary Key : bus_id

Functional dependencies :

bus_id → bus_type

bus_id → arrival_time

bus_id → departure_time

bus_id → boarding_point

bus_id → destination_location

- Train(**train_id**, train_type, arrival_time, departure_time, boarding_point, destination_location)

Primary Key : train_id

Functional dependencies :

train_id → train_type

train_id → arrival_time

train_id → departure_time

train_id → boarding_point

train_id → destination_location

- Flight(**flight_id**, flight_type, arrival_time, departure_time, boarding_point, destination_location)

Primary Key: flight_id

Functional dependencies :

flight_id → flight_type

flight_id → arrival_time

flight_id → departure_time

flight_id → boarding_point

flight_id → destination_location

- Cancellation(**cancellation_id**, booking_id, cancel_status, payment_id)

Primary Key : cancellation_id

Foreign Key : booking_id,payment_id

Functional dependencies :

cancellation_id → cancel_status

- Payment(**payment_id**, booking_id, c_id, payment_method, amount)

Primary Key : payment_id

Foreign Key : booking_id,c_id

Functional dependencies :

payment_id → payment_method

payment_id → amount

- Emergency(**emergecy_id**, type, place_id, contact)

Primary Key : emergency_id, place_id

Foreign Key : place_id

Functional dependencies :

emergency_id → type

emergency_id → contact

- Tour_Package(**package_id**,package_name, package_type, no_of_days, package_amount, package_description)

Primary Key : package_id

Functional dependencies :

package_id → package_name

package_id → package_type

package_id → no_of_days

package_id → package_amount

package_id → package_description

- Tour_detail(**tourpackage_id, place_id, hotel_id**)

Primary Key : tourpackage_id, place_id,hotel_id

Foreign Key : tourpackage_id,place_id,hotel_id

- Tourist_Places(**place_id**, place_name, place_type, place_description, place_images, safety_description, weather)

Primary Key : place_id

Functional dependencies :

place_id → place_name

place_id → place_type

place_id → place_description

place_id → place_images

place_id → safety_description

place_id → weather

- Hotel(**hotel_id**, hotel_name, hotel_address, hotel_type, hotel_image)

Primary Key : hotel_id

Functional dependencies :

hotel_id → hotel_address

hotel_id → hotel_name

hotel_id → hotel_type

hotel_id → hotel_image

- bus_ticket_confirm(**bus_id, payment_id, booking_id**)

Primary Key : bus_id, payment_id, booking_id

Foreign Key : bus_id, payment_id, booking_id

- flight_ticket_confirm(**flight_id, payment_id, booking_id**)

Primary Key : flight_id, payment_id, booking_id

Foreign Key : flight_id, payment_id, booking_id

- train_ticket_confirm(**train_id, payment_id, booking_id**)

Primary Key : train_id, payment_id, booking_id

Foreign Key : train_id, payment_id, booking_id

5.3 Redundancies:

Redundancies occur when we have multiple copies of the same data in our database. Basically, this problem arises when data is not normalised. In our database, we have stored all our data in different types of tables, so we don't have any redundancies in our database.

5.4 Update, insert and delete anomalies:

Insertion deletion and update anomalies mean If the user performing the update does not realize the data is stored redundantly the update will not be done properly. A deletion anomaly is the unintended loss of data due to deletion of other data. ... An insertion anomaly is the inability to add data to the database due to the absence of other data.

But in our database, we have stored all necessary information in different tables and we don't have any non-prime attributes functional dependencies, so we don't have any type of anomalies in our database.

5.5 Normalization to 1NF:

In 1NF form, there should be no repeated columns inside a row or multivalued columns in any database relation, and all attributes should be reliant on the primary key.

In our database design, all the listed attributes in the relations are atomic. We're also just taking one entry for each of the attributes. As a result, the preceding 1NF normalized form is used in database design.

5.6 Normalization to 2NF:

Each and every relationship in the 2NF normalized form should be in the 1NF form, and there should be no partial dependencies in the relations. No nonprime attributes are dependent on any proper subset of the relation's candidate key, according to partial dependency. Any database design relations that have only one Primary key (not composite) are already in the 2NF form.

In our database design there are no partial dependencies, as can be seen from the functional dependencies listed above. As a result, this design has already been normalized to 2NF.

5.7 Normalization to 3NF:

To be in the 3NF, all of the relations must first be in the 2NF form, and there should be no transitive functional dependencies in the relations for non-prime attributes in the 3NF. That means, functional dependencies such as non-prime \rightarrow non-prime does not exist.

In our database design, there are no transitive functional dependencies because all dependencies are defined by the primary key (candidate key) only. As a result, this database design is already 3NF normalized.

5.8 Normalization to BCNF :

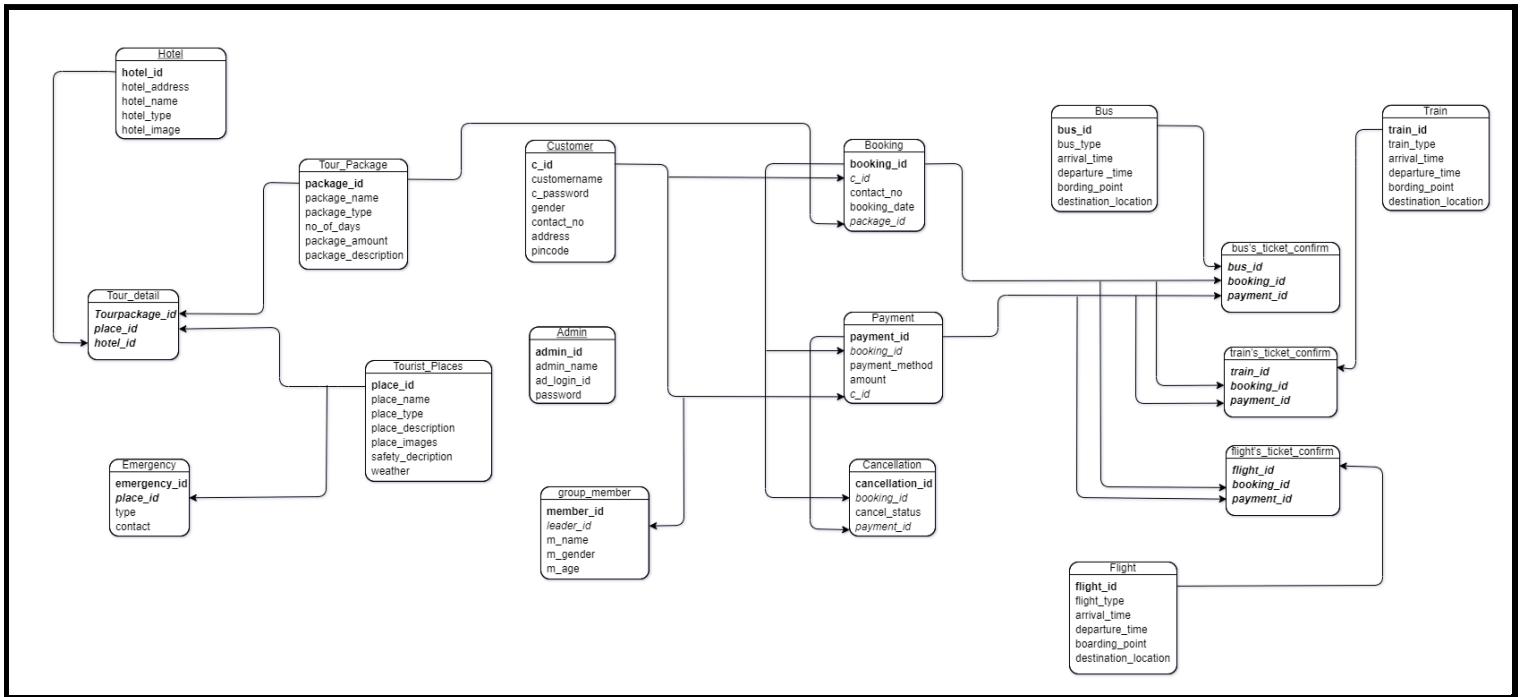
For the relation to be in BCNF form, if all the functional dependencies have a super key or candidate key on their left-hand side, no non-prime attribute can be on the left-hand side.

In our database, the only candidate key, which is also a primary key and it can define all the other attributes present in the relation. Hence all the functional dependencies of type A \rightarrow B are such that A is a name that is a super key hence it is already normalized to BCNF.

NOTE :

The Bus's_ticket_confirm, train's_ticket_confirm and flight's_ticket_confirm tables have only primary keys so there are no functional dependencies as such in any of the attributes so this relation is already normalized. So, the table is in 1NF,2NF,3NF, and BCNF.

5.9 Final relations with the schema



- Admin(admin_id, admin_name, ad_login_id, password)
- Customer(c_id, customername, c_password, gender, contact_no, address, pincode)
- group_member(member_id, leader_id, m_name, m_age, m_gender)
- Booking(booking_id, c_id, contact_no, booking_date, package_id)
- Bus(bus_id, bus_type, arrival_time, departure_time, boarding_point, destination_location)
- Train(train_id, train_type, arrival_time, departure_time, boarding_point, destination_location)
- Flight(flight_id, flight_type, arrival_time, departure_time, boarding_point, destination_location)
- Cancellation(cancellation_id, booking_id, cancel_status, payment_id)
- Payment(payment_id, booking_id, c_id, payment_method, amount)
- Emergency(emergency_id, type, place_id, contact)
- Tour_Package(package_id, package_name, package_type, no_of_days, package_amount, package_description)
- Tour_detail(tourpackage_id, place_id, hotel_id)
- Tourist_Places(place_id, place_name, place_type, place_description, place_images, safety_description, weather)
- Hotel(hotel_id, hotel_name, hotel_address, hotel_type, hotel_image)
- bus_ticket_confirm(bus_id, payment_id, booking_id)
- flight_ticket_confirm(flight_id, payment_id, booking_id)
- train_ticket_confirm(train_id, payment_id, booking_id)

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

6.1 Final DDL Scripts

```

CREATE TABLE Admin
(
    admin_id integer,
    admin_name varchar(30) not null,
    ad_login_id varchar(30),
    password varchar(30),
    primary key (admin_id)
);

CREATE TABLE Customer
(
    c_id integer,
    customername varchar(30) not null,
    c_password varchar(30),
    gender varchar(6),
    contact_no varchar(30),
    address varchar(300),
    pincode integer,
    primary key (c_id),
    check(gender in ('Female','Male'))
);

CREATE TABLE group_member
(
    member_id integer,
    leader_id integer not null,
    m_name varchar(30) not null,
    m_age integer,
    m_gender varchar(6),
    primary key (member_id),
    check(m_gender in ('Female','Male')),
    foreign key (leader_id) references Customer(c_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

```

```
CREATE TABLE Booking
(
    booking_id varchar(20),
    c_id integer,
    contact_no varchar(30),
    booking_date varchar(50),
    package_id integer,
    primary key (booking_id),
    foreign key (c_id) references Customer
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    foreign key (package_id) references Tour_Package
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```

```
CREATE TABLE Bus
(
    bus_id varchar(20) ,
    bus_type varchar(30),
    arrival_time varchar(50),
    departure_time varchar(50),
    boarding_point varchar(30),
    destination_location varchar(30),
    primary key (bus_id)
);
```

```
CREATE TABLE Train
(
    train_id varchar(20),
    train_type varchar(30),
    arrival_time varchar(50),
    departure_time varchar(50),
    boarding_point varchar(30),
    destination_location varchar(30),
    primary key (train_id)
);
```

```
CREATE TABLE Flight
(
    flight_id varchar(20),
    flight_type varchar(30),
    arrival_time varchar(50),
    departure_time varchar(50),
    boarding_point varchar(30),
    destination_location varchar(30),
    primary key (flight_id)
);

CREATE TABLE Cancellation
(
    cancellation_id varchar(20),
    booking_id varchar(20) not null,
    cancel_status varchar(20),
    payment_id varchar(20) not null,
    primary key (cancellation_id),
    foreign key (booking_id) references Booking
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    foreign key (payment_id) references Payment
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE Payment
(
    payment_id varchar(20),
    booking_id varchar(20) not null,
    c_id integer not null,
    payment_method varchar(20),
    amount integer check(amount > 0),
    primary key (payment_id),
    foreign key (booking_id) references Booking
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    foreign key (c_id) references Customer
);
```

```
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
);
```

```
CREATE TABLE Emergency  
(  
    emergency_id varchar(20),  
    type varchar(50) ,  
    place_id varchar(20),  
    contact varchar(30),  
    primary key (emergency_id, place_id),  
    foreign key (place_id) references Tourist_places  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
);
```

```
CREATE TABLE Tour_Package  
(  
    package_id integer,  
    package_name varchar(30),  
    package_type varchar(30),  
    no_of_days integer,  
    package_amount integer check(package_amount > 0),  
    package_description varchar(2000),  
    primary key (package_id)  
);
```

```
CREATE TABLE Tour_detail  
(  
    tourpackage_id integer,  
    place_id varchar(20),  
    hotel_id varchar(20),  
    primary key (tourpackage_id, place_id, hotel_id),  
    foreign key (tourpackage_id) references tour_package(package_id)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
    foreign key (place_id) references Tourist_Places
```

```
ON DELETE CASCADE  
ON UPDATE CASCADE,  
foreign key (hotel_id) references Hotel  
ON DELETE CASCADE  
ON UPDATE CASCADE  
);
```

```
CREATE TABLE Tourist_Places  
(  
    place_id varchar(20),  
    place_name varchar(30) not null,  
    place_type varchar(30),  
    place_description varchar(3000),  
    place_images varchar(200),  
    safety_decription varchar(3000),  
    weather varchar(30),  
    primary key (place_id)  
);
```

```
CREATE TABLE Hotel  
(  
    hotel_id varchar(20),  
    hotel_name varchar(30),  
    hotel_address varchar(100),  
    hotel_type varchar(30),  
    hotel_image varchar(200),  
    primary key(hotel_id)  
);
```

```
CREATE TABLE bus_ticket_confirm  
(  
    bus_id varchar(20),  
    payment_id varchar(20),  
    booking_id varchar(20),  
    primary key(bus_id,payment_id,booking_id),  
    foreign key (bus_id) references Bus  
    ON DELETE CASCADE
```

```
    ON UPDATE CASCADE,  
    foreign key (payment_id) references Payment  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
    foreign key (booking_id) references Booking  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
);
```

```
CREATE TABLE train_ticket_confirm  
(  
    train_id varchar(20),  
    payment_id varchar(20),  
    booking_id varchar(20),  
    primary key(train_id,payment_id,booking_id),  
    foreign key (train_id) references Train  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
    foreign key (payment_id) references Payment  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
    foreign key (booking_id) references Booking  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
);
```

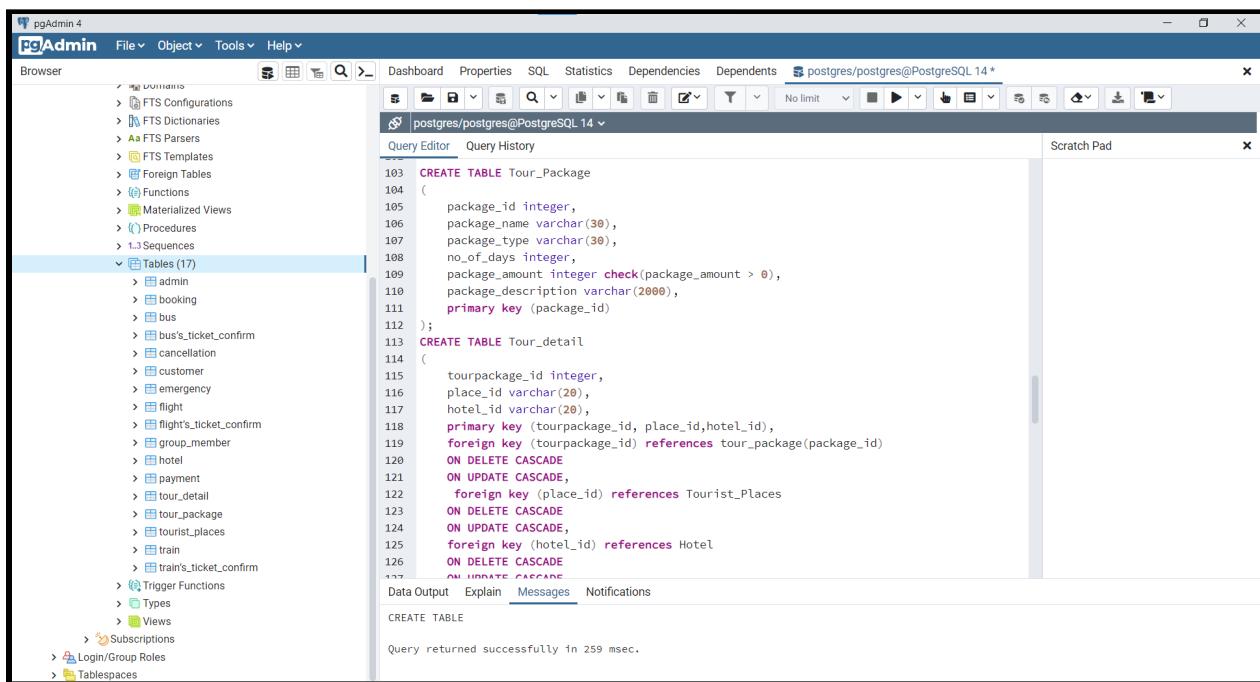
```
CREATE TABLE flight_ticket_confirm  
(  
    flight_id varchar(20),  
    payment_id varchar(20),  
    booking_id varchar(20),  
    primary key(flight_id ,payment_id ,booking_id),  
    foreign key(flight_id ) references Flight  
    ON DELETE CASCADE  
    ON UPDATE CASCADE,  
    foreign key(payment_id) references Payment  
    ON DELETE CASCADE
```

```

    ON UPDATE CASCADE,
    foreign key (booking_id) references Booking
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

```

6.2 Creation of Tables



The screenshot shows the pgAdmin 4 interface. The left sidebar (Browser) lists various database objects: Schemas, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, and Tables (17). The Tables section contains: admin, booking, bus, bus_ticket_confirm, cancellation, customer, emergency, flight, flight_ticket_confirm, group_member, hotel, payment, tour_detail, tour_package, tourist_places, train, train_ticket_confirm, Trigger Functions, Types, Views, Subscriptions, Login/Group Roles, and Tablespaces. The main area (Query Editor) displays the SQL code for creating the 'Tour_Package' and 'Tour_detail' tables. The code includes primary key constraints, foreign key references, and cascade options for update and delete operations. The status bar at the bottom indicates the query was executed successfully in 259 msec.

```

103 CREATE TABLE Tour_Package
104 (
105     package_id integer,
106     package_name varchar(30),
107     package_type varchar(30),
108     no_of_days integer,
109     package_amount integer check(package_amount > 0),
110     package_description varchar(2000),
111     primary key (package_id)
112 );
113 CREATE TABLE Tour_detail
114 (
115     tourpackage_id integer,
116     place_id varchar(20),
117     hotel_id varchar(20),
118     primary key (tourpackage_id, place_id, hotel_id),
119     foreign key (tourpackage_id) references tour_package(package_id)
120     ON DELETE CASCADE
121     ON UPDATE CASCADE,
122         foreign key (place_id) references Tourist_Places
123     ON DELETE CASCADE
124     ON UPDATE CASCADE,
125         foreign key (hotel_id) references Hotel
126     ON DELETE CASCADE
127     ON UPDATE CASCADE
Data Output Explain Messages Notifications
CREATE TABLE
Query returned successfully in 259 msec.

```

6.3 Insert statements

We have created all tables and inserted all tuples using the following SQL query

```

COPY t4.Admin
FROM 'E:\SEM5\DBMS\database\admin.csv' DELIMITER ',' CSV HEADER;

```

Admin

The screenshot shows the pgAdmin 4 interface with the 'admin' database selected. The left sidebar lists various database objects like FTS Configurations, Tables, and Views. The main area shows a query editor with the following SQL:

```
1 select *
2 from admin
```

The results table displays 10 tuples from the 'admin' table:

admin_id	admin_name	ad_login_id	password
1	Clementius Iacopo	ciaofo@booking.com	hJKmlx
2	Joyous Lundon	jlundon1@toplist.cz	JxBSEkTQ
3	Vincenz Bramstom	vbramstom2@dailymotion.com	fxya31Fj
4	Bobette Devin	bdevin3@album.net	lZMUIjQLNINc
5	Susannah Janous	sjanous4@reverbnation.com	6nVeEK
6	Wyrndham Mallinar	wmallinar5@cargo collective.com	obfNRWp6bFt
7	Natala Kolla	nkolla6@vistaprint.com	KNTLGO
8	Joan Kolakovic	jkolakovic7@toplist.cz	NfZWxPAFH
9	Hartwell Coulebeck	hcoulebeck8@cbnews.com	toZm2g9Vo
10	Eugenius Canin	e canin9@pnnov.no	v5v0uNikX1

Number of tuples: 10

Customer

The screenshot shows the pgAdmin 4 interface with the 'customer' database selected. The left sidebar lists various database objects. The main area shows a query editor with the following SQL:

```
1 select *
2 from customer
```

The results table displays 10 tuples from the 'customer' table:

c_id	customername	c_password	gender	contact_no	address	pincode
1	Sarge Jenkins	vdGCjh6ZtmF	Male	+86 251 239 1354	3668 Johnson Circle	197652
2	Adina Brettoner	VfxrSJ	Male	+216 715 258 6245	1302 Summer Ridge Drive	639200
3	Veronike Colleck	re1T93mG5b	Female	+48 881 110 7673	17331 Algoma Road	847614
4	Noe Burlingham	rUz25tykB	Female	+30 203 872 1839	9605 Mayfield Junction	391485
5	Aryln Coyne	RNBHL4o	Male	+93 460 585 8663	8385 Ruskin Plaza	452871
6	Hendrika McKeeman	OlvOli	Female	+92 943 238 2470	870 Crest Line Junction	670784
7	Cchaddie Shoobridge	foTuJYhFrY	Male	+52 631 832 2744	82 Butterfield Hill	754744
8	Randi Muller	WnyTfBw	Female	+52 711 268 8608	3643 Kings Trail	738676
9	Hube See	U5Hhr6	Female	+232 370 643 5344	555 O'Neill Alley	680616
10	Mattheus Ventris	vgRR607	Female	+57 239 657 4379	93809 Sullivan Circle	717881

Number of tuples: 300

Group Member

The screenshot shows the pgAdmin 4 interface with the 'group_member' table selected. The browser pane on the left lists various database objects, and the query editor pane contains the following SQL code:

```
1 select *
2 from group_member
```

The results are displayed in a data grid with the following columns: member_id, leader_id, m_name, m_age, and m_gender. The data consists of 10 rows of member information.

	member_id	leader_id	m_name	m_age	m_gender
1	1	23	Crissy Carlton	23	Female
2	2	88	Maximilian McCaughey	44	Male
3	3	253	Loutitia Coleby	34	Female
4	4	149	Iolande Tolhurst	16	Male
5	5	123	Pebrook Heibel	41	Male
6	6	96	Wenona Barajas	18	Female
7	7	205	Tova Cheeld	18	Male
8	8	38	Hilliard Peter	27	Female
9	9	269	Inglis Froschauer	47	Male
10	10	118	Bonnielle Colouyte	45	Female

Number of tuples: 500

Tourist Place

The screenshot shows the pgAdmin 4 interface with the 'tourist_places' table selected. The browser pane on the left lists various database objects, and the query editor pane contains the following SQL code:

```
1 select *
2 from tourist_places
```

The results are displayed in a data grid with the following columns: place_id, place_name, place_type, and place_description. The data consists of 9 rows of tourist place information.

	place_id	place_name	place_type	place_description
1	1	The Red Fort, Delhi	Agricultural land	Established as Shahjahanabad in 1648, The Red Fort was known as the capital of the Mughal Empire.
2	2	The Taj Mahal, Agra	Beach	The name Taj Mahal, translates to the 'Crown Palace' and is one of the most famous and marvelous
3	3	Pangong Lake, Ladakh	City	With its location between India and Tibet, Pangong Lake is an enchanting saltwater lake located in t
4	4	Valley of Flowers, Nainital	Forest	As evident by its name, the Valley of Flowers is known for its vast diversity of alpine flowering shrub
5	5	Jaisalmer Fort, Jaisalmer	Cultural quarter	Located amidst the golden sands of the Thar Desert, the Jaisalmer Fort was built by the Bhati Rajpu
6	6	Ruins of Hampi, Karnataka	rural	Hampi was the capital of the erstwhile Vijayanagar Empire and known to be one of the richest cities
7	7	Ghats at Varanasi	Entrapment place	Also known by the names Kashi and Benares, Varanasi is said to have been continuously inhabited s
8	8	Backwaters, Kerala	Mountain	The Backwaters of Kerala are essentially a group of 5 lagoons that are linked by natural and manna
9	9	Old Goa, Goa	desert	Located in the Northern Goa district of Goa, Old Goa refers to the historic town that served as the se

Number of tuples: 100

Tour Package

The screenshot shows the pgAdmin 4 interface with the 'tourist_agencies' database selected. The 'Tables' browser pane shows 17 tables, and the 'tour_package' table is selected. The 'Query Editor' pane displays the SQL query: 'select * from tour_package'. The 'Data Output' pane shows the results of the query, which consists of 100 tuples. The columns are: package_id [PK] integer, package_name character varying (30), package_type character varying (30), no_of_days integer, package_amount integer, and package_description character varying (200). The data includes various tourism packages like Adventure, Wildlife, Medical, Pilgrimage, Eco, Cultural, Cruise, Wellness, and Family packages.

package_id	package_name	package_type	no_of_days	package_amount	package_description
1	Adventure Tourism Package	Adventure	2	1500	Adventure tourism, in general, is defined as the type of tourism that encou...
2	Wildlife Tourism Package	Wildlife	8	6400	If you are a wildlife explorer or photographer and wish to learn and take pic...
3	Medical Tourism Package	Medical	13	12000	The medical tourism in India, for the past few years, has gained the attenti...
4	Pilgrimage Tourism Package	Pilgrimage	10	9000	The major reason that attracts tourists to India is the traditional temple ar...
5	Eco Tourism Package	Eco	3	1500	Although, ecotourism is more of a travel philosophy and as India has a rich...
6	Cultural Tourism Package	Cultural	7	6900	The social richness of the country draws the visitors from every corner of t...
7	Cruise Tourism Package	Cruise	3	1999	Cruise tourism permits tourists to explore beautiful riverside villages, und...
8	Wellness Tourism Package	Wellness	10	6999	Renowned as the cradle of medical science like Ayurveda, Yoga, Naturopat...
9	Family Tourism Package	Family	14	23999	The family tourism in India offers all kind of activities like visiting the cult...

Number of tuples: 100

Hotel

The screenshot shows the pgAdmin 4 interface with the 'tourist_agencies' database selected. The 'Tables' browser pane shows 17 tables, and the 'hotel' table is selected. The 'Query Editor' pane displays the SQL query: 'select * from hotel'. The 'Data Output' pane shows the results of the query, which consists of 10 tuples. The columns are: hotel_id [PK] character varying (20), hotel_name character varying (30), hotel_address character varying (100), hotel_type character varying (30), and hotel_image character varying (200). The data includes various hotel names and types across different locations.

hotel_id	hotel_name	hotel_address	hotel_type	hotel_image
H1	Grand Hyatt Goa	goa	Chain hotels	http://dummyimage.co...
H2	Planet Hollywood Goa	goa	Motels	http://dummyimage.co...
H3	Ronil Royale	goa	Resorts	http://dummyimage.co...
H4	Taj Fort Aguada Resort	goa	Extended stay hotels	http://dummyimage.co...
H5	Nahar Singh Mahal	Haryana	Casino hotels	http://dummyimage.co...
H6	The Oberoi, Gurgaon	Haryana	Pop-up hotels	http://dummyimage.co...
H7	Hotel sakari place,guj...	Gujarat	Resorts	http://dummyimage.co...
H8	Peterhoff, Shimla	Himachal Pradesh	Chain hotels	http://dummyimage.co...
H9	Hari Niwas Palace, Ja...	Jammu and Kashmir	Motels	http://dummyimage.co...
H10	Jayamahal Palace Ho...	Karnataka	Resorts	http://dummyimage.co...

Number of tuples: 80

Booking

The screenshot shows the pgAdmin 4 interface with the 'Booking' table selected in the left sidebar. The main area displays the following SQL query and its results:

```
1 select *
2 from booking
```

	booking_id	c_id	contact_no	booking_date	package_id
1	B1	178	+46 567 122 3006	3/27/2021	7
2	B2	152	+62 957 813 8552	01-06-2021	57
3	B3	234	+33 984 754 6501	07-01-2021	17
4	B4	285	+86 493 711 1807	4/25/2021	81
5	B5	67	+56 649 878 5388	3/27/2021	28
6	B6	204	+504 477 961 3704	2/23/2021	2
7	B7	168	+86 896 838 2753	5/23/2021	66
8	B8	29	+359 418 779 0350	8/15/2021	63
9	B9	167	+46 551 495 2821	2/19/2021	94
10	B10	285	+46 877 523 9615	03-08-2021	54

Number of tuples: 200

Payment

The screenshot shows the pgAdmin 4 interface with the 'Payment' table selected in the left sidebar. The main area displays the following SQL query and its results:

```
1 select *
2 from payment
```

	payment_id	booking_id	c_id	payment_method	amount
1	P1	B1	178	googlepay	1500
2	P2	B2	152	easypay	6400
3	P3	B3	234	phonipay	12000
4	P4	B4	285	razorpay	9000
5	P5	B5	67	upi	1500
6	P6	B6	204	paytm	6900
7	P7	B7	168	debit card	1999
8	P8	B8	29	credit card	6999
9	P9	B9	167	googlepay	23999
10	P10	B10	285	easynav	30999

Number of tuples: 200

Cancellation

```
1 select *
2 from cancellation
```

cancellation_id	booking_id	cancel_status	payment_id
c1	B3	yes	P3
c2	B4	no	P4
c3	B5	no	P5
c4	B6	yes	P6
c5	B7	no	P7
c6	B8	yes	P8
c7	B9	yes	P9
c8	B17	no	P17
c9	B18	no	P18
c10	B19	yes	P19

Number of tuples: 30

Emergency

```
1 select *
2 from emergency
```

emergency_id	type	place_id	contact
E1	Medicine	85	+86 251 239 1354
E2	Fire	52	+216 715 258 6245
E3	Lost child	69	+48 881 110 7673
E4	Loss of personal money/luggage	4	+30 203 872 1839
E5	Earthquake	83	+93 460 585 8663
E6	Hurricanes	24	+92 943 238 2470
E7	Fire	5	+52 631 832 2744
E8	Lost child	34	+52 711 268 8608
E9	Loss of personal money/luggage	61	+232 370 643 5344
E10	Flood	71	+57 239 657 4379

Number of tuples: 80

Tour_detail

The screenshot shows the pgAdmin 4 interface with the 'tour_detail' table selected. The query in the Query Editor is:

```
1 select *
2 from tour_detail
```

The resulting table has three columns: tourpackage_id, place_id, and hotel_id. The data is as follows:

	tourpackage_id	place_id	hotel_id
1	1	26	H3
2	1	52	H61
3	1	62	H42
4	2	6	H4
5	2	71	H62
6	2	21	H43
7	2	79	H17
8	3	17	H5
9	3	41	H63
10	3	39	H44

Number of tuples: 375

Bus

The screenshot shows the pgAdmin 4 interface with the 'bus' table selected. The query in the Query Editor is:

```
1 select *
2 from bus
```

The resulting table has seven columns: bus_id, bus_type, arrival_time, departure_time, bonding_point, destination_location, and a timestamp column. The data is as follows:

	bus_id	bus_type	arrival_time	departure_time	bonding_point	destination_location	
1	bs1	Non-AC sleeper	7:48 AM	6:45 AM	Stryków	Tangwang	2023-10-15 10:58:42.000
2	bs2	AC seater	5:50 AM	2:46 AM	Rahayu Dua	Lahad Datu	2023-10-15 10:58:42.000
3	bs3	Non-AC seater-sleeper	5:06 AM	8:12 PM	Landvetter	Dziadkowice	2023-10-15 10:58:42.000
4	bs4	Non-AC seater	11:15 PM	4:46 AM	La Labor	Vila Chã	2023-10-15 10:58:42.000
5	bs5	AC sleeper	10:31 AM	7:11 AM	Blois	Chang'an	2023-10-15 10:58:42.000
6	bs6	Non-AC seater-sleeper	12:55 AM	5:11 AM	Lyasny	Washington	2023-10-15 10:58:42.000
7	bs7	AC seater	9:43 AM	6:50 PM	Przewóz	Lille	2023-10-15 10:58:42.000
8	bs8	Non-AC sleeper	12:19 PM	9:45 AM	São Mateus do Maranhão	Kantemirovka	2023-10-15 10:58:42.000
9	bs9	Non-AC seater	8:11 AM	5:07 PM	Sortavalá	Götene	2023-10-15 10:58:42.000
10	bs10	AC seater	11:31 AM	11:41 PM	Waiak	Shanhou	2023-10-15 10:58:42.000

Number of tuples: 100

Train

The screenshot shows the pgAdmin 4 interface with the 'Browser' tab selected. In the left sidebar, under 'Tables (17)', the 'train' table is highlighted. The main area displays a query editor with the following SQL code:

```
1 select *
2 from train
```

Below the query editor is a data grid showing 10 tuples from the 'train' table. The columns are: train_Id, train_type, arrival_time, departure_time, boarding_point, and destination_location. The data is as follows:

	train_Id	train_type	arrival_time	departure_time	boarding_point	destination_location
1	t1	AC Express	10:57 AM	7:38 PM	Dazaifu	Almelo
2	t2	Intercity Express	12:04 PM	10:07 AM	Lame	Meliti
3	t3	Express	6:59 AM	9:40 PM	Tandayag	Tamandaré
4	t4	Superfast Express	1:56 PM	9:45 AM	Providencia	Baizhang
5	t5	Passenger	6:54 AM	11:16 PM	Pontal	Huangtan
6	t6	Luxury Trains	1:20 PM	9:47 AM	Galügäh	Älvbyn
7	t7	Mail	1:27 PM	10:51 AM	Los Angeles	Gareba
8	t8	Double Decker Express	5:48 AM	9:48 AM	Hoàn Kiếm	Bamenda
9	t9	AC Express	6:19 AM	6:07 PM	KapuskaSing	Adstock
10	t10	Intercity Express	2:51 AM	6:20 AM	Dauanoviriva	Channav

Number of tuples: 150

Flight

The screenshot shows the pgAdmin 4 interface with the 'Browser' tab selected. In the left sidebar, under 'Tables (17)', the 'flight' table is highlighted. The main area displays a query editor with the following SQL code:

```
1 select *
2 from flight
```

Below the query editor is a data grid showing 10 tuples from the 'flight' table. The columns are: flight_Id, flight_type, arrival_time, departure_time, boarding_point, and destination_location. The data is as follows:

	flight_Id	flight_type	arrival_time	departure_time	boarding_point	destination_location
1	f1	Economy class	5:13 PM	2:05 PM	Raškovice	Shinpokh
2	f2	Business class	10:11 PM	2:24 AM	Nelson	Matamey
3	f3	Economy class	6:00 AM	8:57 PM	Magay	Ratenggoji
4	f4	Premium economy class	1:33 AM	6:56 AM	Homa Bay	Acchu
5	f5	Second class	1:57 AM	12:16 AM	Kupiskis	Majie
6	f6	Business class	4:23 PM	3:16 PM	Luoshan	Cimenga
7	f7	First class	2:38 PM	1:11 AM	Al Qadarif	Busungblu
8	f8	Business class	7:00 AM	3:02 AM	Bantawora	Novosemyokino
9	f9	Economy class	7:15 AM	6:38 PM	Klina	Radomir
10	f10	Premium economy class	1:04 AM	2:11 AM	Ianuan	Yannian

Number of tuples: 100

bus_ticket_confirm

The screenshot shows the pgAdmin 4 interface with the 'Browser' tab selected. Under the 'Tables' section, the 'bus_ticket_confirm' table is highlighted. The 'Query Editor' tab contains the SQL query: 'select * from bus_ticket_confirm'. The results table has columns: bus_id, payment_id, and booking_id. The data shows 10 rows of ticket confirmations.

	bus_id	payment_id	booking_id
1	bs15	P50	B50
2	bs20	P51	B51
3	bs25	P52	B52
4	bs16	P53	B53
5	bs21	P54	B54
6	bs26	P55	B55
7	bs17	P56	B56
8	bs23	P57	B57
9	bs27	P58	B58
10	bs18	P59	B59

Number of tuples: 79

train_ticket_confirm

The screenshot shows the pgAdmin 4 interface with the 'Browser' tab selected. Under the 'Tables' section, the 'train_ticket_confirm' table is highlighted. The 'Query Editor' tab contains the SQL query: 'select * from train_ticket_confirm'. The results table has columns: train_id, payment_id, and booking_id. The data shows 10 rows of ticket confirmations.

	train_id	payment_id	booking_id
1	t50	P16	B16
2	t51	P17	B17
3	t52	P18	B18
4	t53	P19	B19
5	t54	P20	B20
6	t10	P21	B21
7	t11	P22	B22
8	t12	P23	B23
9	t13	P24	B24
10	t14	P25	B25

Number of tuples: 73

flight_ticket_confirm

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under the 'Tables (17)' section, including tables like admin, booking, bus, bus_ticket_confirm, cancellation, customer, emergency, flight, flight_ticket_confirm, group_member, hotel, payment, tour_detail, tour_package, tourist_places, train, and train_ticket_confirm. The main query editor window contains the following SQL code:

```
1 select *
2 from flight_ticket_confirm
```

The results pane shows a table with three columns: flight_id, payment_id, and booking_id. The data is as follows:

	flight_id	payment_id	booking_id
1	f2	P161	B161
2	f36	P162	B162
3	f8	P163	B163
4	f78	P164	B164
5	f32	P165	B165
6	f31	P166	B166
7	f91	P167	B167
8	f67	P168	B168
9	f100	P169	B169
10	f56	P170	R170

Number of tuples: 48

6.4 40 SQL Queries with Snapshots of output of each query

Q1. List all the tourist place's details that are safe to visit.

```
select *
from tourist_places
where safety_decription='safe';
```

place_id	place_name	place_type	place_description
1	The Taj Mahal, Agra	Beach	The name Taj Mahal, translates to the 'Crown Palace' and is one of the most famous and marvelous buildings
2	Jaisalmer Fort, Jaisalmer	Cultural quarter	Located amidst the golden sands of the Thar Desert, the Jaisalmer Fort was built by the Bhati Rajput King Rav
3	Backwaters, Kerala	Mountain	The Backwaters of Kerala are essentially a group of 5 lagoons that are linked by natural and manmade canals
4	Jama Masjid, Delhi	Beach	Known formally as 'Masjid-i-Jahan-Numa', the Jama Masjid was built by the Mughal Emperor Shah Jahan in th
5	Ajanta and Ellora	Cultural quarter	Ajanta and Ellora Caves in Aurangabad are rock cut Buddhist caves that date back as far as the 2nd century B
6	Amber Fort, Jaipur	Mountain	The medieval town of Amer was the former capital of the Kachwaha Rajputs, with the fort serving as their sea
7	Meenakshi Temple	Beach	Dedicated to the Hindu deities Parvati and Lord Shiva, Meenakshi Amman Temple in Madurai is a typical Dravi
8	City Palace, Udaipur	Cultural quarter	After leaving Chittor due to its strategic disadvantage, the Sisodiya Rajputs established their new kingdom as
9	Tawang Monastery	Mountain	Known as the largest monastery in India, Tawang Monastery was founded in the 17th century, under the order
10	Palitana Temples	Beach	The Jain temples of Palitana are located in the Bhavnagar district of Gujarat and are known for its marvelous i
11	Rani Ki Vav, Patan	Cultural quarter	Constructed by the king of the Solanki Dynasty in Gujarat, Rani Ki Vav in Patan, is a stone step-well that is know
12	Lotus Temple, Delhi	Cultural quarter	The only Bahai place of worship in India, the Lotus temple is an exquisite work of architecture built in 1986. Ti
13	Borra Caves	Mountain	The Borra Caves in Vishakhapatnam are arguably the deepest cave in India that goes to a depth of 80 meters.
14	Mysore Palace, Mysore	Beach	The Mysore palace was built between 1897 and 1912, by the Wodeyar Kings of Mysore and is known for its re
15	Bhimbetka Rock Shelters	Cultural quarter	A UNESCO World Heritage Site, the Bhimbetka Rock Shelters are known to be one of the oldest archeological .

Successfully run. Total query runtime: 336 msec. 3

Q2. Display customer_id(leader_id) who have a group of more than 5 members.

```
select leader_id
from group_member
group by leader_id
having count(leader_id)>=5;
```

```
postgres/postgres@PostgreSQL:14 ~
Query Editor  Query History
Scratch Pad x

1 select leader_id
2   from group_member
3  group by leader_id
4 having count(leader_id)>=5;
```

Data Output Explain Messages Notifications

	leader_id
1	35
2	6
3	68
4	108
5	124
6	286
7	123

Q3. Print the customer's name, id, contact no. and booking id who have booked their package in the year 2021.

```
select customer.c_id,booking.booking_id, customer.customername, customer.contact_no
from customer,booking
where customer.c_id = booking.c_id and booking_date like '%2021%';
```

```
postgres/postgres@PostgreSQL:14 ~
Query Editor  Query History
Scratch Pad x

1 select customer.c_id,booking.booking_id, customer.customername, customer.contact_no
2  from customer,booking
3 where customer.c_id = booking.c_id and booking_date like '%2021%';
```

Data Output Explain Messages Notifications

	c_id	booking_id	customername	contact_no
1	178	B1	Andrei Mullany	+1 269 618 1498
2	152	B2	Tomaso Brosini	+84 194 204 3247
3	234	B3	Sibylle Cordeix	+358 185 503 8447
4	285	B4	Vincenty Rosenvasser	+225 408 953 0296
5	67	B5	Gar Girling	+504 181 108 6454
6	204	B6	Yankee Carlick	+46 113 712 4655
7	168	B7	Amberly Piggins	+62 215 327 8241
8	29	B8	Lanie Finci	+1 173 871 0721
9	167	B9	Kevin McReynold	+57 484 660 0748
10	285	R10	Vincenty Rosenvasser	+225 408 953 0296

Q4. Show all the customer lists who booked the package of type Adventure.

```
select customer.customername  
from customer,booking,tour_package  
where customer.c_id = booking.c_id and booking.package_id = tour_package.package_id and package_type =  
'Adventure';
```

The screenshot shows a PostgreSQL query editor interface. The query window contains the following SQL code:

```
1 select customer.customername  
2 from customer,booking,tour_package  
3 where customer.c_id = booking.c_id and booking.package_id = tour_package.package_id and package_type = 'Adventure';
```

The results pane displays a table titled "customername" with 10 rows of data:

	customername
1	Vincenty Rosenvasser
2	Denice Lemin
3	Donnie Strette
4	Sauder Collyear
5	Chadd Strangwood
6	Jana Erett
7	Halette Reinhard
8	Natassia Lethbridge
9	Aylmar Fulker
10	Naomi Yaakov

Q5. List all customer's details who had to cancel their payment.

```
select customer.*  
from customer, (select payment.c_id from payment, cancellation where cancellation.payment_id =  
payment.payment_id) as cp  
where cp.c_id = customer.c_id
```

TMS/postgres@PostgreSQL 13 ▾

Query Editor Query History

```
1 select customer.*  
2 from customer, (select payment.c_id from payment, cancellation where cancellation.payment_id = payment.payment_id) as cp  
3 where cp.c_id = customer.c_id
```

Data Output Explain Messages Notifications

c_id	[PK] integer	customername	character varying (30)	c_password	character varying (30)	gender	character varying (6)	contact_no	character varying (30)	address	character varying (300)	pincode	integer
1	234	Sibylle Cordex	xhCFjHJD	Male	+358 185 503 8447	9 Sunnyside Place		446008					
2	285	Vincenty Rosenvasser	O0WNm2elDezN	Male	+225 408 953 0296	2 Merry Plaza		710731					
3	67	Gar Girling	Y0m7NSVb	Male	+504 181 108 6454	50758 Green Alley		652396					
4	204	Yankee Carlick	Hap3koA491c	Male	+46 113 712 4655	5521 Sundown Court		265613					
5	168	Amberly Piggins	fR49ABL	Female	+62 215 327 8241	0567 Mosinee Junction		468814					
6	29	Lanie Finci	amTnmwW	Female	+1 173 871 0721	452 Haas Trail		489094					
7	167	Kevin McReynold	rrJgLN	Female	+57 484 660 0748	578 Columbus Crossing		189032					
8	35	Denice Lemin	7VecsDSFKm	Female	+63 823 182 8101	30069 Goodland Point		585982					
9	239	Eada Jillins	bOvZfUzs3j	Male	+7 395 904 1061	5 Bluejay Place		151170					
10	99	Carolan Dugmore	86P0zy	Male	+86 928 164 1549	43 Springview Plaza		813155					
11	226	Erina Delamaine	JXFRO8	Female	+86 947 106 8364	57 Westend Alley		161451					

Q6. List all available flights details from Pune to Nashik on 14-10-2021.

```
select *  
from flight  
where boarding_point='Pune' and destination_location='Nashik' and arrival_time like '14-10-2021%'
```

TMS/postgres@PostgreSQL 13 ▾

Query Editor Query History

```
1 select *  
2 from flight  
3 where boarding_point='Pune' and destination_location='Nashik' and arrival_time like '14-10-2021%'
```

Data Output Explain Messages Notifications

flight_id	[PK] character varying (20)	flight_type	character varying (30)	arrival_time	character varying (50)	departure_time	character varying (50)	boarding_point	character varying (30)	destination_location	character varying (30)
1	f20	Business class		14-10-2021 16:10		2021-10-14 17:10:48 UTC		Pune		Nashik	

Q7. Show all emergency contact numbers at Jaisalmer Fort.

```
select contact
from tourist_places natural join emergency
where place_name='Jaisalmer Fort, Jaisalmer';
```

The screenshot shows a PostgreSQL query editor window. The title bar says "postgres/postgres@PostgreSQL 14". The Query Editor tab is selected, displaying the SQL query:

```
1 select contact
2 from tourist_places natural join emergency
3 where place_name='Jaisalmer Fort, Jaisalmer';
```

Below the query, there are tabs for Data Output, Explain, Messages, and Notifications. The Data Output tab is selected, showing the results of the query:

contact
+52 631 832 2744
+62 975 760 9510

Q8. Find the total amount of profit. (total_profit = sum_of_payment * 20%).

```
select 0.2*sum(amount)
from payment
```

The screenshot shows a PostgreSQL query editor window. The title bar says "TMS/postgres@PostgreSQL 13". The Query Editor tab is selected, displaying the SQL query:

```
1 select 0.2*sum(amount)
2 from payment
```

Below the query, there are tabs for Data Output, Explain, Messages, and Notifications. The Data Output tab is selected, showing the results of the query:

?column?
numeric
466292.2

Q9. List the details of the tour package which is suitable for ‘Family’ and is more than 5 days long.

```
select *
from tour_package
where no_of_days >= 5 and package_type = 'Family';
```

The screenshot shows a PostgreSQL query editor window. The query in the editor is:

```
1 select *
2 from tour_package
3 where no_of_days >= 5 and package_type = 'Family';
4
```

Below the editor, the "Data Output" tab is selected, showing the results of the query:

package_id	package_name	package_type	no_of_days	package_amount	package_description
1	9 Family Tourism Package	Family	14	23999	The family tourism in India offers all kind of activities like visiting the cultural destinations, a cruise ride, enjoying the scenic beau
2	98 Family Tourism Package	Family	5	4569	The family tourism in India offers all kind of activities like visiting the cultural destinations, a cruise ride, enjoying the scenic beau

Q10. List the name, description, and no of days of the tour package which amount is between 9000 to 13000.

```
select package_name, no_of_days ,package_description
from tour_package
where package_amount between 9000 and 13000;
```

The screenshot shows a PostgreSQL query editor window. The query in the editor is:

```
1 select package_name, no_of_days ,package_description
2 from tour_package
3 where package_amount between 9000 and 13000;
4
```

Below the editor, the "Data Output" tab is selected, showing the results of the query:

package_name	no_of_days	package_description
1 Medical Tourism Package	13	The medical tourism in India, for the past few years, has gained the attention of the people around the world. The ancient medical science – Ayurveda and other alternative medical practices w
2 Pilgrimage Tourism Package	10	The major reason that attracts tourists to India is the traditional temple architecture, art forms and rituals performed. Varanasi is the major hub for all the devotees to explore the Hindu history &
3 Medical Tourism Package	14	The medical tourism in India, for the past few years, has gained the attention of the people around the world. The ancient medical science – Ayurveda and other alternative medical practices w
4 Eco Tourism Package	14	Although, ecotourism is more of a travel philosophy and as India has a rich ecology, it attracts a lots of tourists. Being a non destination-oriented project, some of the eco tourism destinations in
5 Wellness Tourism Package	15	Renowned as the cradle of medical science like Ayurveda, Yoga, Naturopathy, India has a philosophy of healing the patient with the natural ways. The wellness tourism in India offers a host of tr
6 Cultural Tourism Package	11	The social richness of the country draws the visitors from every corner of the world to witness sheer celebrations. Grand monuments delineating structural brightness of past time alongside th
7 Eco Tourism Package	8	Although, ecotourism is more of a travel philosophy and as India has a rich ecology, it attracts a lots of tourists. Being a non destination-oriented project, some of the eco tourism destinations in
8 Wildlife Tourism Package	12	If you are a wildlife explorer or photographer and wish to learn and take pictures of the untamed wildlife, then pack your bags and fly over to India. You can choose the right wildlife tourism pack
9 Wellness Tourism Package	15	Renowned as the cradle of medical science like Ayurveda, Yoga, Naturopathy, India has a philosophy of healing the patient with the natural ways. The wellness tourism in India offers a host of tr
10 Adventure Tourism Package	15	Adventure tourism, in general, is defined as the type of tourism that encourages an individual to come out of his/her comfort zone by engaging in thrilling physical, natural or cultural activities.
11 Adventure Tourism Darjeeling	9	Adventure tourism in general is defined as the type of tourism that encourages an individual to come out of his/her comfort zone by engaging in thrilling physical, natural or cultural activities.

Q11. List all places name where the weather is Partially cloudy.

```
select place_name  
from tourist_places  
where weather='Cloudy';
```

The screenshot shows a PostgreSQL query editor interface. The top bar displays the connection information: postgres/postgres@PostgreSQL 14. Below the bar, there are tabs for 'Query Editor' and 'Query History'. On the right side, there is a 'Scratch Pad' tab with a close button. The main area contains the SQL query:

```
1 select place_name  
2 from tourist_places  
3 where weather='Cloudy';
```

Below the query, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected, showing a table with the results of the query:

place_name
Pangong Lake, Ladakh
Akshardham Temple, Delhi
Khajuraho Temples
City Palace, Jaipur
Bangalore Palace
Chhatrapati Sangrahalay
The Great Living Chola Temples
St. Paul's Cathedral
Sariska Tiger Reserve, Alwar
Jainarh Fort, Jainur

Q12. List all customer's cancellation details for which cancellation is being processeing.

```
select cancellation  
from cancellation  
where cancel_status = 'no'
```

The screenshot shows a PostgreSQL query editor window. The query in the editor is:

```

1 select cancellation.*  

2 from cancellation  

3 where cancel_status = 'no';

```

The results section displays the data output:

cancellation_id	booking_id	cancel_status	payment_id
c2	B4	no	P4
c3	B5	no	P5
c5	B7	no	P7
c8	B17	no	P17
c9	B18	no	P18
c12	B21	no	P21
c13	B22	no	P22
c14	B23	no	P23
c18	B27	no	P27
c19	B28	no	P28
c22	B31	no	P31

Q13. Show how many percent of customers traveled by flight.

```

select round(booking.flight_booking*100.0/bookings.total_booking,2)
from (select count(booking_id)as flight_booking from flight_ticket_confirm ) as booking ,(select
count(booking_id)as total_booking from booking) as bookings

```

The screenshot shows a PostgreSQL query editor window. The query in the editor is:

```

1 select round(booking.flight_booking*100.0/bookings.total_booking,2)  

2 from (select count(booking_id)as flight_booking from flight_ticket_confirm ) as booking ,(select
count(booking_id)as total_booking from booking) as bookings
3

```

The results section displays the data output:

round
24.00

Q14. Show how many hotels are in the resorts.

```
select count(hotel_type)
from hotel
where hotel_type = 'Resorts'
```

The screenshot shows the pgAdmin 4 interface. On the left is the 'Browser' pane, which displays a tree view of database objects. Under the 'Tables' section, there is a node for 'hotel' which is expanded, showing columns like 'hotel_id', 'hotel_name', etc. In the center is the 'Query Editor' pane, which contains the following SQL code:

```
1 select count(hotel_type)
2 from hotel
3 where hotel_type = 'Resorts'
```

Below the query editor, the 'Data Output' tab is selected, showing the results of the query:

count	bigint
1	13

Q15. Give all tour package details which are more than 3 days long and whose package amount is less than 10000.

```
select *
from tour_package
where package_amount < 10000 and no_of_days > 3
```

The screenshot shows a PostgreSQL query editor window. The query in the editor is:

```

1 select *
2 from tour_package
3 where package_amount < 10000 and no_of_days > 3

```

The results are displayed in a table titled "Data Output". The columns are:

package_id	package_name	package_type	no_of_days	package_amount	package_description
1	Wildlife Tourism Package	Wildlife	8	6400	If you are a wildlife explorer or photographer and wish to learn and take pictures of the untamed wildlife, then pack your bags and head to India.
2	Pilgrimage Tourism Package	Pilgrimage	10	9000	The major reason that attracts tourists to India is the traditional temple architecture, art forms and rituals performed. Varanasi, the holiest city of India, is a must-visit destination.
3	Cultural Tourism Package	Cultural	7	6900	The social richness of the country draws the visitors from every corner of the world to witness sheer celebrations. Grand Indian festivals like Diwali, Holi, and Dussehra are a sight to behold.
4	Wellness Tourism Package	Wellness	10	6999	Renowned as the cradle of medical science like Ayurveda, Yoga, Naturopathy, India has a philosophy of healing the patient through holistic approaches.
5	Pilgrimage Tourism Package	Pilgrimage	4	1599	The major reason that attracts tourists to India is the traditional temple architecture, art forms and rituals performed. Varanasi, the holiest city of India, is a must-visit destination.
6	Cruise Tourism Package	Cruise	5	3999	Cruise tourism permits tourists to explore beautiful riverside villages, undisturbed sacred islands and sights that will captivate your imagination.
7	Adventure Tourism Package	Adventure	5	2399	Adventure tourism, in general, is defined as the type of tourism that encourages an individual to come out of his/her comfort zone and experience new challenges.
8	Wildlife Tourism Package	Wildlife	7	4599	If you are a wildlife explorer or photographer and wish to learn and take pictures of the untamed wildlife, then pack your bags and head to India.
9	Medical Tourism Package	Medical	13	8999	The medical tourism in India, for the past few years, has gained the attention of the people around the world. The ancient Indian medical system, Ayurveda, offers unique treatments for various health conditions.
10	Pilgrimage Tourism Package	Pilgrimage	8	7799	The major reason that attracts tourists to India is the traditional temple architecture, art forms and rituals performed. Varanasi, the holiest city of India, is a must-visit destination.
11	Eco Tourism Package	Eco	11	8200	Although environmentalism is more of a lifestyle than a tourism category, it is an integral part of sustainable tourism.

A green status bar at the bottom right indicates: "Successfully run. Total query runtime: 80 msec. 42 rows affected."

Q16. Show how many packages are available of type adventure.

```

select count(package_id)
from tour_package
where package_type='Adventure';

```

The screenshot shows a PostgreSQL query editor window. The query in the editor is:

```

1 select count(package_id)
2 from tour_package
3 where package_type='Adventure';

```

The results are displayed in a table titled "Data Output". The column is:

count
11

Q17. Show list of all hotels name that is from Goa.

```
select hotel_name
from hotel
where hotel_address = 'goa'
```

The screenshot shows a PostgreSQL query editor window titled "TMS/postgres@PostgreSQL 13". The query in the editor is:

```
1 select hotel_name
2 from hotel
3 where hotel_address = 'goa'
```

The results are displayed in a table under the "Data Output" tab:

hotel_name
Grand Hyatt Goa
Planet Hollywood Goa
Ronil Royale
Taj Fort Aguada Resort

Q18. List the places name which place type is Mountain.

```
select place_name
from tourist_places
where place_type = 'Mountain';
```

The screenshot shows a PostgreSQL query editor window titled "postgres/postgres@PostgreSQL 14". The query in the editor is:

```
1 select place_name
2 from tourist_places
3 where place_type = 'Mountain';
```

The results are displayed in a table under the "Data Output" tab:

place_name
Backwaters, Kerala
Amber Fort, Jaipur
Tawang Monastery
Borra Caves
Lingaraja Temple Complex
Fatehpur Sikri, Agra
Junagarh Fort, Bikaner
Ramanathaswamy Temple
Munnar Hills, Idukki
Marine National Park

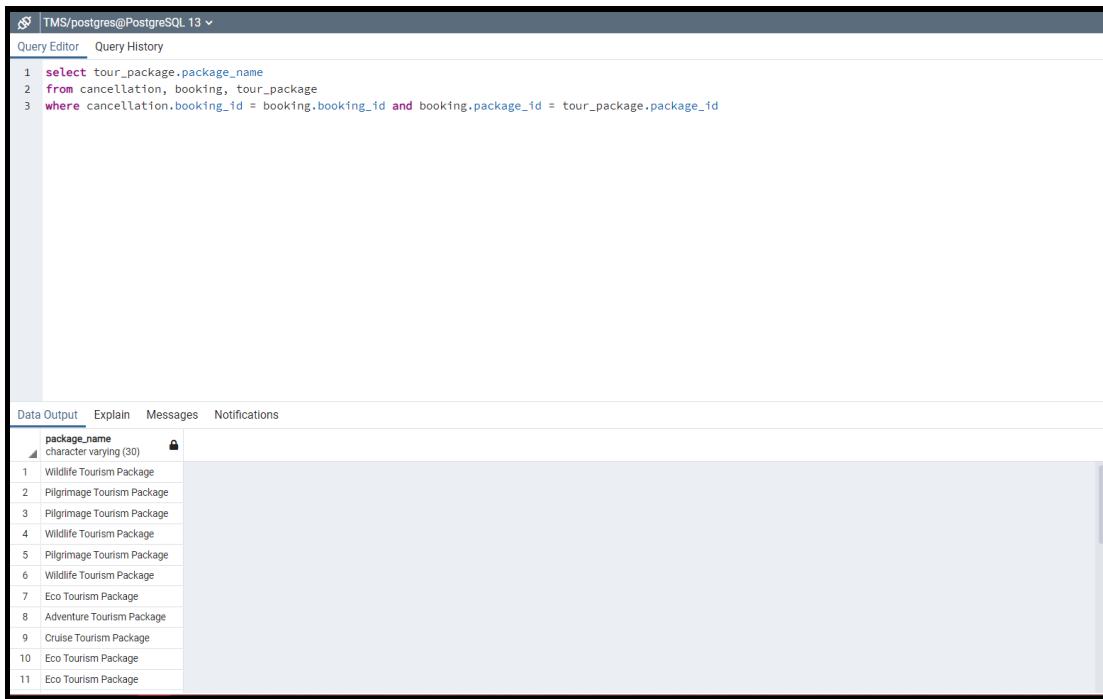
Q19. Show all booking information of payments which amount is greater than 20000.

```
select booking.*  
from booking natural join payment  
where amount > 20000
```

	booking_id	c_id	contact_no	booking_date	package_id
1	B9	167	+46 551 495 2821	2/19/2021	94
2	B10	285	+46 877 523 9615	03-08-2021	54
3	B20	226	+421 934 774 0571	10/20/2021	34
4	B24	1	+47 556 705 6006	5/15/2021	19
5	B26	254	+86 435 422 9678	3/24/2021	70
6	B39	250	+86 365 507 0773	12/31/2020	21
7	B49	82	+86 858 289 3232	07-07-2021	26
8	B50	126	+1 915 708 2374	06-11-2021	29
9	B53	255	+52 806 675 8500	4/18/2021	50
10	B55	273	+1 405 415 6068	2/13/2021	93
11	B56	42	+55 493 822 1850	1/22/2021	67

Q20. Show the list of all canceled package_name.

```
select tour_package.package_name  
from cancellation, booking, tour_package  
where cancellation.booking_id = booking.booking_id and booking.package_id = tour_package.package_id
```



The screenshot shows a PostgreSQL query editor window titled 'TMS/postgres@PostgreSQL 13'. The query in the editor is:

```

1 select tour_package.package_name
2 from cancellation, booking, tour_package
3 where cancellation.booking_id = booking.booking_id and booking.package_id = tour_package.package_id

```

The results are displayed in a table under the 'Data Output' tab:

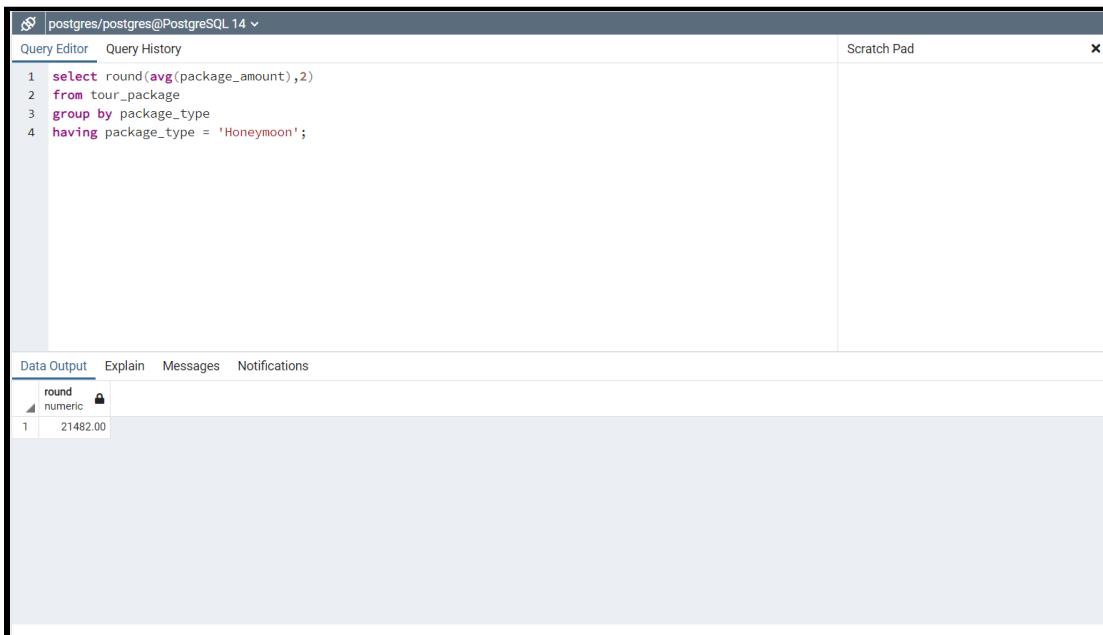
package_name
Wildlife Tourism Package
Pilgrimage Tourism Package
Pilgrimage Tourism Package
Wildlife Tourism Package
Pilgrimage Tourism Package
Wildlife Tourism Package
Eco Tourism Package
Adventure Tourism Package
Cruise Tourism Package
Eco Tourism Package
Eco Tourism Package

Q21. Show the average amount of Honeymoon packages?

```

select round(avg(package_amount),2)
from tour_package
group by package_type
having package_type = 'Honeymoon';

```



The screenshot shows a PostgreSQL query editor window titled 'postgres/postgres@PostgreSQL 14'. The query in the editor is:

```

1 select round(avg(package_amount),2)
2 from tour_package
3 group by package_type
4 having package_type = 'Honeymoon';

```

The results are displayed in a table under the 'Data Output' tab:

round
21482.00

Q22. Show all admin names.

```
select admin_name  
from admin
```

The screenshot shows a PostgreSQL query editor window. The title bar says "postgres/postgres@PostgreSQL 14". The Query Editor tab is active, displaying the SQL query:

```
1 select admin_name  
2 from admin  
3
```

The Data Output tab is selected, showing the results of the query:

admin_name
Clementius Iacofa
Joyous Lundon
Vincenz Bramstom
Bobbette Devin
Susannah Janous
Wyndham Mallinar
Natala Kolin
Joan Kolakovic
Hartwell Coulbeck

Q23. Display all member names of customer id 10.

```
select m_name  
from group_member  
where leader_id=6;
```

The screenshot shows a PostgreSQL query editor window. The title bar says "postgres/postgres@PostgreSQL 14". The Query Editor tab is active, displaying the SQL query:

```
1 select m_name  
2 from group_member  
3 where leader_id=6;  
4
```

The Data Output tab is selected, showing the results of the query:

m_name
Odella Lewing
Stanislaus Tasseler
Sherwin Baggarley
Cory Bapty
Dennie Plose

Q24. Show the ratio of male and female customers.

with girls(value) as

(select count(c_id) from customer where gender='Female'),

boys(value) as

(select count(c_id) from customer where gender='Male')

```
select round(1.0*boys.value / girls.value , 2)
from boys, girls;
```

The screenshot shows a PostgreSQL query editor window. The title bar says "postgres/postgres@PostgreSQL 14". The main area is the "Query Editor" tab, which contains the following SQL code:

```
1 with girls(value) as
2   (select count(c_id) from customer where gender='Female'),
3
4   boys(value) as
5   (select count(c_id) from customer where gender='Male')
6
7 select round(1.0*boys.value / girls.value , 2)
8 from boys,girls;
9
10
11
```

Below the code, there are tabs for "Data Output", "Explain", "Messages", and "Notifications". The "Data Output" tab is selected, showing a single row of results:

round	numeric
1	1.04

Q25. Display the highest paid amount till now.

```
select max(amount)
from payment
```

The screenshot shows a PostgreSQL query editor window titled "TMS/postgres@PostgreSQL 13". The query in the editor is:

```
1 select max(amount)
2 from payment
```

The results pane shows a single row of data:

	max
1	51999

Q26. Show how many customers have canceled their booking.

```
select count(cancellation_id)
from cancellation
where cancel_status = 'yes';
```

The screenshot shows a PostgreSQL query editor window titled "postgres/postgres@PostgreSQL 14". The query in the editor is:

```
1 select count(cancellation_id)
2 from cancellation
3 where cancel_status = 'yes';
```

The results pane shows a single row of data:

	count
1	15

Q27. Show the arrival time and departure time of the train whose train id is t46.

```
select arrival_time,departure_time
from train
where train_id='t46';
```

The screenshot shows a PostgreSQL query editor window. The title bar says "postgres/postgres@PostgreSQL 14". The main area contains the following SQL code:

```
1 select arrival_time,departure_time
2 from train
3 where train_id='t46';
4
5
6
```

Below the code, there are tabs for "Data Output", "Explain", "Messages", and "Notifications". The "Data Output" tab is selected, showing a table with two columns: "arrival_time" and "departure_time". The data is as follows:

	arrival_time	departure_time
1	character varying (50)	character varying (50)
1	2:18 PM	3:50 PM

Q28. List all casino hotels' names that are situated in Rajasthan.

```
select hotel_name
from hotel
where hotel_type = 'Casino hotels' and hotel_address = 'Rajasthan'
```

The screenshot shows a PostgreSQL query editor window. The title bar says "TMS/postgres@PostgreSQL 13". The main area contains the following SQL code:

```
1 select hotel_name
2 from hotel
3 where hotel_type = 'Casino hotels' and hotel_address = 'Rajasthan'
```

Below the code, there are tabs for "Data Output", "Explain", "Messages", and "Notifications". The "Data Output" tab is selected, showing a table with one column: "hotel_name". The data is as follows:

hotel_name
Bissau Palace Hotel
LMB Hotel, Jaipur
Umaid Bhawan Palace

Q29. Show how many percent of customers traveled by train.

```
select round(booking.train_booking*100.0/bookings.total_booking,2)
from (select count(booking_id)as train_booking from train_ticket_confirm ) as booking ,(select
count(booking_id)as total_booking from booking) as bookings
```

The screenshot shows a PostgreSQL query editor window titled "TMS/postgres@PostgreSQL 13". The query is:

```
1 select round(booking.train_booking*100.0/bookings.total_booking,2)
2 from (select count(booking_id)as train_booking from train_ticket_confirm ) as booking ,(select
3 count(booking_id)as total_booking from booking) as bookings
```

The results are displayed in a table:

	round	numeric
1	36.50	

Q30. List all the package names which are booked in october month.

```
select package_name
from tour_package, (select * from booking where booking_date like '10%' or booking_date like '10%') as
oct_booking
where oct_booking.package_id = tour_package.package_id
```

The screenshot shows a PostgreSQL query editor window titled "TMS/postgres@PostgreSQL 13". The query is:

```
1 select package_name
2 from tour_package, (select * from booking where booking_date like '10%' or booking_date like '10%') as oct_booking
3 where oct_booking.package_id = tour_package.package_id
```

The results are displayed in a table:

package_name	character varying (30)
1	Cultural Tourism Package
2	Family Tourism Package
3	Cruise Tourism Package
4	Cruise Tourism Package
5	Medical Tourism Package
6	Medical Tourism Package
7	Pilgrimage Tourism Package
8	Eco Tourism Package
9	Eco Tourism Package
10	Wellness Tourism Package
11	Cruise Tourism Package

A green message bar at the bottom right indicates: "Successfully run. Total query runtime: 214 msec. 17 rows affected."

Q31. List all the tourist places details which are beach and safe to visit and their weather is Partially cloudy.

```
select *
from tourist_places
where safety_description='safe' and place_type='Beach' and weather='Partially cloudy';
```

The screenshot shows a PostgreSQL query editor window titled 'TMS/postgres@PostgreSQL 13'. The 'Query Editor' tab is selected. The query entered is:

```
1 select *
2 from tourist_places
3 where safety_description='safe' and place_type='Beach' and weather='Partially cloudy';
```

Below the query, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected, displaying the results of the query:

place_id	place_name	place_type	place_description
1	The Taj Mahal, Agra	Beach	The name Taj Mahal, translates to the 'Crown Palace' and is one of the most famous and marvelous buildings in India.
2	Jama Masjid, Delhi	Beach	Known formally as 'Masjid-i-Jahan-Numa', the Jama Masjid was built by the Mughal Emperor Shah Jahan in the mid-17th century.
3	Meenakshi Temple	Beach	Dedicated to the Hindu deities Parvati and Lord Shiva, Meenakshi Amman Temple in Madurai is a typical Dravidian architecture.
4	Palitana Temples	Beach	The Jain temples of Palitana are located in the Bhavnagar district of Gujarat and are known for its marvelous rock cut architecture.
5	Mysore Palace, Mysore	Beach	The Mysore palace was built between 1897 and 1912, by the Wodeyar Kings of Mysore and is known for its remarkable architecture.
6	Jallianwala Bagh, Amritsar	Beach	One of the major sites of the Indian struggle for Independence, the Jallianwala Bagh is a monument of National importance.
7	Jantar Mantar Observatory	Beach	Home to the world's largest stone sundial, Jantar Mantar in Jaipur is one of the 5 astronomical observatories in India.
8	Belur Math, Belur	Beach	Marvelously combining artistic elements from Hindu, Islamic and European traditions, Belur Math is also the headquarter of the Ramakrishna Mission.
9	Gagron Fort, Jhalawar	Beach	Bounded on three sides by Rivers, the Gagron Fort is also a UNESCO World Heritage site and is located in Jhalawar.
10	Nainital Lake, Nainital	Beach	Nainital Lake is located in the Nainital District of Uttarakhand and serves as the major tourist attraction in the hill town.
11	Bishnupur Temples, Bankura	Beach	Initially a small region under the Gupta Empire, the town of Bishnupur flourished during the reign of the Malla Kings.

Q32. Show how many percent of customers traveled by bus.

```
select round(booking.bus_booking*100.0/bookings.total_booking,2)
from (select count(booking_id) as bus_booking from bus_ticket_confirm ) as booking , (select
count(booking_id)as total_booking from booking) as bookings
```

The screenshot shows a PostgreSQL query editor window titled 'TMS/postgres@PostgreSQL 13'. The 'Query Editor' tab is selected. The query entered is:

```
1 select round(booking.bus_booking*100.0/bookings.total_booking,2)
2 from (select count(booking_id) as bus_booking from bus_ticket_confirm ) as booking , (select
count(booking_id)as total_booking from booking) as bookings
```

Below the query, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected, displaying the results of the query:

round
39.50

Q33. Give the package description of Wildlife and Cruise package type.

```
select distinct(package_name),package_description
from tour_package
where package_type='Wildlife' or package_type='Cruise'
```

The screenshot shows a PostgreSQL query editor window titled 'TMS/postgres@PostgreSQL 13'. The query is:

```
1 select distinct(package_name),package_description
2 from tour_package
3 where package_type='Wildlife' or package_type='Cruise'
```

The results are displayed in a table:

package_name	package_description
Wildlife Tourism Package	If you are a wildlife explorer or photographer and wish to learn and take pictures of the untamed wildlife, then pack your bags and fly over to India. You can choose the right wild
Cruise Tourism Package	Cruise tourism permits tourists to explore beautiful riverside villages, undisturbed sacred islands and sights that will captivate and intrigue their senses.

Q34. List all buses details from Mumbai to Agra.

```
select *
from bus
where boarding_point='Mumbai' and destination_location='Agra'
```

The screenshot shows a PostgreSQL query editor window titled 'TMS/postgres@PostgreSQL 13'. The query is:

```
1 select *
2 from bus
3 where boarding_point='Mumbai' and destination_location='Agra'
```

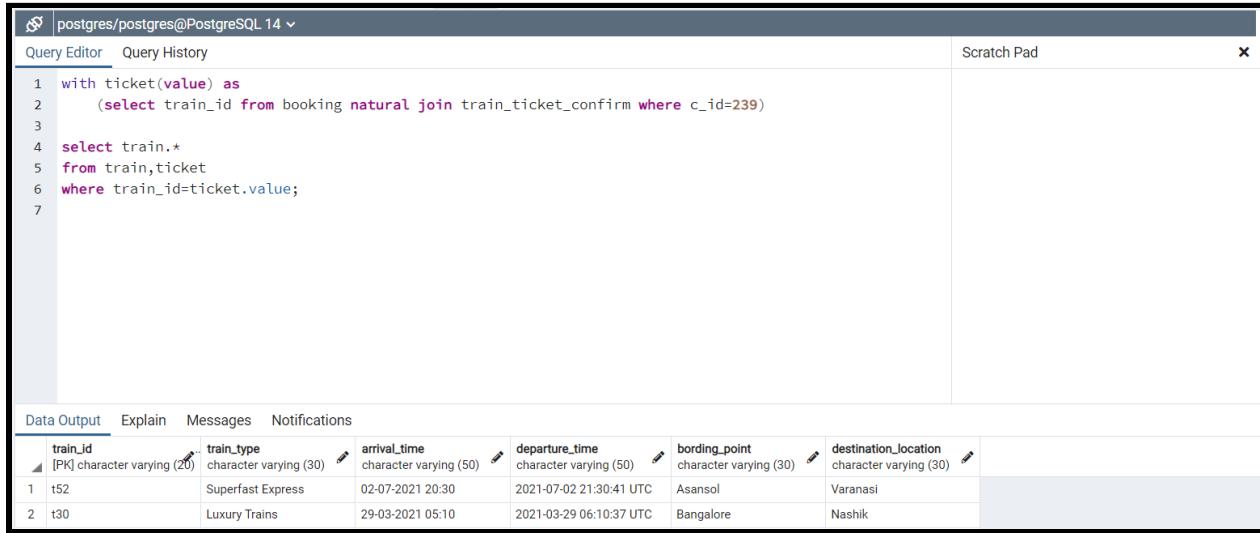
The results are displayed in a table:

bus_id	bus_type	arrival_time	departure_time	boarding_point	destination_location
bs6	Non-AC seater-sleeper	03-10-2021 21:00	2021-10-03 22:00:13 UTC	Mumbai	Agra
bs36	Non-AC sleeper	14-09-2021 02:12	2021-09-14 03:12:45 UTC	Mumbai	Agra
bs70	AC seater	27-06-2021 06:01	2021-06-27 07:01:50 UTC	Mumbai	Agra
bs91	Non-AC seater-sleeper	28-10-2021 06:58	2021-10-28 07:58:33 UTC	Mumbai	Agra
bs95	AC seater	21-12-2020 18:25	2020-12-21 19:25:23 UTC	Mumbai	Agra

Q35. Display information about the train of customer id 239.

with ticket(value) as

```
(select train_id from booking natural join train_ticket_confirm where c_id=239)
select train.*
from train,ticket
where train_id=ticket.value;
```



The screenshot shows a PostgreSQL query editor window. The query is:

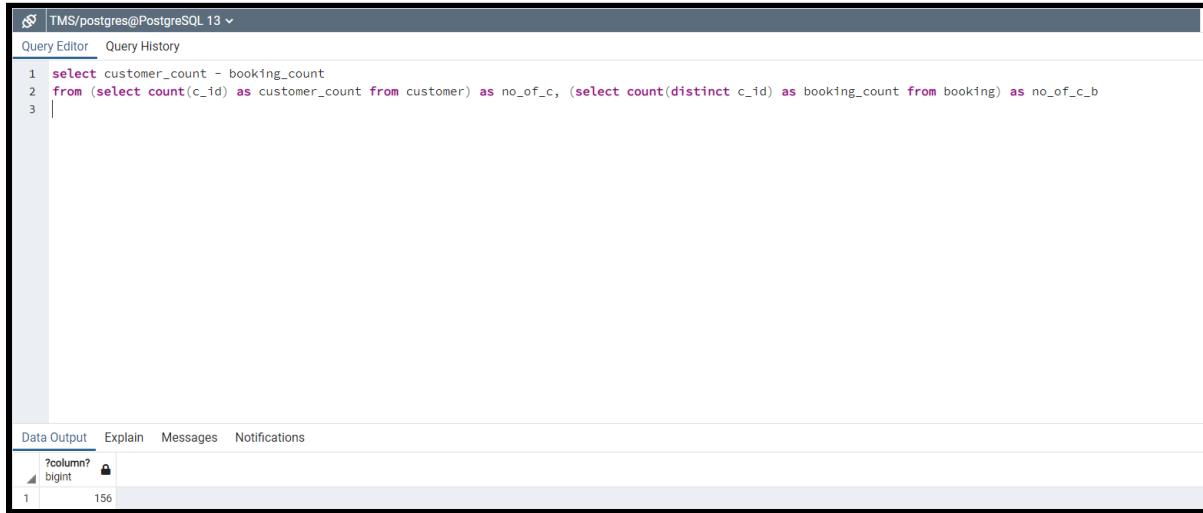
```
1 with ticket(value) as
2   (select train_id from booking natural join train_ticket_confirm where c_id=239)
3
4 select train.*
5   from train,ticket
6 where train_id=ticket.value;
7
```

The results table has the following data:

	train_id	train_type	arrival_time	departure_time	bording_point	destination_location
1	t52	Superfast Express	2021-07-02 21:30:41 UTC	Asansol	Varanasi	
2	t30	Luxury Trains	2021-03-29 06:10:37 UTC	Bangalore	Nashik	

Q36. How many customers have not booked any package yet?

```
select customer_count - booking_count
from (select count(c_id) as customer_count from customer) as no_of_c, (select count(distinct c_id) as
booking_count from booking) as no_of_c_b
```



The screenshot shows a PostgreSQL query editor window. The query is:

```
1 select customer_count - booking_count
2 from (select count(c_id) as customer_count from customer) as no_of_c, (select count(distinct c_id) as booking_count from booking) as no_of_c_b
3
```

The results table has the following data:

	column?	lock
1	156	

Q37. Show Second class Flight from Ahmedabad to Delhi.

```
select flight.*  
from flight  
where flight_type = 'Second class' and boarding_point = 'Ahmedabad' and destination_location = 'Delhi'
```

flight_id	flight_type	arrival_time	departure_time	boarding_point	destination_location
f11	Second class	09-04-2021 7:49	2021-04-09 08:49:50 UTC	Ahmedabad	[...] Delhi

Q38. List details of the customer which doesn't have any members with it.

```
select customer.*  
from customer  
where c_id not in (select distinct leader_id from group_member)
```

c_id	customername	c_password	gender	contact_no	address	pincode
1	Cchadie Shoobridge	foTuJYEHfYY	Male	+52 631 832 2744	82 Butterfield Hill	754744
2	Meggy Windrass	Jai9OEHR9	Male	+86 965 309 2987	403 Kim Hill	771357
3	Hildagarde Streak	5yHE31cA218	Female	+33 314 722 2534	8 Lerdahl Terrace	217929
4	Bernardina Sivorn	oEauXFNwQ	Female	+63 348 246 4887	1112 Golf Course Trail	796295
5	Dasi Yetts	PGaW3tJrs6Z	Female	+81 733 690 4627	97 Monument Crossing	139268
6	Gordy Samuels	OND8Ejhxt	Male	+86 796 494 8267	1 Bonner Road	484425
7	Beatrice Girdlestone	NYmP5QxRD	Male	+237 955 446 7297	227 Springview Plaza	465307
8	Pammie Stochart	M1LWxs2XlhQ	Male	+504 382 778 8390	29 Mitchell Place	354518
9	Brittine Daynter	TIVs3ks	Male	+370 388 511 2113	58770 Dottie Point	644088
10	Adrian Sansbury	tjOJAdlP65yJ	Male	+7 990 544 2909	760 Bartillon Alley	671434
11	Brinna Moggach	oX1nw0Rp9	Male	+380 836 596 4107	09 Lyons Road	588562

Q39. Show all places name visited by customer id 204.

with places(value) as

```
(select place_id from booking,tour_detail where tourpackage_id=package_id and c_id=204)
```

```
select place_name  
from tourist_places,places  
where place_id=value;
```

The screenshot shows a PostgreSQL query editor interface. The top bar displays the connection information: postgres/postgres@PostgreSQL 14. Below the bar, there are tabs for 'Query Editor' (which is selected) and 'Query History'. On the right side, there is a 'Scratch Pad' tab and a close button. The main area contains the SQL code for the query:

```
1 with places(value) as  
2     (select place_id from booking,tour_detail where tourpackage_id=package_id and c_id=204)  
3  
4 select place_name  
5 from tourist_places,places  
6 where place_id=value;  
7
```

Below the code, there are four tabs: 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected, showing a table with one column, 'place_name'. The table contains four rows of data:

place_name
character varying (30)
1 Ruins of Hampi, Karnataka
2 Ramanathaswamy Temple
3 Khajuraho Temples
4 Mehrangarh Fort, Jodhpur

Q40. List the name of the place which is most visited.

```

create view packageplaces(tourpackagecount,pid) as
select count(tourpackage_id), place_id
    from (select tour_detail.tourpackage_id, tour_detail.place_id
          from tour_detail, booking
         where booking.package_id = tour_detail.tourpackage_id) as spp
       group by place_id;

select tourist_places.*
  from packageplaces,tourist_places
 where packageplaces.pid = tourist_places.place_id and tourpackagecount = (select max(tourpackagecount) from
packageplaces);

```

The screenshot shows a PostgreSQL query editor window titled "TMS/postgres@PostgreSQL 13". The query tab contains the following code:

```

1 create view packageplaces(tourpackagecount,pid) as
2 select count(tourpackage_id), place_id
3     from (select tour_detail.tourpackage_id, tour_detail.place_id
4           from tour_detail, booking
5          where booking.package_id = tour_detail.tourpackage_id) as spp
6        group by place_id;
7
8
9 select tourist_places.*
10   from packageplaces ,tourist_places
11  where packageplaces.pid = tourist_places.place_id and tourpackagecount = (select max(tourpackagecount) from packageplaces);
12

```

The results tab shows the output of the query:

place_id	place_name	place_type	place_description
93	Monuments at Mandu, Dhar	City	Mandu or Mandavgarh is located in the Dhar district of Madhya Pradesh. It was said to have been established in the 6th century BC and later came under

Function

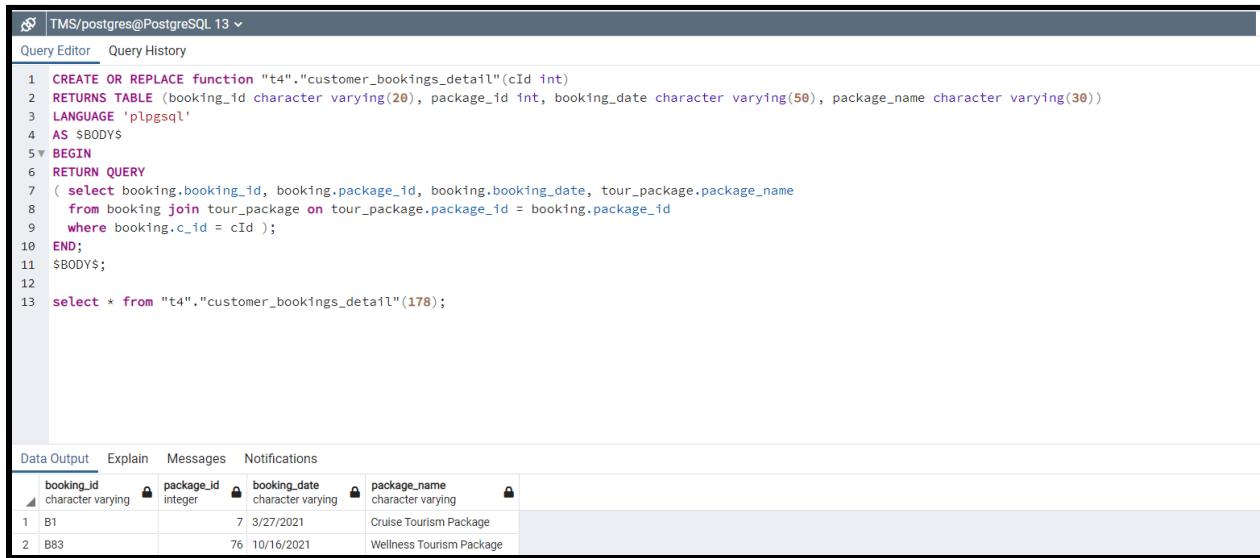
1. Function which shows booking details of customers.

```

CREATE OR REPLACE function "t4"."customer_bookings_detail"(cId int)
RETURNS TABLE (booking_id character varying(20), package_id int, booking_date character varying(50),
package_name character varying(30))
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
RETURN QUERY
( select booking.booking_id, booking.package_id, booking.booking_date, tour_package.package_name
from booking join tour_package on tour_package.package_id = booking.package_id
where booking.c_id = cId );
END;
$BODY$;

select * from "t4"."customer_bookings_detail"(178);

```



The screenshot shows a PostgreSQL query editor window titled 'TMS/postgres@PostgreSQL 13'. The 'Query Editor' tab is active, displaying the SQL code for creating a function and executing it. The code is identical to the one provided above. Below the code, the 'Data Output' tab is selected, showing a table with two rows of data. The table has four columns: 'booking_id' (character varying), 'package_id' (integer), 'booking_date' (character varying), and 'package_name' (character varying). The first row corresponds to the query 'select * from "t4"."customer_bookings_detail"(178);', and the second row corresponds to the query 'select * from "t4"."customer_bookings_detail"(178);' again.

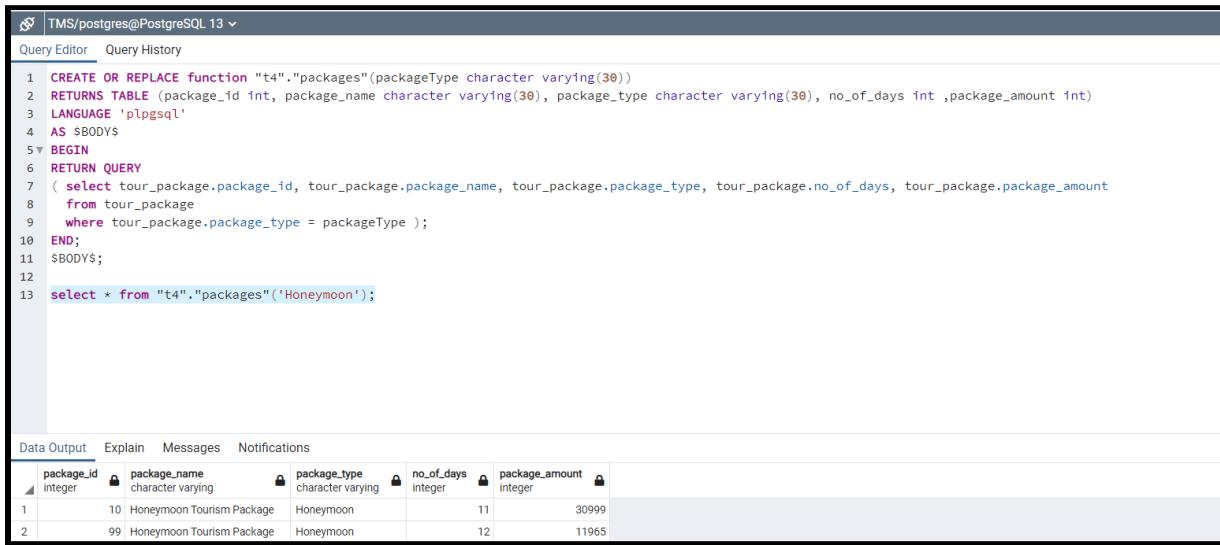
booking_id	package_id	booking_date	package_name
B1	7	3/27/2021	Cruise Tourism Package
B83	76	10/16/2021	Wellness Tourism Package

2. Function which show package type's information

```

CREATE OR REPLACE function "t4"."packages"(packageType character varying(30))
RETURNS TABLE (package_id int, package_name character varying(30), package_type character varying(30),
no_of_days int ,package_amount int)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
RETURN QUERY
( select tour_package.package_id, tour_package.package_name, tour_package.package_type,
tour_package.no_of_days, tour_package.package_amount
from tour_package
where tour_package.package_type = packageType );
END;
$BODY$;
select * from "t4"."packages"('Honeymoon');

```



The screenshot shows a PostgreSQL query editor window titled 'TMS/postgres@PostgreSQL 13'. The query editor contains the SQL code for creating a function named 'packages'. The function takes a parameter 'packageType' of type character varying(30) and returns a table with columns: package_id (integer), package_name (character varying), package_type (character varying), no_of_days (integer), and package_amount (integer). The function body uses a RETURN QUERY clause to select data from the 'tour_package' table where the package type matches the input parameter. The code ends with an END; statement and a \$BODY\$ placeholder. A final select statement is shown to demonstrate the function: 'select * from "t4"."packages"('Honeymoon');'. Below the code, there are tabs for Data Output, Explain, Messages, and Notifications. The Data Output tab displays a table with two rows of data:

	package_id	package_name	package_type	no_of_days	package_amount
1	10	Honeymoon Tourism Package	Honeymoon	11	30999
2	99	Honeymoon Tourism Package	Honeymoon	12	11965

Triggers function

1. Trigger function and trigger call for inserting new tuples in customer.

```
CREATE OR REPLACE FUNCTION "t4"."Customer_Update"()
RETURNS trigger
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
c_ID numeric;
BEGIN
SELECT customer.c_id INTO c_ID FROM customer
WHERE customer.c_id=NEW.c_id;
IF(c_ID=NEW.c_id) THEN
RAISE NOTICE 'C_ID already exists. Violates the primary key constraints.';
RETURN null;
ELSE
RAISE NOTICE 'Data inserted successfully.';
RETURN new;
END IF;
END
$BODY$
```

```
CREATE TRIGGER "Customer_Trigger"
BEFORE INSERT
ON "t4".customer
FOR EACH ROW
EXECUTE PROCEDURE "t4"."Customer_Update"();
```

SQL Query

Violation Case

The screenshot shows a PostgreSQL query editor window titled "TMS/postgres@PostgreSQL 13". The code area contains a function definition and an insert statement:

```

1 CREATE OR REPLACE FUNCTION "t4"."Customer_Update"()
2 RETURNS trigger
3 LANGUAGE 'plpgsql'
4 AS $BODY$
5 DECLARE
6   c_ID numeric;
7 BEGIN
8   SELECT customer.c_id INTO c_ID FROM customer
9   WHERE customer.c_id=NEW.c_id;
10  IF(c_ID=NEW.c_id) THEN
11    RAISE NOTICE 'C_ID already exists. Violates the primary key constraints.';
12    RETURN null;
13  ELSE
14    RAISE NOTICE 'Data inserted successfully.';
15    RETURN new;
16  END IF;
17 END
$BODY$
```

Below the code, the message "NOTICE: C_ID already exists. Violates the primary key constraints." is displayed.

Successful Case

The screenshot shows a PostgreSQL query editor window titled "TMS/postgres@PostgreSQL 13". The code area contains the same function definition and insert statement as the previous screenshot, but with different values:

```

1 CREATE OR REPLACE FUNCTION "t4"."Customer_Update"()
2 RETURNS trigger
3 LANGUAGE 'plpgsql'
4 AS $BODY$
5 DECLARE
6   c_ID numeric;
7 BEGIN
8   SELECT customer.c_id INTO c_ID FROM customer
9   WHERE customer.c_id=NEW.c_id;
10  IF(c_ID=NEW.c_id) THEN
11    RAISE NOTICE 'C_ID already exists. Violates the primary key constraints.';
12    RETURN null;
13  ELSE
14    RAISE NOTICE 'Data inserted successfully.';
15    RETURN new;
16  END IF;
17 END
$BODY$
```

Below the code, the message "NOTICE: Data inserted successfully." is displayed.

2. Trigger function for inserting new group members.

```
CREATE OR REPLACE function "t4".Member_Update()
RETURNS trigger
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
c_Id numeric;
BEGIN
SELECT customer.c_id INTO c_Id FROM customer
where customer.c_id=NEW.leader_id;
IF(c_id=NEW.leader_id) THEN
RAISE NOTICE 'Data Inserted Successfully';
RETURN NEW;
ELSE
RAISE NOTICE 'Customer is not available. Violates Foreign key constraints';
RETURN NULL;
END IF;
END
$BODY$
```

```
CREATE TRIGGER "Member_Trigger"
BEFORE INSERT
ON "t4".group_member
FOR EACH ROW
EXECUTE PROCEDURE "t4".Member_Update();
```

Unsuccessful Case (violation case)

The screenshot shows a PostgreSQL Query Editor window with the following details:

- Connection:** TMS/postgres@PostgreSQL 13
- Editor Tabs:** Query Editor, Query History
- SQL Code:**

```

1 CREATE OR REPLACE function "t4".Member_Update()
2 RETURNS trigger
3 LANGUAGE 'plpgsql'
4 AS $BODY$
5 DECLARE
6   c_Id numeric;
7 BEGIN
8   SELECT customer.c_id INTO c_Id FROM customer
9   WHERE customer.c_id=NEW.leader_id;
10  IF(c_id!=NEW.leader_id) THEN
11    RAISE NOTICE 'Data Inserted Successfully';
12    RETURN NEW;
13  ELSE
14    RAISE NOTICE 'Customer is not available. Violates Foreign key constraints';
15    RETURN NULL;
16  END IF;
17  END
$BODY$
```

Line 28: `CREATE TRIGGER "Member_Trigger"`

Line 29: `BEFORE INSERT`

Line 30: `ON "t4".group_member`

Line 31: `FOR EACH ROW`

Line 32: `EXECUTE PROCEDURE "t4".Member_Update();`

Line 28: `INSERT INTO t4.group_member(`

Line 29: `member_id, leader_id, m_name, m_age, m_gender)`

Line 30: `VALUES (1001,305, 'Kishan', 10, 'Male');`
- Data Output:** Shows the error message: `NOTICE: Customer is not available. Violates Foreign key constraints`
- Notifications:** Shows `INSERT 0 0`
- Message:** `Query returned successfully in 181 msec.`

Successful Case

The screenshot shows a PostgreSQL Query Editor window with the following details:

- Connection:** TMS/postgres@PostgreSQL 13
- Editor Tabs:** Query Editor, Query History
- SQL Code:**

```

1 CREATE OR REPLACE function "t4".Member_Update()
2 RETURNS trigger
3 LANGUAGE 'plpgsql'
4 AS $BODY$
5 DECLARE
6   c_Id numeric;
7 BEGIN
8   SELECT customer.c_id INTO c_Id FROM customer
9   WHERE customer.c_id=NEW.leader_id;
10  IF(c_id!=NEW.leader_id) THEN
11    RAISE NOTICE 'Data Inserted Successfully';
12    RETURN NEW;
13  ELSE
14    RAISE NOTICE 'Customer is not available. Violates Foreign key constraints';
15    RETURN NULL;
16  END IF;
17  END
$BODY$
```

Line 28: `CREATE TRIGGER "Member_Trigger"`

Line 29: `BEFORE INSERT`

Line 30: `ON "t4".group_member`

Line 31: `FOR EACH ROW`

Line 32: `EXECUTE PROCEDURE "t4".Member_Update();`

Line 28: `INSERT INTO t4.group_member(`

Line 29: `member_id, leader_id, m_name, m_age, m_gender)`

Line 30: `VALUES (1000,16, 'Kishan', 10, 'Male');`
- Data Output:** Shows the success message: `NOTICE: Data Inserted Successfully`
- Notifications:** Shows `INSERT 0 1`
- Message:** `Query returned successfully in 282 msec.`

Section7 : Project Code with output screenshots

- We used python Django to connect our Postgres database and build the following website.
- We have made a short video for working on our web page containing all functionalities and zip files of our codes file.

7.1 Video Link:

<https://drive.google.com/file/d/1umoUuxgXvZT9GIai56ijEwbI0dz3VMYN/view?usp=sharing>

7.2 Codes File Link:

https://drive.google.com/file/d/1bnb5TYmAv7yN1jLOAIE_-NGJ-13sMH6X/view?usp=sharing

Views.py

We use the below backhand handling code to connect python Django with our Postgres database. To see all other files go to the google drive link.

```
from django import http
from django.http.response import HttpResponseRedirect
from django.shortcuts import render
from TMS.models import EmpModel
from django.contrib import messages
from TMS.forms import Empforms

import psycopg2
connection =
psycopg2.connect(host="localhost",database="TMS",user="postgres",password="admin",options='--c search_path=t4')
cursor = connection.cursor()

def home(request):
    return render(request, 'index.html')

def showemp(request):
    showall=EmpModel.objects.all()
    return render(request, 'c_index.html', {"data":showall})

def Insertemp(request):
    if request.method == "POST":
        if request.POST.get('c_id') and request.POST.get('customername') and
request.POST.get('c_password') and request.POST.get('gender') and
request.POST.get('contact_no') and request.POST.get('address') and
request.POST.get('pincode'):
            saverecord=EmpModel()
            saverecord.c_id=request.POST.get('c_id')
            saverecord.customername=request.POST.get('customername')
            saverecord.c_password=request.POST.get('c_password')
```

```
        saverecord.gender=request.POST.get('gender')
        saverecord.contact_no=request.POST.get('contact_no')
        saverecord.address=request.POST.get('address')
        saverecord.pincode=request.POST.get('pincode')
        saverecord.save()
        messages.success(request,'Customer ' +saverecord.customername+ ' is added
successfully.....')
        return render(request,'c_insert.html')
    else:
        return render(request,'c_insert.html')

def Editemp(request,c_id):
    Editempobj=EmpModel.objects.get(c_id=c_id)
    return render(request,'c_edit.html',{"EmpModel":Editempobj})

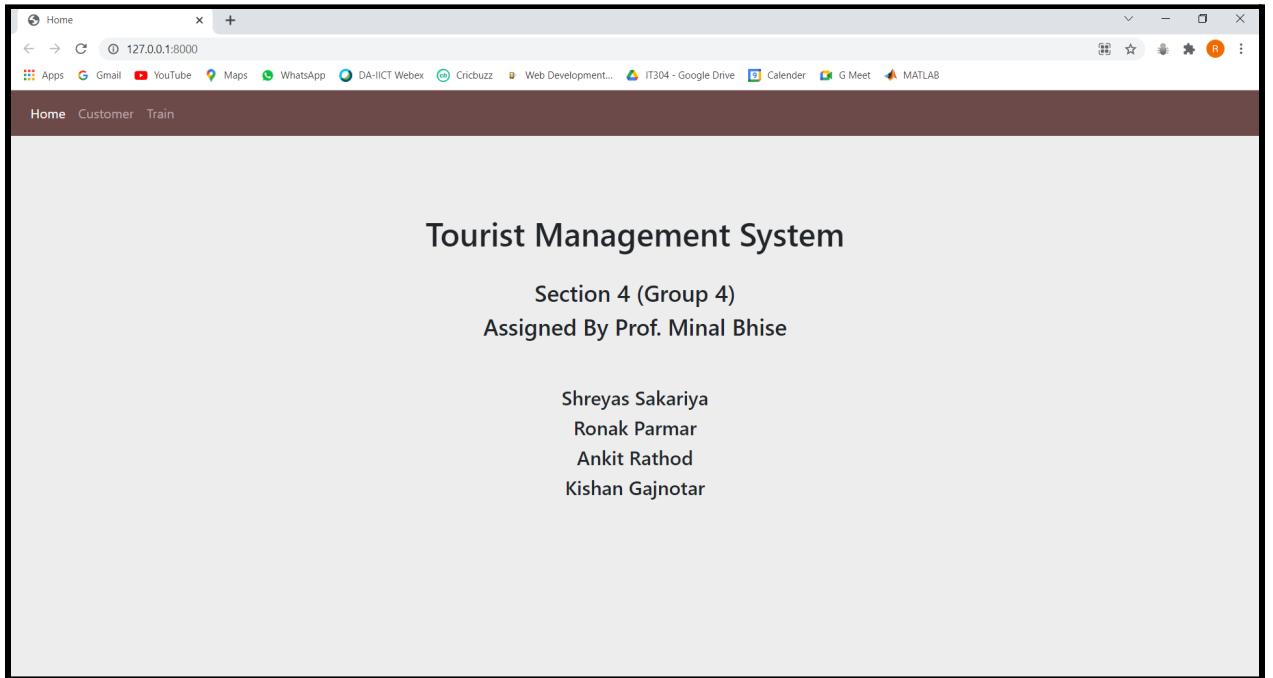
def updateemp(request,c_id):
    Updateemp=EmpModel.objects.get(c_id=c_id)
    form=Empforms(request.POST,instance=Updateemp)
    if form.is_valid():
        form.save()
        messages.success(request,'Record Update Successfully....!')
        return render(request,'c_edit.html',{"EmpModel":Updateemp})

def Delemp(request,c_id):
    delemployer=EmpModel.objects.get(c_id=c_id)
    delemployer.delete()
    showdata=EmpModel.objects.all()
    return render(request,'c_index.html',{"data": showdata})

def booking(request):
    if request.POST.get('find'):
        boarding_point= "%" +request.POST.get('boarding_point')+"%"
        destination_location= "%" +request.POST.get('destination_location')+"%"
        q=f"select * from train where (boarding_point like {boarding_point} and
destination_location like {destination_location})"
        cursor.execute(q)
        connection.commit()
        tuples = cursor.fetchall()
        return render(request,'t_index.html',{"data":tuples})
    else:
        q=f"select * from train "
        cursor.execute(q)
        connection.commit()
        tuples = cursor.fetchall()
        return render(request,'t_index.html',{"data":tuples})
```

7.3 Screenshot of output or web pages with data

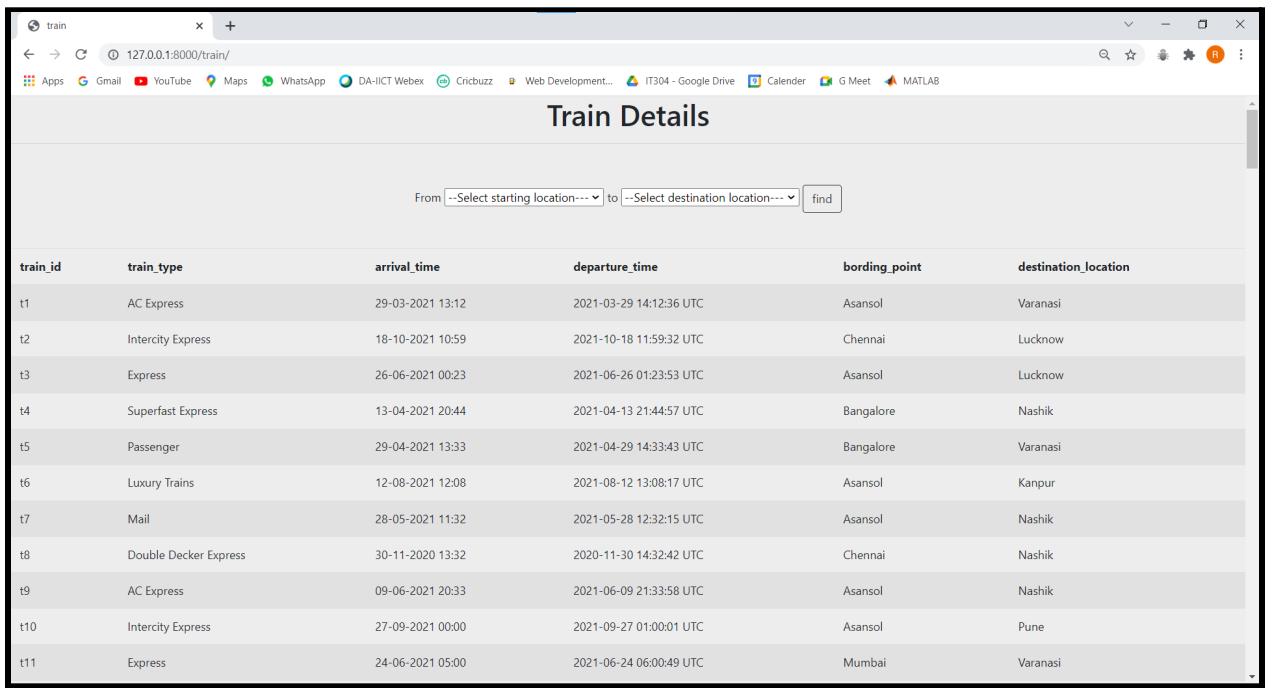
❖ The Homepage of our website



❖ Customer interface on our website

Customer's List						
c_id	name	password	gender	number	add	pincode
1	Sarge Jeakins	vdGCjh6ZttmF	Male	+86 251 239 1354	3668 Johnson Circle	197652
2	Adina Brettoner	Vfxr5Cj	Male	+216 715 258 6245	1302 Summer Ridge Drive	639200
3	Veronike Colleck	re1T93mG5b	Female	+48 881 110 7673	17331 Algoma Road	847614
4	Noe Burlingham	rUuZ5tYdB	Female	+30 203 872 1839	9605 Mayfield Junction	391485
5	Ariyn Coyne	RNIBtl4o	Male	+93 460 585 8663	8385 Ruskin Plaza	452871
6	Hendrika McKeeman	0LvQli	Female	+92 943 238 2470	870 Crest Line Junction	670784
7	Cchaddie Shoobridge	foTuYEhFrY	Male	+52 631 832 2744	82 Butterfield Hill	754744

❖ Train interface on our website

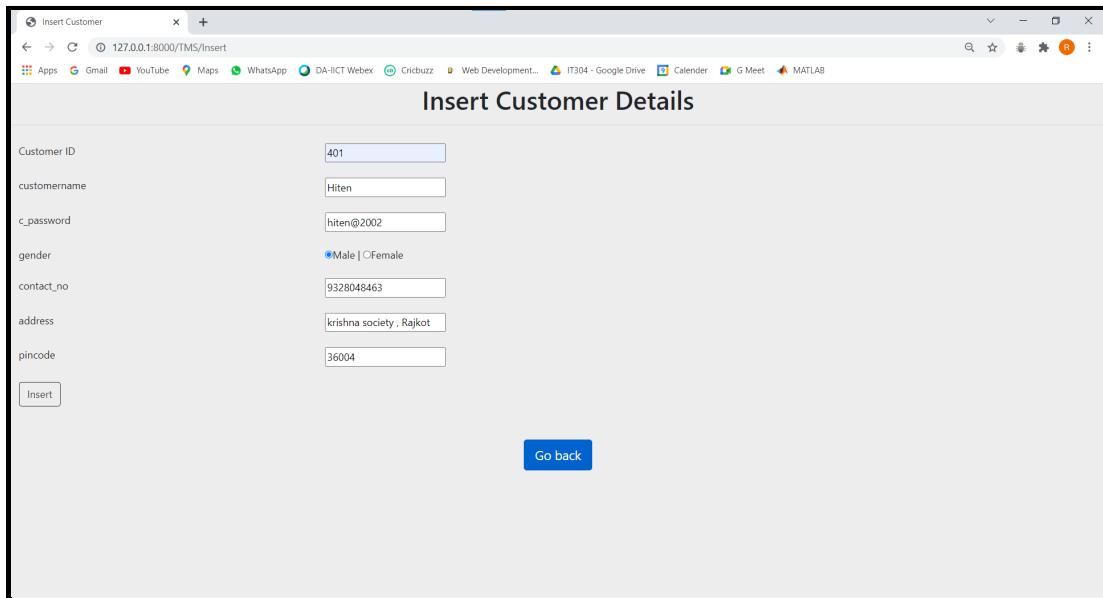


train_id	train_type	arrival_time	departure_time	bording_point	destination_location
t1	AC Express	29-03-2021 13:12	2021-03-29 14:12:36 UTC	Asansol	Varanasi
t2	Intercity Express	18-10-2021 10:59	2021-10-18 11:59:32 UTC	Chennai	Lucknow
t3	Express	26-06-2021 00:23	2021-06-26 01:23:53 UTC	Asansol	Lucknow
t4	Superfast Express	13-04-2021 20:44	2021-04-13 21:44:57 UTC	Bangalore	Nashik
t5	Passenger	29-04-2021 13:33	2021-04-29 14:33:43 UTC	Bangalore	Varanasi
t6	Luxury Trains	12-08-2021 12:08	2021-08-12 13:08:17 UTC	Asansol	Kanpur
t7	Mail	28-05-2021 11:32	2021-05-28 12:32:15 UTC	Asansol	Nashik
t8	Double Decker Express	30-11-2020 13:32	2020-11-30 14:32:42 UTC	Chennai	Nashik
t9	AC Express	09-06-2021 20:33	2021-06-09 21:33:58 UTC	Asansol	Nashik
t10	Intercity Express	27-09-2021 00:00	2021-09-27 01:00:01 UTC	Asansol	Pune
t11	Express	24-06-2021 05:00	2021-06-24 06:00:49 UTC	Mumbai	Varanasi

Functionality

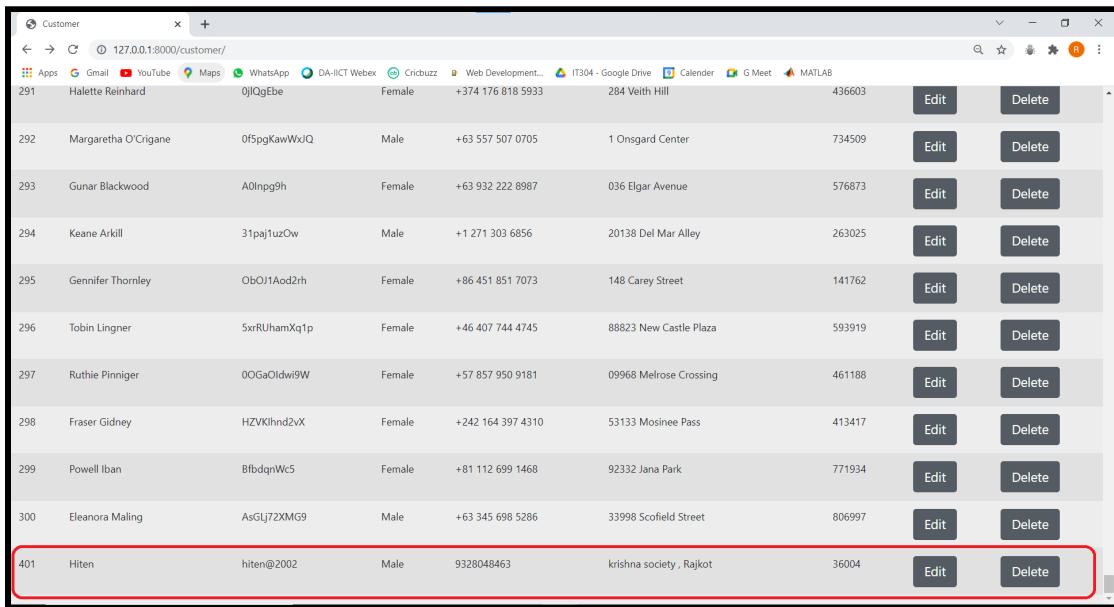
1. Insert

❖ Inserting Customer's detail



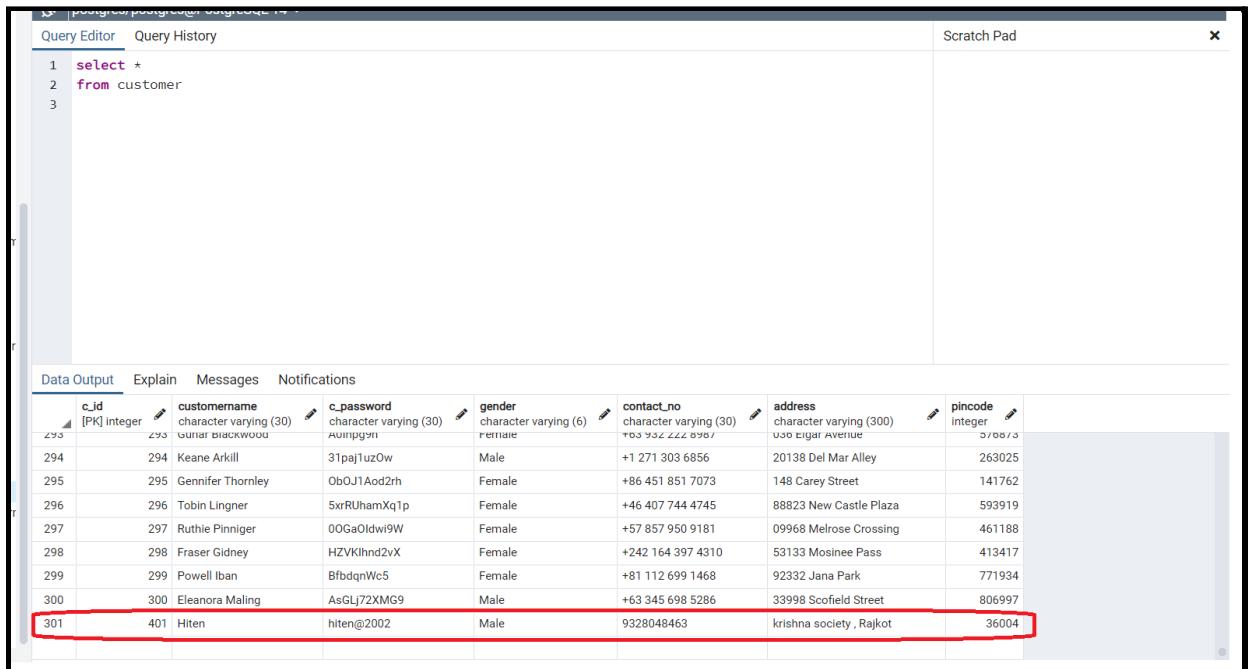
Customer ID	401
customername	Hiten
c_password	hiten@2002
gender	<input checked="" type="radio"/> Male <input type="radio"/> Female
contact_no	9328048463
address	krishna society, Rajkot
pincode	36004
<input type="button" value="Insert"/> <input type="button" value="Go back"/>	

❖ Customer record after inserting



c_id	customername	c_password	gender	contact_no	address	pincode	Edit	Delete
291	Halette Reinhard	0jIQgEbe	Female	+374 176 818 5933	284 Veith Hill	436603	Edit	Delete
292	Margaretha O'Crigane	0f5pgKawWxJQ	Male	+63 557 507 0705	1 Onsgard Center	734509	Edit	Delete
293	Gunar Blackwood	A0lnppg9h	Female	+63 932 222 8987	036 Elgar Avenue	576873	Edit	Delete
294	Keane Arkill	31paJ1uzOw	Male	+1 271 303 6856	20138 Del Mar Alley	263025	Edit	Delete
295	Gennifer Thornley	ObOJ1Aod2rh	Female	+86 451 851 7073	148 Carey Street	141762	Edit	Delete
296	Tobin Lingner	5xrRUhamXq1p	Female	+46 407 744 4745	88823 New Castle Plaza	593919	Edit	Delete
297	Ruthie Pinniger	0OGaOldwi9W	Female	+57 857 950 9181	09968 Melrose Crossing	461188	Edit	Delete
298	Fraser Gidney	HZVKlhnd2vX	Female	+242 164 397 4310	53133 Mosinee Pass	413417	Edit	Delete
299	Powell Iban	BfbdqnWc5	Female	+81 112 699 1468	92332 Jana Park	771934	Edit	Delete
300	Eleanora Maling	AsGLj72XMG9	Male	+63 345 698 5286	33998 Scofield Street	806997	Edit	Delete
401	Hiten	hiten@2002	Male	9328048463	krishna society , Rajkot	36004	Edit	Delete

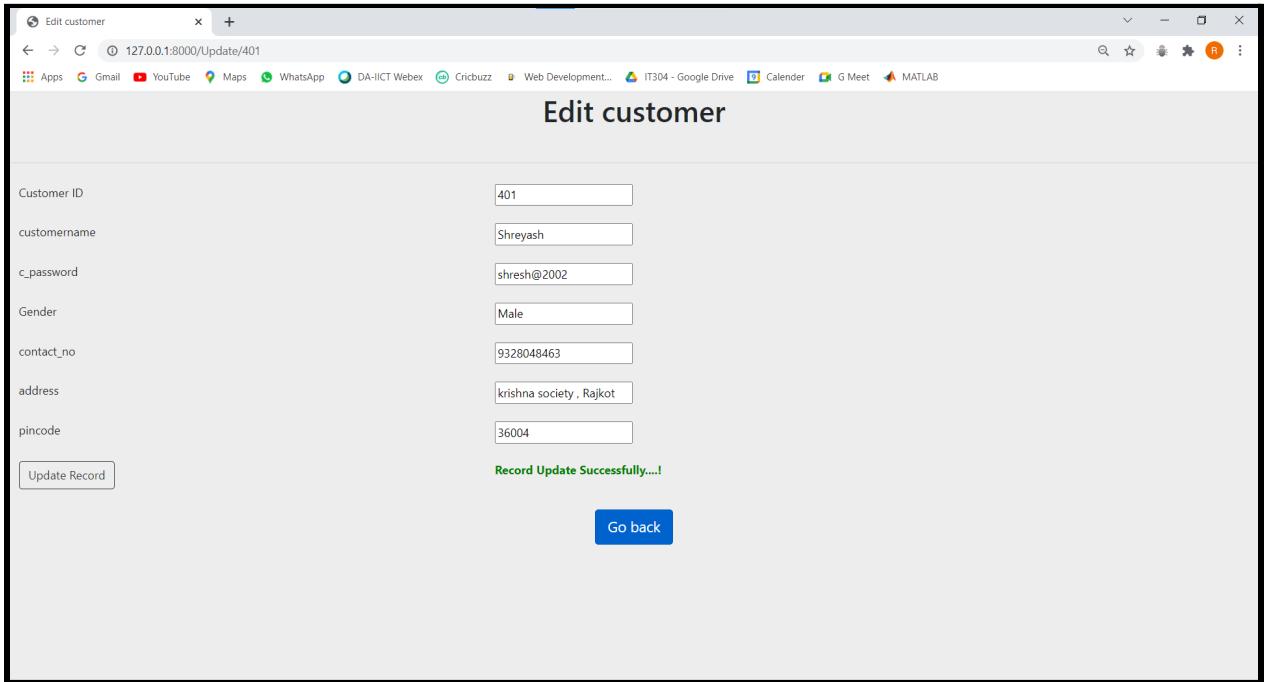
❖ Postgres updated data after inserting it on the website



c_id	customername	c_password	gender	contact_no	address	pincode
293	Gunar Blackwood	A0lnppg9h	Female	+63 932 222 8987	036 Elgar Avenue	576873
294	Keane Arkill	31paJ1uzOw	Male	+1 271 303 6856	20138 Del Mar Alley	263025
295	Gennifer Thornley	ObOJ1Aod2rh	Female	+86 451 851 7073	148 Carey Street	141762
296	Tobin Lingner	5xrRUhamXq1p	Female	+46 407 744 4745	88823 New Castle Plaza	593919
297	Ruthie Pinniger	0OGaOldwi9W	Female	+57 857 950 9181	09968 Melrose Crossing	461188
298	Fraser Gidney	HZVKlhnd2vX	Female	+242 164 397 4310	53133 Mosinee Pass	413417
299	Powell Iban	BfbdqnWc5	Female	+81 112 699 1468	92332 Jana Park	771934
300	Eleanora Maling	AsGLj72XMG9	Male	+63 345 698 5286	33998 Scofield Street	806997
301	Hiten	hiten@2002	Male	9328048463	krishna society , Rajkot	36004

2. Update

- ❖ Updating customer's details



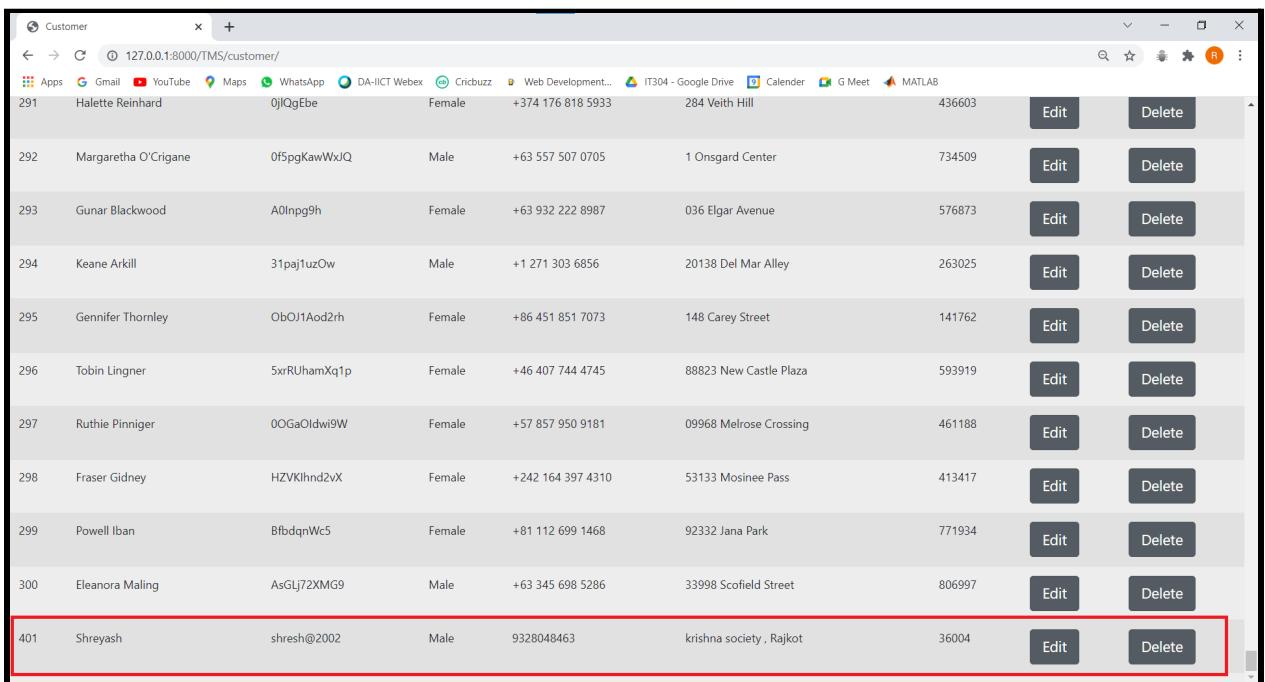
The screenshot shows a web browser window titled "Edit customer". The URL is 127.0.0.1:8000/Update/401. The form contains the following fields:

Customer ID	401
customername	Shreyash
c_password	shresh@2002
Gender	Male
contact_no	9328048463
address	krishna society , Rajkot
pincode	36004

Record Update Successfully....!

[Update Record](#) [Go back](#)

- ❖ Updated Customer record after updating



The screenshot shows a web browser window titled "Customer" with a table listing customer records. The URL is 127.0.0.1:8000/TMS/customer/. The table has columns: ID, Name, Address, Gender, Contact, Address, Pincode, Edit, and Delete.

ID	Name	Address	Gender	Contact	Address	Pincode	Edit	Delete
291	Halette Reinhard	0jIQgEbe	Female	+374 176 818 5933	284 Veith Hill	436603	Edit	Delete
292	Margaretha O'Crigane	0f5pgKawWxJQ	Male	+63 557 507 0705	1 Onsgard Center	734509	Edit	Delete
293	Gunar Blackwood	A0lnpg9h	Female	+63 932 222 8987	036 Elgar Avenue	576873	Edit	Delete
294	Keane Arkill	31paj1uzOw	Male	+1 271 303 6856	20138 Del Mar Alley	263025	Edit	Delete
295	Gennifer Thornley	ObOJ1Aod2rh	Female	+86 451 851 7073	148 Carey Street	141762	Edit	Delete
296	Tobin Lingner	5xrRUhamXq1p	Female	+46 407 744 4745	88823 New Castle Plaza	593919	Edit	Delete
297	Ruthie Pinniger	0OGaOldwi9W	Female	+57 857 950 9181	09968 Melrose Crossing	461188	Edit	Delete
298	Fraser Gidney	HZVKlhnd2vX	Female	+242 164 397 4310	53133 Mosinee Pass	413417	Edit	Delete
299	Powell Iban	BfbdqNWC5	Female	+81 112 699 1468	92332 Jana Park	771934	Edit	Delete
300	Eleanora Maling	AsGLj72XMG9	Male	+63 345 698 5286	33998 Scofield Street	806997	Edit	Delete
401	Shreyash	shresh@2002	Male	9328048463	krishna society , Rajkot	36004	Edit	Delete

❖ **Postgres: Updated Customer record after updating**

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Tables (17)', the 'train' table is selected. In the main area, a query editor window displays the following SQL code:

```

1 select *
2 from customer
3

```

The results of this query are shown in a table titled 'Data Output'. The table contains 12 rows of customer data. The last row, which corresponds to the updated record, is highlighted with a red border. The columns in the table are:

c_id	customername	c.password	gender	contact_no	address	pincode
294	Gunnar Blackwood	aunpynp	Female	+1 271 303 6856	20138 Del Mar Alley	263025
295	Keane Arkill	31paj1uzOw	Male	+86 451 851 7073	148 Carey Street	141762
296	Gennifer Thornley	Ob0J1Aod2fh	Female	+46 407 744 4745	88823 New Castle Plaza	593919
297	Tobin Lingner	5xrUHamXq1p	Female	+57 857 950 9181	09968 Melrose Crossing	461188
298	Ruthie Pinniger	0OGaOldwi9W	Female	+242 164 397 4310	53133 Mosinee Pass	413417
299	Fraser Gidney	HZVKlhnd2vX	Female	+81 112 699 1468	92332 Jana Park	771934
300	Powell Iban	BfbdqnWc5	Female	+63 345 698 5286	33998 Scofield Street	806997
301	Eleanor Maling	AsGLj72XMG9	Male	9328048463	krishna society , Rajkot	36004
401	Shreyash	shresh@2002	Male			

3. Delete

❖ Click on the delete button which customer detail you want to delete then press "OK"

The screenshot shows a web browser displaying a list of customer records. A modal dialog box is centered over the table, asking 'Are You Sure Want to Delete the Record?'. Two buttons are visible: 'OK' (highlighted with a red box) and 'Cancel'. The customer records listed are:

c_id	customername	c.password	gender	contact_no	address	pincode
291	Halette Reinhard	0jIQgEbe	Female			
292	Margaretha O'Crigane	0f5pgKawWxJQ	Male			
293	Gunnar Blackwood	A0lrrpg9h	Female	+63 932 222 8987	036 Elgar Avenue	576873
294	Keane Arkill	31paj1uzOw	Male	+1 271 303 6856	20138 Del Mar Alley	263025
295	Gennifer Thornley	Ob0J1Aod2fh	Female	+86 451 851 7073	148 Carey Street	141762
296	Tobin Lingner	5xrUHamXq1p	Female	+46 407 744 4745	88823 New Castle Plaza	593919
297	Ruthie Pinniger	0OGaOldwi9W	Female	+57 857 950 9181	09968 Melrose Crossing	461188
298	Fraser Gidney	HZVKlhnd2vX	Female	+242 164 397 4310	53133 Mosinee Pass	413417
299	Powell Iban	BfbdqnWc5	Female	+81 112 699 1468	92332 Jana Park	771934
300	Eleanor Maling	AsGLj72XMG9	Male	+63 345 698 5286	33998 Scofield Street	806997
401	Shreyash	shresh@2002	Male	9328048463	krishna society , Rajkot	36004

- ❖ Customer record after deleting

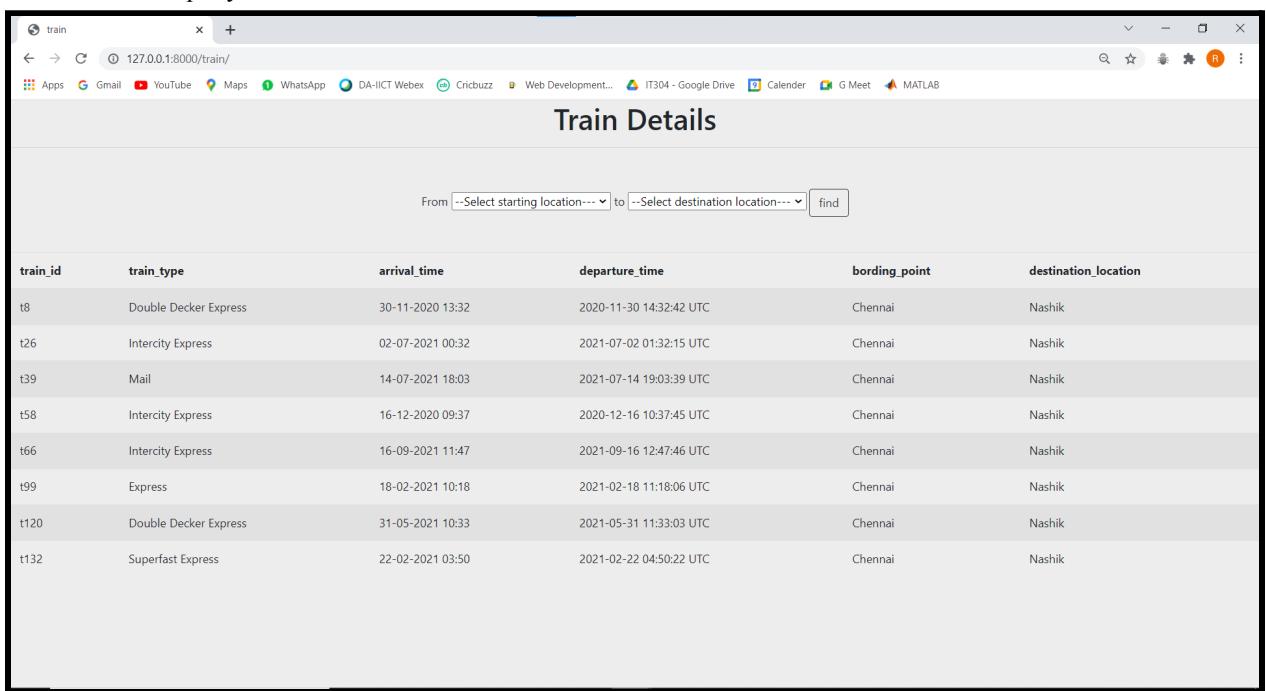
ID	Name	Email	Gender	Phone	Address	Zip	Edit	Delete
290	Enid Fallawie	PBB4jP	Male	+55 337 976 4538	38 Gina Hill	165865	<button>Edit</button>	<button>Delete</button>
291	Halette Reinhard	0jIQgEbe	Female	+374 176 818 5933	284 Veith Hill	436603	<button>Edit</button>	<button>Delete</button>
292	Margaretha O'Crigane	0f5pgKawWxJQ	Male	+63 557 507 0705	1 Onsgard Center	734509	<button>Edit</button>	<button>Delete</button>
293	Gunar Blackwood	A0lnpg9h	Female	+63 932 222 8987	036 Elgar Avenue	576873	<button>Edit</button>	<button>Delete</button>
294	Keane Arkill	31paj1uzOw	Male	+1 271 303 6856	20138 Del Mar Alley	263025	<button>Edit</button>	<button>Delete</button>
295	Gennifer Thornley	ObOJ1Aod2h	Female	+86 451 851 7073	148 Carey Street	141762	<button>Edit</button>	<button>Delete</button>
296	Tobin Lingner	5xrRUhamXq1p	Female	+46 407 744 4745	88823 New Castle Plaza	593919	<button>Edit</button>	<button>Delete</button>
297	Ruthie Pinniger	0OGaOldwi9W	Female	+57 857 950 9181	09968 Melrose Crossing	461188	<button>Edit</button>	<button>Delete</button>
298	Fraser Gidney	HZVKlhnd2vX	Female	+242 164 397 4310	53133 Mosinee Pass	413417	<button>Edit</button>	<button>Delete</button>
299	Powell Iban	BfbdqnWc5	Female	+81 112 699 1468	92332 Jana Park	771934	<button>Edit</button>	<button>Delete</button>
300	Eleanora Maling	AsGLj72XMG9	Male	+63 345 698 5286	33998 Scofield Street	806997	<button>Edit</button>	<button>Delete</button>

4. Search

- ❖ Here we use Train table for searching operation in which you select boarding point(starting location) and destination location it will give also train information from starting location to destination location

Train Details					
train_id	train_type	arrival_time	departure	boarding_point	destination_location
t1	AC Express	29-03-2021 13:12	2021-03-	Asansol	Varanasi
t2	Intercity Express	18-10-2021 10:59	2021-10-	Chennai	Lucknow
t3	Express	26-06-2021 00:23	2021-06-26 01:23:53 UTC	Asansol	Lucknow
t4	Superfast Express	13-04-2021 20:44	2021-04-13 21:44:57 UTC	Bangalore	Nashik
t5	Passenger	29-04-2021 13:33	2021-04-29 14:33:43 UTC	Bangalore	Varanasi
t6	Luxury Trains	12-08-2021 12:08	2021-08-12 13:08:17 UTC	Asansol	Kanpur
t7	Mail	28-05-2021 11:32	2021-05-28 12:32:15 UTC	Asansol	Nashik
t8	Double Decker Express	30-11-2020 13:32	2020-11-30 14:32:42 UTC	Chennai	Nashik
t9	AC Express	09-06-2021 20:33	2021-06-09 21:33:58 UTC	Asansol	Nashik
t10	Intercity Express	27-09-2021 00:00	2021-09-27 01:00:01 UTC	Asansol	Pune
t11	Express	24-06-2021 05:00	2021-06-24 06:00:49 UTC	Mumbai	Varanasi

❖ Result of search query



The screenshot shows a web browser window titled "train" with the URL "127.0.0.1:8000/train/". The page is titled "Train Details" and contains a search bar with fields "From" and "To" followed by a "find" button. Below the search bar is a table with columns: train_id, train_type, arrival_time, departure_time, boarding_point, and destination_location. The table lists eight train entries, all originating from Chennai and terminating at Nashik.

train_id	train_type	arrival_time	departure_time	boarding_point	destination_location
t8	Double Decker Express	30-11-2020 13:32	2020-11-30 14:32:42 UTC	Chennai	Nashik
t26	Intercity Express	02-07-2021 00:32	2021-07-02 01:32:15 UTC	Chennai	Nashik
t39	Mail	14-07-2021 18:03	2021-07-14 19:03:39 UTC	Chennai	Nashik
t58	Intercity Express	16-12-2020 09:37	2020-12-16 10:37:45 UTC	Chennai	Nashik
t66	Intercity Express	16-09-2021 11:47	2021-09-16 12:47:46 UTC	Chennai	Nashik
t99	Express	18-02-2021 10:18	2021-02-18 11:18:06 UTC	Chennai	Nashik
t120	Double Decker Express	31-05-2021 10:33	2021-05-31 11:33:03 UTC	Chennai	Nashik
t132	Superfast Express	22-02-2021 03:50	2021-02-22 04:50:22 UTC	Chennai	Nashik