

Internet Service Provider System

DBMS - IT 214

Team S3_T8

Members (T8):

Sachin Parekh	201901176
Tvesha Srivastava	201901182
Richa Sharma	201901190
Archana Todkar	201901294

**INSTRUCTOR- MINAL BHISE
MENTOR TA- MAYANK PATEL**



**DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND
COMMUNICATION TECHNOLOGY**

Table of Contents

A. Introduction	5
Purpose	5
Intended Audience and Reading Suggestions	6
Product Scope	6
Description:	7
B. Requirements Collection Phase	8
Background Reading	8
Interviews	9
Interview - 1	9
Purpose of Interview:	9
Agenda:	10
Documents to be brought to the interview:	10
Interview - 2	10
Purpose of Interview:	10
Agenda:	11
Documents to be brought to the interview:	11
Interview - 3	11
Combined requirements gathered from the interviews	12
Questionnaire	13
Combined Requirements gathered from the response	25
Observations	26
Combined Requirements gathered from the observations	27
C. Create Fact-Finding Chart	28
D. List Requirements	29
E. User Classes and Characteristics	30
F. Operating Environment	30
Hardware, software and connectivity requirements	30
External Interface Requirements	30
G. Product Functions	31
H. Privileges	31
I. Assumptions	32
J. Business Constraints	32
I. Noun (& Verb) Analysis.	34
Extracted Nouns and Verbs from Problem Description	34

Accepted Noun & Verbs list	36
II. Develop the ER Diagram (ERD).	39
Version 1 of ER Diagram	39
Version 2 of ER Diagram	40
I.) Mapping of ER model to Relational model.	42
1. With the help of mapping rules discussed in the class, map all the entity and relationship sets to the corresponding relational representation.	42
2) Write down all the relations with the schema.	43
1) List all the Relations & Schemas with all details (Original Design of Database)	45
2) Identify and list all types of dependencies (PK, FK, Functional Dependencies) for each relation	47
DDL Scripts for Celltech Database	66
SQL QUERIES	70
Front-end Development	95
Screenshot of output or web/app pages with data.	96

Section 1:

Final Version of SRS

A. Introduction

Purpose

The main purpose is to provide access to the Internet to both personal and business customers. Our product provides customers with internet access that gives them the ability to work and perform basic technological actions with an uninterrupted connection. Also, by maintaining an efficient database of our users we have solved any management problems and continue to recognize patterns, which help us in understanding our users and their needs better.

Intended Audience and Reading Suggestions

- Developers
- Marketing teams
- Documentation writer
- Database Administrator(DBA)
- Database designers
- System analyst
- Application programmers
- Testers
- Sophisticated Users - They can be engineers, scientists, business analysts, who are familiar with the database and can develop their own database applications according to their requirements.
- End users(General public)

There is no specific order to read and whoever the reader is can proceed and skip to the part they find relevant by referring to the table of contents above.

Product Scope

With billions of internet users and each having their own purpose and needs, the primary goal is to satisfy every user's needs, be it an individual user or corporate user. To meet everyone's needs, the product will offer a variety of Internet plans, accessibility options, and payment features to have a personalized experience. The scope of this project also includes the implementation of the ISP database and to levitate the efficiency of ISP management with respect to various factors like speed, range etc. This product will consider almost all aspects which an ISP app is required to deliver. It will keep count of billing details, variety of plans etc. Several security concerns with the product and database system that arise from a large number of Internet users will also be taken into account. Another goal would be to resolve these issues and maintain user and data confidentiality. Along with these, every user will have access to relevant information.

Description:

- In compliance with supplying our product which is an ISP, we had to make a proper database for it that deals with most of the issues that might arise in our management.
- Design must be able to handle large data uploading loads as well as keep data updating in real-time so that accurate information reaches our company each time a user gets added to the database or is removed from it.
- The notion of an ISP that handles providing proper connections in an efficient manner and dealing with privacy and storage issues with the help of an accurate database.
- The major role of our tech support and development team comes when we deal with the security levels of our database because it's very necessary to keep all the user information private and classified to prevent questionable activities which can alter with the original information causing the system to fail and our company to suffer.
- We, as a company needs to ensure that our product is up to date with the latest technology and provides it to our users at a comparatively affordable rate.
- Having a proper ISP is important in every work environment whether it is a big corporation or a small school. With the advancing technology, it seems impossible to survive nowadays with an unstable broadband network, fiber or any type of network connection even at home.
- Through the interviews we have conducted and observations we have taken note of, we have assessed the following concerns:-
 1. Technical glitches
 2. Security threats
 3. Loss of data and how to recover it.
- In relation to these and keeping them in mind, we have formed a database for our company such that we only keep the information relevant to us in our database and forms since bigger the sea, larger the number of fishes being caught, in explicit terms we have tried to minimize the details we hold of our users so that even in case of a threat, they cannot be targeted.

- We have included several tables and functions in our database to give us a crystal-like clarity on our product and how much people are comfortable with paying for it.
- In order to stay in touch with our existing and potential customers, we have conducted surveys to understand their preferences and disinterests.
- The users of our product include people requiring a stable connection like workplace environments inclusive of big and small corporations, schools and colleges and the general public.
- We have classified our users into two groups, while one group consists of the general public installing our ISP in their homes, the other one includes big business companies and authorities which have installed our product in their office campus.
- One important issue in working with corporations is lack of information, since we are providing a proper connection to an ample number of floors and buildings, we require a well built database for each one which is strong enough to hold all the entries.
- We have tried to make our product as unique as we can by constituting it with high speed, large range and next to no glitches.
- We have added tables to our database specifically for regions that get low connectivity so we can monitor how many users on average have to deal with such issues and come up with solutions to their problems.
- We have taken proper measures to ensure the restoration of data in case of data loss or glitches.
- With these constant modifications, we have tried to deal with maintaining a proper database for our product and its sale.

B. Requirements Collection Phase

Background Reading

- In this section we have tried to collect data from various other Internet Service Providers, big companies like Jio and Airtel have been a source of our knowledge and reading their developments and issues has helped us in formulating an optimal app and database.
- Providing internet services is inclusive of a very large domain of users, whether it be for whatever use. This increases our areas to work on and benefits respectively, which is why it has been important for us to go through this segment.
- To provide our users with an ISP that is satisfactory, we need accurate and real time data, for example:-
 1. How have big corporations like Jio and Airtel dealt with data leaks in the past?.
 2. Back-up and recovery of this data in the database and the process behind it.
 3. Ensuring the users of a “no privacy breach” policy and minimizing our data on their personal records.

4. Making our app more easy-to-use and understandable by providing several,customizable fields in it.
- In a field as competitive as ours,it's important to look up to bigger companies for inspiration and clarity on what/what is not necessary to be a part of our app and database.
- We have tried to avoid most of the issues that could arise within our database,including:-
 1. A proper history of payments and selected plans by the users.
 2. Adding more functions to each table to maximize the benefit and scope.
 3. Making sure that user information is stored under their reference number and at a maximum,their name to prevent us from having to ask for personal details.
- We have understood some of these concepts and modified them in our own way by going through the following articles:-
 1. <https://www.trustradius.com/business-internet-service-providers-isps>
 2. <https://www.trustradius.com/business-internet-service-providers-isps>
 3. <https://www.jio.com/fiber.html>

Interviews

Interview - 1

IT Solutions: Mock Interview Plan & Summary.

System: Internet Service Provider

Project Reference: T8/S3/2021/10

(**Role Play**)

Interviewee: 1) Archana Todkar

Designation: CEO at Cell tech

Interviewer: 1) Sachin Parekh

Designation: Developer - IT Solutions

2) Richa Sharma

Designation: Developer - IT Solutions

Date: 06/10/2021

Time: 11:30AM

Duration: 45 minutes

Place: Google Meet

Purpose of Interview:

This meeting is to identify the issues in the current existing database system of our company and how we plan on resolving them. We plan on coming up with a better model for our database which will be inclusive of all fields.

Agenda:

- Making the database more inclusive and thorough.
- User-specific access.
- Security concerns and backup strategy

Documents to be brought to the interview:

Network plans of our product and its details.

Documents containing important information like reference numbers, product plans, etc. of the users.

Documents containing our security protocols.

Documents containing an outline of our project.

Interview - 2

IT Solutions: Mock Interview Plan & Summary.

System: Internet Service Provider

Project Reference: T8/S3/2021/10

(Roleplay)

Interviewee: 1) Sachin Parekh **Designation:** resident of East Santa Cruz (hub of highest data usage)

Interviewer: 1) Richa Sharma
2) Tvesha Srivastava

Designation: Developer-IT Solutions
Designation: Developer-IT Solutions

Date: 06/10/2021

Time: 5:00PM

Duration: 35 mins

Place: Google Meet

Purpose of Interview:

This meeting is to highlight difficulties and requirements with the existing ISP that the end-users are using, as well as how we can assist them in resolving these issues.

- Providing easy to understand user interface to people.
- To address issues regarding the range and also the strength of their connectivity inside their household.
- To determine whether or not the individuals in the neighborhood have access to local ISP centers.
- We need to know what users would expect from a new internet service provider.
- Notify customers by sending alerts before their plan expiry and provide a window (for x days) for the renewal of their plan.
- The customer services and technician must be available 24x7.

Agenda:

To understand the issues faced by the current residents of this neighborhood, to ensure that our ISP can resolve them, and to provide the area with a better broadband connection.

Documents to be brought to the interview:

Valid ID Proof , Brochure

Interview - 3

IT Solutions: Mock Interview Plan & Summary.

System: Internet Service Provider

Project Reference: T8/S3/2021/10

(**Role Play**)

Interviewee: 1) Tvesha Srivastava

Designation: Technical head at STAR Uni

Interviewer: 1) Archana Todkar

Designation: Developer ITSolutions

2) Richa Sharma

Designation: Developer ITSolutions

Date: 06/10/2021

Time: 10:30AM

Duration: 45 minutes

Place: Google Meet

Purpose of Interview:

A brief discussion about our ISP and how it provides better range, speed, and connectivity in contrast to our other competitors belonging in the same genre.

We plan to provide STAR Uni with our latest product and the network plan we feel

might be the most optimal for them given their history of usage.

Agenda:

- Providing a memo of our network plans.
- Area restriction
- Registering STAR Uni to the user database by successfully incorporating the broadband connection in their campus.
- Revoke access from inappropriate sites to maintain discipline

Documents to be brought to the interview:

Network plans of our product and its details.

Documents containing our security protocols.

Documents containing an outline of our project.

A brochure of our recent developments.

Combined requirements gathered from the interviews

- User Specific Access - The database needs to be able to configure different access levels and rights based on different users.
- Security - Measures to be taken to avoid security breaches like
 - i) Unauthorized access
 - ii) Data loss/corruption
 - iii) User privacy
- Backup and Recovery - Maintaining backups is essential in case of crash/loss and recovering data is necessary.
- Simple User Interface - Easy to understand and easy to use for every age group.
- Notification alerts - About plan expiration and reminders along with renewal window for next plan.
- Recommendations - According to customers' history of internet plans, we can suggest optimum plans for their respective needs.
- Access Revoke - Restrict specific user privileges.

Questionnaire

Internet Service provider Survey

We would love to hear your thoughts or feedback on how we can improve your experience!

* Required

1. Name *

2. Email

3. Age group *

Mark only one oval.

- 15-18
- 18-30
- 30-50
- 50 or above

4. What are you using currently? *

Mark only one oval.

- Broadband connection
- Cellular network connection

5. What is your operator circle? *

Mark only one oval.

- Andhra Pradesh & Telangana
- Assam
- Bihar & Jharkhand
- Delhi
- Gujarat
- Haryana
- Himachal pradesh
- Jammu & Kashmir
- Karnataka
- Kerala & Laksh
- Kolkata
- MP & Chhattisgarh
- Maharashtra & Goa
- Mumbai
- North East
- Odisha
- Punjab
- Rajasthan
- Tamil Nadu
- UP (East)
- UP (West)
- West bengal

6. What is your medium for accessing internet? *

Check all that apply.

- WiFi
- Satellite
- Mobile phone data
- USB Dongle
- Optic Fibre

Other:

7. Which company's internet service you are using? *

Check all that apply.

- Jio
- Airtel
- BSNL
- Idea(Vi)
- Tikona
- Hathway

Other:

8. What Network speed you are getting in your circle? *

Check all that apply.

- 2G
- 3G
- 4G

9. How is the Quality of service provided by your ISP? *

Mark only one oval.

- Excellent
- Very good
- Good
- Fair
- Poor

10. What do you usually use the internet for? *

Check all that apply.

- Internet Surfing
- Streaming videos
- Downloading Movies, songs and games
- Social Media
- Email
- Online studies
- Online gaming

Other:

11. What speed options you prefer? *

Check all that apply.

- Upto 40 Mbps
- Upto 100 Mbps
- Upto 200 Mbps
- Upto 500 Mbps
- Upto 1 Gbps

Other:

12. Are there any problems you are facing with your current internet provider? *

Check all that apply.

- Network issue
- Coverage issue
- Speed issue
- Billing issue

Other:

13. Are you planning to switch from your current Internet service provider? *

Mark only one oval.

- Yes
- No
- Maybe

14. If yes, what is the reason?

15. Would you like follow ups from your internet provider? *

Mark only one oval.

- Yes
- No
- Maybe

16. If yes, how frequent follow ups you need?

Mark only one oval.

- Weekly
- Monthly
- Quarterly
- Yearly

17. Are you getting the services that were advertised ? *

Mark only one oval.

- Yes
- No
- Maybe

18. Would you please check the speed by going to www.fast.com, wait for few seconds and write it down here?

19. Feedback Type

Mark only one oval.

- Comments
- Questions
- Bug Reports
- Feature Request

20. Feedback

21. Suggestions for improvement

This content is neither created nor endorsed by Google.

Google Forms

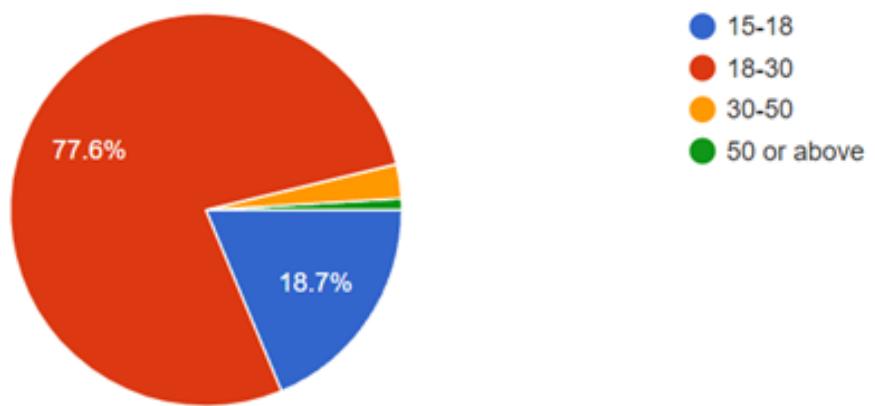
Link to our google form:<https://forms.gle/81tgy4KkRzvNrJ9>

Summary

According to our statistics, we can see that our major user base belongs to the age group of 18-30.

Age group

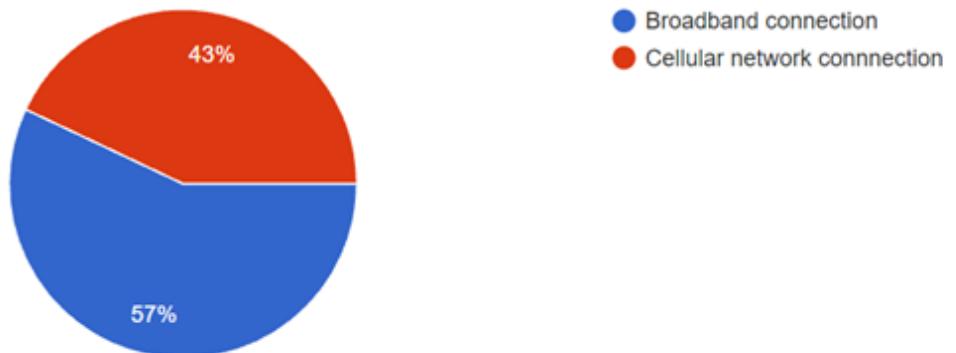
107 responses



We can see people are relying on both and preferences are insignificant.

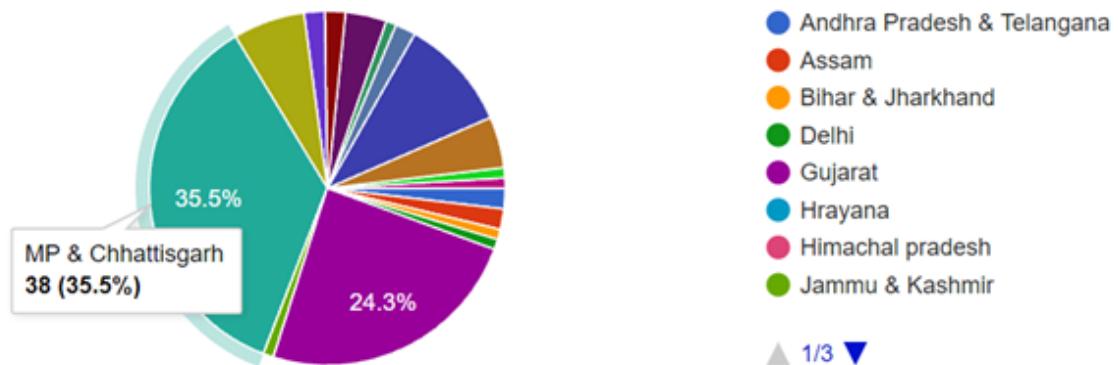
What are you using currently?

107 responses



What is your operator circle?

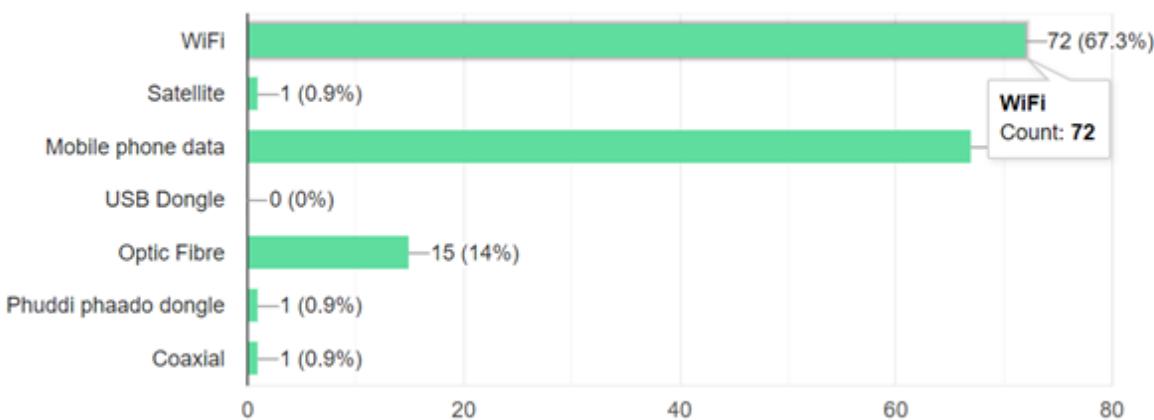
107 responses



▲ 1/3 ▼

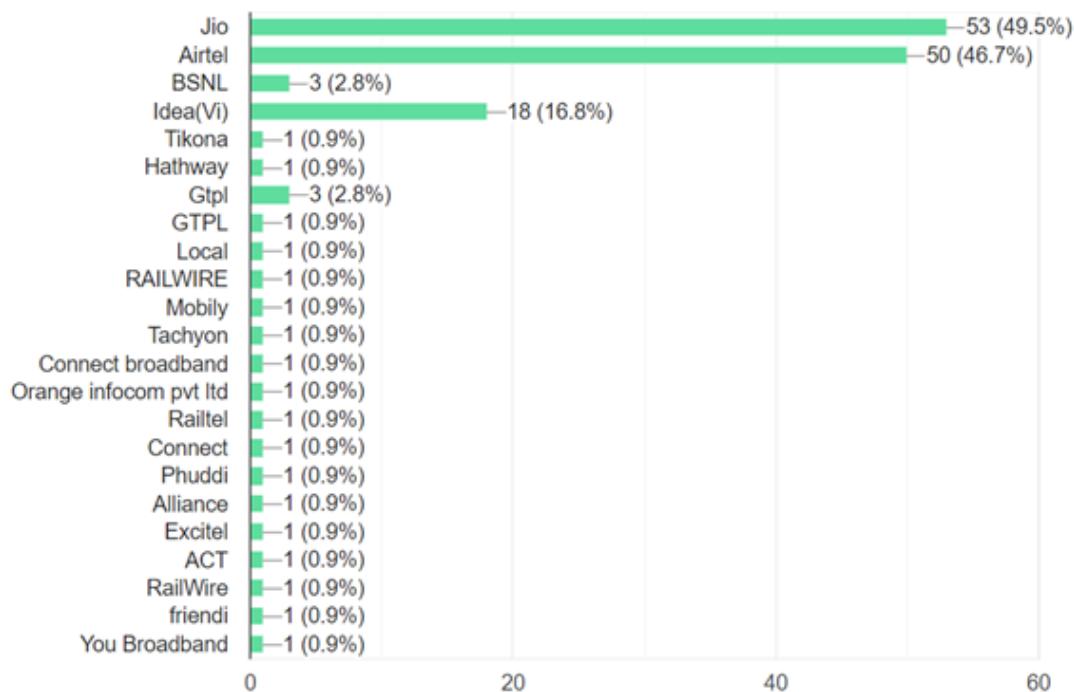
What is your medium for accessing internet?

107 responses



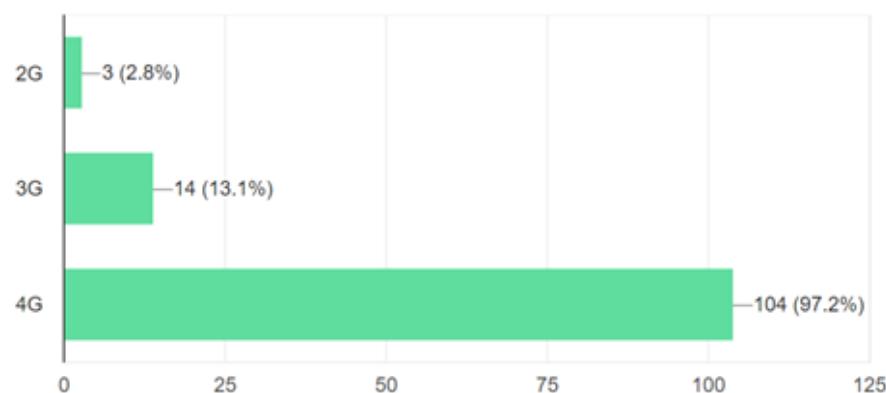
Which company's Internet service you are using?

107 responses



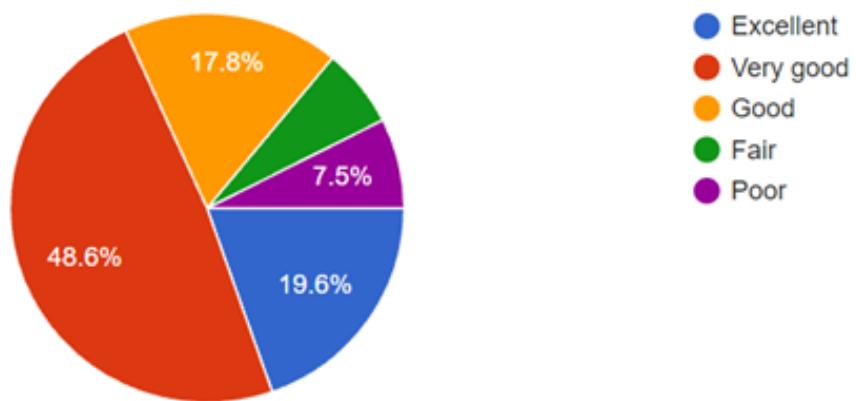
What Network speed you are getting in your circle?

107 responses



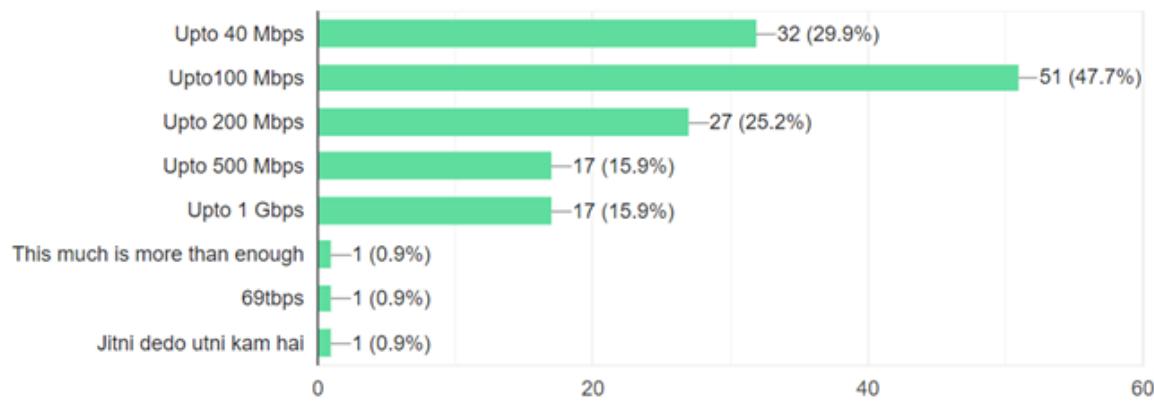
How is the Quality of service provided by your ISP?

107 responses



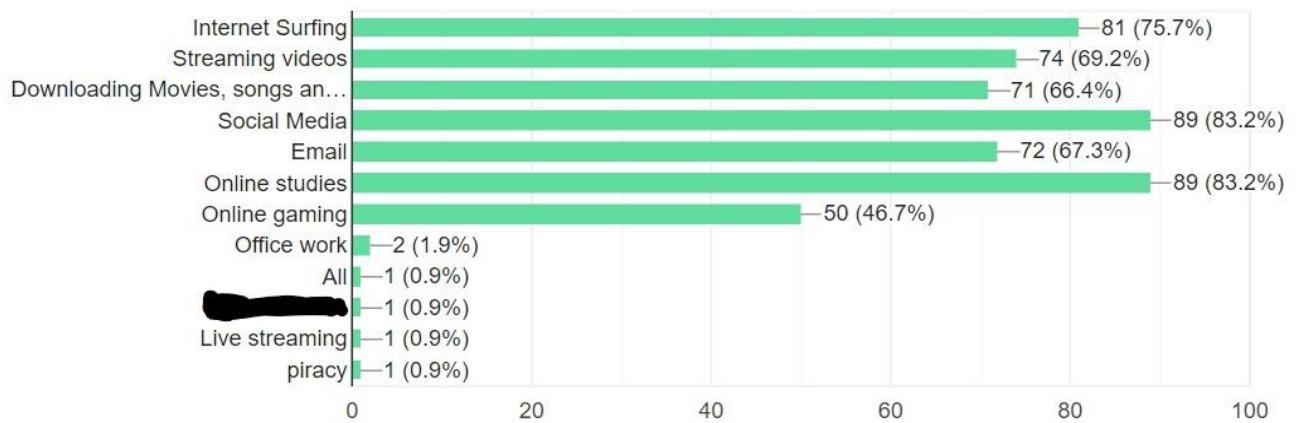
What speed options you prefer?

107 responses



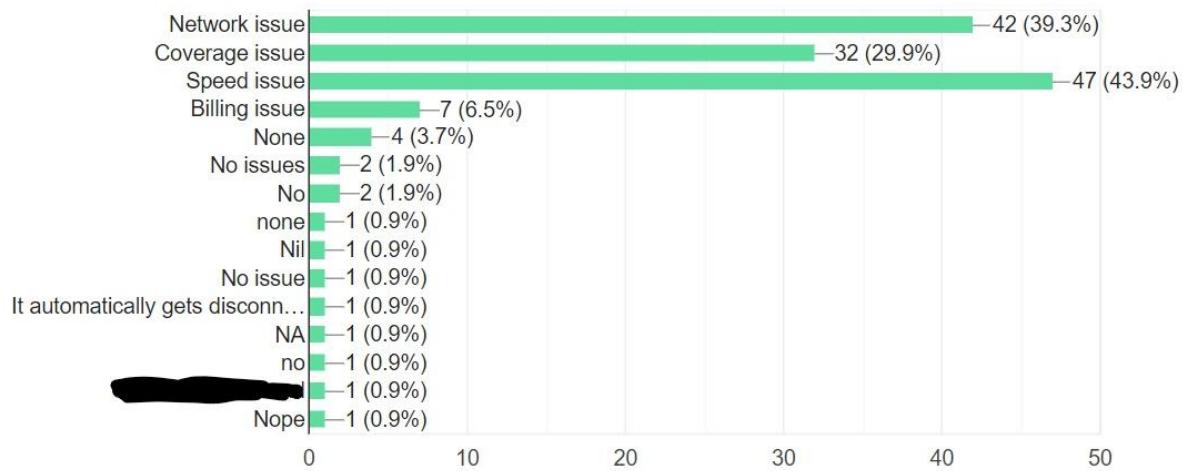
What do you usually use the internet for?

107 responses



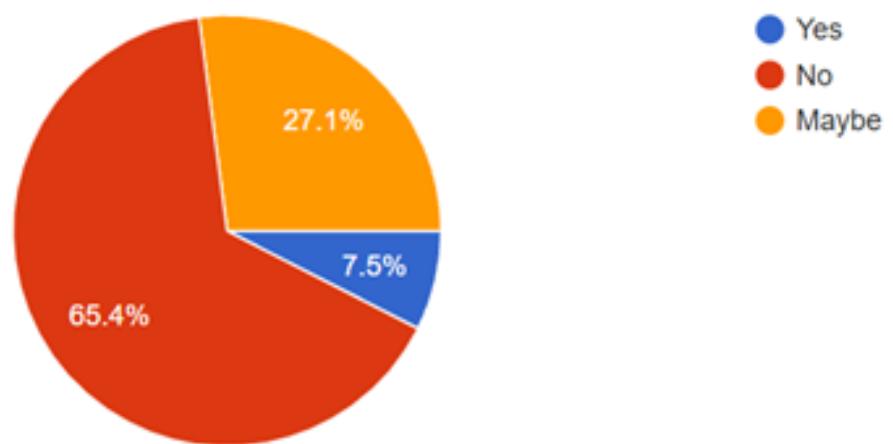
Are there any problems you are facing with your current internet provider?

107 responses



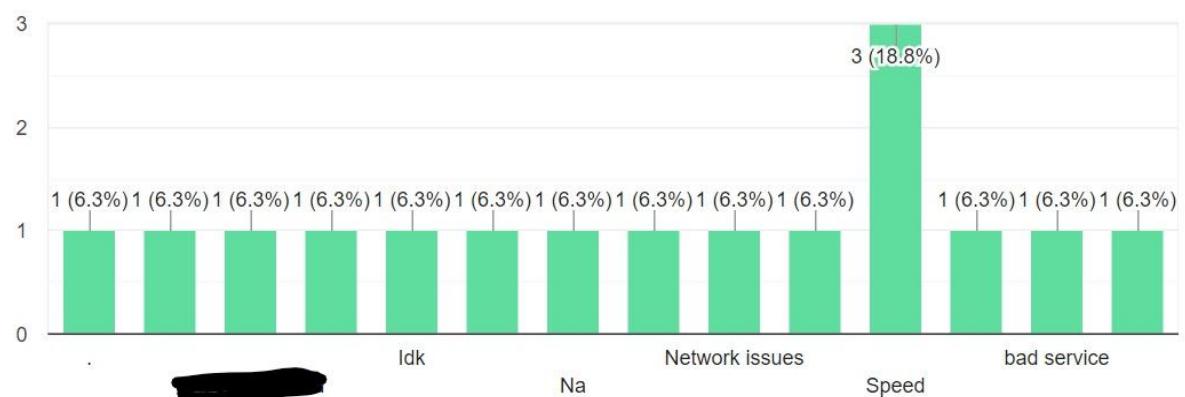
Are you planning to switch from your current Internet service provider?

107 responses



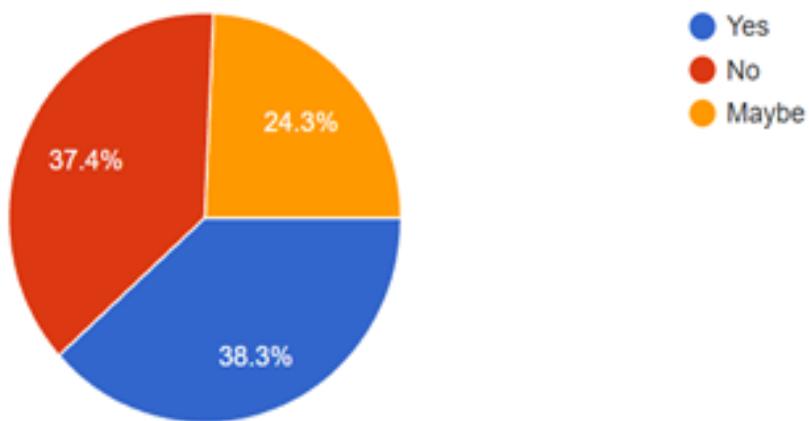
If yes, what is the reason?

16 responses



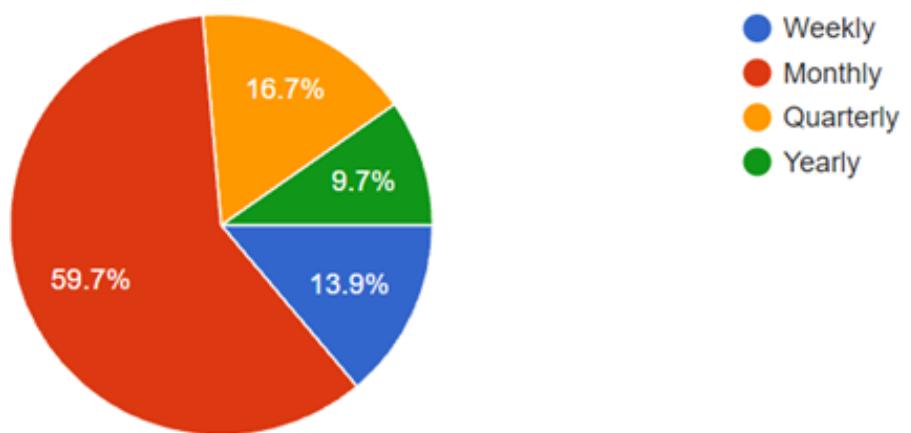
Would you like follow ups from your internet provider?

107 responses



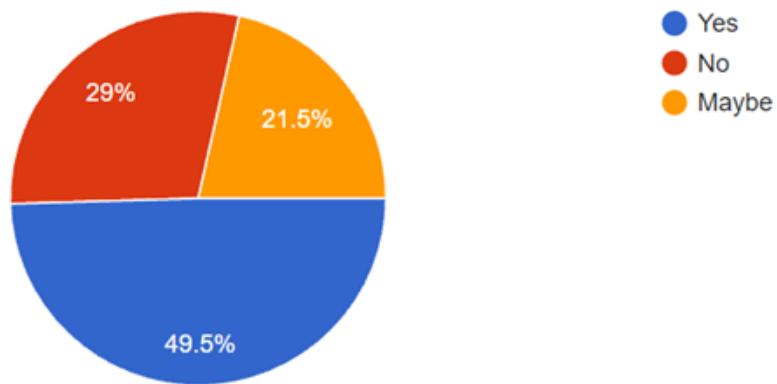
If yes, how frequent follow ups you need?

72 responses



Are you getting the services that were advertised ?

107 responses



Would you please check the speed by going to www.fast.com, wait for few seconds and write it down here?

71 responses

110 Mbps

31 Mbps

28

7.2 mbps

90 Mbps

6.2 Mbps

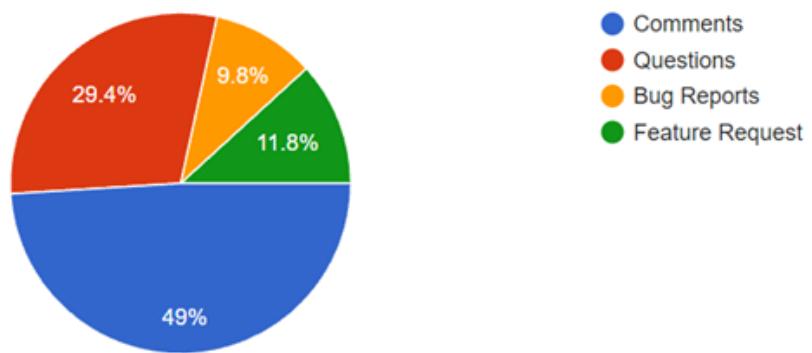
4.2mbps

85

1byte

Feedback Type

51 responses



Feedback

22 responses

The internet service provided is good but if network is low, good internet makes no sense

It's going good, I just hope it was cheaper and I'd get more network coverage in underdeveloped areas

Idk

I think the services are pretty good as they were promised

Noice

Nothing

My internet is good aur kuch nahi chiye

Poor connection in my area

looking forward for a good Internet coverage

Suggestions for improvement

21 responses

They can make some cheap plans

It's really noiceee, i mean I enjoyed flexing my internet lol-

More network stability sholud be there!!!

Just keep the internet connection stable

Weekly follow up from the customers should be taken

No improvement required

Improve the internet speed

Wi-Fi should never stop working

increase the coverage area

Combined Requirements gathered from the response

- Coverage Speed
- Cheaper Plans
- Monthly follow-ups
- Network stability
- Tracking poor connections

Observations

IT Solutions: Observations

Project Reference: S3/T8/2021/10

(**Role Play**)

Observations by: Archana Todkar (IT Solutions- Company employee)

Date: 07/10/2021

Time: 12:00PM

Duration: 45 minutes

Observations:

We have tried to note down our observations sent by our observer on site. We have concluded that though our ISP is working well and faring well with the audience, there still remain some issues. We have tried to list them all out including some rules and regulations that we noticed the company wants to adhere to strictly.

- There are glitches observed when payment is processed.
In some cases, payments haven't been received once they've been debited from the user's end, and in order to ensure transparency in the payment aspect, working on our software along with suggested collaboration with some payment companies.
- Payment and plan details don't get updated when paying from elsewhere.
- As the technology has grown, we need to make sure from our end that we have used those growing technologies to our best so that our system can be easily used by the users.
- Service availability during lockdown was not upto the mark.

- It is not known to the users when a network outage occurs or how long it will take to resolve that.
- Some people have their own significant dependents who need supervision(kids) and guidance(elderly) by tracking and having access to their data plans as well. So we think this feature should be incorporated in the new systems by which they can have dependant's account access added with one's own login account.
- By adding the above feature they will be able to make changes to current plans, recharges and control the data limits of their dependents.
- Online payments which are done through banks should provide additional benefits to the users.

Combined Requirements gathered from the observations

- Provide frequent updates to solve bugs and glitches.
- Collaboration with companies like paytm, to ensure the update of payment information done from our application.
- Tie Ups with different banks in order to provide extra benefits, when they use their portal for payment.
- More than one user access from one device.
- Child supervision feature for data usage limit.
- Notification alert on registered mobile number regarding outage information.

C. Create Fact-Finding Chart

Objective	Technique	Subject	Time
Variation in speed	Interview, background reading	Resident	35 minutes
To get a brief idea of the network range our ISP providers	Interview, observations	Institutional staff	45 minutes
To investigate occasional internet glitches	Interview, observations	Observer	45 minutes
To establish how organizations want their database to be maintained	Background reading, Interview	Employees	45 minutes
To understand how the payment takes place	Interview	CEO of Cell Tech, Institute	50 minutes
How the process of payment takes place	Interview	Resident	35 minutes

D. List Requirements

1. Simple User Interface - The product should be easy to understand and operate by each and every user.
2. Security - There should be a proper database management system to preserve database integrity , confidentiality and availability.
3. Privacy - All personal information regarding users should be kept confidential to protect them.
4. Backup and Recovery - Establishing a backup and recovery system helps organizations avoid data loss.
5. Frequent Updates - To detect and resolve bugs and introduce new features related to the product.
6. Access Revoke -
7. User Specific Access - Every user class should have different access control according to their given roles, user privileges, and object privileges
8. Multiple user login access -
9. Collaboration with payment companies
10. Collaboration with banks
11. Dependant user supervision

E. User Classes and Characteristics

1. Technical Team / Administrations

These users need full access to the database to create and maintain the system according to their needs and the end-user needs. Some might need to view data while others need to be able to modify the data.

2. Application users / Customers

These users need to know how to use the product or the menu-driven application to perform their desired task regardless of the knowledge of how the system works.

F. Operating Environment

Hardware, software and connectivity requirements

Our database runs on all of the major operating systems like Linux, Windows, etc. But for that, as a prerequisite, PGAdmin must be installed in the system.

External Interface Requirements

1. Our product and the database which is inclusive of all the things related to it has been made such that it is easy to understand, we can access and note all the plans that a user has opted for just by entering their customer ID without disclosure of any personal details.
2. Since it is a huge database, it can run on any laptop or computer with good storage and a minimum of 4GB RAM.
3. Our only software requirements include the installation of PGAdmin since we need it to run the queries and any browser compatible with it.

G. Product Functions

- We have installed a proper database that stores all the information relevant to our product and its users, right from their customer ID to the purchased plan.
- We have made our database such that it is easy to modify the user's purchased plans, delete customer IDs in case of termination of services and add new customer IDs.
- Our function for payment resolution is such that we find it easy to deal with payment issues in case there are any since we can easily track which payments are done, due and pending.
- The database has been built with putting the security of users as our topmost priority and so can only be used and seen by authorized officials.

H. Privileges

1. User data can only be accessed by the user holding that customer ID and he can see his payment status.
2. Most of the details of the user are only related to the plan purchased, payment, and their preferences, eliminating the personal information available on the user since that can cause security threats.
3. The database can only be updated by an official having the unique, required login ID and password.
4. Recovery of data in case of any glitches has already been taken care of by our technical support and development team.
5. We have also attached a function to view which employees of our technical team helped out a user deal with issues when in need to reward them.

I. Assumptions

- Most of our users expect utmost privacy and security since the cybercrime rates have been skyrocketing.
- In case of some confusion, the user might want to rethink his selection of plan details, etc. For that, he might want and expect the company to reach out and resolve their technical issues instead of suggesting newer and more expensive plans.
- In the case of corporations, it is expected from our end that the user has the specific hardware and software to run this application and if not is willing to install it.

J. Business Constraints

Our ISP mostly does not exhibit constraint related limitations when first installed. Over time, as technology, bandwidth-consuming applications, and subscriber expectations mature, constraints are eventually discovered. We and our team have tried to deal with these issues as much as we can and plan on improving them by constantly conducting surveys and understanding the plans other ISPs have used. Based on these observations, we have tried to add more and more competitive schemes in the plans table of our database, all the while constantly monitoring how many users/reference IDS are present in each one.

Another constraint we often suffer from includes issues regarding security, when big corporations like Cell-Tech approach us, their main cause of hesitation is security regarding our database. We have tried to face this problem head-on by taking high-security measures and giving access to the database to only a few certified and trusted employees. In case of any mishap, we will provide assurance of data recovery.

The security management systems we use are governed by legitimate security policies which include all the data security techniques we have for protection from

malicious software, database threats like human error, malware attack, physical damage, liquid damage, hard drive damage, power outages, system thefts, natural disasters and many more.

Lastly, finance has to be discussed between both parties to execute the project.

Section 2:

Noun Analysis

I. Noun (& Verb) Analysis.

Table.1

Extracted Nouns and Verbs from Problem Description

NOUN	VERB
Proper database	Production
Design	Accuracy
Product	Efficiency
User	Privacy
tech support	Security
Information	Classified
Price	Affordability
Usage	Work
Technology	Advancement
Security	Protection
Efficient manner	Properly
Storage issue	Risk
Large data	Space

Development team	Responsibility
Questionable activities	Dangerous
Major role	Duty
Up to date	Punctual
Company	Management
Glitch	Restoration
Uploading load	Speed
Proper connection	Clarity
Updation	Accuracy
Back up	Recovery
Database	Informative
People	Concern
Data loss	Breach
Document	Verification
ISP(big corporations)	Well-built
Payment	Facility
Speed	Fast
Range	Connectivity
Product	Uniqueness
Management	Handling

Crystal like clarity	Transparency
Customers	Reach
Constant modifications	Updates
Workplace environment	Friendly
Table	Represent

Table.2

Accepted Noun & Verbs list

Candidate entity set	Candidate attribute set	Candidate relationship set
Centre	<ul style="list-style-type: none"> ● State ● Pin code ● City ● Centre_id ● Centre_name 	Company information
Customer support	<ul style="list-style-type: none"> ● Complaint_id ● Complaint status ● Helpline number ● Customer Feedback 	Service help, relationship, complaint resolution, two-way communication

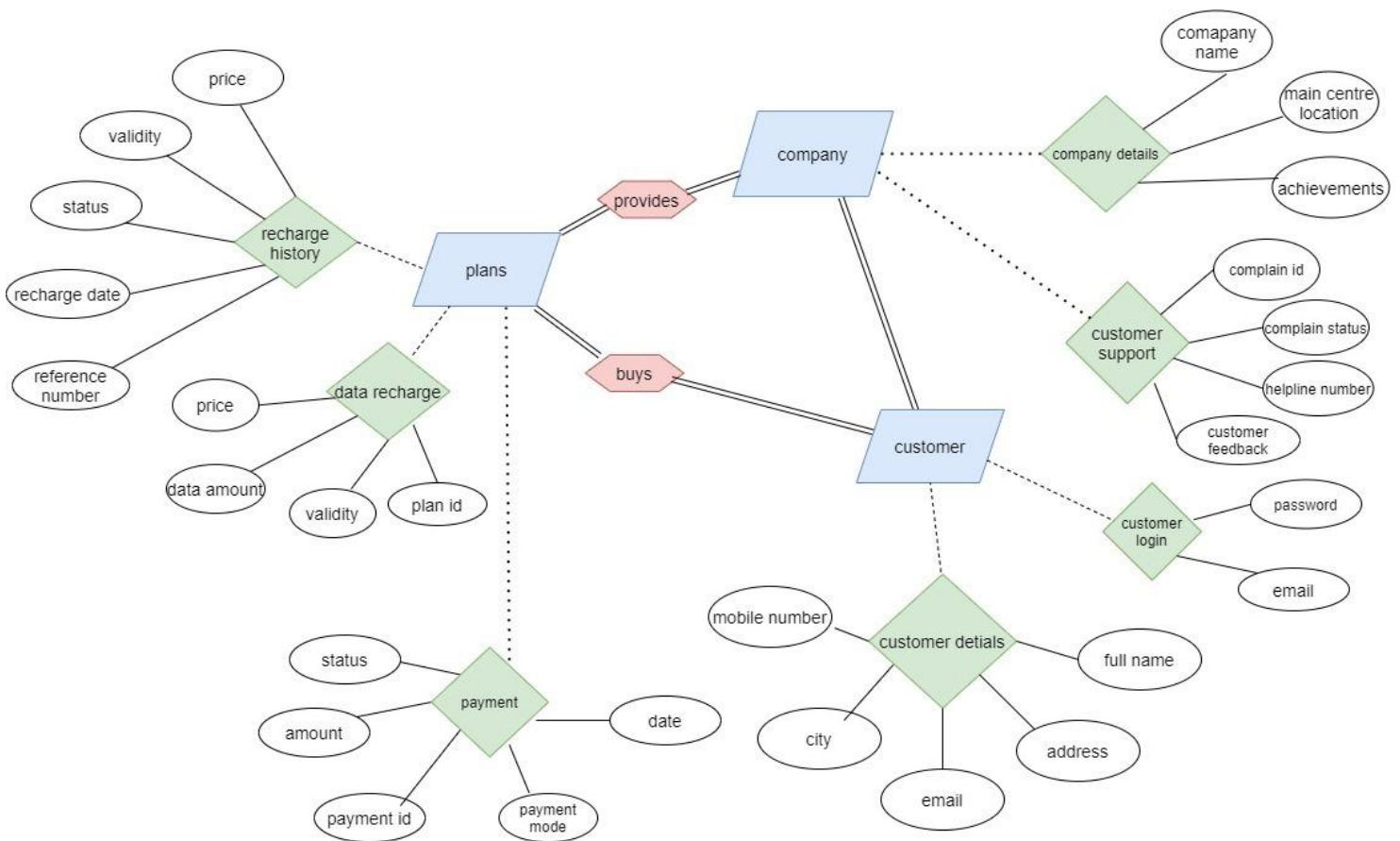
Recharge history	<ul style="list-style-type: none"> ● reach_history_id 	Recharge history
Recharge Plans	<ul style="list-style-type: none"> ● reach_plan_id ● Data amount ● Validity 	Recharge plans and offers
Customer	<ul style="list-style-type: none"> ● First Name ● Mobile Number ● Email ● pincode ● City ● Last Name ● Customer_id ● Active ● state 	Customer information
Technician	<ul style="list-style-type: none"> ● Vaccinated ● First name ● Last name ● Employee_id 	Service
Payment	<ul style="list-style-type: none"> ● Payment_id ● Status ● Amount ● Date ● Payment mode 	Payment information

Section3:

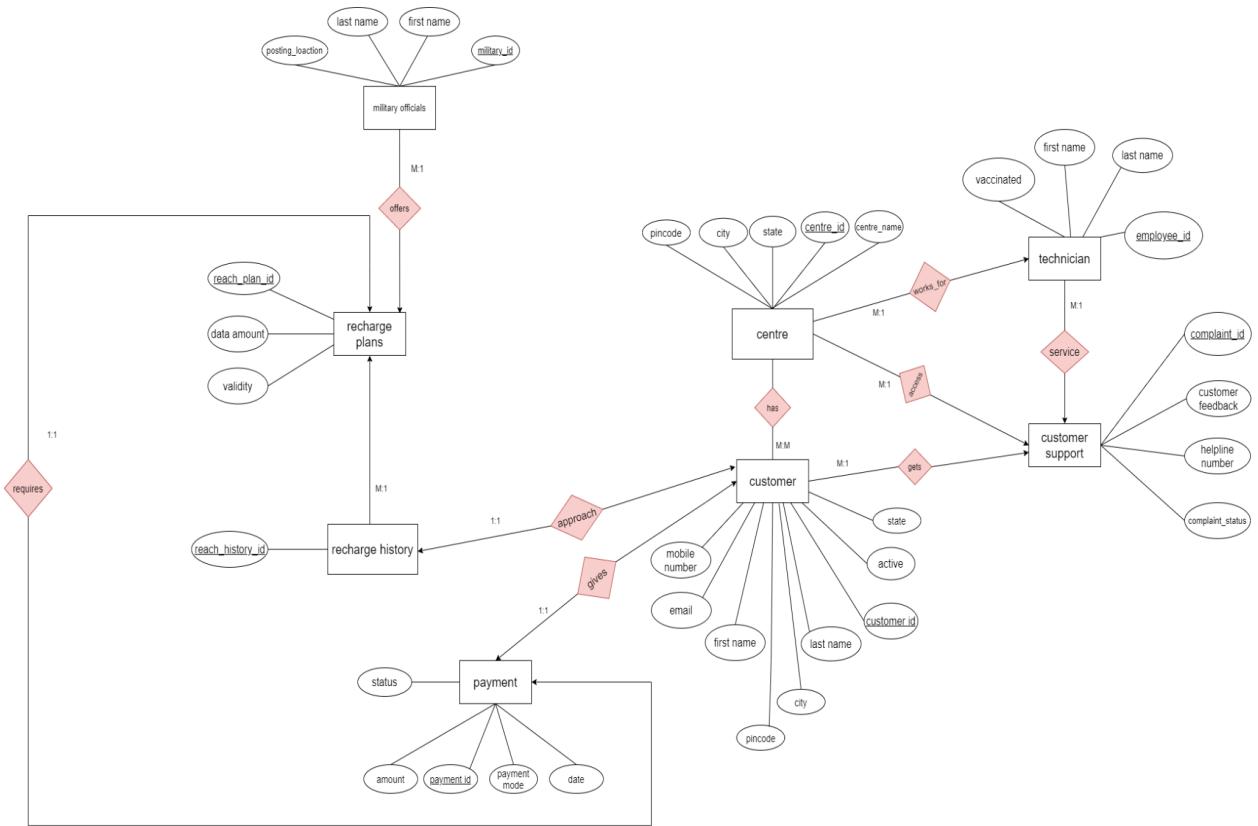
ER-Diagrams all versions

II. Develop the ER Diagram (ERD).

Version 1 of ER Diagram



Version 2 of ER Diagram

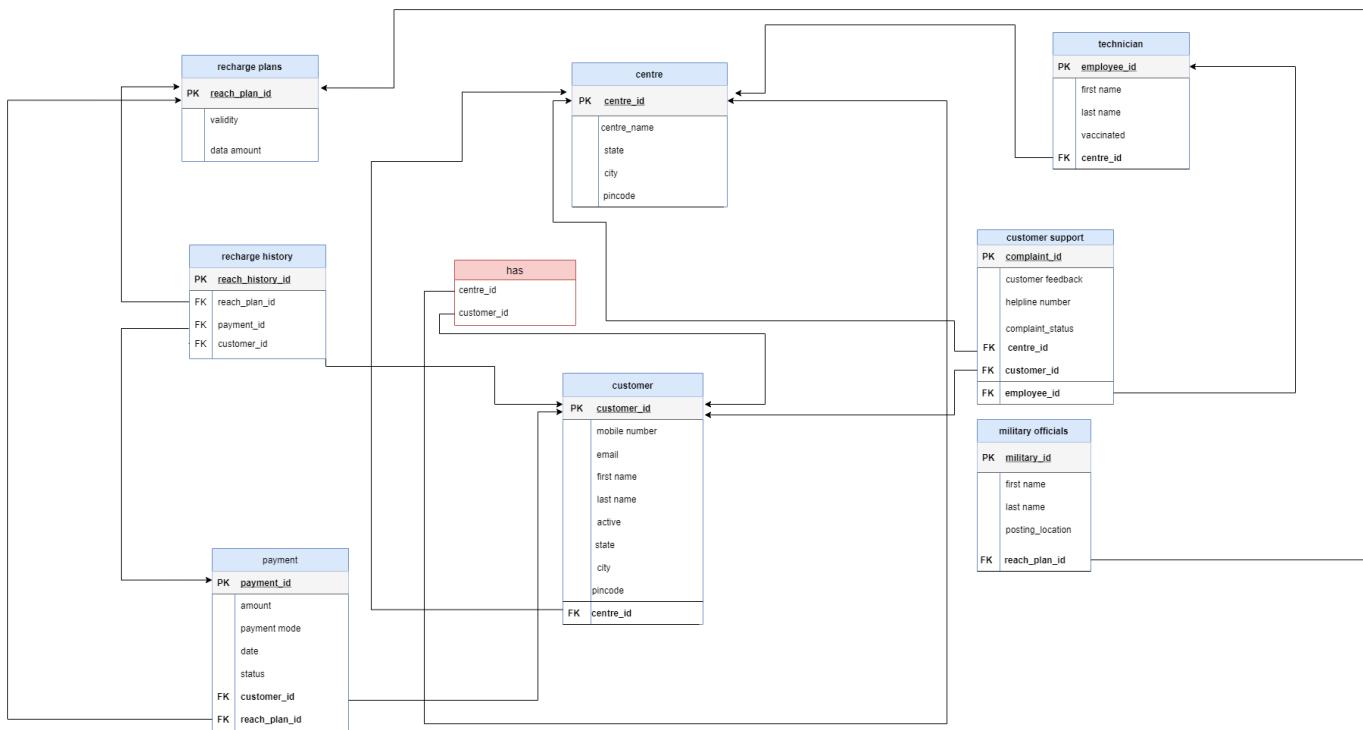


Section4:

Conversion of Final ER-Diagram to Relational Model

I.) Mapping of ER model to Relational model.

1. With the help of mapping rules discussed in the class, map all the entity and relationship sets to the corresponding relational representation.



2) Write down all the relations with the schema.

1. Recharge plans(reach_plan_id, data amount, validity)
reach_plan_id is the primary key.
2. Recharge_history(reach_history_id, reach_plan_id, payment_id,

customer_id)

reach_history_id is the primary key.

reach_plan_id, payment_id, customer_id are the foreign keys.

3. Payment (**payment_id**, amount, payment mode, date, status, reach_plan_id, customer_id)

payment_id is the primary key.

reach_plan_id is used as foreign key from recharge plans.

Customer_id is used as foreign key from customer.

4. Customer (**customer_id**, mobile number, email, first name, last name, active, state, city, Pincode, centre_id)

customer_id is the primary key.

centre_id is the foreign key from centre.

5. Customer_support (**complaint_id**, customer_feedback, helpline_number, complaint_status, center_id, customer_id, employee_id)

complaint_id is the primary key.

customer_id is the foreign key from customer.

centre_id is foreign key from centre.

employee_id is the foreign key from technician.

6. Technician(**employee_id**, centre_id, first name, last name, vaccinated)

employee_id is the primary key.

Centre_id is foreign key from centre.

7. Centre(**centre_id**, centre_name, state, city, pin code)

centre_id is a primary key.

8. Military officials(**military_id**, reach_plan_id, first name, last name , posting location)

military_id is a primary key.

reach_plan_id is used as a foreign key from recharge_plans.

9. Has(centre_id, customer_id)

Section5:

Normalisation and Schema Refinement

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

All the relations are as follows:

1. Recharge plans(**reach_plan_id**, data amount, validity)
reach_plan_id is the primary key.

2. Recharge_history(**reach_history_id**, reach_plan_id, payment_id, customer_id)
reach_history_id is the primary key.
reach_plan_id, payment_id, customer_id are the foreign keys.

3. Payment (**payment_id**, amount, payment mode, date, status, reach_plan_id , customer_id)
payment_id is the primary key.
reach_plan_id is used as foreign key from recharge plans.
Customer_id is used as foreign key from customer.

4. Customer (**customer_id**, mobile number, email, first name, last name, active, state, city, Pincode, centre_id)
customer_id is the primary key.
centre_id is the foreign key from centre.

5. Customer_support (**complaint_id**, customer_feedback, helpline_number, complaint_status, center_id, customer_id, employee_id)

complaint_id is the primary key.
customer_id is the foreign key from customer.
centre_id is foreign key from centre.
employee_id is the foreign key from technician.

6. Technician(employee_id, centre_id, first name, last name, vaccinated)

employee_id is the primary key.
Centre_id is foreign key from centre.

7. Centre(centre_id, centre_name, state, city, pin code)

centre_id is a primary key.

8. Military officials(military_id, reach_plan_id, first name, last name , posting location)

military_id is a primary key.
reach_plan_id is used as a foreign key from recharge_plans.

9. Has(centre_id, customer_id)

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

1. Recharge_plans (reach_plan_id, data amount, validity)

- (reach_plan_id) \rightarrow validity, data_amount

reach_plan_id is considered as price amount of that particular data plan in this table.

**2. Recharge_history(reach_history_id, reach_plan_id, payment_id,
customer_id)**

- Primary key: (reach_history_id) -> reach_plan_id
- Foreign key:(reach_plan_id, payment_id, customer_id)

**3. Payment (payment_id, amount, payment mode, date, status,
, reach_plan_id , customer_id)**

- Primary key:(payment_id) -> amount, payment mode, date, status
- Foreign key:(customer_id, reach_plan_id)

**4. Customer (customer_id, mobile number, email, first name, last name, active, state,
city, Pincode, centre_id)**

- Primary key:(customer_id) ->mobile number, email, first name, last name, active, pincode
- Pincode -> city, state
- Foreign key: (centre_id)

**5. Customer_support (complaint_id, customer_feedback, helpline_number,
complaint_status, center_id, customer_id, employee_id)**

- Primary key:(complaint_id) -> customer_feedback, helpline_number, complaint_status
- Foreign key:(customer_id, center_id, employee_id)

6. Technician(employee_id, centre_id, first name, last name, vaccinated)

- Primary key:(employee_id -> vaccinated, first_name, last_name)
- Foreign key:(centre_id)

7. Centre(centre_id, centre_name, state, city, pin code)

- Primary key:(centre_id) -> pin_code
- Pincode -> city, state

8. Has (centre_name, Customer_id)

9. Military officials(military_id, reach_plan_id, first name, last name , posting location)

- Primary key:(military_id) -> first name, last name, posting location
- Foreign key:(reach_plan_id)

Documentation of normalization & Schema Refinement Process upto 3NF/BCNF: This document should contain:

- **List of redundancies existing for every schema which is part of the database.**

Redundancy

After reviewing all the tables and attributes present in our database, We found address, city and state to be redundant and so we made a separate table named address which will have pincode as PK and city and state.

- **List of update, delete, and insert anomalies for every schema.**

Anomalies

- Insert: For every table, a particular ID is made such that it must be inserted and associated with other attributes in order to insert other values in that particular table.

- Update: update function anomaly does not exist as of now since each value may or may not be updated depending upon the changes either made by our company or our users.
- Delete: If the primary or Foreign key of a particular tuple is deleted then loss of data occurs since all attributes are functionally dependent on the primary key.

First Normal Form

On revisiting the list of attributes and dependencies, we have been able to conclude that our database is 1NF since there are no multi-valued attributes and have atomic values.

Second Normal Form

We have already concluded that our database is in 1NF, we have further observed that all the non-key attributes are dependent on the primary key and hence, no partial dependencies exist. Thus, the entire database is in 2NF.

Third Normal Form

Customer Table:

Here, state and city are dependent on pincode and pincode is dependent on customer_id.

1	customer_id	mobile number	email	first name	last name	active	city	state	pin code	centre_id
2	201	3883909049	lisha@centini.org	Lisha	Centini	Yes	Hosapete	Karnataka	925345	475
3	202	5101617924	arlene_klusman@gmail.com	Arlene	Klusman	Yes	Habra	West Bengal	405252	402
4	203	1295676492	alease@buemi.com	Alease	Buemi	Yes	Silvassa	Dadara and nagar haveli	733382	403
5	204	9317467010	louisa@cronauer.com	Louisa	Cronauer	Yes	Thiruvvarur	Tamil nadu	272472	419
6	205	6306162058	angella.cetta@hotmail.com	Angella	Cetta	Yes	Palakkad	Kerala	889609	405
7	206	3364849934	cgoldammer@cox.net	Cyndy	Goldammer	Yes	Port Blair	Andaman and Nicobar	877368	406
8	207	6812125351	rosio.cork@gmail.com	Rosio	Cork	Yes	Agartala	Tripura	228327	407
9	208	6976640502	ckorando@hotmail.com	Celeste	Korando	Yes	Aizwal	Mizoram	735276	408
10	209	3030568911	twana.felger@felger.org	Twana	Felger	Yes	Bhopal	Madhya Pradesh	257884	409

The non-prime attributes (city and state) transitively depend on super key(customer_id) which violates the rule of third normal form.

Hence we need to move the city and state to the new <address> table, with pincode as a Primary key.

Customer Table:

customer_id	mobile number	email	first name	last name	active	pin code	centre_id
201	3883909049	lisha@centini.org	Lisha	Centini	Yes	925345	475
202	5101617924	arlene_klusman@gmail.com	Arlene	Klusman	Yes	405252	402
203	1295676492	alease@buemi.com	Alease	Buemi	Yes	733382	403
204	9317467010	louisa@cronauer.com	Louisa	Cronauer	Yes	272472	419
205	6306162058	angella.cetta@hotmail.com	Angella	Cetta	Yes	889609	405
206	3364849934	cgoldammer@cox.net	Cyndy	Goldammer	Yes	877368	406
207	6812125351	rosio.cork@gmail.com	Rosio	Cork	Yes	228327	407
208	6976640502	ckorando@hotmail.com	Celeste	Korando	Yes	735276	408
209	3030568911	twana.felger@felger.org	Twana	Felger	Yes	257884	409

Address Table:

pincode	city	state
691792	Hosapete	Karnataka
405252	Habra	West Bengal
733382	Silvassa	Dadara and nagar haveli
615177	Thiruvarur	Tamil nadu
889609	Palakkad	Kerala
877368	Port Blair	Andaman and Nicobar
228327	Agartala	Tripura
735276	Aizwal	Mizoram
257884	Bhopal	Madhya Pradesh

Centre Table:

The same is done for Centre table.

centre_id	centre_name	pincode
401	celltech_Hosapete	691792
402	celltech_Habra	405252
403	celltech_Silvassa	733382
404	celltech_Tiruvarur	615177
405	celltech_Palakkad	889609
406	celltech_Port Blair	877368
407	celltech_Agartala	228327
408	celltech_Aizwal	735276
409	celltech_Bhopal	257884

After decomposing these tables our entire database is in 3NF.

- Write down final relations with the schema. i.e., R1(A1, A2, A3,... An). Make sure to underline the PK attributes.

1. Recharge_plans (reach_plan_id, data amount, validity)

- (reach_plan_id) -> validity, data_amount

reach_plan_id is considered as the price amount of that particular data plan in this table.

2. Recharge_history (reach_history_id, reach_plan_id, payment_id, date, payment_mode, validity, customer_id)

- Primary key: (reach_history_id)
- Foreign key:(reach_plan_id, payment_id, customer_id)

3. Payment (payment_id, reach_plan_id, amount, payment mode, date, status, customer_id)

- Primary key:(payment_id) -> amount, payment mode, date, status
- Foreign key:(customer_id,reach_plan_id)

4. Customer (customer_id, mobile number , email, first name, last name, active, centre_id, pincode)

- Primary key:(customer_id) -> mobile number , email, first name, last name, active
- Foreign key: centre_id, pincode

5. Address (pin code, city , state)

- Primary key:(pin code) -> city, state

6. Customer_support (complaint_id, customer_id, customer_feedback, helpline_number, complaint_status, center_id, employee_id)

- Primary key:(complaint_id) -> customer_feedback, helpline_number, complaint_status
- Foreign key:(customer_id, center_id, employee_id)

7. Technician (employee_id, centre_id, first name, last name

, vaccinated)

- Primary key:(employee_id -> first name, last name, vaccinated.
- Foreign key:(centre_id,)

8. Centre(centre_id, centre_name, pincode)

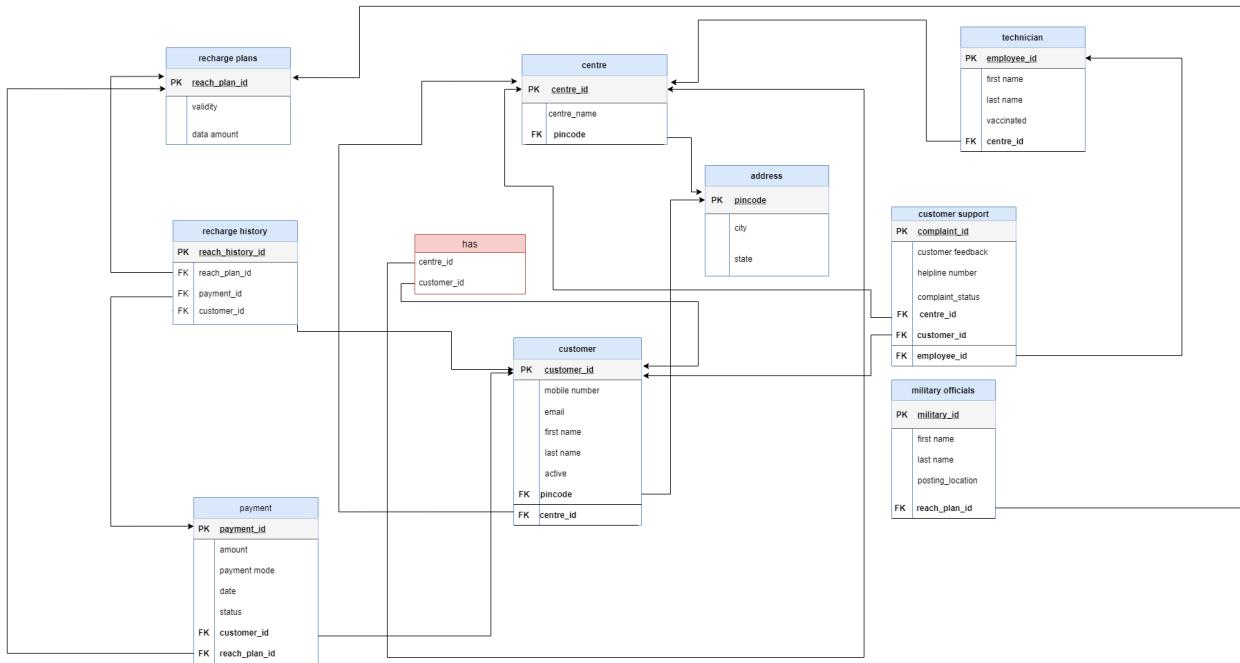
- Primary key:(centre_id) -> centre_name
- Foreign key:(pincode)

10. Has (centre_id, Customer_id)

11. Military officials (military_id, reach_plan_id, first name, last name , posting location)

- Primary key:(military_id) -> first name, last name, posting location
- Foreign key:(reach_plan_id)

→ Refined Schema:



II. Re-write DDL Scripts.

→ DDL Snapshots: Put the snapshot of all tables after creating them inside Postgres with DDL.

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

- Schemas:** The current schema is `ISP`.
- Tables:** The table `recharge plans` is selected in the left sidebar under the `Tables (1)` section.
- Query Editor:** The SQL code for creating the table is displayed:

```

1 SET SEARCH_PATH TO ISP;
2 CREATE TABLE IF NOT EXISTS "ISP"."recharge plans"
3 (
4     reach_plan_id bigint NOT NULL,
5     validity character varying(40) COLLATE pg_catalog."default" NOT NULL,
6     "data amount" character varying(40) COLLATE pg_catalog."default" NOT NULL,
7     PRIMARY KEY (reach_plan_id)
8 );
9

```
- Messages:** The message indicates the query was executed successfully: `Query returned successfully in 1 secs 473 msec.`

pgAdmin 4

File Object Tools Help

Browser

- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Sequences
- > Tables (2)
 - > address
 - > Columns (3)
 - pincode
 - city
 - state
 - > Constraints (1)
 - address_pkey
 - > Indexes
 - > RLS Policies
 - > Rules
 - > Triggers
 - > recharge plans
- > Trigger Functions
- > Types
- > Views
- > mobile_dealership_db

Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```

1
2 CREATE TABLE IF NOT EXISTS "ISP".address
3 (
4     pincode bigint NOT NULL,
5     city character(50) COLLATE pg_catalog."default" NOT NULL,
6     state character(50) COLLATE pg_catalog."default" NOT NULL,
7     PRIMARY KEY (pincode)
8 );
9
10

```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 109 msec.

pgAdmin 4

File Object Tools Help

Browser

- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Sequences
- > Tables (3)
 - > address
 - > centre
- > centre
 - > Columns (3)
 - centre_id
 - centre_name
 - pincode
 - > Constraints (2)
 - centre_pincode_fkey
 - centre_pkey
 - > Indexes
 - > RLS Policies
 - > Rules
 - > Triggers
- > recharge plans
- > Trigger Functions
- > Types
- > Views
- > mobile_dealership_db

Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```

1 CREATE TABLE IF NOT EXISTS "ISP".centre
2 (
3     centre_id bigint NOT NULL,
4     centre_name character varying(50) COLLATE pg_catalog."default" NOT NULL,
5     pincode bigint NOT NULL,
6
7     PRIMARY KEY (centre_id),
8     FOREIGN KEY (pincode)
9         REFERENCES "ISP".address (pincode)
10        ON UPDATE CASCADE
11        ON DELETE CASCADE
12 );
13
14

```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 180 msec.

pgAdmin 4

File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```

1 CREATE TABLE IF NOT EXISTS "ISP".customer
2 (
3     customer_id bigint NOT NULL,
4     "mobile number" bigint NOT NULL,
5     email character varying(50) COLLATE pg_catalog."default" NOT NULL,
6     "first name" character(50) COLLATE pg_catalog."default" NOT NULL,
7     "last name" character(50) COLLATE pg_catalog."default" NOT NULL,
8     active character(10) COLLATE pg_catalog."default" NOT NULL,
9     pincode bigint NOT NULL,
10    centre_id bigint NOT NULL,
11
12    PRIMARY KEY (customer_id),
13    FOREIGN KEY (centre_id)
14        REFERENCES "ISP".centre (centre_id)
15        ON UPDATE CASCADE
16        ON DELETE CASCADE
17
18 );

```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 115 msec.

pgAdmin 4

File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```

1 CREATE TABLE IF NOT EXISTS "ISP".technician
2 (
3     employee_id bigint NOT NULL,
4     "first name" character(50) COLLATE pg_catalog."default" NOT NULL,
5     "last name" character(50) COLLATE pg_catalog."default" NOT NULL,
6     vaccinated character(50) COLLATE pg_catalog."default" NOT NULL,
7     centre_id bigint NOT NULL,
8
9     PRIMARY KEY (employee_id),
10    FOREIGN KEY (centre_id)
11        REFERENCES "ISP".centre (centre_id)
12        ON UPDATE CASCADE
13        ON DELETE CASCADE
14
15 );

```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 1 secs 274 msec.

pgAdmin 4

File Object Tools Help

Browser

- Sequences
- Tables (6)
 - address
 - centre
 - customer
 - customer support**
 - Columns (7)
 - complaint_id
 - customer feedback
 - helpline number
 - complaint status
 - centre_id
 - customer_id
 - employee_id
 - Constraints (4)
 - customer support_centre_id_fkey
 - customer support_customer_id_fkey
 - customer support_employee_id_fkey
 - customer support_pkey
 - Indexes
 - RLS Policies
 - Rules
 - Triggers
- recharge plans
- technician
- Trigger Functions

Query Editor Query History

```

1 CREATE TABLE IF NOT EXISTS "ISP"."customer support"
2 (
3     complaint_id bigint NOT NULL,
4     "customer feedback" integer NOT NULL,
5     "helpline number" bigint NOT NULL,
6     "complaint status" character(50) COLLATE pg_catalog."default" NOT NULL,
7     centre_id bigint NOT NULL,
8     customer_id bigint NOT NULL,
9     employee_id bigint NOT NULL,
10
11    PRIMARY KEY (complaint_id),
12    FOREIGN KEY (centre_id)
13        REFERENCES "ISP".centre (centre_id)
14        ON UPDATE CASCADE
15        ON DELETE CASCADE,
16    FOREIGN KEY (customer_id)
17        REFERENCES "ISP".customer (customer_id)
18        ON UPDATE CASCADE

```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 111 msec.

pgAdmin 4

File Object Tools Help

Browser

- address
- centre
- customer
- customer support
- payment**
- Columns (7)
 - payment_id
 - amount
 - payment_mode
 - date
 - status
 - customer_id
 - reach_plan_id
- Constraints (3)
 - payment_customer_id_fkey
 - payment_pkey
 - payment_reach_plan_id_fkey
- Indexes
- RLS Policies
- Rules
- Triggers
- recharge plans
- technician
- Trigger Functions
- Types
- Views
- mobile_dasharchin dh

Query Editor Query History

```

1 CREATE TABLE IF NOT EXISTS "ISP".payment
2 (
3     payment_id bigint NOT NULL,
4     amount integer NOT NULL,
5     payment_mode character varying COLLATE pg_catalog."default" NOT NULL,
6     date date NOT NULL,
7     status character varying COLLATE pg_catalog."default" NOT NULL,
8     customer_id bigint NOT NULL,
9     reach_plan_id bigint NOT NULL,
10
11    PRIMARY KEY (payment_id),
12    FOREIGN KEY (customer_id)
13        REFERENCES "ISP".customer (customer_id)
14        ON UPDATE CASCADE
15        ON DELETE CASCADE,
16    FOREIGN KEY (reach_plan_id)
17        REFERENCES "ISP"."recharge plans" (reach_plan_id)
18        ON UPDATE CASCADE

```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 518 msec.

pgAdmin 4

File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Scratch Pad

```

1 CREATE TABLE IF NOT EXISTS "ISP"."recharge history"
2 (
3     reach_history_id bigint NOT NULL,
4     reach_plan_id integer NOT NULL,
5     payment_id bigint NOT NULL,
6     customer_id bigint NOT NULL,
7     PRIMARY KEY (reach_history_id)
8 );
9

```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 549 msec.

pgAdmin 4

File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Scratch Pad

```

1 CREATE TABLE IF NOT EXISTS "ISP"."military officials"
2 (
3     military_id character varying COLLATE pg_catalog."default" NOT NULL,
4     "first name" character varying COLLATE pg_catalog."default" NOT NULL,
5     "last name" character varying COLLATE pg_catalog."default" NOT NULL,
6     posting_location character varying COLLATE pg_catalog."default" NOT NULL,
7     reach_plan_id bigint NOT NULL,
8
9     PRIMARY KEY (military_id),
10    FOREIGN KEY (reach_plan_id)
11        REFERENCES "ISP"."recharge plans" (reach_plan_id)
12        ON UPDATE CASCADE
13        ON DELETE CASCADE
14 );
15

```

Data Output Explain Messages Notifications

CREATE TABLE

Query returned successfully in 302 msec.

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane is open, displaying a tree view of database objects. A table named 'has' is selected. The 'Query Editor' tab in the center contains the SQL code for creating the 'has' table:

```

1 CREATE TABLE IF NOT EXISTS "ISP".has
2 (
3     centre_id bigint NOT NULL,
4     customer_id bigint NOT NULL,
5     PRIMARY KEY (centre_id, customer_id)
6 );
7

```

The 'Messages' tab at the bottom shows the message: 'CREATE TABLE'. Below it, it says 'Query returned successfully in 157 msec.'

→ Data Snapshots: Put the snapshot of select * queries of all the tables after insertion of data. Mention the number of records of each table.

The screenshot shows the pgAdmin 4 interface. The 'Browser' pane on the left has 'Tables (10)' selected. The 'Query Editor' tab in the center contains the SQL code: 'SELECT * FROM "ISP"."recharge_plans"'. The 'Data Output' tab is selected, showing the results of the query:

	reach_plan_id	validity	data amount
1	800149	30	1 GB/day
2	800199	45	1 GB/day
3	800248	60	1 GB/day
4	800499	90	1 GB/day
5	801798	365	1 GB/day
6	800249	30	1.5 GB/day
7	800319	45	1.5 GB/day
8	800398	60	1.5 GB/day
9	800598	90	1.5 GB/day
10	800298	30	2 GB/day
11	800359	45	2 GB/day
12	800449	60	2 GB/day
13	800599	90	2 GB/day
14	802408	365	2 GB/day

pgAdmin 4

File Object Tools Help

Browser

- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Sequences
- > Tables (10)
 - > address
 - > centre
 - > customer
 - > customer support
 - > has
 - > military officials
 - > payment
 - > recharge history
 - > recharge plans
 - > technician
- > Trigger Functions
- > Types
- > Views
- > mobile_dealership_db
- > public
- > sv_db
- > Subscriptions

Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```
1 SELECT * FROM "ISP".address
```

Data Output Explain Messages Notifications

	pincode	city	state
1	691792	Hosapete	Karnataka
2	405252	Habra	West Bengal
3	733382	Silvassa	Dadara and nagar haveli
4	615177	Thiruvanur	Tamil nadu
5	889609	Palakkad	Kerala
6	877368	Port Blair	Andaman and Nicobar
7	228327	Agartala	Tripura
8	735276	Aizwal	Mizoram
9	257884	Bhopal	Madhya Pradesh
10	944272	Gangtok	sikkim
11	266890	Nagpur	Maharastra
12	398879	Panaji	Goa
13	668180	Daman	Daman and Diu
14	R06126	Shimla	Himachal Pradesh

pgAdmin 4

File Object Tools Help

Browser

- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Sequences
- > Tables (10)
 - > address
 - > centre
 - > customer
 - > customer support
 - > has
 - > military officials
 - > payment
 - > recharge history
 - > recharge plans
 - > technician
- > Trigger Functions
- > Types
- > Views
- > mobile_dealership_db
- > public
- > sv_db
- > Subscriptions

Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```
1 SELECT * FROM "ISP".centre
```

Data Output Explain Messages Notifications

	centre_id	centre_name	pincode
1	401	celitech_Hosapete	691792
2	402	celitech_Habra	405252
3	403	celitech_Silvassa	733382
4	404	celitech_Tiruvanur	615177
5	405	celitech_Palakkad	889609
6	406	celitech_Port Blair	877368
7	407	celitech_Agartala	228327
8	408	celitech_Aizwal	735276
9	409	celitech_Bhopal	257884
10	410	celitech_Gangtok	944272
11	411	celitech_Nagpur	266890
12	412	celitech_Panaji	398879
13	413	celitech_Daman	668180
14	414	celitech_Shimla	R06126

pgAdmin 4

File Object Tools Help

Browser

- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > 1.3 Sequences
- > Tables (10)
 - > address
 - > centre
 - > customer
 - > customer support
 - > has
 - > military officials
 - > payment
 - > recharge history
 - > recharge plans
 - > technician
- > Trigger Functions
- > Types
- > Views
- > mobile_dealership_db
- > public
- > sv_db
- > Subscriptions

Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```
1 SELECT * FROM "ISP"."customer"
```

Data Output Explain Messages Notifications

	customer_id	mobile_number	email	first name	last name	active	pincode	centre_id
1	201	3883909049	lisha@centini.org	Lisha	Centini	Yes	925345	475
2	202	5101617924	arlene_klusman@gmail.com	Arlene	Klusman	Yes	405252	402
3	203	1295676492	alease@buemi.com	Alease	Buemi	Yes	733382	403
4	204	9317467010	louisa@cronauer.com	Louisa	Cronauer	Yes	272472	419
5	205	6306162058	angella.cetta@hotmail.com	Angella	Cetta	Yes	889609	405
6	206	3364849934	cgoldammer@cox.net	Cyndy	Goldammer	Yes	877368	406
7	207	6812125351	rosio.cor@gmail.com	Rosio	Cork	Yes	228327	407
8	208	6976640502	ckorando@hotmail.com	Celeste	Korando	Yes	735276	408
9	209	3030568911	twana.felger@felger.org	Twana	Felger	Yes	257884	409
10	210	4344776536	estrella@aol.com	Estrella	Samu	Yes	944272	410
11	211	5455345213	dkines@hotmail.com	Donte	Kines	Yes	266890	411
12	212	6955422079	tiffny_steffensmeier@cox.net	Tiffny	Steffensmeier	Yes	398879	412
13	213	8674214997	emiceli@miceli.org	Edna	Miceli	Yes	668180	413
14	214	5203772047	sue@anl.com	Sue	Knownacki	Yes	806126	414

pgAdmin 4

File Object Tools Help

Browser

- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > 1.3 Sequences
- > Tables (10)
 - > address
 - > centre
 - > customer
 - > customer support
 - > has
 - > military officials
 - > payment
 - > recharge history
 - > recharge plans
 - > technician
- > Trigger Functions
- > Types
- > Views
- > mobile_dealership_db
- > public
- > sv_db
- > Subscriptions

Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```
1 SELECT * FROM "ISP"."technician"
```

Data Output Explain Messages Notifications

	employee_id	first name	last name	vaccinated	centre_id
1	101	Sheryl	Seymark	Yes	401
2	102	Ira	Toffetto	Yes	402
3	103	Adriana	Peacop	Yes	403
4	104	Nikita	Arnold	No	404
5	105	Simmonds	Burfield	Yes	405
6	106	Kimberli	Thurborn	Yes	406
7	107	Danie	Attersoll	Yes	407
8	108	Daniela	Paske	Yes	408
9	109	Kenneth	MacKill	Yes	409
10	110	Petronia	Assad	Yes	410
11	111	Karina	Scrane	Yes	411
12	112	Klemens	Cowing	Yes	412
13	113	Rae	Barock	Yes	413
14	114	H'arrv	Scrutter	Yes	414

pgAdmin 4

File Object Tools Help

Browser

- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Sequences
- Tables (10)
 - address
 - centre
 - customer
 - customer support
 - has
 - military officials
 - payment
 - recharge history
 - recharge plans
 - technician
- Trigger Functions
- Types
- Views
- mobile_dealership_db
- public
- sv_db
- Subscriptions

Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```
1 SELECT * FROM "ISP"."customer support"
```

Data Output Explain Messages Notifications

complaint_id	customer_feedback	helpline_number	complaint_status	centre_id	customer_id	employee_id
1	501	8	7472683523	resolved	406	206
2	502	3	6857327110	resolved	418	218
3	503	10	3019550115	resolved	405	205
4	504	2	1789802957	resolved	436	236
5	505	9	6838789722	pending	421	221
6	506	5	4982530760	in process	462	262
7	507	10	3200517367	pending	441	241
8	508	1	7111457376	in process	458	258
9	509	10	2616139008	resolved	438	238
10	510	9	7742549235	resolved	475	275
11	511	5	5082541140	pending	460	260
12	512	10	9166663587	in process	407	207
13	513	6	1983439369	resolved		
14	514	9	1477117110	resolved		

Successfully run. Total query runtime: 129 msec. 30 rows affected.

pgAdmin 4

File Object Tools Help

Browser

- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Sequences
- Tables (10)
 - address
 - centre
 - customer
 - customer support
 - has
 - military officials
 - payment
 - recharge history
 - recharge plans
 - technician
- Trigger Functions
- Types
- Views
- mobile_dealership_db
- public
- sv_db
- Subscriptions

Dashboard Properties SQL Statistics Dependencies Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

```
1 SELECT * FROM "ISP"."payment"
```

Data Output Explain Messages Notifications

payment_id	amount	payment_mode	date	status	customer_id	reach_plan_id
1	6001	149	UPI	2021-01-06	successful	201
2	6002	199	Card	2021-01-05	successful	202
3	6003	248	Wallet	2021-11-12	successful	203
4	6004	499	UPI	2021-08-15	successful	204
5	6005	798	Card	2021-09-28	successful	205
6	6006	249	UPI	2021-11-04	successful	206
7	6007	319	Card	2021-07-25	successful	207
8	6008	398	Card	2021-03-06	successful	208
9	6009	598	UPI	2021-09-02	successful	209
10	6010	298	Card	2021-04-01	successful	210
11	6011	359	Card	2021-01-30	successful	211
12	6012	449	Wallet	2021-10-05	successful	212
13	6013	599	UPI	2021-09-03	successful	213
14	6014	408	Card	2021-07-15	successful	214

pgAdmin 4

File Object Tools Help

Browser

- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Sequences
- Tables (10)
 - address
 - centre
 - customer
 - customer support
 - has
 - military officials
 - payment
 - recharge history
 - recharge plans
 - technician
- Trigger Functions
- Types
- Views
- mobile_dealership_db
- public
- sv_db
- Subscriptions

Query Editor Query History

```
1 SELECT * FROM "ISP"."recharge history"
```

Data Output Explain Messages Notifications

	reach_history_id	reach_plan_id	payment_id	customer_id
1	9001	800149	6001	201
2	9002	800199	6002	202
3	9003	800248	6003	203
4	9004	800499	6004	204
5	9005	801798	6005	205
6	9006	800249	6006	206
7	9007	800319	6007	207
8	9008	800398	6008	208
9	9009	800598	6009	209
10	9010	800298	6010	210
11	9011	800359	6011	211
12	9012	800449	6012	212
13	9013	800599	6013	213
14	9014	802408	6014	214

pgAdmin 4

File Object Tools Help

Browser

- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Sequences
- Tables (10)
 - address
 - centre
 - customer
 - customer support
 - has
 - military officials
 - payment
 - recharge history
 - recharge plans
 - technician
- Trigger Functions
- Types
- Views
- mobile_dealership_db
- public
- sv_db
- Subscriptions

Query Editor Query History

```
1 SELECT * FROM "ISP"."military officials"
```

Data Output Explain Messages Notifications

	military_id	first name	last name	posting_location	reach_plan_id
1	9RHA21GCUH	Jolanda	Hanafan	Chishtian Mandi	800100
2	J00509152C	Barrett	Toyama	Shillong	800300
3	NIXV6L6JFO	Helga	Fredicks	Naihati	800140
4	KCNQTQPWKF	Ashlyn	Pinilla	Pondicherry	800350
5	1552SX9G6K	Fausto	Agramonte	Dehri	802000
6	T8UOL5SSFC	Ronny	Caiafa	Rohtak	800150
7	6TC10X6U6	Marge	Limmel	Amravati	800350
8	D9F1UOLZYC	Norah	Waynire	Dehradun	800700
9	2247X0CQJ	Aliza	Baltimore	Pondicherry	802000
10	3CGPLP9B8J	Mozell	Pelkowski	Aizwal	800140
11	L1QWL5M3IX	Viola	Bitsule	Champdani	800350
12	YURYSFASJ0	Franklyn	Emard	Pali	802000
13	43ZQG450L9	Willodean	Konopacki	Jodhpur	802000
14	PY101WVHNR	Roskie	Silvestrini	Rawah	800140

pgAdmin 4

File Object Tools Help

Browser

- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Sequences
- > Tables (10)
 - > address
 - > centre
 - > customer
 - > customer support
 - > has
 - > military officials
 - > payment
 - > recharge history
 - > recharge plans
 - > technician
- > Trigger Functions
- > Types
- > Views

Dependents 201901190_db/postgres@PostgreSQL 10*

Query Editor Query History

1 SELECT * FROM "ISP".has

Data Output Explain Messages Notifications

centre_id	customer_id
1	201
2	202
3	203
4	204
5	205
6	206
7	207
8	208
9	209
10	210
11	211
12	212
13	213
14	214

Section6:

SQL: Final DDL Scripts, 40 SQL

Queries with

Snapshots of output of each query.

DDL Scripts for Celltech Database

1. Recharge plans

```
CREATE TABLE IF NOT EXISTS "ISP"."recharge plans"
(
    reach_plan_id bigint NOT NULL,
    validity character varying(40) COLLATE pg_catalog."default" NOT NULL,
    "data amount" character varying(40) COLLATE pg_catalog."default" NOT NULL,
    PRIMARY KEY (reach_plan_id)
);
```

2. Address

```
CREATE TABLE IF NOT EXISTS "ISP".address
(
    pincode bigint NOT NULL,
    city character(50) COLLATE pg_catalog."default" NOT NULL,
    state character(50) COLLATE pg_catalog."default" NOT NULL,
    PRIMARY KEY (pincode)
);
```

3. Centre

```
CREATE TABLE IF NOT EXISTS "ISP".centre
(
    centre_id bigint NOT NULL,
    centre_name character varying(50) COLLATE pg_catalog."default" NOT NULL,
    pincode bigint NOT NULL,
    PRIMARY KEY (centre_id),
    FOREIGN KEY (pincode)
        REFERENCES "ISP".address (pincode)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

4. Customer

```
CREATE TABLE IF NOT EXISTS "ISP".customer
(
    customer_id bigint NOT NULL,
    "mobile number" bigint NOT NULL,
    email character varying(50) COLLATE pg_catalog."default" NOT NULL,
    "first name" character(50) COLLATE pg_catalog."default" NOT NULL,
    "last name" character(50) COLLATE pg_catalog."default" NOT NULL,
    active character(10) COLLATE pg_catalog."default" NOT NULL,
    pincode bigint NOT NULL,
    centre_id bigint NOT NULL,
    PRIMARY KEY (customer_id),
    FOREIGN KEY (centre_id)
        REFERENCES "ISP".centre (centre_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

5. Technician

```
CREATE TABLE IF NOT EXISTS "ISP".technician
(
    employee_id bigint NOT NULL,
    "first name" character(50) COLLATE pg_catalog."default" NOT NULL,
    "last name" character(50) COLLATE pg_catalog."default" NOT NULL,
    vaccinated character(50) COLLATE pg_catalog."default" NOT NULL,
    centre_id bigint NOT NULL,
    PRIMARY KEY (employee_id),
    FOREIGN KEY (centre_id)
        REFERENCES "ISP".centre (centre_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

6. Customer support

```
CREATE TABLE IF NOT EXISTS "ISP"."customer support"
(
    complaint_id bigint NOT NULL,
    "customer feedback" integer NOT NULL,
    "helpline number" bigint NOT NULL,
    "complaint status" character(50) COLLATE pg_catalog."default" NOT NULL,
    centre_id bigint NOT NULL,
    customer_id bigint NOT NULL,
    employee_id bigint NOT NULL,
    PRIMARY KEY (complaint_id),
    FOREIGN KEY (centre_id)
        REFERENCES "ISP".centre (centre_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (customer_id)
        REFERENCES "ISP".customer (customer_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (employee_id)
        REFERENCES "ISP".technician (employee_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

7. Payment

```
CREATE TABLE IF NOT EXISTS "ISP".payment
(
    payment_id bigint NOT NULL,
    amount integer NOT NULL,
    payment_mode character varying COLLATE pg_catalog."default" NOT NULL,
    date date NOT NULL,
    status character varying COLLATE pg_catalog."default" NOT NULL,
    customer_id bigint NOT NULL,
    reach_plan_id bigint NOT NULL,
    PRIMARY KEY (payment_id),
    FOREIGN KEY (customer_id)
        REFERENCES "ISP".customer (customer_id)
        ON UPDATE CASCADE
```

```

    ON DELETE CASCADE,
FOREIGN KEY (reach_plan_id)
    REFERENCES "ISP"."recharge plans" (reach_plan_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);

```

8. Recharge history

```

CREATE TABLE IF NOT EXISTS "ISP"."recharge history"
(
    reach_history_id bigint NOT NULL,
    reach_plan_id integer NOT NULL,
    payment_id bigint NOT NULL,
    customer_id bigint NOT NULL,
    PRIMARY KEY (reach_history_id)
);

```

9. Military officials

```

CREATE TABLE IF NOT EXISTS "ISP"."military officials"
(
    military_id character varying COLLATE pg_catalog."default" NOT NULL,
    "first name" character varying COLLATE pg_catalog."default" NOT NULL,
    "last name" character varying COLLATE pg_catalog."default" NOT NULL,
    posting_location character varying COLLATE pg_catalog."default" NOT NULL,
    reach_plan_id bigint NOT NULL,
    PRIMARY KEY (military_id),
    FOREIGN KEY (reach_plan_id)
        REFERENCES "ISP"."recharge plans" (reach_plan_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

```

10. Has

```

CREATE TABLE IF NOT EXISTS "ISP".has
(
    centre_id bigint NOT NULL,
    customer_id bigint NOT NULL,
    PRIMARY KEY (centre_id, customer_id)
);

```

SQL QUERIES

Q1. Show user details for all the users that use card payment mode.

```
select*from "ISP".payment  
where payment_mode='Card'
```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects: functions, materialized views, procedures, sequences, and tables. Under 'Tables (10)', the 'payment' table is selected, showing its columns: payment_id, amount, payment_mode, date, status, customer_id, and reach_plan_id. The right pane is the 'Query Editor' where the query is typed. Below the editor is the 'Explain' tab, which provides a detailed execution plan for the query. The 'Data Output' tab is active, displaying the results of the query in a table format. The table has 17 rows, each representing a payment record with columns: payment_id, amount, payment_mode, date, status, customer_id, and reach_plan_id.

payment_id	amount	payment_mode	date	status	customer_id	reach_plan_id	
1	6002	199	Card	2021-01-05	successful	202	800199
2	6005	798	Card	2021-09-28	successful	205	801798
3	6007	319	Card	2021-07-25	successful	207	800319
4	6008	398	Card	2021-03-06	successful	208	800398
5	6010	294	Card	2021-04-01	successful	210	800298
6	6011	359	Card	2021-01-30	successful	211	800359
7	6014	498	Card	2021-07-15	successful	214	802498
8	6017	558	Card	2021-07-22	successful	217	800558
9	6020	78	Card	2021-05-05	successful	220	800078
10	6023	251	Card	2021-03-31	successful	223	800251
11	6026	140	Card	2021-02-22	successful	226	800150
12	6029	400	Card	2021-07-18	successful	229	800400
13	6031	700	Card	2021-06-02	successful	231	800700
14	6032	2000	Card	2021-04-10	successful	232	802000
15	6035	199	Card	2021-05-20	successful	235	800199
16	6037	499	Card	2021-04-29	successful	237	800499
17	6039	700	Card	2021-06-26	successful	239	801700

Q2. Selecting names and contact numbers of customers who have /has Lisha as their name/surname.

```
select "first name", "mobile number"  
from "ISP".customer  
Where "first name" Like 'L%';
```

```

1 select "first name", "mobile number"
2 from "ISP".customer
3 Where "first name" Like 'L%';
4

```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects like Materialized Views, Procedures, Sequences, Tables (10), and others. The 'Tables (10)' section is expanded, showing tables such as address, centre, customer, customer support, has, military officials, payment, recharge history, recharge plans, technician, and others. The 'customer' table is selected. The right pane is the 'Query Editor' where the provided SQL query is run. Below the query, the 'Explain' tab is active, followed by 'Data Output'. The 'Data Output' tab displays the results of the query:

	first name	mobile number
1	Lisha	3883909049
2	Louisa	9317467010
3	Lindsey	1502351825
4	Lucina	3828706954
5	Leatha	3492944916
6	Laticia	2595793345
7	Lezile	7677724076
8	Lizette	4200716761
9	Larae	5823420381
10	Latrice	1656250937
11	Layla	8666663595

Q3. To understand the details of recharge plans bought by every customer over the last 365 days.

```

SELECT*
FROM "ISP"."recharge plans"
INNER JOIN "ISP".payment
ON payment.reach_plan_id="recharge plans".reach_plan_id
ORDER BY date LIMIT 365

```

```

1 SELECT *
2 FROM "ISP"."recharge plans"
3 INNER JOIN "ISP".payment
4 ON payment.reach_plan_id= "recharge plans".reach_plan_id
5 ORDER BY date LIMIT 365
6
7
8

```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects like mobile number, email, first name, last name, active, pincode, centre_id, and others. The 'customer support' table is selected. The right pane is the 'Query Editor' where the provided SQL query is run. Below the query, the 'Data Output' tab is active, displaying the results of the query:

	reach_plan_id	validity	data amount	payment_id	amount	payment_mode	date	status	customer_id
1	800449	60	2 GB/day	6045	449	Wallet	2021-01-03	successful	245
2	800199	45	1 GB/day	6002	199	Card	2021-01-05	successful	202
3	800149	30	1 GB/day	6001	149	UPI	2021-01-06	successful	201
4	800398	60	1.5 GB/day	6041	398	Card	2021-01-09	successful	241
5	800599	90	2 GB/day	6046	599	UPI	2021-01-15	successful	246
6	800098	existing	12 GB	6021	98	Wallet	2021-01-15	successful	221
7	800398	60	1.5 GB/day	6074	398	Card	2021-01-17	successful	274

Q4. Select id of the customer who has paid minimum amount –sq

```
SELECT customer_id
FROM "ISP".payment
WHERE amount=(SELECT MIN(amount)
FROM "ISP".payment)
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like 'customer', 'payment', and 'centre'. The main window shows the Query Editor with the following SQL code:

```
1 SELECT customer_id
2 FROM "ISP".payment
3 WHERE amount=(SELECT MIN(amount)
4 FROM "ISP".payment)
5
```

The Data Output tab shows the results:

customer_id
219
252

Q5. Same customers have been added to the customer table by mistake, we are required to do proper indexing.

```
SELECT DISTINCT "first name" FROM ISP.customer;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like 'customer', 'centre', and 'payment'. The main window shows the Query Editor with the following SQL code:

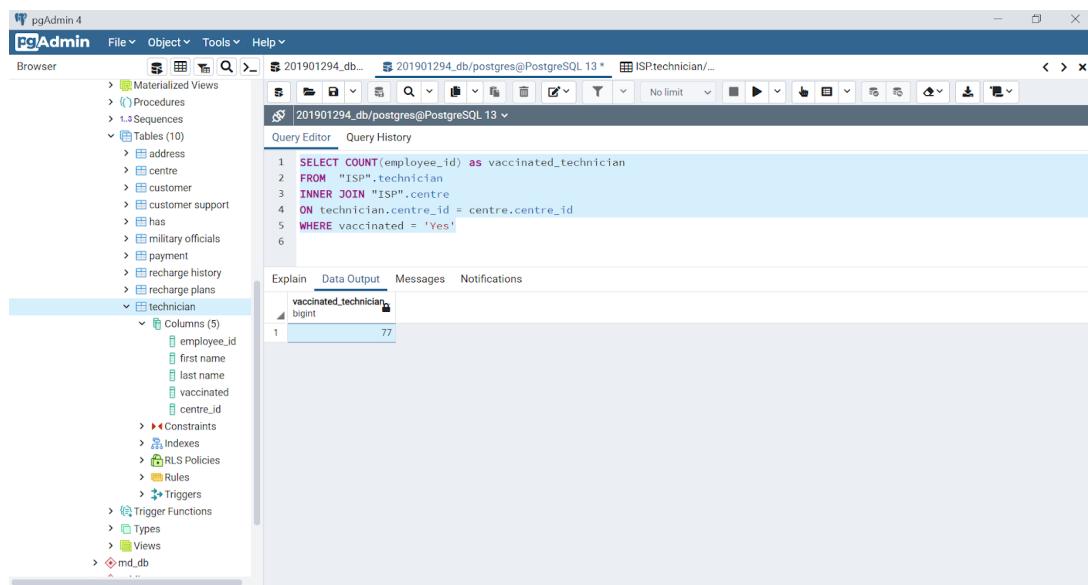
```
1 SELECT DISTINCT "first name" FROM "ISP".customer;
```

The Data Output tab shows the results:

first name
Louisa
Ty
Van
Rosio
Kimberly
Chau
Rolande
Clay
Rory
Leatha
Dorthy
Cheryl
Donte
Melissa
Bok

Q6. Count the number of vaccinated technicians over all the centers

```
SELECT COUNT(employee_id) as vaccinated_technician  
FROM "ISP".technician  
INNER JOIN "ISP".centre  
ON technician.centre_id = centre.centre_id  
WHERE vaccinated= 'Yes'
```

A screenshot of the pgAdmin 4 interface. The left sidebar shows a tree view of database objects: Materialized Views, Procedures, Sequences, Tables (10), address, centre, customer, customer support, has, military officials, payment, recharge history, recharge plans, technician (selected), Columns (5), employee_id, first name, last name, vaccinated, centre_id, Constraints, Indexes, RLS Policies, Rules, Triggers, Trigger Functions, Types, Views, and md_db. The main area is the Query Editor with the following content:

```
1 SELECT COUNT(employee_id) as vaccinated_technician  
2 FROM "ISP".technician  
3 INNER JOIN "ISP".centre  
4 ON technician.centre_id = centre.centre_id  
5 WHERE vaccinated = 'Yes'  
6
```

Below the query editor is an Explain plan table:

vaccinated_technician	bigint
1	77

Q7. Number of pending request at particular center

```
SELECT COUNT ("complaint status") as pending_request  
FROM "ISP"."customer support"  
INNER JOIN "ISP".centre  
ON "customer support".centre_id = centre.centre_id  
Where "complaint status" = 'pending'
```

```

1 SELECT COUNT ("complaint status") as pending_request
2 FROM "ISP"."customer support"
3 INNER JOIN "ISP".centre
4 ON "customer support".centre_id = centre.centre_id
5 WHERE "complaint status" = 'pending'
6

```

Q8. Print recharge details of last years.

```

SELECT *
FROM "ISP"."recharge plans"
INNER JOIN "ISP".payment
ON payment.reach_plan_id= "recharge plans".reach_plan_id
ORDER BY date LIMIT 365

```

reach_plan_id	validity	data amount	payment_id	amount	payment_mode	date	status	customer_id
800449	60	2 GB/day	6045	449	Wallet	2021-01-03	successful	245
800199	45	1 GB/day	6002	199	Card	2021-01-05	successful	202
800149	30	1 GB/day	6001	149	UPI	2021-01-06	successful	201
800398	60	1.5 GB/day	6041	398	Card	2021-01-09	successful	241
800599	90	2 GB/day	6046	599	UPI	2021-01-15	successful	246
800098	existing	12 GB	6021	98	Wallet	2021-01-15	successful	221
8000308	60	1.5 GB/day	6074	200	Card	2021-01-17	successful	272

Q9. Get the first name and pincode of customers from customer table in ascending order.

SELECT first name and pincode

FROM "ISP".customer

ORDER BY first name and pincode

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'customer' schema, the 'Columns' section is expanded, showing columns like 'customer_id', 'mobile number', 'email', 'first name', 'last name', 'active', 'pincode', and 'centre_id'. The main window contains a query editor with the following SQL code:

```
1 SELECT "first name",pincode
2 FROM "ISP"."customer"
3 ORDER BY "first name"
```

Below the query editor is a data output table with the following results:

	first name	pincode
1	Alesea	733382
2	Angella	889609
3	Annabelle	887888
4	Arlene	405252
5	Arminda	310477
6	Audra	637809
7	Beatriz	191449

A green message at the bottom right indicates: **Successfully run. Total query runtime: 387 msec. 80 rows affected.**

Q10. Find reach_plan_id whose data amount = 2GB/day using Grouping.

select validity, "data amount", reach_plan_id

from "ISP"."recharge plans"

Group by "data amount", "recharge plans".validity, "recharge plans".reach_plan_id

having "data amount" = '2 GB/day'

order by validity, "data amount"

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'ISP' schema, the 'Tables' section is expanded, showing tables like 'address', 'collations', 'domains', 'fts configurations', 'fts dictionaries', 'fts parsers', 'fts templates', 'foreign tables', 'functions', 'materialized views', 'procedures', 'sequences', and 'tables'. The main window contains a query editor with the following SQL code:

```
1 select validity, "data amount", reach_plan_id
2 from "ISP"."recharge plans"
3 Group by "data amount", "recharge plans".validity, "recharge plans".reach_plan_id
4 having "data amount" = '2 GB/day'
5 order by validity, "data amount"
```

Below the query editor is a data output table with the following results:

	validity	data amount	reach_plan_id
1	30	2 GB/day	800200
2	30	2 GB/day	800298
3	365	2 GB/day	802498
4	365	2 GB/day	802000
5	45	2 GB/day	800359
6	60	2 GB/day	800449
7	90	2 GB/day	800599
8	on	2 GB/day	800nnnn

Q11. Print recharge details of last 20 days

```
SELECT *
FROM "ISP"."recharge plans"
INNER JOIN "ISP".payment
ON payment.reach_plan_id= "recharge plans".reach_plan_id
ORDER BY date LIMIT 20
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
1 SELECT *
2 FROM "ISP"."recharge plans"
3 INNER JOIN "ISP".payment
4 ON payment.reach_plan_id= "recharge plans".reach_plan_id
5 ORDER BY date LIMIT 20
```

The results table has the following columns and data:

reach_plan_id	validity	data amount	payment_id	amount	payment_mode	date	status	customer_id	reach_plan_id
800449	60	2 GB/day	6045	449	Wallet	2021-01-03	successful	245	800449
800199	45	1 GB/day	6002	199	Card	2021-01-05	successful	202	800199
800149	30	1 GB/day	6001	149	UPI	2021-01-06	successful	201	800149
800398	60	1.5 GB/day	6041	398	Card	2021-01-09	successful	241	800398
800098	existing	12 GB	6021	98	Wallet	2021-01-15	successful	221	800098
800599	90	2 GB/day	6046	599	UPI	2021-01-15	successful	246	800599
800398	60	1.5 GB/day	6074	398	Card	2021-01-17	successful	274	800398
800359	45	2 GB/day	6011	359	Card	2021-01-30	successful	211	800359
800100	30	1 GB/day	6024	100	Wallet	2021-01-30	successful	224	800100
800150	existing	25 GB	6026	140	Card	2021-02-22	successful	226	800150
800319	45	1.5 GB/day	6040	319	UPI	2021-02-26	successful	240	800319
800398	60	1.5 GB/day	6008	398	Card	2021-03-06	successful	208	800398

Q12. Minimum validity for military officials'

```
SELECT validity
FROM "ISP"."military officials"
INNER JOIN "ISP"."recharge plans"
ON "military officials".reach_plan_id= "recharge plans".reach_plan_id
ORDER BY validity LIMIT 10'
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
1 SELECT validity
2 FROM "ISP"."military officials"
3 INNER JOIN "ISP"."recharge plans"
4 ON "military officials".reach_plan_id= "recharge plans".reach_plan_id
5 ORDER BY validity LIMIT 10'
```

The results table has the following column and data:

validity
30
30
30
30
30
30
30
30
30
30

Q13. In which centre was maximum feedback received and complaint status is pending,in order of centre ID.

```
SELECT complaint_id,MAX("customer feedback")
FROM "ISP"."customer support"
GROUP BY complaint_id
HAVING "customer support"."complaint status" = 'pending'
ORDER BY complaint_id
```

```
1 SELECT complaint_id,MAX("customer feedback")
2 FROM "ISP"."customer support"
3 GROUP BY complaint_id
4 HAVING "customer support"."complaint status" = 'pending'
5 ORDER BY complaint_id
6
```

complaint_id	max
505	9
507	10
511	5
517	10
523	10
525	5

Q14. To get a list of which recharge plans the military officials have opted for and validity of those plans.

```
SELECT n."first name", r.reach_plan_id,r.validity
FROM "ISP"."military officials" AS n
LEFT JOIN "ISP"."recharge plans" AS r
ON n.reach_plan_id = r.reach_plan_id
ORDER BY r."reach_plan_id";
```

```
1 SELECT n."first name", r.reach_plan_id,r.validity
2 FROM "ISP"."military officials" AS n
3 LEFT JOIN "ISP"."recharge plans" AS r
4 ON n.reach_plan_id = r.reach_plan_id
5 ORDER BY r."reach_plan_id";
```

first name	reach_plan_id	validity
Portia	800100	30
Taryn	800100	30
Jolanda	800100	30
Glendora	800100	30
Alex	800100	30
Salena	800100	30
Cristy	800140	30

Q15 Inner join the technician and centre table to find out all the details of centre a technician belongs to.

```
SELECT c.centre_id,c.centre_name,c.pincode,t.employee_id
FROM "ISP".centre AS c
INNER JOIN "ISP".technician as t
ON c.centre_id=t.centre_id
```

```
1 SELECT c.centre_id,c.centre_name,c.pincode,t.employee_id
2 FROM "ISP".centre AS c
3 INNER JOIN "ISP".technician as t
4 ON c.centre_id=t.centre_id
5
```

centre_id	centre_name	pincode	employee_id
401	celitech_Hosapete	691792	101
402	celitech_Habra	405252	102
403	celitech_Silvassa	733382	103
404	celitech_Tiruvurur	615177	104
405	celitech_Palakkad	889609	105
406	celitech_Port Blair	877368	106
407	celitech_Agartala	228327	107

Q16 To find the average amount that is paid for each of the recharge plans given.

```
SELECT r.reach_plan_id,AVG(p.amount)
FROM "ISP".recharge history AS r
INNER JOIN "ISP".payment as p
ON r.reach_plan_id=p.reach_plan_id
GROUP BY r.reach_plan_id
```

```
1
2 SELECT r.reach_plan_id,AVG(p.amount)
3 FROM "ISP"."recharge history" AS r
4 INNER JOIN "ISP".payment as p
5 ON r.reach_plan_id=p.reach_plan_id
6 GROUP BY r.reach_plan_id;
7
```

reach_plan_id	avg
800599	599.0000000000000
800100	100.0000000000000
800558	558.0000000000000
800300	300.0000000000000
800248	248.0000000000000
800198	198.0000000000000
800048	48.0000000000000

Q17 To understand which payment mode is being used with the amount details,to understand which mode of payment do the users prefer for plans with larger amount.

```
SELECT r.reach_plan_id,p.amount,p.payment_mode
```

```
FROM "ISP"."recharge history" AS r
```

```
INNER JOIN "ISP".payment as p
```

```
ON r.reach_plan_id=p.reach_plan_id
```

reach_plan_id	amount	payment_mode
800149	149	UPI
800149	149	UPI
800149	149	UPI
800199	199	Card
800199	199	Card
800199	199	Card
800248	248	Wallet

Q18 To get the total amount spent per plan id in the past three months.

```
SELECT r.reach_plan_id,SUM(p.amount)
```

```
FROM recharge history
```

```
INNER JOIN payment AS p
```

```
ON r.reach_plan_id=p.reach_plan_id
```

```
GROUP BY reach_plan_id
```

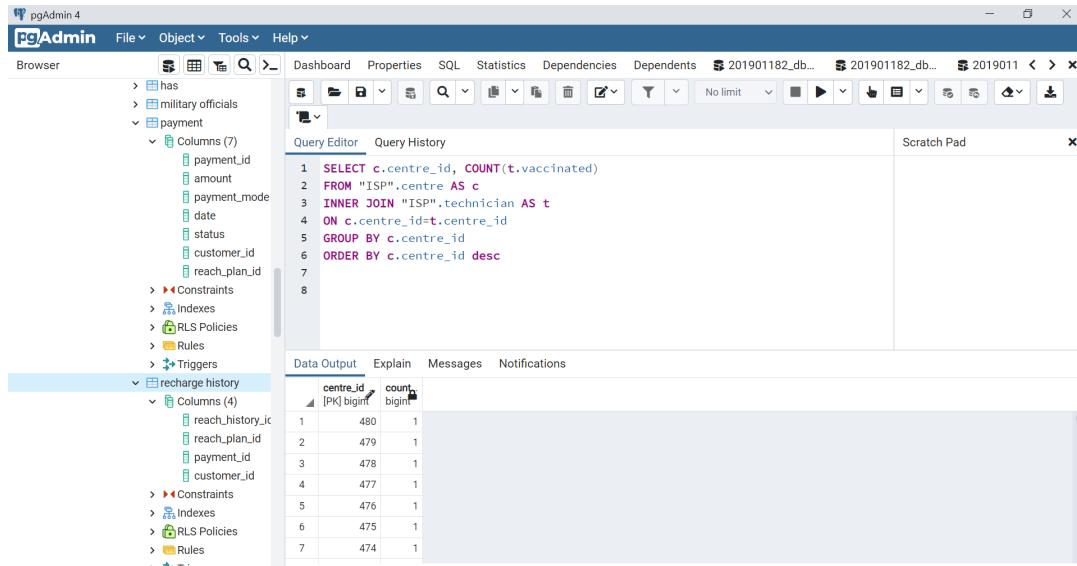
```
ORDER BY SUM(p.amount)
```

reach_plan_id	sum
800048	192
800078	312
800098	392
800100	400
800198	792
800200	800
800251	1004

Q19 To fetch the centres of our company where at least one technician has been vaccinated to notice how many centres have not even begun with their vaccination drive.

SELECT c.centre_id, COUNT(t.vaccinated)

FROM "ISP".centre AS c
INNER JOIN "ISP".technician AS t
ON c.centre_id=t.centre_id
GROUP BY c.centre_id
ORDER BY c.centre_id desc



The screenshot shows the pgAdmin 4 interface. The left sidebar displays a database browser with tables like 'has', 'military officials', 'payment', and 'recharge history'. The 'recharge history' table is currently selected. The central area contains a 'Query Editor' tab with the following SQL code:

```
1 SELECT c.centre_id, COUNT(t.vaccinated)
2 FROM "ISP".centre AS c
3 INNER JOIN "ISP".technician AS t
4 ON c.centre_id=t.centre_id
5 GROUP BY c.centre_id
6 ORDER BY c.centre_id desc
7
8
```

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

	centre_id	count
1	480	1
2	479	1
3	478	1
4	477	1
5	476	1
6	475	1
7	474	1

20. To fetch the helpline no. details of each centre equipped with customer support for the annual company directory.

SELECT c.centre_id,s."helpline number"

FROM "ISP".centre AS c

INNER JOIN "ISP"."customer support" as s

ON c.centre_id=s.centre_id;

```

1 SELECT c.centre_id,s."helpline number"
2 FROM "ISP".centre AS c
3 INNER JOIN "ISP"."customer support" as s
4 ON c.centre_id=s.centre_id;
5

```

centre_id	helpline number
406	7472683523
418	6857327110
405	3019550115
436	1789802957
421	6838789722
462	4982530760
441	3200531767

Q21. To find out how much feedback do all the centres get in total.

```

SELECT c.centre_id, COUNT(s."customer feedback")
FROM "ISP".centre as c
INNER JOIN "ISP"."customer support" as s
ON c.centre_id=s.centre_id
GROUP BY c.centre_id
ORDER BY c.centre_id desc

```

```

1 SELECT c.centre_id, COUNT(s."customer feedback")
2 FROM "ISP".centre as c
3 INNER JOIN "ISP"."customer support" as s
4 ON c.centre_id=s.centre_id
5 GROUP BY c.centre_id
6 ORDER BY c.centre_id desc
7
8

```

centre_id	count
475	2
474	1
472	1
469	2
467	1
462	1
460	1

Successfully run. Total query runtime: 74 msec. 28 rows affected.

Q22. Find top up plans with existing plan's validity.

Select *
from "ISP"."recharge plans"
where validity = 'existing'

```
1 Select *
2 from "ISP"."recharge plans"
3 where validity = 'existing'
4
```

recharge_plan_id	validity	data_amount
800048	existing	3GB
800078	existing	6 GB
800098	existing	12 GB
800198	existing	25 GB
800251	existing	50 GB
800150	existing	25 GB

Q23. Find Customer's full name and pincode who live in the city Bhopal.

Select "first name", "last name", city, state, address.pincode As pin
from "ISP".customer join "ISP".address
on address.pincode = customer.pincode
Where city = 'Bhopal'

```
1 Select "first name", "last name", city, address.pincode As pin
2 from "ISP".customer join "ISP".address
3 "ISP".customer join "ISP".address
4 on address.pincode = customer.pincode
5 Where city = 'Bhopal'
6
7
```

first_name	last_name	city	state	pin
Twana	Felger	Bhopal	Madhya Pradesh	257884
Pamella	Schmieder	Bhopal	Madhya Pradesh	257884

Q24. Find customer_id and complaint_id whose complaint status are in process and the employee_id.

```
Select customer_id, complaint_id, "complaint status",technician.employee_id As emp, "first name", vaccinated
From "ISP"."customer support" join "ISP"."technician"
on "customer support".employee_id = technician.employee_id
where "complaint status" = 'in process'
```

```
1 Select customer_id, complaint_id, "complaint status",technician.employee_id As emp, "first name", vaccinated
2 From
3 "ISP"."customer support" join "ISP"."technician"
4 on "customer support".employee_id = technician.employee_id
5 where "complaint status" = 'in process'
```

	customer_id	complaint_id	complaint status	emp	first name	vaccinated
1	207	512	in process	107	Danie	Yes
2	224	526	in process	124	Janine	Yes
3	258	508	in process	158	Roxana	Yes
4	262	506	in process	162	Brett	Yes
5	272	524	in process	172	Adrian	Yes

Q.25 Find maximum customer feedback and centre_id showing city and state where the complaints are registered and which employee is assigned to the complaint.

```
SELECT "customer support".centre_id as c , "customer feedback", "first name", city, state,
technician.employee_id as EMP
FROM "ISP"."customer support" join "ISP"."technician"
on "customer support".employee_id = "technician".employee_id
join "ISP".centre
on "customer support".centre_id = "centre".centre_id
join "ISP".address
on "centre".pincode = "address".pincode
where "customer feedback" > '5'
ORDER BY "customer feedback" desc
```

```

1 SELECT "customer support".centre_id as c , "customer feedback", "first name", city, state, technician.employee_id as emp
2 FROM
3 "ISP"."customer support" join "ISP"."technician"
4 on "customer support".employee_id = "technician".employee_id
5 join "ISP".centre
6 on "customer support".centre_id = "centre".centre_id
7 join "ISP".address
8 on "centre".pincode = "address".pincode
9 where "customer feedback" > '5'
10 ORDER BY "customer feedback" desc

```

c	customer feedback	first name	city	state	emp
1	438	10 Melodee	Burhanpur	Madhya Pradesh	138
2	405	10 Simmonds	Palakkad	Kerala	105
3	475	10 Boycie	Bengaluru	Karnataka	175
4	407	10 Dannie	Agartala	Tripura	107
5	451	10 Sam	Patna	Bihar	151
6	441	10 Augy	Damoh	Madhya Pradesh	141
7	416	10 Emanuele	Beawar	Rajasthan	116
8	402	9 Ira	Habra	West Bengal	102
9	421	9 Ginni	Purulia	West Bengal	121
10	469	9 Agatha	Mokokchung	Nagaland	169

Q26. Find reach_plan_id whose data amount = 1GB/day using Grouping.

select validity, "data amount", reach_plan_id
from "ISP"."recharge plans"

Group by "data amount", "recharge plans".validity, "recharge plans".reach_plan_id
having "data amount" = '1 GB/day'
order by validity, "data amount"

```

1 select validity, "data amount", reach_plan_id
2 from "ISP"."recharge plans"
3 Group by "data amount", "recharge plans".validity, "recharge plans".reach_plan_id
4 having "data amount" = '1 GB/day'
5 order by validity, "data amount"

```

validity	data amount	reach_plan_id
1	1 GB/day	800149
2	1 GB/day	800100
3	1 GB/day	801798
4	1 GB/day	800199
5	1 GB/day	800248
6	1 GB/day	800499
7	1 GB/day	800300

Successfully run. Total query runtime: 136 msec. 7 rows affected.

Q27. Find amount for which payment mode is wallet

```
select amount, "payment_mode"
from "ISP".payment
Group by amount, "payment_mode"
having "payment_mode" = 'Wallet'
```

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables (10)' section, the 'payment' table is selected. The 'Columns (7)' section is expanded, showing columns: payment_id, amount, payment_mode, date, status, customer_id, and reach_plan_id. The main area is the 'Query Editor' tab, containing the following SQL code:

```
1 select amount, "payment_mode"
2   from "ISP".payment
3   Group by amount, "payment_mode"
4   having "payment_mode" = 'Wallet'
5
6
```

The 'Data Output' tab shows the results of the query:

	amount	payment_mode
1	450	Wallet
2	98	Wallet
3	249	Wallet
4	598	Wallet
5	150	Wallet
6	798	Wallet
7	350	Wallet
8	449	Wallet
9	100	Wallet
10	248	Wallet

Q28. Show all recharge plans in descending order including those not associated with military IDs

```
SELECT "military officials".military_id , "recharge plans".reach_plan_id as rpi
FROM "ISP"."military officials"
FULL OUTER JOIN "ISP"."recharge plans"
ON "military officials".reach_plan_id = "recharge plans".reach_plan_id
WHERE "recharge plans".reach_plan_id > '800150'
ORDER BY "recharge plans".reach_plan_id desc
```

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

- Browser:** Shows the database structure with the 'military' table selected.
- Query Editor:** Contains the following SQL query:


```
1 SELECT "military officials".military_id , "recharge plans".reach_plan_id as rpi
2 FROM "ISP"."military officials"
3 FULL OUTER JOIN "ISP"."recharge plans"
4 ON "military officials".reach_plan_id = "recharge plans".reach_plan_id
5 WHERE "recharge plans".reach_plan_id > '800150'
6 ORDER BY "recharge plans".reach_plan_id desc
7 |
```
- Data Output:** Displays the results of the query, showing columns: military_id (character varying) and rpi (bigint). The data includes 11 rows of military IDs and their corresponding reach plan IDs.

Q29. Write a query to update mobile number of a given customer

UPDATE "ISP".customer

```
SET "mobile number"='1234567890'
WHERE "last name"='Shin';
```

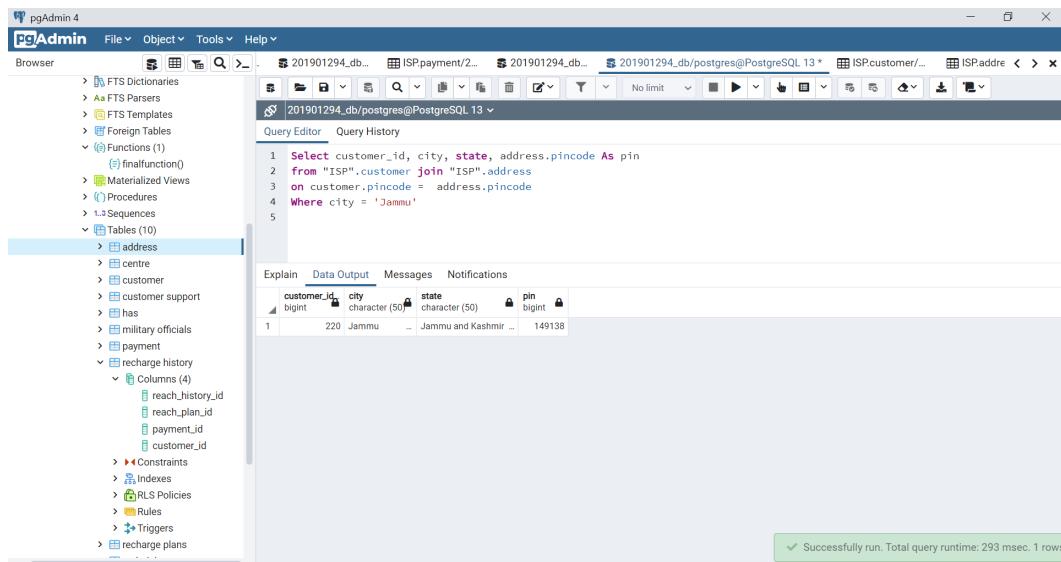
The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database structure with the 'customer' table selected.
- Query Editor:** Contains the following SQL query:


```
1 UPDATE "ISP".customer
2 SET "mobile number"='1234567890'
3 WHERE "last name"='Shin';
```
- Data Output:** Displays the results of the query, showing the message: "Query returned successfully in 482 msec."

Q30 We are yet to target a good audience in Jammu, check how many people from Jammu are our customers.

Select customer_id, city, state, address.pincode As pin
from "ISP".customer join "ISP".address
on customer.pincode = address.pincode
Where city = 'Jammu'



```

1 Select customer_id, city, state, address.pincode As pin
2 from "ISP".customer join "ISP".address
3 on customer.pincode = address.pincode
4 Where city = 'Jammu'
5

```

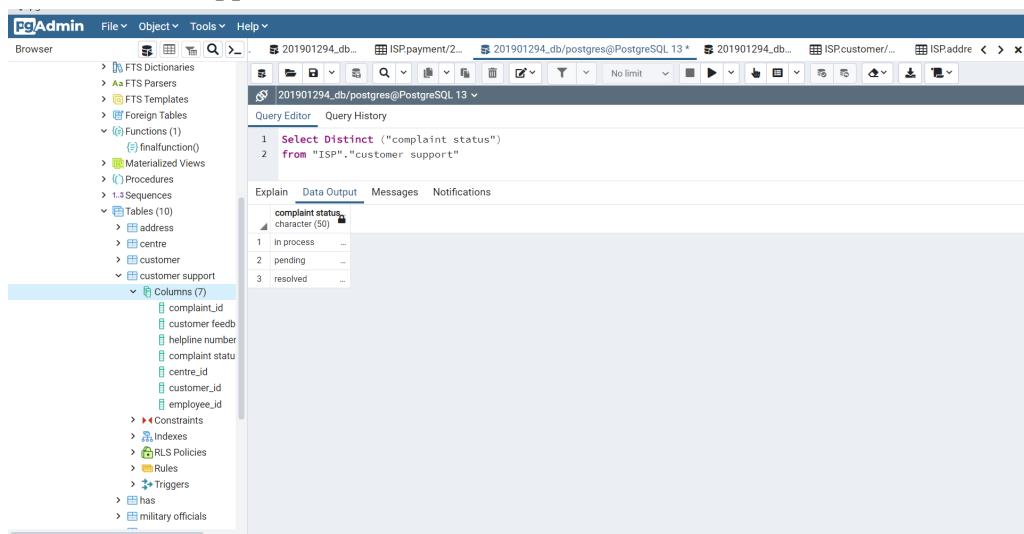
The screenshot shows the pgAdmin 4 interface with the following details:
- **Browser**: Shows the database structure with tables like address, centre, customer, etc.
- **Query Editor**: Contains the SQL query provided above.
- **Data Output**: Displays the results of the query in a table:

	customer_id	city	state	pin
1	220	Jammu	Jammu and Kashmir	149138

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 293 msec. 1 rows"

Q31 Check what are the types categories for complaint status

Select Distinct ("complaint status")
from "ISP"."customer support"



```

1 Select Distinct ("complaint status")
2 from "ISP"."customer support"

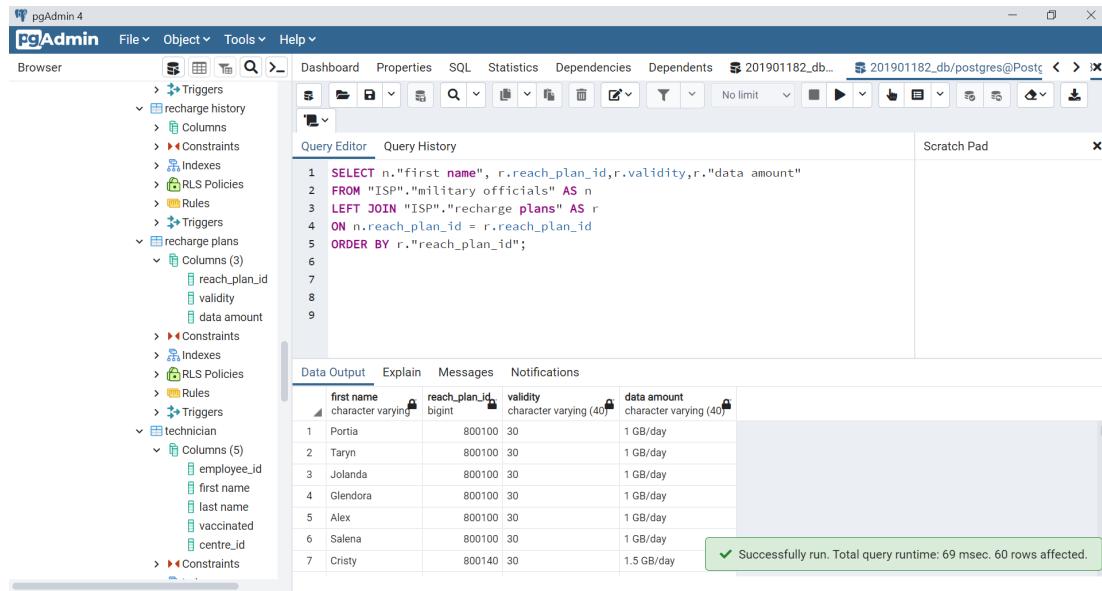
```

The screenshot shows the pgAdmin 4 interface with the following details:
- **Browser**: Shows the database structure with tables like address, centre, customer, etc., and a specific table for customer support with columns like complaint_id, customer_feedback, helpline_number, etc.
- **Query Editor**: Contains the SQL query provided above.
- **Data Output**: Displays the results of the query in a table:

	complaint status
1	in process
2	pending
3	resolved

Q32 To get a list of which recharge plans the military officials have opted for and validity of those plans.

```
SELECT n."first name", r.reach_plan_id,r.validity,r."data amount"
FROM "ISP"."military officials" AS n
LEFT JOIN "ISP"."recharge plans" AS r
ON n.reach_plan_id = r.reach_plan_id
ORDER BY r."reach_plan_id";
```



first name	reach_plan_id	validity	data amount
Portia	800100	30	1 GB/day
Taryn	800100	30	1 GB/day
Jolanda	800100	30	1 GB/day
Glendora	800100	30	1 GB/day
Alex	800100	30	1 GB/day
Salena	800100	30	1 GB/day
Cristy	800140	30	1.5 GB/day

Q33 Find the number of customers whose data amount usage > 3GB/day using Grouping.

```
select validity, "data amount", reach_plan_id
from "ISP"."recharge plans"
Group by "data amount", "recharge plans".validity, "recharge plans".reach_plan_id
having "data amount" > '3 GB/day'
order by validity, "data amount"
```

```

1 select validity, "data amount", reach_plan_id
2 from "ISP"."recharge plans"
3 Group by "data amount", "recharge plans".validity, "recharge plans".reach_plan_id
4 having "data amount" > '3 GB/day'
5 order by validity, "data amount"
6

```

validity	data amount	reach_plan_id
existing	3GB	800248
existing	50 GB	800251
existing	6 GB	800078

Q34 Inner join the technician and centre table to find out all the the city each technician is allotted to via pincode.

```

SELECT c.city,c.state,t.employee_id
FROM "ISP".centre AS c
INNER JOIN "ISP".technician as t
ON c.centre_id=t.centre_id

```

```

1 SELECT c.pincode,t.employee_id
2 FROM "ISP".centre AS c
3 INNER JOIN "ISP".technician as t
4 ON c.centre_id=t.centre_id
5
6

```

pincode	employee_id
691792	101
405252	102
733382	103
615177	104
889609	105
877368	106
228327	107

✓ Successfully run. Total query runtime: 144 msec. 80 rows affected.

Q35 Select id of the customer who has paid maximum amount -sq

```
SELECT customer_id
FROM "ISP".payment
WHERE amount=(SELECT MAX(amount)
FROM "ISP".payment)
```

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects: Columns (7), Constraints, Indexes, RLS Policies, Rules, Triggers, has, military officials, payment, recharge history, recharge plans, technician, Trigger Functions, Types, Views, md_db, public, sv_db, Subscriptions, postgres, Login/Group Roles, and Tablespaces. The 'payment' table is currently selected. The main area has tabs for 'Query Editor' and 'Query History'. The 'Query Editor' tab contains the SQL code. The 'Data Output' tab shows the results of the query:

customer_id	bigint
1	232
2	265

Q36 In which centre was minimum feedback received and complaint status is pending, in order of centre ID.

```
SELECT complaint_id, MIN("customer feedback")
FROM "ISP"."customer support"
GROUP BY complaint_id
HAVING "customer support"."complaint status" = 'pending'
ORDER BY complaint_id
```

```

1
2 SELECT complaint_id,MIN("customer feedback")
3 FROM "ISP"."customer support"
4 GROUP BY complaint_id
5 HAVING "customer support"."complaint status" = 'pending'
6 ORDER BY complaint_id
7

```

Successfully run. Total query runtime: 53 msec. 6 rows affected.

Q37 To fetch the complaint status of each customer along with their mobile number.

```

SELECT c.customer_id,c."mobile number",s."complaint status"
FROM "ISP".customer AS c
INNER JOIN "ISP"."customer support" as s
ON c.customer_id=s.customer_id;

```

```

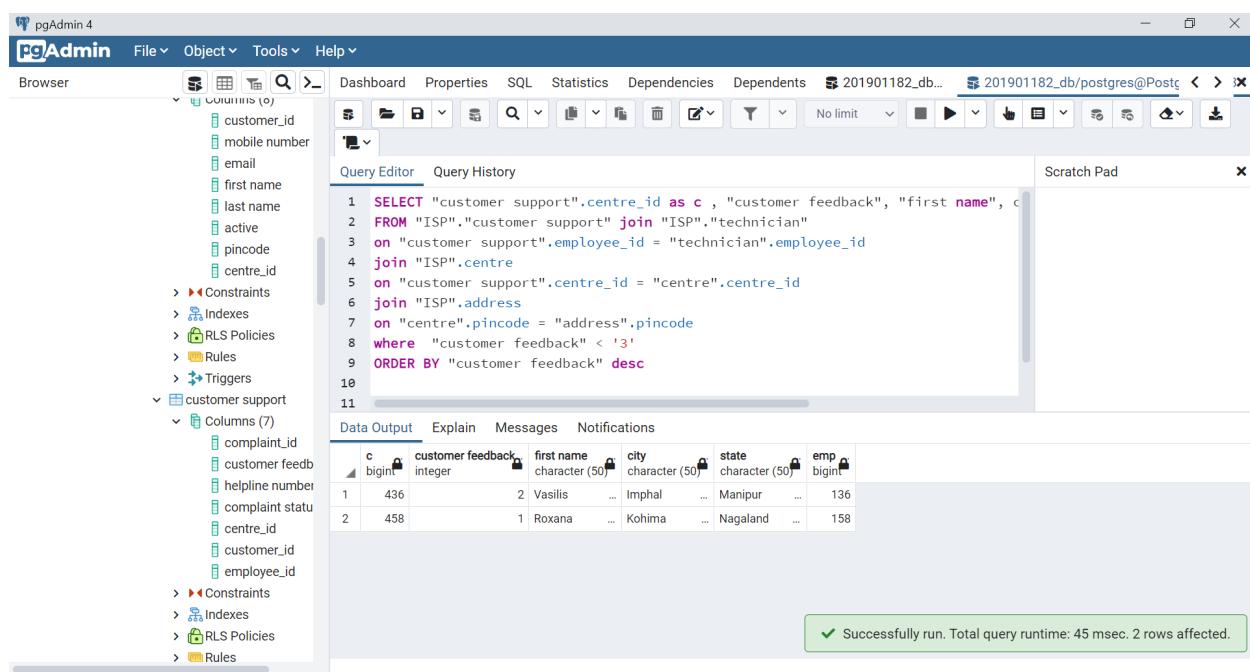
1 SELECT c.customer_id,c."mobile number",s."complaint status"
2 FROM "ISP".customer AS c
3 INNER JOIN "ISP"."customer support" as s
4 ON c.customer_id=s.customer_id;
5

```

Successfully run. Total query runtime: 53 msec. 7 rows affected.

Q38. Find minimum customer feedback and centre_id showing city and state where the complaints are registered and which employee is assigned to the complaint.

```
SELECT "customer support".centre_id as c , "customer feedback", "first name", city, state,
technician.employee_id as EMP
FROM "ISP"."customer support" join "ISP"."technician"
on "customer support".employee_id = "technician".employee_id
join "ISP".centre
on "customer support".centre_id = "centre".centre_id
join "ISP".address
on "centre".pincode = "address".pincode
where "customer feedback" < '3'
ORDER BY "customer feedback" desc
```



The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Customer Support' schema, there is a 'customer support' table with 7 columns: complaint_id, customer feedback, first name, city, state, and emp. The 'customer feedback' column is of type integer, and the other five are character(50). The emp column is of type bigint. The data returned by the query is:

c	customer feedback	first name	city	state	emp
1	436	Vasilis	Imphal	Manipur	136
2	458	Roxana	Kohima	Nagaland	158

A green message at the bottom right indicates: 'Successfully run. Total query runtime: 45 msec. 2 rows affected.'

Q39. Triggers function

Create a trigger on the technician table to display a message when a new employee_id is added but already exists.

SQL Query: insert into “ISP”.technician(employee_id) values (105);

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'Tables (10)'. A specific table named 'centre' is selected. Under the 'centre' table, the 'Triggers' section is expanded, showing one trigger named 'trigger_tech'. The 'Query Editor' tab at the top has the following SQL code:

```

1 insert into "ISP".technician(employee_id)
2 values (105)

```

The 'Messages' tab below the query editor shows a notice:

NOTICE: ID is used. Please try other IDs.
INSERT 0 0

The message also states: 'Query returned successfully in 246 msec.'

Q40. Function

Function to display centre_name by giving pincode from the centre table

SQL Query: Select "ISP"."finalfunction"(257884)

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'Functions (1)'. A function named 'finalfunction' is selected. The 'Query Editor' tab at the top has the following SQL code:

```

1 Select "ISP"."finalfunction"(257884)
2

```

The 'Messages' tab below the query editor shows the result of the function execution:

finalfunction
character varying
1 celitech_Bhopal

Section7: Project Code with output screenshots

Front-end Development

For setting up a connection with Postgresql and python we used anaconda navigator. Then we have to run the ‘conda terminal’ and install these packages. After installing the packages, we now open Jupyter Notebook and access our database from Postgresql using python. Below are the steps to set up the database connection.

```
%load_ext sql
sqlalchemy import
create_engine
%sql postgresql://postgres:admin@localhost:5432/201901182_db
```

After performing these 3 steps, we have established a connection between our database and the Jupyter Notebook. Next, we have created an object ‘engine’ using the ‘create_engine’ function: engine =
create_engine('postgresql://postgres:admin@localhost:5432/201901182_db').

Packages to install:

- conda install -y -c conda-forge ipython-sql
- conda install -y -c conda-forge postgresql
- conda install -y -c anaconda psycopg2
- conda install -y -c conda-forge pgspecial

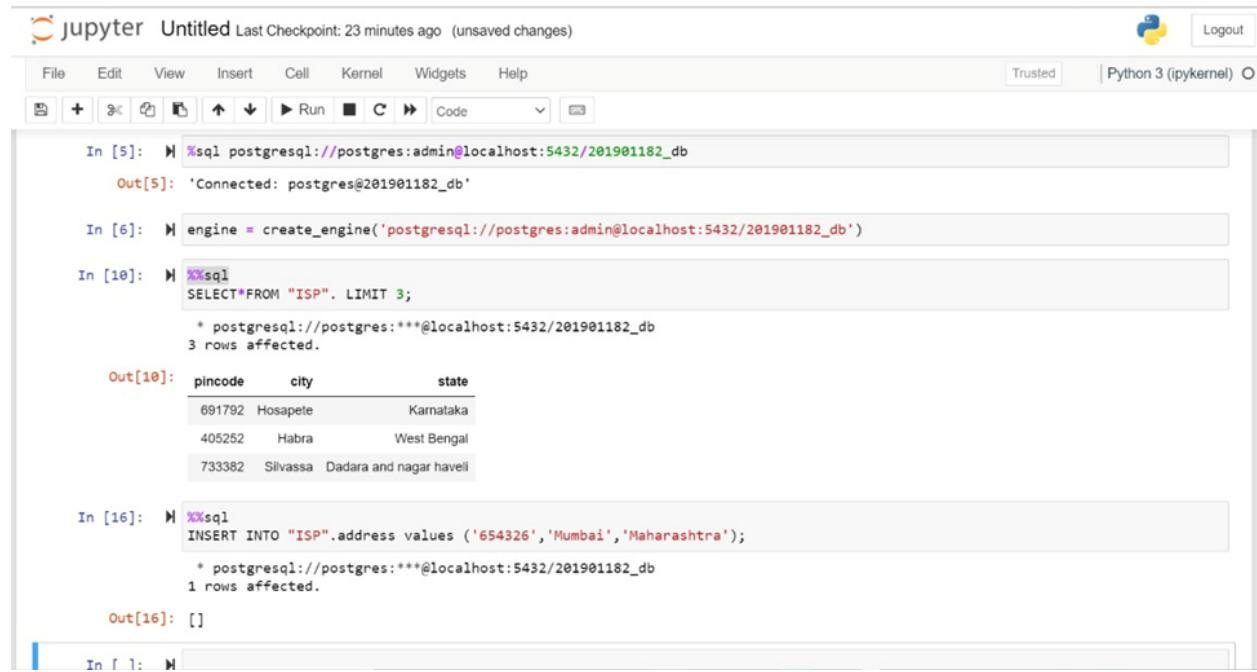
Code:

```
%load_ext sql
sqlalchemy import
create_engine
%sql postgresql://postgres:admin@localhost:5432/201901182_db
```

```
'connected: postgres@201901182_db'  
Function: engine =  
create_engine('postgresql://postgres:admin@localhost:5432/201901182_db')
```

Screenshot of output or web/app pages with data.

1. Display rows from the address table to see all address details



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [5]: %sql postgres://postgres:admin@localhost:5432/201901182_db  
Out[5]: 'Connected: postgres@201901182_db'  
  
In [6]: engine = create_engine('postgresql://postgres:admin@localhost:5432/201901182_db')  
  
In [10]: %%sql  
SELECT * FROM "ISP". LIMIT 3;  
* postgresql://postgres:***@localhost:5432/201901182_db  
3 rows affected.  
  
Out[10]:

| pincode | city     | state                   |
|---------|----------|-------------------------|
| 691792  | Hosapete | Karnataka               |
| 405252  | Habra    | West Bengal             |
| 733382  | Silvassa | Dadara and nagar haveli |

  
In [16]: %%sql  
INSERT INTO "ISP".address values ('654326','Mumbai','Maharashtra');  
* postgresql://postgres:***@localhost:5432/201901182_db  
1 rows affected.  
  
Out[16]: []
```

2. Display first 3 customer's complete details from the customer table

```

In [1]: %load_ext sql
In [4]: from sqlalchemy import create_engine
In [5]: %sql postgresql://postgres:admin@localhost:5432/201901182_db
Out[5]: 'Connected: postgres@201901182_db'
In [6]: engine = create_engine('postgresql://postgres:admin@localhost:5432/201901182_db')
In [7]: %%sql
SELECT*FROM "ISP".customer LIMIT 3;
* postgresql://postgres:***@localhost:5432/201901182_db
3 rows affected.
Out[7]:
customer_id  mobile number    email   first name  last name  active  pincode  centre_id
201          3883909049  lisha@centini.org  Lisha   Centini   Yes    925345   475
202          5101617924  arlene_klusman@gmail.com  Arlene  Klusman   Yes    405252   402
203          1295676492  alease@buemi.com    Alease   Buemi     Yes    733382   403
In [8]: %%sql
INSERT INTO "ISP".address values('654321','Lucknow','Uttar Pradesh');

```

3. Insert a new address detail row in the address table

```

In [16]: %%sql
INSERT INTO "ISP".address values ('654326','Mumbai','Maharashtra');
* postgresql://postgres:***@localhost:5432/201901182_db
1 rows affected.
Out[16]: []

```

This is the PgAdmin window showing the new row number 51 inserted with pincode '654326', city 'Mumbai' and state 'Maharashtra'.

PgAdmin File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents 201901182_db... ISP.address/20... ISP.addn < > x1

Foreign Tables Functions Materialized Views Procedures Sequences Tables (10) address Columns (3) pincode city state Constraints Indexes RLS Policies Rules Triggers centre Columns (3) centre_id centre_name pincode Constraints Indexes RLS Policies Rules Triggers

Query Editor Query History

```
1 SELECT * FROM "ISP".address
2 ORDER BY pincode ASC
```

Data Output Explain Messages Notifications

	pincode	city	state
47	627520	Ballari	Karnataka
48	632015	Kollam	Kerala
49	637809	Vellore	Tamil nadu
50	654321	Lucknow	Uttar Pradesh
51	654326	Mumbai	Maharashtra
52	662905	Swardj dweep	Andaman and Nicobar Islands
53	668180	Daman	Daman and Diu
54	673964	Gurugram	Haryana
55	685445	Agonda	Goa
56	687116	Manali	Himachal Pradesh
57	691792	Hosapete	Karnataka
58	716938	Dhanbad	Jharkhand
59	723664	Kota	Rajasthan

4. Update any customer's mobile number in the customer table

```

Out[16]: []

In [17]: %sql
UPDATE "ISP".customer
SET "mobile number"='3456789765'
WHERE customer_id='201';

* postgresql://postgres:***@localhost:5432/201901182_db
1 rows affected.

Out[17]: []

```

In []:

This is the PgAdmin window showing the customer 201, Lisha's new number being updated.

New

customer_id	mobile number	email	first name	last name	active	pincode	centre_id
201	3456789765	lisha@centini.org	Lisha	Centini	Yes	925345	475
202	5101617924	arlene_klusman@gmail.com	Arlene	Klusman	Yes	405252	402
203	1295676492	alease@buerni.com	Alease	Buerni	Yes	733382	403
204	9317467010	louisa@cronauer.com	Louisa	Cronauer	Yes	272472	419
205	6306162058	angella.cetta@hotmail.com	Angella	Cetta	Yes	899609	405

Old

customer_id	mobile number	email	first name	last name	active	pincode	centre_id
201	3883909049	lisha@centini.org	Lisha	Centini	Yes	925345	475
202	5101617924	arlene_klusman@gmail.com	Arlene	Klusman	Yes	405252	402
203	1295676492	alease@buerni.com	Alease	Buerni	Yes	733382	403
204	9317467010	louisa@cronauer.com	Louisa	Cronauer	Yes	272472	419

5. Delete any customer's details from the customer table

```
* postgresql://postgres:***@localhost:5432/201901182_db
1 rows affected.

Out[16]: []

In [18]: M XXsql
        DELETE FROM "ISP".customer WHERE customer_id='201';

* postgresql://postgres:***@localhost:5432/201901182_db
1 rows affected.

Out[18]: []
```

This is the PgAdmin window showing the customer 201 is now deleted.

Old

The screenshot shows the PgAdmin interface with the 'customer' table selected in the left sidebar. The main pane displays a query editor with the following SQL code:

```
1 SELECT * FROM "ISP".customer
2 ORDER BY customer_id ASC
```

Below the query editor is a data grid showing four rows of customer data:

	customer_id	mobile number	email	first name	last name	active	pincode	centre_id
1	201	3883909049	lisha@centini.org	Lisha	Centini	Yes	925345	475
2	202	5101617924	arlene_klusman@gmail.com	Arlene	Klusman	Yes	405252	402
3	203	1295676492	alease@buemi.com	Alese	Buemii	Yes	733382	403
4	204	9317467010	louisa@cronauer.com	Louisa	Cronauer	Yes	272472	419

New

The screenshot shows the PgAdmin interface with the 'customer' table selected in the left sidebar. The main pane displays a query editor with the same SQL code as before:

```
1 SELECT * FROM "ISP".customer
2 ORDER BY customer_id ASC
```

Below the query editor is a data grid showing five rows of customer data, including the newly added row:

	customer_id	mobile number	email	first name	last name	active	pincode	centre_id
1	202	5101617924	arlene_klusman@gmail.com	Arlene	Klusman	Yes	405252	402
2	203	1295676492	alease@buemi.com	Alese	Buemii	Yes	733382	403
3	204	9317467010	louisa@cronauer.com	Louisa	Cronauer	Yes	272472	419
4	205	6306162058	angella.cetta@hotmail.com	Angella	Cetta	Yes	889609	405
5	206	3364849934	cgoldammer@cox.net	Cyndy	Goldammer	Yes	877368	406

6. Update any customer's complaint status from the customer support table

The screenshot shows a Jupyter Notebook window with the following content:

```

In [17]: 
DELETE FROM "ISP".customer WHERE customer_id='201';
* postgresql://postgres:***@localhost:5432/201901182_db
1 rows affected.

Out[17]: []

In [18]: 
UPDATE "ISP"."customer support"
SET "complaint status"='resolved'
WHERE complaint_id='505';
* postgresql://postgres:***@localhost:5432/201901182_db
1 rows affected.

Out[18]: []

In [19]: 
UPDATE "ISP"."customer support"
SET "complaint status"='resolved'
WHERE complaint_id='505';
* postgresql://postgres:***@localhost:5432/201901182_db
1 rows affected.

Out[19]: []

```

This is the PgAdmin window showing the complaint_id 505's status being updated from pending to resolved.

New

The screenshot shows a PgAdmin 4 interface with the following details:

- Browser:** Shows various database objects like FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, and Tables (10).
- Query Editor:** Contains the following SQL query:


```
1 SELECT * FROM "ISP"."customer support"
2 ORDER BY complaint_id ASC
```
- Data Output:** Displays the results of the query in a table format. The table has columns: complaint_id, customer_feedback, helpline_number, complaint_status, centre_id, customer_id, employee_id. The data shows 6 rows, with the last row (complaint_id 505) highlighted in blue.

	complaint_id	customer_feedback	helpline_number	complaint_status	centre_id	customer_id	employee_id
1	501	8	7472683523	resolved	...	406	206
2	502	3	6857327110	resolved	...	418	218
3	503	10	3019550115	resolved	...	405	205
4	504	2	1789802957	resolved	...	436	236
5	505	9	6838789722	pending	...	421	221
6	506	5	4982530760	in process	...	462	262

Old

The screenshot shows a PgAdmin 3 interface with the following details:

- Browser:** Shows various database objects like Indexes, RLS Policies, Rules, Triggers, and tables like customer.
- Query Editor:** Contains the following SQL query:


```
1 SELECT * FROM "ISP"."customer support"
2 ORDER BY complaint_id ASC
```
- Data Output:** Displays the results of the query in a table format. The table has columns: complaint_id, customer_feedback, helpline_number, complaint_status, centre_id, customer_id, employee_id. The data shows 6 rows, with the last row (complaint_id 505) highlighted in blue.

	complaint_id	customer_feedback	helpline_number	complaint_status	centre_id	customer_id	employee_id
1	501	8	7472683523	resolved	...	406	206
2	502	3	6857327110	resolved	...	418	218
3	503	10	3019550115	resolved	...	405	205
4	504	2	1789802957	resolved	...	436	236
5	505	9	6838789722	resolved	...	421	221
6	506	5	4982530760	in process	...	462	262