



Dhirubhai Ambani
Institute of Information and Communication Technology

Vaccination or Other Health Drives Database

IT214 - Database Management System (2022)
Prof. Minal Bhise

NAME	STUDENT ID
Rons Adhaduk	202001214
Hardi Sanghani	202001106

Mentor TA: Devanshi (201911060)

Index

Section 1: Final version of SRS	3
Section 2: Noun Analysis	24
Section 3: ER Diagrams all versions	35
Section 4: Conversion of Final ER-Diagram to Relational Model	38
Section 5: Normalisation and Schema Refinement	40
Section 6: SQL	46
Section 7: Product Code with Output Screenshots	100

Section 1

Final version of SRS

A. Introduction

1.1 Purpose:

The main purpose of this system is to create an improved and efficient management system for the vaccination drives and other disease control drives. The system will be able to manage all the details about users, vaccine centers, employees & managers of respective vaccine centers, etc. The system can be used in general-purpose for other types of vaccines or disease control systems also.

1.2 Intended Audience:

This srs document is intended for database engineers, project managers, section managers, the system design team, and developers. This will also be available as a reference for Health scientists and researchers and government officers.

1.3 Product Scope:

The software intends to make the process of Vaccination Drives and Disease control Drives simple. It has the following functionalities :

- Keeps track of all vaccination centers, stock of the vaccines, vaccine requirements per center, and also other disease control drives.
- Improves the efficiency of the vaccination process or Disease control drives.
- Keeps track of the user details who are already vaccinated or registered.
- Managing the database, related to the details of the users, strength of the vaccinated persons.

Description

In earlier times when we didn't have enough technologies like today, so global diseases or pandemic situations lead to world crises and economic falls, hospitals can be flooded by patients and people would also lose financial support, due to these technological limitations, we were not able to organize well-managed vaccination drives. At that time people managed to take vaccines from the available hospitals or from local health centers. At that time all the people were not able to take vaccines, also there was not a well-managed database for the people who have already been vaccinated, there was not any management about stocks of vaccines, also there was not any kind of schedule for vaccination, etc.

If we take the reference of the COWIN online vaccination drive, if some users will request a vaccine slot then the system will provide all the available centers in an unordered way, but we can implement some functionality in the system such that the system will always provide the nearest available center with the available slots, whenever the user request for a slot. Also in the COWIN vaccination system, the user is not able to pre-register the slot for vaccines. These would lead to the idea of creating a better and well-managed efficient vaccination system.

This system will keep the database of the vaccination drives or disease control programs. The system will keep track of the details of all the vaccination centers along with their locations and stock of the vaccines at respective vaccination centers, also this will keep records of all the managers and employees working at respective vaccination centers.

The system will also keep a record of all the user information of the patients along with the age groups, and other details. The manager of the respective vaccination center will also be able to generate the report about all the users who are vaccinated at that particular vaccination center, and also the system can generate the report about all the vaccinated persons.

Prominent features/working flow:

- **Administration signup** - The administration (Government officials) first have to sign up in the system using their details such as admin name, admin gender, admin city, etc. Here the admins have all the privileges regarding all the functionalities and features in the system.
- **Slot booking through the query** - First of all, the users have to sign up into the system, using their details such as user-name, user age, user-gender, user-city, etc. After signing up the Users can book slots according to their convenience and availability of slots, and the system will provide information about the booked slot such as slot timings.
- **Displaying the nearest available vaccination centers** - The system will be able to display the vaccination centers with available slots along with their details (i.e center-name, center-city, etc.) based on their ascending order of the distance from the particular user's location who requested a vaccination slot.
- **Certification and User report** - Each center will have some employees, the manager of the particular center will assign the employees. At first, the employees assigned to the respective vaccination centers have to sign up into the system with their details such as employee-name, employee-gender, etc. After that, the employees will be given some privileges in the system. The employees of a particular center will also be able to issue the certificates and also be able to generate the report about all the users who have been vaccinated or registered at the particular center.
- **Manager details** - All the managers first have to sign up in the system using their details such as manager-name, manager-gender, manager-center name, etc. Each center will have a manager, and The administration assigns the managers and also will be able to generate the details about all the managers of particular city centers. Also
- **Latest Guidelines and registration process** - The system will contain updated guidelines about the vaccines or medicines for particular diseases, proper steps for the registration process for booking particular vaccine slots.
- **Ordering of vaccines based on center requirements** - The system will take care of the total availability of the vaccines per center and managers of the respective centers can request the administration about the requirement of the vaccines and administration can order the vaccines from the inventory based on requirements.
- **Displaying the nearest inventories** - The system will be able to display the inventories details such as inventory name, inventory city, etc. based on their ascending order of distance from the particular administration location, which has requested the vaccine stock.

- **Slot vaccine quantity updation** - The system will be able to update the center's vaccine stock based on the order and the number of vaccines per slot on a daily basis.
- **Well-managed Inventory** - The system has a well-managed inventory tracking system for ordering, delivering, classifying the classes of the vaccines.

Ordering of the vaccines for the particular center will be done by the manager, he has decided the particular threshold values to reorder the vaccines, if the current stock of the vaccines will drop to that threshold value then the manager will request the upper level (i.e Administration) for more stocks about the vaccines, and the Administration (Government official) has all the order details such as order id, vaccine-type, etc.. and admin is allowed to order the vaccines directly from the main inventory.

To distinguish between the types of vaccines to order, each of the vaccines will be marked by a particular vaccine type. So the manager can get an idea about the particular vaccines to order. According to that, the manager keeps a record of different vaccines.

B. Fact-Finding phase

1. Background readings with Description:

- **Websites:**

1. [CoWIN](#)
2. [COVID-19 vaccination in India](#)

In recent days, India and other countries have encountered the worldwide pandemic of coronavirus. In order to overcome this pandemic situation, governments of all the countries are collaborating with self-finance companies to start vaccination drives all over the world. Here we have discussed the working flow and limitations of, particularly covid-19 vaccination drive.

Specifically talking about India, the Indian government started COVID-19 Vaccination for the frontline workers in phase I, after phase I the government started the vaccination for all the residents through the online platform named COWIN. COWIN is the Indian government web portal for COVID19 vaccine registration and it is operated by India's Ministry of Health and Health Welfare.

- **Working flow and limitations of COWIN vaccination system**

For getting vaccinated every individual has to first register himself/herself to the COWIN portal, after that he/she has to be authenticated by OTP sent in one's phone number, after that one has to book a slot for the vaccines at a particular vaccination center on the COWIN web portal, the individual will get the vaccination date and the name of the vaccine with which he/she will be vaccinated, and after the vaccination, the individual will be provided a certificate of vaccination on the COWIN web portal itself. But here this system has some limitations, when an individual requests a vaccination slot, the system will sometimes provide all the vaccination centers without any concern of the availability or non-availability of a particular vaccination center's vaccines availability, also regardless of the vaccination center's distance from the individual's location. Also, the individual was not able to pre-register the vaccination slots based on their convenience.

Document References:

- [Software Requirements Specification Personal Health Monitoring Application \(PHMA\)](#)
This document describes SRS for a project for the personal health monitoring application (PHMA). This document gives a broad idea about the functional and non-functional requirements along with design phases of the system, assumptions regarding the system, features, etc. for developing such health systems.
- [\(PDF\) Software Requirements Specification of E-Health Architecture for Nepal](#)
This document was prepared for developing the complete E-Health architecture for Nepal, this SRS document gives the general idea about the functional and non-functional requirements for developing efficient E-Health systems.

Requirements Gathered from Background readings

- The system should not display the unavailable slots for the vaccination.
- The system should show the nearest vaccination centers.
- The user should be able to pre-register for the slots.

2 Interviews:

2.1 Interview1:

System: Vaccination Drive Management

Project Reference: SF/SJ/2021/10

Interviewee: Dhairyra Rupala

Designation: Head at civil health center

Interviewer: Satyam Bhut

Designation: Developer

Date: 05/10/2021

Duration: 40 mins

Mode: Online (Google meet)

Purpose of Interview: Meeting for sharing our ideas of the project and discussion about the requirements and features required as an admin(Head of programs) of the system.

Agenda:

- Problems regarding the currently existing system as an admin.
- What are the challenges faced or can rise, in conducting the disease control drives or vaccination drives?
- What are the expectations or privileges as an admin, in conducting the drives?
- Any new ideas regarding the system.

Documents to be brought to the interview:

- Rough plan of resources required for the system
- Rough plan of features and functionalities we will be implementing in the system.

Interview1 Summary:

System: Vaccination Drive Management

Project Reference: SF/SJ/2021/10

Participants: Dhairy Rupala (Head at civil health center)
Satyam Bhut (Developer)

Date: 05/10/2021

Time: 40 mins

Place: Online (Google meet)

Purpose of Interview:

1. The UI should be more user-friendly and should be more efficient in terms of slot booking and vaccine information.
 2. The User faces authentication issues sometimes (i.e OTP detection,...)
 3. Backend-related problems.
 4. Improvement is needed in authentication.
-

2.2 Interview2:

System: Vaccination Drive Management

Project Reference: SF/SJ/2021/10

Interviewee: Hiten Padaliya

Designation: Manager at Civil Health center

Interviewer: Jenish Rathod

Designation: Developer

Date: 05/10/2021

Duration: 40 mins

Mode: Online (Zoom)

Purpose of Interview: Meeting for sharing our ideas of the project and discussion about the requirements and features required as a manager of the system.

Agenda:

- Problems regarding the currently existing system as a manager.
- Current development in the Vaccination management system
- How technology can help to manage the disease control drives or vaccination drives better
- Any new ideas regarding the system.

Documents to be brought to the interview:

- Rough plan of resources required for the system
- Rough plan of features and functionalities we will be implementing in the system.

Interview2 Summary:

System: Vaccination Drive Management

Project Reference: SF/SJ/2021/10

Participants: Hiten Padaliya (Manager at Civil Health center)
Jenish Rathod (Developer)

Date: 05/10/2021 **Time:** 40 mins

Place: Online (Zoom)

Purpose of Interview:

1. Improve the features like current guidelines and news updates about the diseases or vaccines.
 2. The interface should be more comprehensive.
 3. Some features are cluttered.
-

2.3 Interview3:

System: Vaccination Drive Management

Project Reference: SF/SJ/2021/10

Interviewee: Hiten Padaliya

Designation: Employee at a health center

Interviewer: Dhairyra Rupala

Designation: Developer

Date: 06/10/2021

Duration: 30 mins

Mode: Online (Google meet)

Purpose of Interview: Meeting for sharing our ideas of the project and discussion about the requirements and features required as an employee of the system.

Agenda:

- Problems regarding the currently existing system as an employee.
- How smoothly can you conduct the drive if you use this system?
- What are the new ideas about the system?

Documents to be brought to the interview:

- Rough plan of resources required for the system
- Rough plan of features and functionalities we will be implementing in the system.

Interview3 Summary:

System: Vaccination Drive Management

Project Reference: SF/SJ/2021/10

Participants: Hiten Padaliya (Employee at health center)
Dhairyra Rupala (Developer)

Date: 06/10/2021

Time: 30 mins

Place: Online (Google meet)

Purpose of Interview:

1. A role-based system should be very helpful.
 2. The roles have their own privileges and features to work with.
 3. The system should have features to handle concurrent access issues.
 4. The system has the ability to handle power failure.
-

2.4 Interview4:

System: Vaccination Drive Management

Project Reference: SF/SJ/2021/10

Interviewee: Jenish Rathod

Designation: User (Civilians)

Interviewer: Hiten Padaliya

Designation: Developer

Date: 07/10/2021

Duration: 40 mins

Mode: Online (Google meet)

Purpose of Interview: Meeting about sharing ideas about the existing system and the need for improvement.

Agenda:

- To discuss the challenges that the user was facing in the current system.
- To take feedback regarding the current system.
- To discuss the new system and their implementation
- Any new ideas regarding the system

Documents to be brought to the interview:

- Pros and cons of the current systems.
- Rough plan of features and functionalities we will be implementing in the system.

Interview4 Summary:

System: Vaccination Drive Management

Project Reference: SF/SJ/2021/10

Participants: Jenish Rathod(User)

Hiten Padaliya (Developer)

Date: 07/10/2021

Time: 40 mins

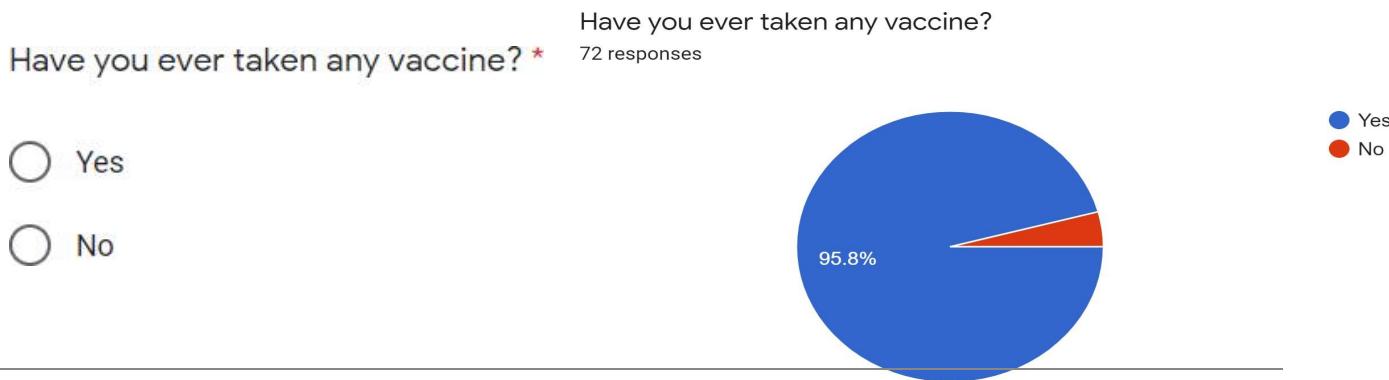
Place: Online (Google meet)

Purpose of Interview:

1. There should be some feature about showing the registration process for the slot bookings.
2. The system should display the nearest available vaccination centers from the user's area.

3. Questionnaires:

Below are the questions and responses gathered in the google form. We collected the responses from 3 types of designations who might possibly use the system, which are Users, Managers of health centers, and Administration (Government) of the vaccination drives/disease control programs.



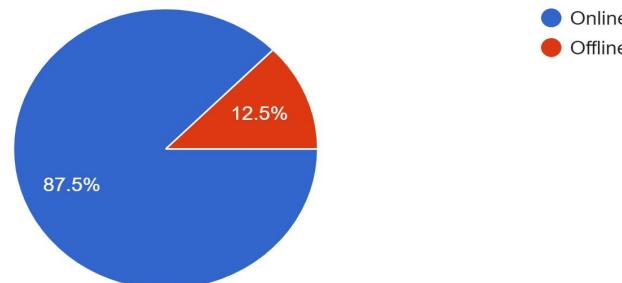
Preferred mode of registration.*

Preferred mode of registration.*

Offline

Online

Preferred mode of registration.
72 responses

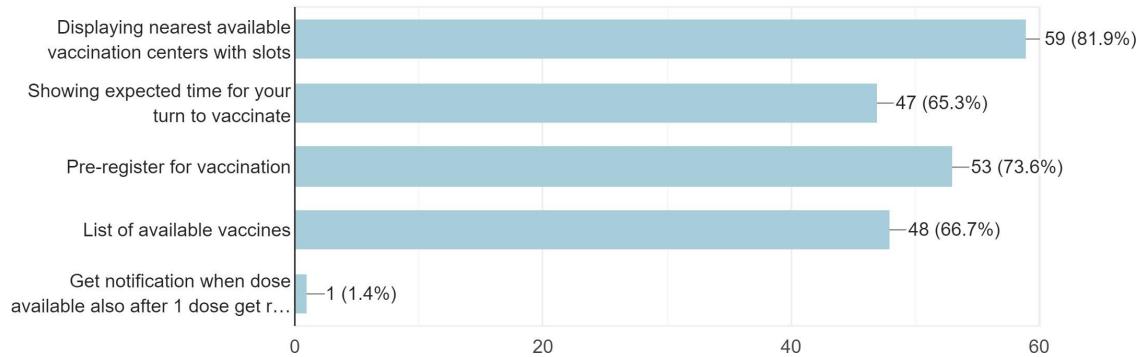


As a user what kind of functionalities do you expect in the system? *

- Displaying nearest available vaccination centers with slots
- Showing expected time for your turn to vaccinate
- Pre-register for vaccination
- List of available vaccines
- Other: _____

As a user what kind of functionalities do you expect in the system?

72 responses

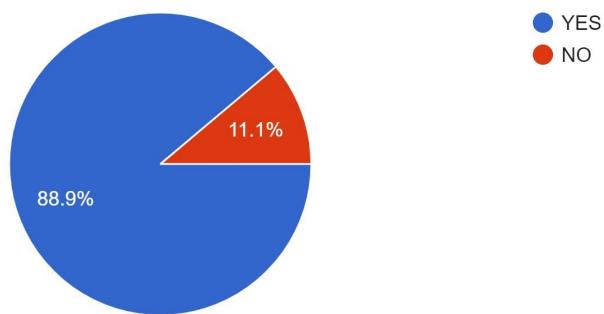


Would you like to receive notification when there is some available slots for vaccines? *

- YES
- NO

Would you like to receive notification when there is some available slots for vaccines?

72 responses

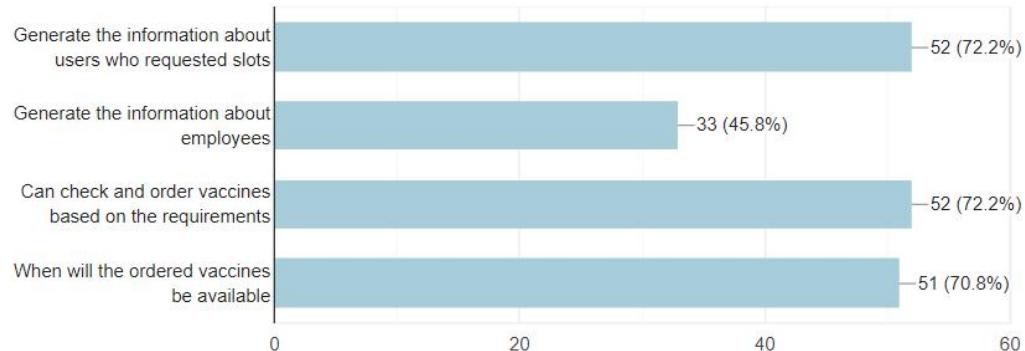


As a manager of particular vaccination center what kind of functionalities do you want in the system? *

- Generate the information about users who requested slots
- Generate the information about employees
- Can check and order vaccines based on the requirements
- When will the ordered vaccines be available
- Other: _____

As a manager of particular vaccination center what kind of functionalities do you want in the system?

72 responses

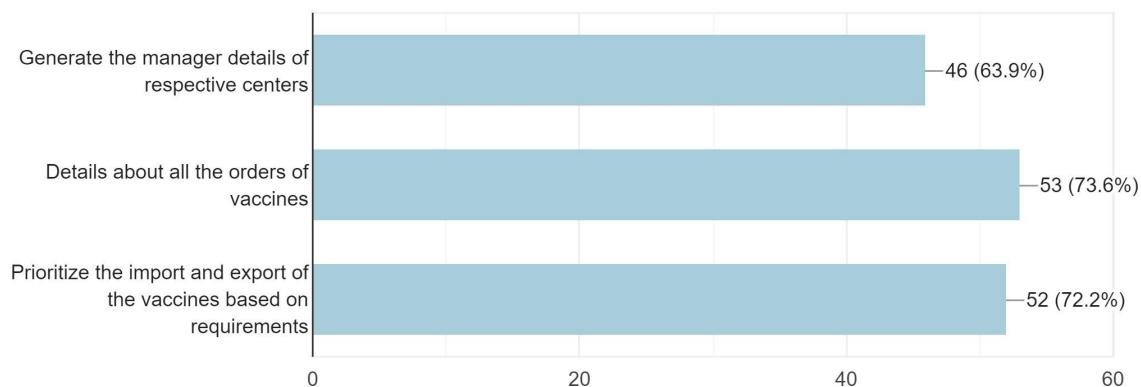


As a administration what kind of functionalities do you want? *

- Generate the manager details of respective centers
- Details about all the orders of vaccines
- Prioritize the import and export of the vaccines based on requirements
- Other: _____

As a administration what kind of functionalities do you want?

72 responses

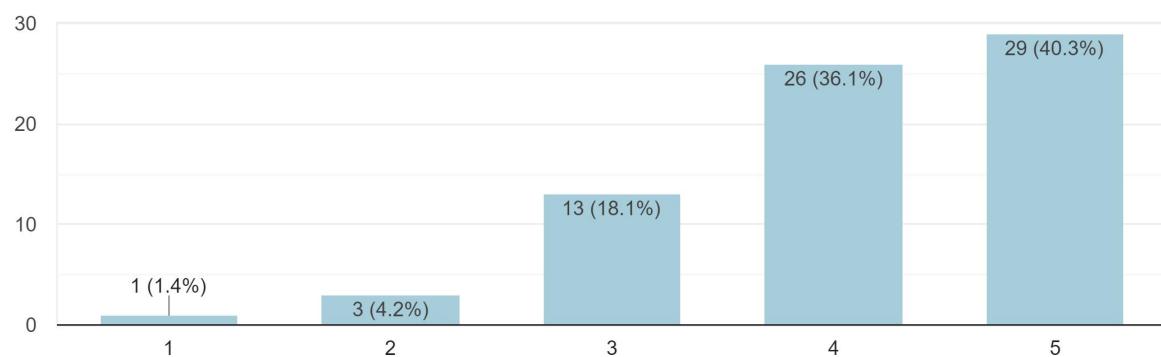


Rate the vaccination process. *

1	2	3	4	5	
Lowest	<input type="radio"/> Highest				

Rate the vaccination process.

72 responses

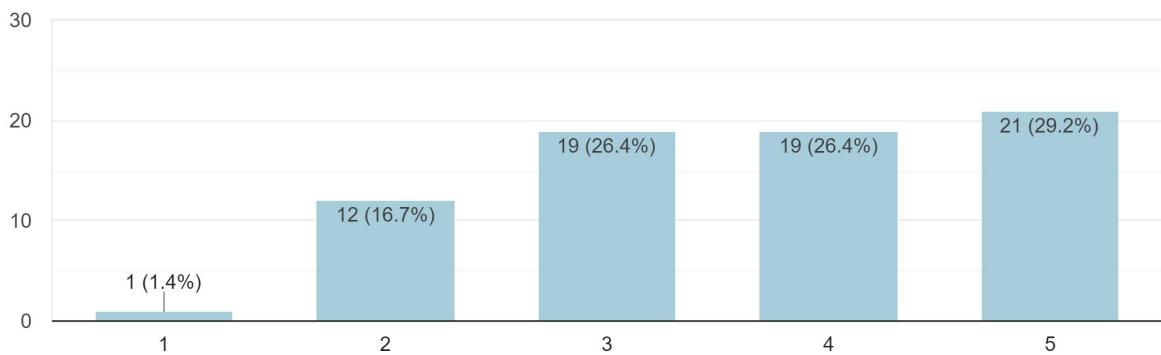


Rate the slot booking for vaccination drive. *

1	2	3	4	5	
Lowest	<input type="radio"/> Highest				

Rate the slot booking for vaccination drive.

72 responses



Requirements Gathered from QnA:

- Most of the users are familiar with the online registration systems and it was also more flexible to use.
- Most of the people are faced with issues in finding the nearest vaccination slots and also recommended the pre-registration process and get notified so they don't have to wait for vaccination availability.
- The majority of the users are quite satisfied with the current slot booking process but few of them are not much satisfied and it has to be improved.
- With respect to the manager, it would be better if the system is able to generate the final list of the users who requested slots and also generate the information regarding the status of the vaccines and the list of the employees working in the center.
- With respect to the administration, it would be better if the system has the functionality to prioritize the orders of the vaccines and can monitor the information about the managers.
- The overall performance of the existing system is good, but it has some considerable issues to take care into account.

4. Observation:

Infosoft Softwares: Observations System:

Vaccination Management System Project

Reference: SF/SJ/2021/10

Observations by: Mitul Dave (Head developer at Infosoft software)

Date: 14/8/2003 **Time:** 14:30

Duration: 45 minutes **Place:** Microsoft Teams (Online)

Observation:

- The core idea of the system is very intuitive and innovative.
- The system can be very useful for the Online management of vaccination drives/disease control programs.
- The current system has some bugs in the authentication procedure for the registration, it should be fixed.
- The overall architecture, performance, and functionality of the system are quite admirable.

Requirements Gathered from Observation

- Solve the issues regarding authentication while registering.

C. Fact-Finding Chart:

Objective	Technique	Subjects(s)	Time commitment
To get the ideas about workflows and management of existing systems	Background readings	Similar readings, few existing projects	1 Day
To gain understanding about the role of the administrator/corporation of the vaccine management drives/disease control programs	Interview	Admin(Head of drives/programs)	40 Minutes
To gain an understanding of the role of the manager of particular vaccination centers	Interview	Manager of a health center	40 Minutes
To gain an understanding of the role of the employees of particular vaccination centers	Interview	An employee of the health center	30 Minutes
To gain an understanding of the role of the users, who might use the system	Interview	Civilians	40 Minutes
To collect the information about the requirements	Questionnaire	Civilians	1 Day

D. Requirements

Requirements in the system for Users

- The system should display the nearest available vaccination center from the user's area.
- The user should be able to pre-register for the vaccination.
- The user should be able to see the list of the available vaccines/medicines and their information, also the user can update their personal details.
- The user should get a notification about the available vaccination slots.
- The system should display the expected time for the user's turn for the vaccination.
- The system's UI should be more user-friendly and comprehensive, also the authentication system should be improved.
- The system should have features like current guidelines or news updates for users.
- The system should show the registration steps to the users for the registration procedure.

Requirements in the system for Managers

- As a manager, the system should generate information about the users and their respective slots which are booked.
- The manager should be able to check the stock of the vaccines/medicines and can order based on the requirements for respective vaccination centers.
- Also, the manager should be able to see the expected arrival time for the ordered vaccines/medicines.
- As a manager, the system should also generate a list of the employees working in the center and can manipulate it. Also, the manager should be able to update his/her personal details.

Requirements in the system for Administrator

- Admin should be able to generate the details about the ordered vaccines/medicines which are requested from the manager and can manipulate with it or which admin ordered from the inventory itself.
- As an administrator, the system should be able to prioritize the orders of the vaccines based on the requirements.
- As an administrator, the system should be able to generate the details of the managers of particular centers and the admin should be able to manipulate it, also the administrator is able to update his/her own personal details.

E. User Classes and Characteristics:

• **Government Officials (Administrator/Head of programs):**

Government Officials use this system, to manage the vaccination drives/disease control programs, to handle the different vaccination centers, to assign managers, and give them privileges of the system to manage their assigned vaccination centers, also government officials use the system to handle the ordering of vaccines, update the stocks and also, update the current news and guidelines for the users.

• **Managers of the program:**

Managers use this system for handling the health programs in their vaccination centers, also managing the resources and ordering the required resources based on the needs in their respective vaccination centers, along with handling all the

employees working in the vaccination centers. Managers provide some privileges to the employee working in the vaccination center to handle some tasks.

- **Employees:**

Employees working in the respective centers use the system for updating the information about the users who are vaccinated, also the employees have the privilege to provide the certificate or vaccination report to the users who are vaccinated.

- **Users (Civilians):**

Users or civilians use the system for registering themselves for vaccination, and they can book the slots based on their convenience and slots availability.

F. Operating Environment:

This system will run on an online platform, the main communication interface between the user's view and the database will be managed by SQL scripts.

1. Hardware, Software or connectivity requirements:

➤ **Hardware:**

- Processor: Intel i3 (minimum)
- Clock speed: 1.2GHz
- Memory: 50GB
- RAM: 2GB
- Operating System: Windows 7/8/10

➤ **Software:**

- Database Handler: SQL
- Software Building Tool/ Host language: Python

➤ **Connectivity:**

- Internet Speed: 15Mbps

2. External Interface Requirements:

- We need a Google Maps API to track the locations and to find the nearest centers of the various vaccination centers, also we will require an API for current guidelines about the diseases and vaccine information.

G. Product Functions

- **Register/ Sign up:**

Through this functionality, users or managers or employees, or admin can register themselves in the system. Users/managers/admin/Employees have to provide the details required for the registration process and password for signing up. The system will also provide the steps as a guideline to register on the system.

- **Sign In:**

Through the details provided in the sign-up process and password, the managers/users/employees or admin can sign in to the system.

- **Current_Guidlines:**

This functionality will provide the current news and guidelines about the diseases.

- **Retrieve_details**

This functionality can be used by the admin/managers or the employees, the admin will be able to see all the details about the managers, the manager will be able to see all the details about the employees, employees will be able to see all the details about the users.

- **Order_vaccines:**

This functionality is for both the managers and admins of the system, the manager can request the order of the vaccine from the admin and the admin can order the vaccines from the inventory based on the requirements.

- **update_vaccineStock:**

This functionality will help the manager and admins to see the vaccine stock and update the stock based on the orders.

- **add_manager:**

Using this functionality the admin can assign or recruit the manager for the respective vaccination center.

- **update_managerDetails :**

With this functionality, an admin can manipulate the information of managers. But the manager can update his/her own personal details only.

- **add_employee:**

This functionality will help the managers of the particular center to recruit or add employees.

- **update_employeeDetails:**

With the functionality Managers can update the information about all the employees but, a particular employee will be able to update his/her own details only.

- **update(userInfo**:

With this functionality, Employees/Users users will be able to update the user details and the employee will be able to update the status as vaccinated or not vaccinated only, but the user can update his/her own details only like name, location, etc.

- **provide_certificate**:

With this functionality, the Employees will be able to provide the certificates to the users who are vaccinated.

- **Search_vaccines**:

With the functionality, the user will be able to search for the required medicines or vaccines.

- **find_nearest**:

This filter/functionality will show the nearest available vaccination slot with center details to the users.

- **Book_slot**:

Using this functionality the user will be able to book the slot, at first, the user has to select the vaccine and the system will show the available slots along with centers, and if the user selects the filter *find_nearest* then the system will show the available slots and sort them according to the distance with respect to the user's location. Also, the user will be able to pre-register the slots.

H. Privileges:

Functions	Privileges
Sign Up, Sign in	Administration, Manager, Employee, User
Search_Vaccines	User
Slot_Booking	User
Provide Certificate	Employee
Manipulating with User_Details	Employee, User
Manipulating with Employees or Employee_Details	Manager, Employee
Manipulating with Managers or Manager details	Administration, Manager
Orders/Update of Vaccines	Administration, Manager
Guidelines related to Disease	Administration, Manager, Employee, User

* here the admin of the system will be able to access all the modules but, there is a specific role assigned for all the small tasks.

I. Assumptions:

- Here the architecture will be focusing on providing better management for vaccination drives/disease control programs online.
- So it is presumed that the internet facility, power supply, internet speed/bandwidth is always up and running, also it is assumed that all the information and data is up to date.
- Also, the server of the system will be live 24x7, and the server has enough capability to handle large amounts of data.
- Also, it is assumed that all the users of this system will have the required hardware and software specifications.

J. Business constraints

- The system will contain all the updated information about the data of the users vaccinated, stocks of vaccines, availability of particular vaccines/medicines, etc.
- Also, all the information/data can be recovered and will not be corrupted in case any system failures or power shortage occurs.
- Also, the data should be flexible and consistent in case any migration occurs.
- The system will be secured in a way, such that all the personal details of the users, managers, or admin will not be compromised.

Section 2

Noun Analysis

Final Problem Description

In earlier times when we didn't have enough technologies like today, so global diseases or pandemic situations lead to world crises and economic falls, hospitals can be flooded by patients and people would also lose financial support, due to these technological limitations, we were not able to organize well-managed vaccination drives. At that time people managed to take vaccines from the available hospitals or from local health centers. At that time all the people were not able to take vaccines, also there was not a well-managed database for the people who have already been vaccinated, there was not any management about stocks of vaccines, also there was not any kind of schedule for vaccination, etc.

If we take the reference of the COWIN online vaccination drive, if some users will request a vaccine slot then the system will provide all the available centers in an unordered way, but we can implement some functionality in the system such that the system will always provide the nearest available center with the available slots, whenever the user request for a slot. Also in the COWIN vaccination system, the user is not able to pre-register the slot for vaccines. These would lead to the idea of creating a better and well-managed efficient vaccination system.

This system will keep the database of the vaccination drives or disease control programs. The system will keep track of the details of all the vaccination centers along with their locations and stock of the vaccines at respective vaccination centers, also this will keep records of all the managers and employees working at respective vaccination centers.

The system will also keep a record of all the user information of the patients along with the age groups, and other details. The manager of the respective vaccination center will also be able to generate the report about all the users who are vaccinated at that particular vaccination center, and also the system can generate the report about all the vaccinated persons.

Prominent features/working flow:

- **Administration signup** - The administration (Government officials) first have to sign up in the system using their details such as admin name, admin gender, admin city, etc. Here the admins have all the privileges regarding all the functionalities and features in the system.
- **Slot booking through the query** - First of all, the users have to sign up into the system, using their details such as user-name, user-age, user-gender, user-city, etc. After signing up the Users can book slots according to their convenience and availability of slots, and the system will provide information about the booked slot such as slot timings.
- **Displaying the nearest available vaccination centers** - The system will be able to display the vaccination centers with available slots along with their details (i.e center-name, center-city, etc.) based on their ascending order of the distance from the particular user's location who requested a vaccination slot.
- **Certification and User report** - Each center will have some employees, the manager of the particular center will assign the employees. At first, the employees assigned to the respective vaccination centers have to sign up into the system with their details such as employee-name, employee-gender, etc. After that, the employees will be given some privileges in the system. The employees of a particular center will also be able to issue the

certificates and also be able to generate the report about all the users who have been vaccinated or registered at the particular center.

- **Manager details** - All the managers first have to sign up in the system using their details such as manager-name, manager-gender, manager-centername, etc. Each center will have a manager, and The administration assigns the managers and also will be able to generate the details about all the managers of particular city centers. Also
- **Latest Guidelines and registration process** - The system will contain updated guidelines about the vaccines or medicines for particular diseases, proper steps for the registration process for booking particular vaccine slots.
- **Ordering of vaccines based on center requirements** - The system will take care of the total availability of the vaccines per center and managers of the respective centers can request the administration about the requirement of the vaccines and administration can order the vaccines from the inventory based on requirements.
- **Displaying the nearest inventories** - The system will be able to display the inventories details such as inventory name, inventory city, etc. based on their ascending order of distance from the particular administration location, which has requested the vaccine stock.
- **Slot vaccine quantity updation** - The system will be able to update the center's vaccine stock based on the order and the number of vaccines per slot on a daily basis.
- **Well-managed Inventory** - The system has a well-managed inventory tracking system for ordering, delivering, classifying the classes of the vaccines.

Ordering of the vaccines for the particular center will be done by the manager, he has decided the particular threshold values to reorder the vaccines, if the current stock of the vaccines will drop to that threshold value then the manager will request the upper level (i.e Administration) for more stocks about the vaccines, and the Administration (Government official) has all the order details such as order id, vaccine-type, etc.. and admin is allowed to order the vaccines directly from the main inventory.

To distinguish between the types of vaccines to order, each of the vaccines will be marked by a particular vaccine type. So the manager can get an idea about the particular vaccines to order. According to that, the manager keeps a record of different vaccines.

I. Noun (& Verb) Analysis:

(1) Nouns (Entities) and verbs (relationships) in the description.

Noun	Verb
Final problem description	can
technologies	Like
Global diseases	read
Pandemic situations	Be
World crisis	Would
Economic falls	lose
hospitals	support
patients	organize
Financial supports	classifying
Technological limitations	take
Vaccination drives	schedule
vaccines	reference
time	drive
time	will
hospitals	request
Local health centers	provide
vaccines	Implement
database	Register
Stocks of vaccines	Control
Schedule for vaccination	Track
reference	Keep
Cowin	Generate
Managers	Assigns
Vaccination drive	Record

users	Report
Order details	allowed
Order id	Delivering
vaccine slots	Flow
Main inventory	Query
Available centers	Display
system	order
Nearest available center	issue
administration	Assigns
Available slots	Process
user	Update
Cowin	Reorder
Vaccination system	Drop
user	Fall
slot	Is
vaccines	Programs
Vaccination system	Groups
database	steps
Vaccination drives	features
Disease control drives	Has
details	values
Vaccination centers	types
The stock of the vaccines	flooded
respective vaccination centers	managed
records	was
Manager	vaccinated
employees	based
Respective vaccination centers	requested

User information	decided
patients	allowed
manager	marked
Respective vaccination centers	ascending
report	creating
users	booking
Particular vaccination center	delivering
Vaccinated person	according
users	working
slot	using
Prominent features	book
slots	displaying
Nearest available vaccination centers	allocate
vaccination centers	
Available slots	
Vaccination slots	
User report	
employees	
Particular center	
certificates	
users	
manager	
administration	
managers	
Latest guidelines	
Registration process	
medicines	
vaccines	

Particular diseases	
Proper steps	
Vaccine slots	
managers	
administrations	
inventory	
administrations	
inventories	
Administration location	
Vaccine stock	
order	
Daily basis	
Classes of vaccines	
manager	
Threshold values	
Current stock of the vaccine	
Manager	
Upper level	
Administration	
stock	
vaccines	
administration	
government official	
Admin name	
Admin gender	
Admin city	
privileges	
functionalities	

User name	
User age	
User gender	
User city	
information	
Slots timings	
Center name	
Center city	
User's location	
Employee name	
Employee gender	
Manager name	
Manager gender	
Manager centername	
Inventory name	
Inventory city	

(2) Accepted Noun and verbs list

Candidate Entity Set	Candidate Attribute Set	Candidate Relationship Set
Admin	Admin name Admin gender Admin city	Order, assigns
Manager	Manager name Manager gender Manager centerName	Order, assigns
Users	User name User gender User age User city	book
Employee	Employee name Employee gender	Certify
Centers	Center name Center city	
Slots	Slots timings	
Inventory	Inventory name Inventory city	
Order details	Order id, vaccine type	

(3) Truncating Initial Noun List:

Noun	Reason
vaccination	Duplicates, General
vaccines	Duplicates, General
Available	Vague
stock	General, duplicate
drives	Vague, duplicate
system	general

respective	Irrelevant
Particular	Irrelevant
diseases	General, duplicates
hospitals	General, duplicates
patients	general
time	vague
database	general
Cowin	Duplicates
Nearest available centers	vague
report	general
Details	Associations
Final problem description	vague

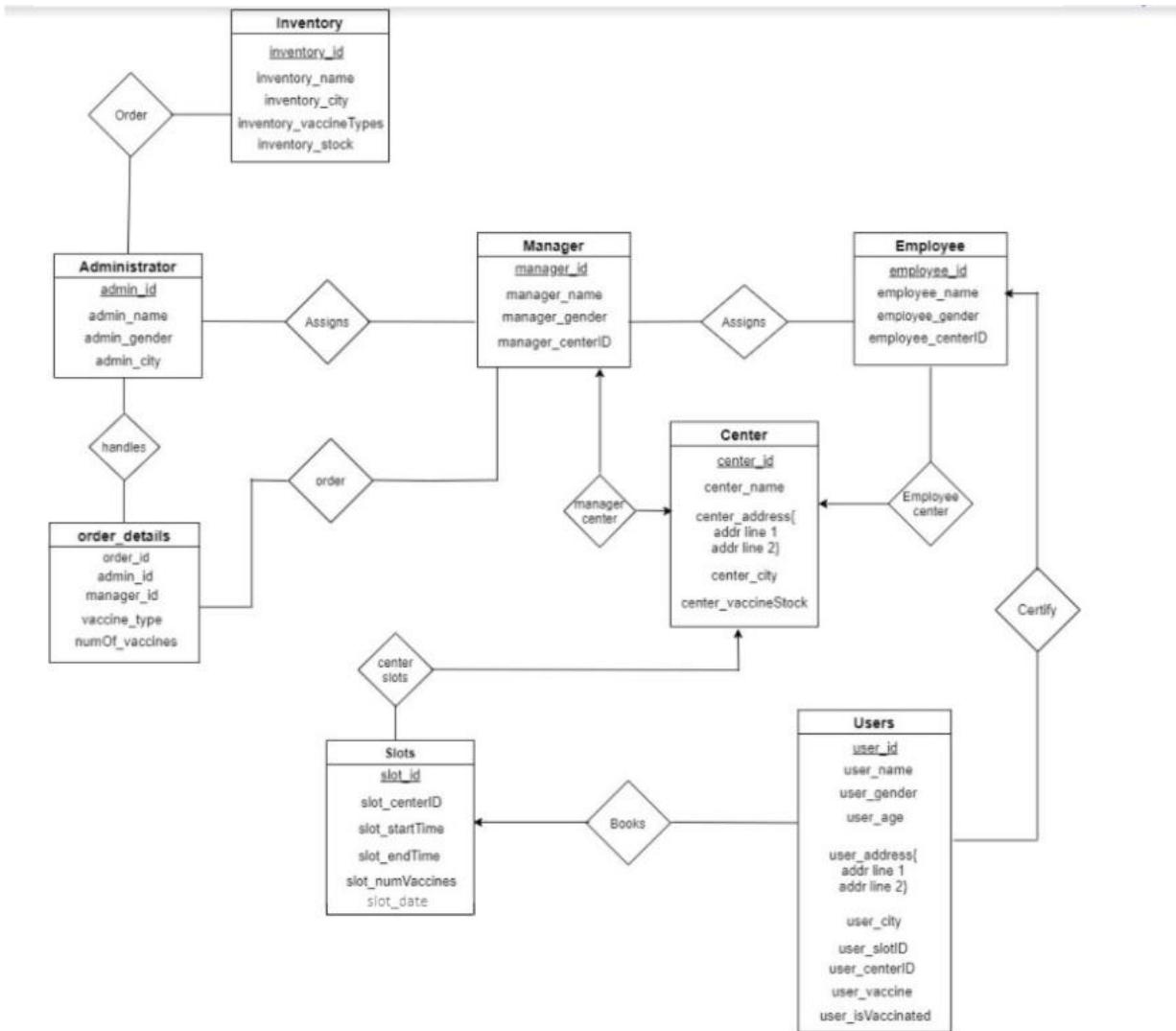
technologies	irrelevant
Pandemic situation	general
World crisis	general
Financial supports	vague
Technological limitations	vague
Local health centers	General
Stocks of vaccines	vague
Reference	vague
details	General
Records	vague
User information	Vague
Prominent features	irrelevant
User report	general
Latest guidelines	General
Registration process	General

Proper steps	Vague
Administration location	attribute
Daily basis	Vague
Threshold values	Irrelevant
Upper level	vague

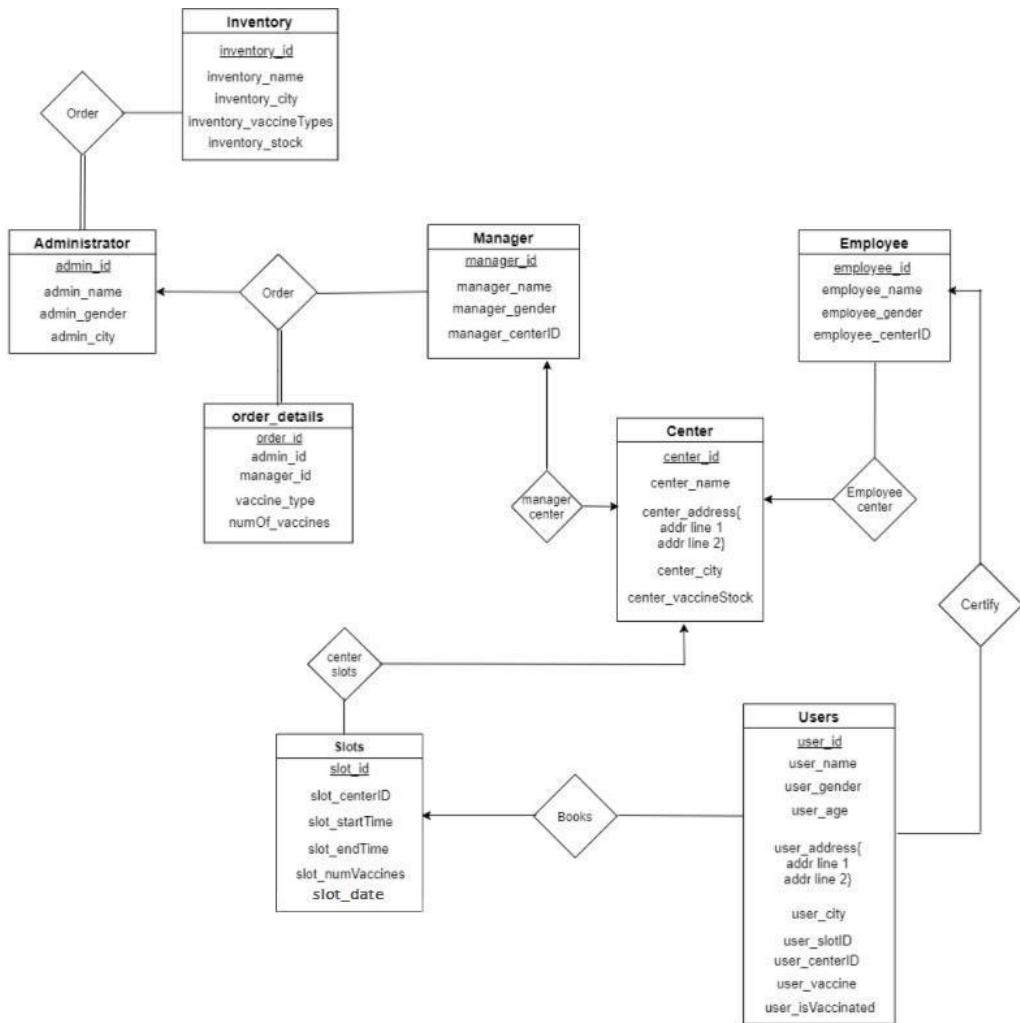
Section 3

ER Diagrams All versions

Develop the ER Diagram (ERD):



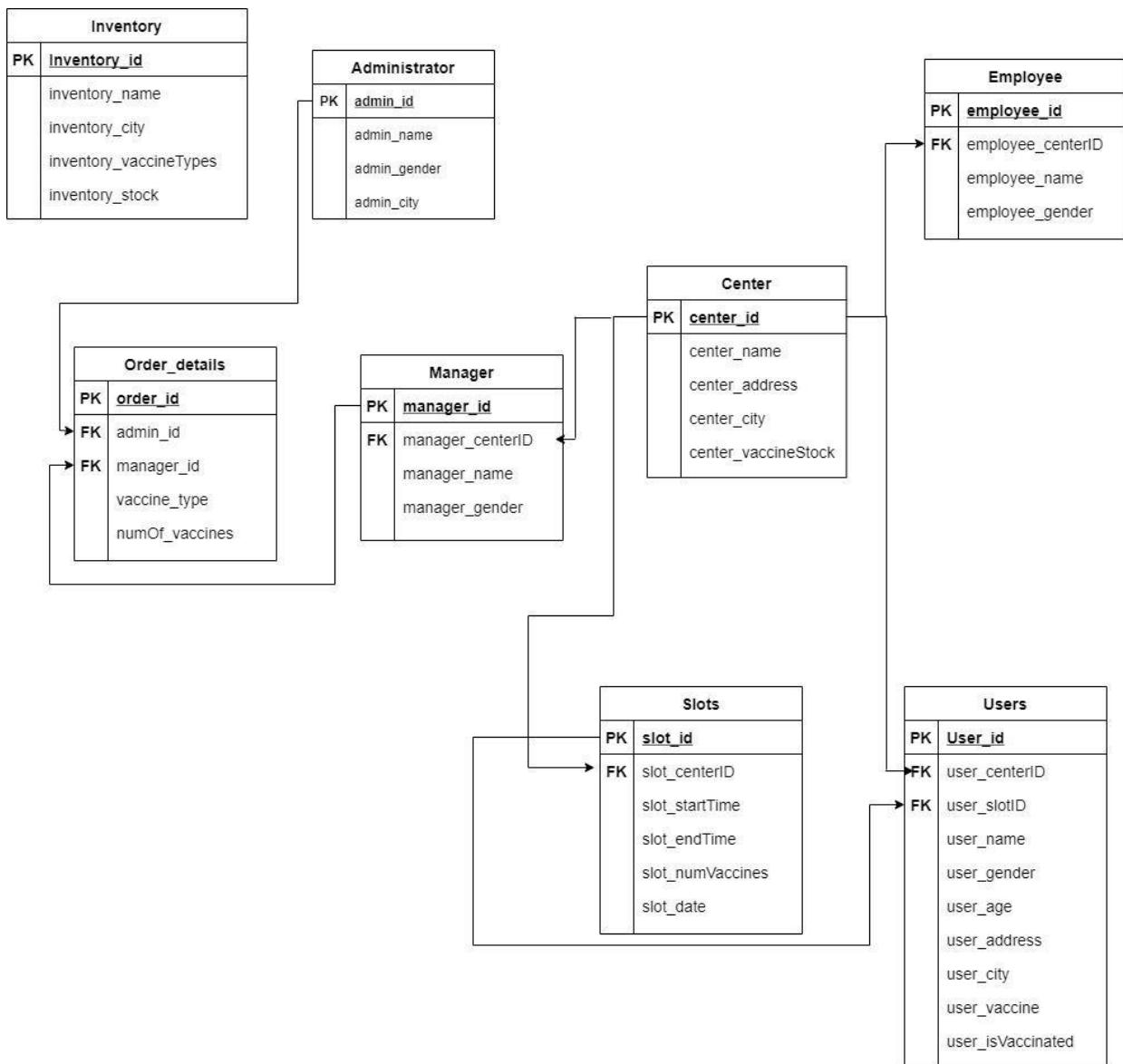
ER Diagram Final:



Section 4

Conversion of Final ER-Diagram to Relational Model

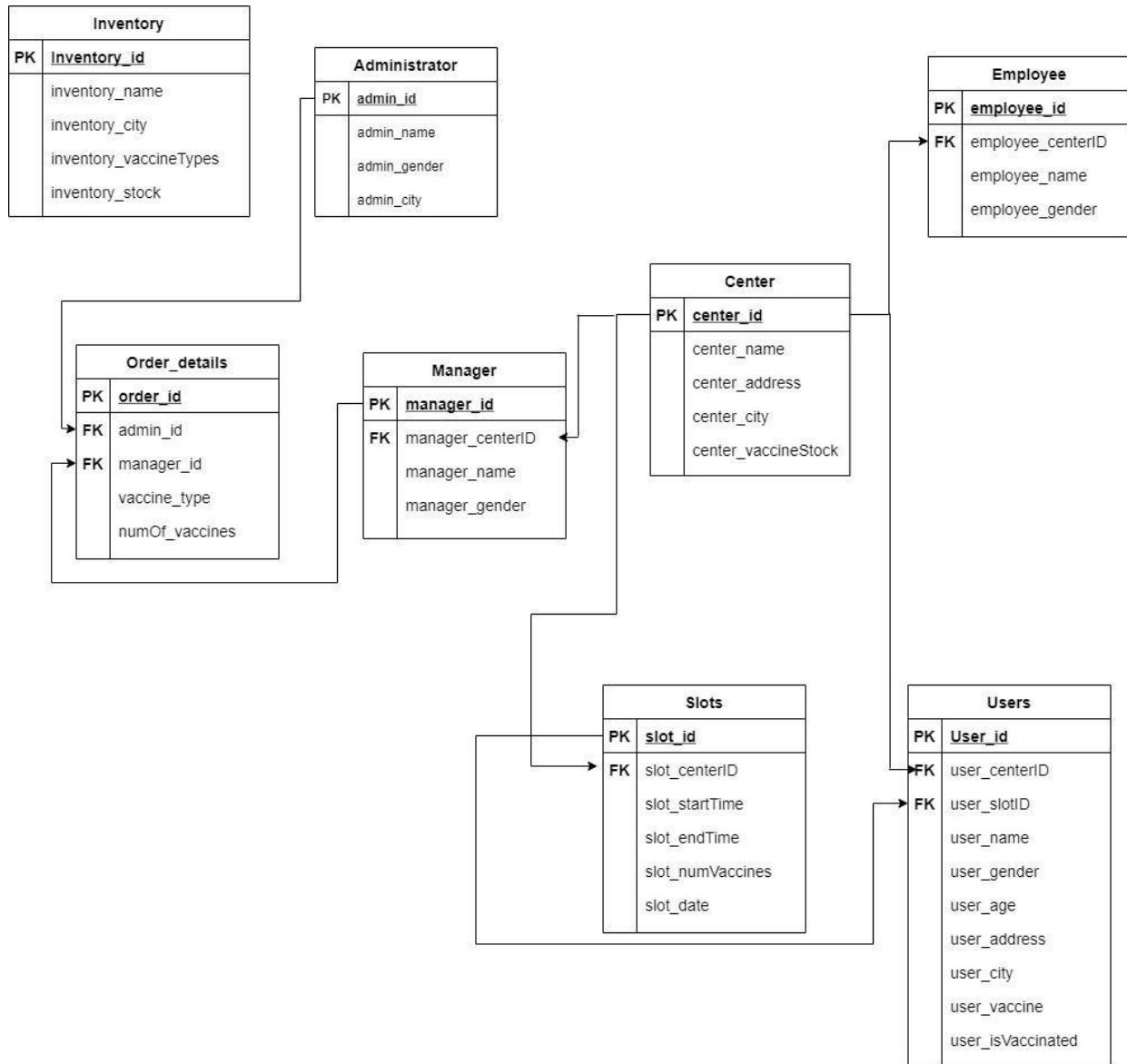
Relational Model:



Section 5
Normalization
And
Schema Refinement

Original Design of Database

Relational Schema



Documentation of Normalization & Schema Refinement

Functional Dependencies:

Administrator

Primary Key = admin_id

Admin_id -> admin_name

Admin_id -> admin_gender

Admin_id -> admin_city

Center

Primary key = center_id

Center_id -> center_name

Center_id -> center_address

Center_id -> center_city

Center_id -> center_vaccineStock

Employee

Primary key = employee_id

Foreign key = employee_centerID

Employee_id -> employee_name

Employee_id -> employee_gender

Manager

Primary key = manager_id

Foreign key = manager_centerID

Manager_id -> manager_name

Manager_id -> manager_gender

Order details

Primary key = order_id

Foreign key = admin_id, manager_id

Order_id -> vaccine_type

Order_id -> numOf_vaccines

Inventory

Primary key = Inventory_id

Inventory_id -> inventory_name

Inventory_id -> inventory_city

Inventory_id -> inventory_vaccineTypes

Inventory_id -> inventory_stock

Slots

Primary key = slot_id

Foreign key = slot_centerID

Slot_id -> slot_startTime

Slot_id -> slot_endTime

Slot_id -> slot_numVaccines

Slot_id -> slot_date

Users

Primary key = user_id

Foreign key = user_centerID, user_slotID

User_id -> user_name

User_id -> user_gender

User_id -> user_age

User_id -> user_address

User_id -> user_city

User_id -> user_vaccine

User_id -> user_isVaccinated

Normalization to 1NF:

In the 1NF form in each and every relation of the Database, there should not be any repeating columns within a row or not multivalued columns.

Here, in our database design, all the listed attributes in the relations are atomic, also wherever we have the address attribute in the relation, we stored the address in a single string. We are also considering only one entry of each of the attributes. So the above database design is the 1NF normalized form.

Normalization to 2NF:

In the 2NF normalized form, each and every relationship should be in the 1NF form also there should not be any partial dependencies in the relations. Partial dependency means that no nonprime attributes are dependent on any proper subset of the candidate key of the relation.

For any database design relations with only one Primary key (not composite) are already in the 2NF form.

Here from the above functional dependencies, it is clear that this database design has no composite primary keys, hence no partial dependencies. So this design is already in the 2NF normalized form.

Normalization to 3NF

For the relations to be in the 3NF, all the relations at first have to be 2NF form, and for the 3NF there should not be any transitive functional dependencies in the relations for the non-prime attributes. So there are not any functional dependencies such as non-prime \rightarrow non-prime. So this database design is already in the 3NF normalized form.

Normalization to BCNF

For the relation to be in BCNF form, iff all the functional dependencies have a super key as its left-hand side, no non-prime attribute can be on the left-hand side. In our database, all the functional dependencies have a primary key(i.e prime attribute) as their left-hand side. Also, all the foreign keys included in all the relations are the primary keys of the referenced tables.

Relation/Schema	Update	Delete
Inventory	It'll not affect the relation	It'll not affect the relation
Administrator	It'll update the foreign keyed entries of the order_detail relation.	It'll delete the foreign keyed entries of the order_detail relation.
Center	It'll update the slot, employee, manager, and user relations.	It'll delete the foreign keyed entries of all the relations except users.
Slots	It'll update the users foreign keyed entries.	It'll set the null value to the foreign keyed entries of the users relation.
Manager	It'll update the foreign keyed entries of the order_detail relation.	It'll delete the foreign keyed entries of the order_detail relation.
Order_details	It'll not affect the relation	It'll not affect the relation
Employee	It'll not affect the relation	It'll not affect the relation
Users	It'll not affect the relation	It'll not affect the relation

NOTE - Here there are no insertion anomalies in any of the relations.

Section 6

SQL

Final DDL Scripts:

• Inventory Table

```
CREATE TABLE Inventory
```

```
(  
    inventory_id int not null,  
    inventory_name varchar(100) not null,  
    inventory_city varchar(100) not null,  
    inventory_vaccineTypes varchar(200) not null,  
    inventory_stock Int not null default 0,  
    primary key (inventory_id),  
    check(inventory_stock>=0)
```

```
);
```

• Administrator Table

```
CREATE TABLE Administrator
```

```
(  
    admin_id int not null,  
    admin_name varchar(50) not null,  
    admin_gender varchar(6) not null,  
    admin_city varchar(30) not null,  
    primary key (admin_id),  
    check(admin_gender in ('Female','Male'))
```

```
);
```

• Center Table

```
CREATE TABLE Center
```

```
(  
    center_id int not null,  
    center_name varchar(50) not null,  
    center_address varchar(200) not null,  
    center_city varchar(30) not null,  
    center_vaccineStock int not null default 0,  
    primary key (center_id),  
    check(center_vaccineStock>=0)
```

```
);
```

• Slots Table

```
CREATE TABLE Slots
```

```
(  
    slot_id int not null,  
    slot_centerID int,  
    slot_startTime varchar not null default '00:00',  
    slot_endTime varchar not null default '00:00',  
    slot_numVaccines int not null default 0,
```

```
Slot_date varchar not null default '00.00.0000',
check(slot_numVaccines>=0),
primary key(slot_id),
foreign key(slot_centerID) references Center(center_id) on delete cascade on update cascade
);
```

● Manager Table

```
CREATE TABLE Manager
(
    manager_id int not null,
    manager_centerID int,
    manager_name varchar(50) not null,
    manager_gender varchar(6) not null,
    primary key(manager_id),
    foreign key(manager_centerID) references Center(center_id) on delete cascade on update cascade,
    check(manager_gender in ('Female','Male'))
);
```

● Order_details Table

```
CREATE TABLE Order_details
(
    order_id int not null,
    admin_id int,
    manager_id int,
    vaccineType varchar(200),
    numOf_vaccines int default 0,
    primary key(order_id),
    foreign key(admin_id) references administrator(admin_id) on delete cascade on update cascade,
    foreign key(manager_id) references manager(manager_id) on delete cascade on update cascade
);
```

● Employee Table

```
CREATE TABLE Employee
(
    employee_id int not null,
    employee_centerID int,
    employee_name varchar(50) not null,
    employee_gender varchar(6) not null,
    primary key(employee_id),
    foreign key(employee_centerID) references Center(center_id) on delete cascade on update cascade,
    check(employee_gender in ('Female','Male'))
);
```

- **Users Table**

```
CREATE TABLE Users
(
    user_id int not null,
    user_centerID int,
    user_slotID int,
    user_name varchar(50) not null,
    user_gender varchar(6) not null,
    user_age int,
    user_address varchar(200) not null,
    user_city varchar(30) not null,
    user_vaccine varchar(200) default null,
    user_isVaccinated char(1) default 'N',
    primary key (user_id),
    foreign key (user_centerID) references Center(center_id) on delete set null on update cascade,
    foreign key (user_slotID) references Slots(slot_id) on delete set null on update cascade,
    check(user_gender in ('Female','Male')),
    check(user_age>0),
    check (user_isVaccinated in ('Y','N','y','n'))
);
```

Final Table Schema

Inventory(Inventory id, inventory_name, inventory_city, inventory_vaccineTypes, inventory_stock)

Administrator(admin id, admin_name ,admin_gender, admin_city)

Center(center id, center_name, center_address, center_city, center_vaccineStock)

Slots(slot id, slot_centerID, slot_startTime, slotendTime, slot_numVaccines)

Manager(manager id, manager_centerID, manager_name, manager_gender)

Order_details(order id, admin_id, manager_id, vaccineType, numOf_vaccines)

Employee(employee id, employee_centerID, employee_name, employee_gender)

Users(user id, user_centerID, user_slotID, user_name, user_gender, user_age, user_address, user_city, user_vaccine, user_isVaccinated)

Creation of Tables

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

- Browser:** Shows the database structure with the following schema and table definitions:
 - Schemas:** SV_DB, mta, proj
 - Tables:** administrator, center, employee, inventory, manager, order_details, slots, users
- Query Editor:** Contains the SQL code for creating the `Inventory` and `Administrator` tables.
- Data Output:** Shows the message "Query returned successfully in 100 msec."

```

1 CREATE TABLE Inventory
2 (
3     inventory_id int not null,
4     inventory_name varchar(100) not null,
5     inventory_city varchar(100) not null,
6     inventory_vaccinatypes varchar(200) not null,
7     inventory_stock int not null default 0,
8     primary key (inventory_id),
9     check(inventory_stock>0)
10 );
11
12 CREATE TABLE Administrator
13 (
14     admin_id int not null,
15     admin_name varchar(50) not null,
16     admin_gender varchar(6) not null,
17     admin_city varchar(30) not null,
18     primary key (admin_id),
19     check(admin_gender in ('Female','Male'))
20 );
21
22
CREATE TABLE
23
Query returned successfully in 100 msec.
  
```

Inventory Data

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

- Browser:** Shows the database structure with the following schema and table definitions:
 - Tables:** administrator, center, employee, inventory, manager, order_details, slots, users
- Query Editor:** Contains the SQL code `select * from inventory`.
- Data Output:** Displays the data from the `Inventory` table.

Inventory_Id	Inventory_name	Inventory_city	Inventory_vaccinatypes
1	Druen-Weters	Sujnpur	ONDANSETRON
2	From: Johnston and Heatholtz	Alandi	Zulpidom Tartrate
3	Rico: Puccochi and Halsic	Indrapuri	LEVODIHYROXYNE SODIUM
4	Lurkin, Mayr and Boller	Dubneri	Silver Maple
5	Jakubowski, Hoeger und Oberholzer	Singki	ENDOCET
6	Cole, Cormier and Nikolaus	Renjapur	Olanzapine
7	Gislason-Bert	Namirra	Sulfacalazine
8	Lomiko and Sons	Mevor	Lorazepam
9	Renged-abadeh	Vadnamurai	Oxyacizine Hydrochloride
10	Reatty Inc.	Dibringalih	Image essentials hair regrowth treatment
11	Cratichchenk and Sons	Desil	SILVAN CHLORSYU
12	Kertezmann-Rechbar	Ajnatpur	Simply Numb Encore
13	Marquardt-Ritsche	Dhasal	Ortho-Klear Wht Capsules

Data Output: Shows the message "Successfully run. Total query runtime: 82 msec. 100 rows affected."

Center Data

The screenshot shows the pgAdmin 4 interface with the 'center' table selected in the left sidebar under the 'Tables' section. The main pane displays the table structure and data. The 'center' table has columns: center_id (PK Integer), center_name (character varying(50)), center_address (character varying(200)), center_city (character varying(30)), and center_vaccinestock (integer). The data shows 13 rows of center information, such as 'Wolf Group' at 77 Lexington Street and 'Gutmann Inc' at 4644 Red Cloud Avenue. A message at the bottom right indicates the query was successfully run.

center_id	center_name	center_address	center_city	center_vaccinestock
1	Wolf Group	77 Lexington Street	Jakarta	19
2	Sparce Crist	84207 School Drive	Smetho	878
3	Gutmann Inc	4644 Red Cloud Avenue	Porecbo	8
4	Cosin Hockt	41064 Milwaukee Plaza	Ostsbang	7
5	Bellinger, Meyer and Crutis	17414 Ferguson Alley	Pittsburgh	37
6	Gleeson and Sons	54219 River Crest Trail	Champira	564
7	O'Connor Design	4 River Crest Parkway	El Tonita	578
8	Hill, Huie and Britsch	1854 Ohio Pass	Vungtyring	496
9	Teller-Koll	27 Dennis Lane	Ilobivo	38
10	Schmitt and Sons	25970 Autumn Leaf Trace	Zelmarie	2
11	Zuluf, feet and Smith	9271 Nieder Trace	Marja	16
12	Sidde and Done	4856 Maguire Head	Masey	43
13	Dickson Biomar	639 Northfield Place	Bodolchabin	

Slots Data

The screenshot shows the pgAdmin 4 interface with the 'slots' table selected in the left sidebar under the 'Tables' section. The main pane displays the table structure and data. The 'slots' table has columns: slot_id (PK Integer), slot_centerid (integer), slot_starttime (character varying), slot_endtime (character varying), slot_unoccupines (integer), and slot_date (character varying). The data shows 18 rows of slot information, such as slot 1 from 07:00:00 to 07:00:00 on 2021-07-20. A message at the bottom right indicates the query was successfully run.

slot_id	slot_centerid	slot_starttime	slot_endtime	slot_unoccupines	slot_date
1	1	07:00:00	07:00:00	5	20.07.2021
2	2	08:02:01	08:01:01	10	06.09.2021
3	3	08:04:02	08:02:02	173	16.01.2021
4	4	08:06:03	08:03:03	953	07.07.2021
5	5	22:08:04	10:04	3	30.11.2020
6	6	09:10:05	12:05	73	17.11.2020
7	7	09:12:06	14:06	56493	24.11.2020
8	8	08:14:07	16:07	65	26.01.2021
9	9	05:16:08	19:08	31	07.07.2021
10	10	07:18:09	20:09	8	10.09.2021
11	11	24:20:10	22:10	3	16.10.2021
12	12	06:22:11	00:11	501	29.03.2021
13	13	04:00:12	02:12	81	04.12.2020
14	14	09:02:13	04:13	5973	16.05.2021
15	15	08:04:14	06:14	53	12.01.2021
16	16	03:06:15	08:15	49	08.10.2021
17	17	04:08:16	10:16	85	26.05.2021
18	18	05:10:17	15:17	21	06.07.2021

Order Details Data

The screenshot shows the PgAdmin interface with the following details:

- Browser:** Displays the database schema structure, including Schemas, Tables, and Views.
- Query Editor:** Contains the SQL query: `select * from order_details;`
- Data Output:** Shows the results of the query in a tabular format. The columns are: order_id, admin_id, manager_id, vaccine_type, and numof_vaccines.

order_id	admin_id	manager_id	vaccine_type	numof_vaccines
1	1	72	11 Unisene	27280
2	2	18	92 ENDOCET	1
3	3	62	64 Allergy	7151
4	4	66	87 Austrian Pine	156
5	5	68	100 Cetepime	936
6	6	34	57 Mosquilo	301
7	7	64	54 CAROLYN INSTANT IT AND SANI ZLUT WITH IALOL AND VITAMIN L	70506
8	8	66	35 Allergy/Itch Headache	45
9	9	57	60 Olanzapine	85904
10	10	100	71 NASOPEN	3
11	11	47	90 ALUSCINA CONSTITUTIA	250
12	12	63	82 Image essentials hair regrowth treatment	3
13	13	1	42 Natur Sunscreen Broad Spectrum SPF-30	732
14	14	18	2 Pain Relief	3

Employee Data

The screenshot shows the PgAdmin interface with the following details:

- Browser:** Displays the database schema structure, including Schemas, Tables, and Views.
- Query Editor:** Contains the SQL query: `select * from employee;`
- Data Output:** Shows the results of the query in a tabular format. The columns are: employee_id, employee_centerid, employee_name, and employee_gender.

employee_id	employee_centerid	employee_name	employee_gender
1	1	59 Malynda	Male
2	2	56 Sergeant	Female
3	3	47 Benni	Male
4	4	50 Illegarde	Female
5	5	32 Lotti	Female
6	6	29 Polcy	Male
7	7	34 Sheena	Male
8	8	59 Horn	Female
9	9	61 Star	Female
10	10	78 Kurtis	Male
11	11	49 Dav	Male
12	12	61 Balle	Female
13	13	11 Ed	Female
14	14	52 Walther	Male

Users Data

The screenshot shows the pgAdmin 4 interface with the 'users' table selected in the left sidebar under the 'Tables' section. The main pane displays the results of the query 'select * from users'.

Table Headers:

- user_id [PK] integer
- user_centerid integer
- user_slotid integer
- user_name character varying (50)
- user_gender character varying (5)
- user_age integer
- user_address character varying (200)
- user_city character varying (30)
- user_vaccine character varying

Table Data:

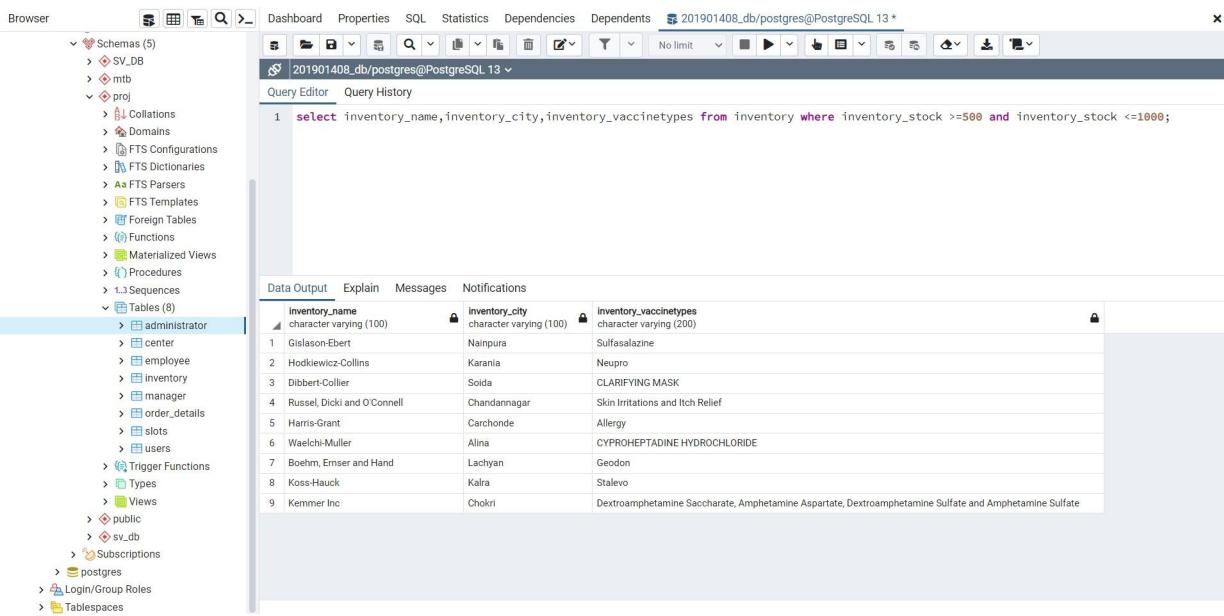
user_id	user_centerid	user_slotid	user_name	user_gender	user_age	user_address	user_city	user_vaccine
1	1	54	Lohare	Male	74	33, Charlie Society, Zeehan Chowk	Singpur	Penter Pan
2	2	25	Phillida	Male	43	82, Jayshee Nagar	Alandi	Hormodendru
3	3	21	65	Christoforo	Male	62	15, Anusala Nagar	Indrapuri
4	4	56	37	Atlante	Male	56	72, Juhl Apartments, MotiGanj	Dabheri
5	5	91	71	Lucius	Male	57	55, Kumbh Pur	Singla
6	6	96	80	Ardibald	Female	74	88, Surya Society, Sudalu	Banjapur
7	7	40	95	Catlee	Female	26	77, Raj n Vilas, Kormangala	Nanpara
8	8	98	15	Amubal	Female	69	37, Richa Apartments, Aundh	Mewar
9	9	85	42	Flynn	Male	40	61, Radh Hagni, Subbarani Chowk	SESAI LIP B
10	10	18	50	Daniel	Female	20	81, Ronval	Dhoniwadi
11	11	2	34	Allanora	Female	23	54, Alex Apartments, Anchur	Desil
12	12	22	5	Johl	Female	32	10, Chhati Chowk,	Almetaur
13	13	77	1	Addul	Male	46	✓ Successfully run. Total query runtime: 61 msec. 100 rows affected.	

SQL Queries

1. List all the Inventories which have vaccine stock in between 500 and 1000.

SQL Query :

```
select inventory_name,inventory_city,inventory_vaccinatypes from inventory where inventory_stock >=500 and inventory_stock <=1000;
```



The screenshot shows the pgAdmin 4 interface. On the left is a tree view of database objects under 'Browser'. In the center is the 'Query Editor' tab where the SQL query is typed. Below the editor is a results grid showing the output of the query. The results grid has columns: 'inventory_name', 'inventory_city', and 'inventory_vaccinatypes'. The data is as follows:

inventory_name	inventory_city	inventory_vaccinatypes
Gislason-Ebert	Nainpura	Sulfsalazine
Hodkiewicz-Collins	Karania	Neupro
Dibbert-Collier	Soida	CLARIFYING MASK
Russel, Dicky and O'Connell	Chandannagar	Skin Irritations and Itch Relief
Harris-Grant	Carchonde	Allergy
Waelchi-Muller	Alina	CYPROHEPTADINE HYDROCHLORIDE
Boehm, Emser and Hand	Lachyan	Geodon
Koss-Hauck	Kalra	Stalevo
Kemmer Inc	Chokri	Dextroamphetamine Saccharate, Amphetamine Aspartate, Dextroamphetamine Sulfate and Amphetamine Sulfate

2. List all the inventories which have the same vaccine Types in stock.

SQL Query :

```
select * from inventory where inventory_vaccinatypes in (select inventory_vaccinatypes  
from inventory group by inventory_vaccinatypes having count(inventory_vaccinatypes) > 1)
```

The screenshot shows the pgAdmin 4 interface. On the left is a tree view of database objects under 'Browser'. In the center is the 'Query Editor' tab with the following SQL code:

```
1 select * from inventory where inventory_vaccinatypes in (select inventory_vaccinatypes  
2 from inventory  
3 group by inventory_vaccinatypes  
4 having count(inventory_vaccinatypes) > 1)  
5
```

Below the code is a 'Data Output' table with the following data:

Inventory_Id	Inventory_name	Inventory_city	Inventory_vaccinatypes	Inventory
1	6 Cole, Cormier and Nikolaus	Ranjapur	Olanzapine	integer
2	97 Shanahan-Leffler	Verdi	Olanzapine	

A green success message at the bottom right says: "Successfully run. Total query runtime: 59 msec. 2 rows affected."

3. Provide the inventory details of the inventory which is located in Ghogha City.

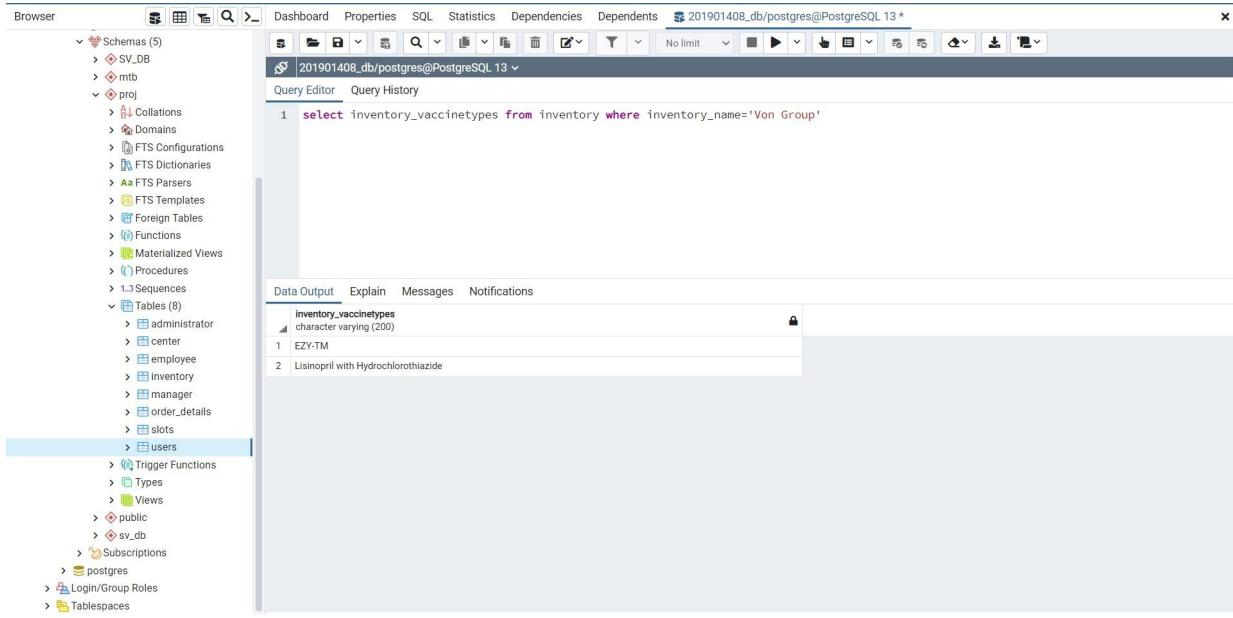
SQL Query = select * from inventory where inventory_city = 'Ghogha'

The screenshot shows the pgAdmin 4 interface. On the left is the Browser pane, which lists various database objects like Schemas, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, and Tables. The 'Tables' node is expanded, showing tables such as administrator, center, employee, inventory, manager, order_details, slots, and users. The 'users' table is currently selected, indicated by a blue selection bar. In the center is the Query Editor pane, containing the SQL query: 'select * from inventory where inventory_city = 'Ghogha''. Below the query is the Data Output pane, which displays the results of the query:

	inventory_id	inventory_name	inventory_city	inventory_vaccinatypes	inventory
1	14	Larkin-Rutherford	Ghogha	Aprodine	Integer

4. List all the vaccine types which are available at Von group inventory.

SQL Query = select inventory_vaccinatypes from inventory where inventory_name='Von Group'



The screenshot shows the pgAdmin 4 interface. On the left is a tree view of database objects under 'Browser'. In the center is the 'Query Editor' window with the query:

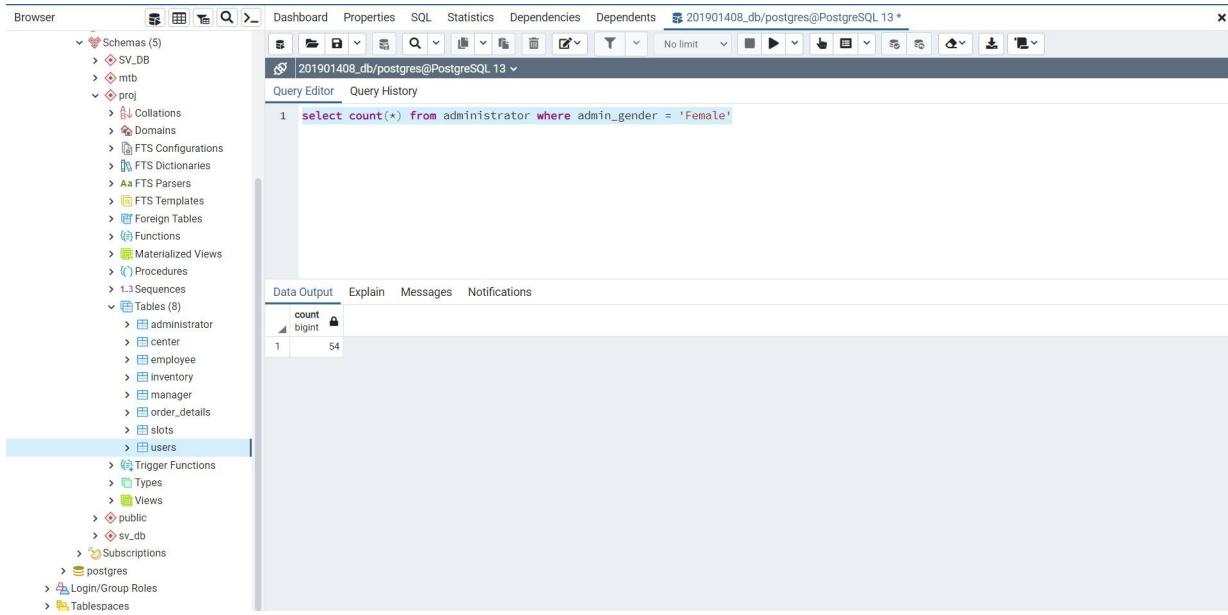
```
1 select inventory_vaccinatypes from inventory where inventory_name='Von Group'
```

Below the query editor is the 'Data Output' tab, which displays the results:

inventory_vaccinatypes
EZY-TM
Lisinopril with Hydrochlorothiazide

5. Give the count of the male administrator.

SQL Query = select count(*) from administrator where admin_gender = 'Male'



The screenshot shows the pgAdmin 4 interface. On the left is a tree view of database objects under the schema 'proj'. In the center is the 'Query Editor' window containing the SQL query:

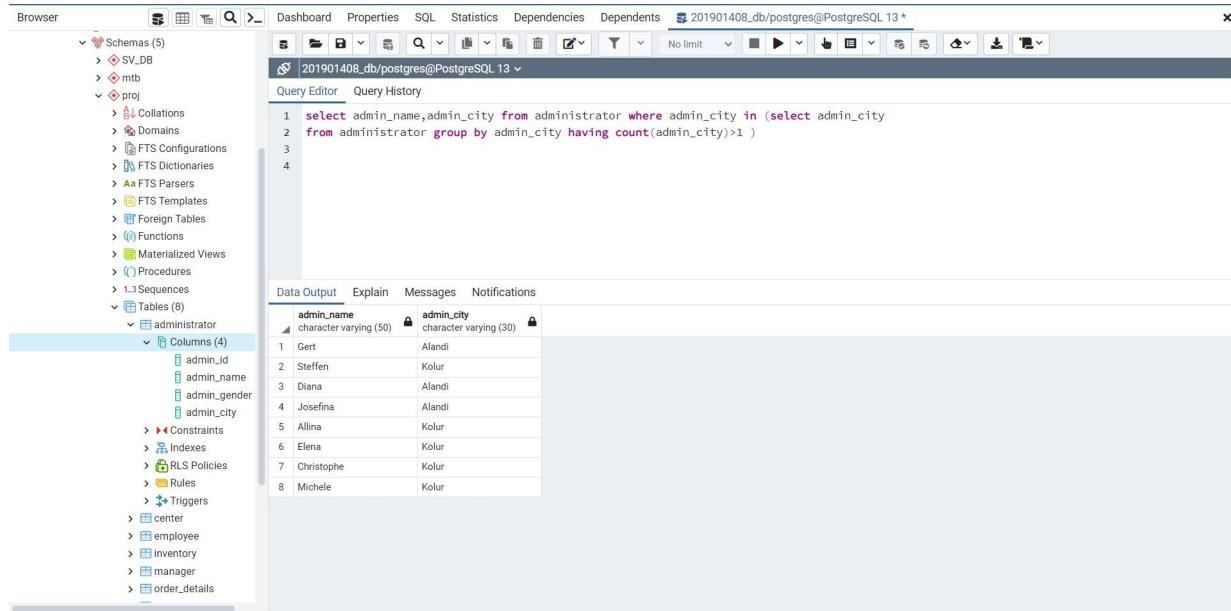
```
1 select count(*) from administrator where admin_gender = 'Male'
```

Below the query editor is the 'Data Output' tab, which displays the result of the query:

count
54

6. List all the admin cities having at least two administrators.

SQL Query - select admin_name,admin_city from administrator where admin_city in (select admin_city from administrator group by admin_city having count(admin_city)>1)



The screenshot shows the pgAdmin 4 interface. On the left is the Browser pane, which lists various database objects like Schemas, Tables, and Columns. The central area is the Query Editor, containing the following SQL code:

```
1 select admin_name,admin_city from administrator where admin_city in (select admin_city
2      from administrator group by admin_city having count(admin_city)>1 )
3
4
```

Below the query editor is the Data Output pane, which displays the results of the query:

admin_name	admin_city
Gert	Alandi
Steffen	Kolur
Diana	Alandi
Josefina	Alandi
Allina	Kolur
Elena	Kolur
Christophe	Kolur
Michele	Kolur

7. Provide the administrator names who have ordered some vaccines.

SQL Query - select admin_name, order_id

from administrator join order_details on administrator.admin_id = order_details.order_id

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects, including Schemas, proj, and Tables (8), with 'administrator' selected. The 'Tables' section lists center, employee, inventory, manager, order_details, slots, and users. On the right, the 'Query Editor' tab is active, showing the following SQL query:

```
1 select admin_name, order_id
2 from administrator join order_details on administrator.admin_id = order_details.order_id
3
4
5
```

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

	admin_name	order_id
1	Tamiko	1
2	Gert	2
3	Dexter	3
4	Tracie	4
5	Bartel	5
6	Millie	6
7	Eula	7
8	Emmy	8
9	Angelo	9
10	Erena	10
11	Charmon	11
12	Dario	13
13	Jerrone	14
14	Allie	15

8. Provide the administrator id and administrator name in ascending order of their names.

SQL Query - select admin_id,admin_name from administrator order by admin_name

The screenshot shows the pgAdmin 4 interface. On the left, the Browser pane displays the database schema, including Schemas, SV_DB, mtb, proj, and Tables. The Tables section is expanded, showing tables like center, employee, inventory, manager, order_details, slots, users, Trigger Functions, Types, Views, public, sv_db, postres, and Logins/Group Roles. The Tablespace section is also visible. The central area shows the Query Editor with the following SQL query:

```
1 select admin_id,admin_name from administrator order by admin_name
```

The Data Output tab shows the results of the query:

admin_id	admin_name
1	37 Adah
2	34 Agatha
3	45 Agnese
4	46 Aldis
5	84 Aleece
6	24 Alic
7	15 Allie
8	16 Allina
9	40 Andre
10	85 Ange
11	35 Angelita
12	98 Angelita
13	9 Angelo
14	78 Aprilette

9. Find out which manager has requested the order of which vaccine and the quantity of the vaccine.

SQL Query = select manager_id, manager_name, order_details.vaccinetype from order_details natural join manager

The screenshot shows a PostgreSQL query editor interface. The title bar reads "201901419_SRS/postgres@PostgreSQL 14". Below the title bar, there are tabs for "Query Editor" and "Query History", with "Query Editor" being the active tab. A single line of SQL code is present in the editor area:

```
1 select manager_id, manager_name, order_details.vaccinetype from order_details natural join manager
```

Below the editor, there are four tabs: "Data Output", "Explain", "Messages", and "Notifications". The "Data Output" tab is selected and displays a table of results. The table has three columns: "manager_id", "manager_name", and "vaccinetype". The data is as follows:

manager_id	manager_name	vaccinetype
83	Candice	Stannous Fluoride
84	Minetta	GIM
85	Bernetta	Sodium Citrate Blood-Pack Units, (PL 146 Plastic)
86	Frederik	ephrine nose
87	Teresa	BABY TINY COLD
88	Beatrisa	Olanzapine
89	Lyman	Healthy Accents hemorrhoidal
90	Nannie	CYPROHEPTADINE HYDROCHLORIDE
91	Elmer	berkley and jensen migraine formula
92	Pierre	Lamotrigine
93	Shirl	Xodol
94	Mickey	Primlev
95	Gabe	Diclofenac Sodium
96	Minetta	Givenchy TEINT COUTURE Long Wearing Fluid Foundation with Sunscreen Broad Spectrum SPF 20 ELEGANT SHELL
97	Merrili	ALMOND
98	Gilbertina	Alendronate Sodium
99	Phillis	Geodon
100	Stefanie	LEVOTHYROXINE SODIUM

10. Find the centers which have female managers.

SQL Query - select center_id,center_name from center where center_id in
(select manager_centerID from manager where manager_gender='Female')

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the session is 201901419_SRS/postgres@PostgreSQL 14. The main area contains the following SQL code:

```
1 select center_id,center_name from center where center_id in
2 (select manager_centerID from manager where manager_gender='Female')
```

Below the code, there are tabs for Data Output, Explain, Messages, and Notifications. The Data Output tab is selected, displaying a table with two columns: center_id and center_name. The data consists of 37 rows, each containing a center ID and its name. The table structure is as follows:

	center_id [PK] integer	center_name character varying (50)
21	61	Upton and Sons
22	63	Lindgren Group
23	66	Gaylord-Sporer
24	69	Kreiger and Sons
25	70	Graham-Spinka
26	72	Towne, Wunsch and Nikolaus
27	74	Deckow and Sons
28	83	Gutkowsky and Sons
29	85	Okuneva LLC
30	86	McLaughlin-Treutel
31	87	Bradtke, Bartell and Kreiger
32	92	Christiansen Inc
33	94	Dicki Group
34	95	Boyer Group
35	96	Powlowski-Rippin
36	97	Mertz-Will
37	100	Hegmann, Streich and Lebsack

11. Count the respective number of employees to the respective centers.

SQL Query = select center_id,center_name,count(employee_id) as Employees from center join employee on center_id=employee_centerid group by center_id

Query Editor Query History

```
1  select center_id,center_name,count(employee_id) as Employees from center join employee on center_id=empl
```

Data Output Explain Messages Notifications

	center_id [PK] integer	center_name character varying(50)	employees bigint
1	87	Bradtke, Bartell and Kreiger	2
2	29	McKenzie-Hammes	1
3	71	Swaniawski, Willms and Huel	2
4	68	Murray, Ratke and Goldner	1
5	34	Braun, Thompson and Leffler	1
6	51	Wilkinson Group	1
7	96	Powlowski-Rippin	1
8	70	Graham-Spinka	1
9	52	Kihn, Cartwright and Waters	1
10	67	Price Group	1
11	63	Lindgren Group	1
12	90	Kautzer-Zieme	1
13	10	Schmidt and Sons	1
14	45	Emard, Waelchi and Howell	1
15	6	Gleason and Sons	1
16	84	Wolf Group	3
17	39	Ortiz Group	1
18	89	Hane LLC	1

12. Provide the details of the users along with their center name and slot start time with end time, date who have taken the vaccination.

SQL Query

```
create or replace view uc_details as
```

```
select user_slotid,user_name, user_gender,user_age,user_vaccine,center_name from users
join center on (users.user_id=center.center_id and (user_vaccine is not null))
```

```
Select user_name,user_gender,user_age,user_vaccine,center_name,slots.slot_date,slots.slot_startTime,
slots.slot_endTime
from uc_details join slots
on slots.slot_id=uc_details.user_slotid
```

201901419_SRS/postgres@PostgreSQL 14 ~

Query Editor Query History

```
1 create or replace view uc_details as
2 select user_slotid,user_name, user_gender,user_age,user_vaccine,center_name from users join center
3 on (users.user_id=center.center_id and (user_vaccine is not null))
4
5 select user_name,user_gender,user_age,user_vaccine,center_name,slots.slot_date,slots.slot_startTime,slots.slot_endTime
6 from uc_details join slots
7 on slots.slot_id=uc_details.user_slotid
```

Data Output Explain Messages Notifications

user_name	user_gender	user_age	user_vaccine	center_name	slot_date	slot_startTime
Jerri	Female	44	Sulfasalazine	Swaniawski, Willms and Huel	06.07.2021	10:29
Brandyn	Male	30	Geodon	Towne, Wunsch and Nikolaus	27.02.2021	11:05
Sherman	Female	53	Neupro	Deckow and Sons	06.09.2021	02:01
Marnie	Female	53	Austrian Pine	Donnelly LLC	31.12.2020	02:37
Chance	Male	71	Healthy Accents hemorrhoidal	Bogisch-MacGyver	01.01.2021	05:26
Gaylord	Female	34	Ibuprofen	Quigley-Anderson	04.09.2021	08:28
Husain	Female	78	Doxetaxel	Kovacek-Sauer	09.05.2021	18:45
Gerty	Female	21	Skin Irritations and Itch Relief	Pagac-Glover	12.05.2021	01:24
Liliane	Female	31	CLARIFYING MASK	Gurtowski and Sons	02.11.2021	00:48
Hi	Male	22	Allergy	McLaughlin-Treutel	14.11.2021	22:47
Teodorico	Male	53	smart sense ibuprofen	Bradthe Bartell and Kreiger	09.07.2021	05:02
Milt	Female	40	Yes To Cucumbers Natural Sunscreen SPF 30	Buckridge-Deckow	02.11.2021	22:23
Evelina	Female	47	ONDANSETRON	Hane LLC	22.08.2021	08:52
Ninnetta	Female	59	Givenchy TEINT COUTURE Long Wearing Fluid Foundation with Sunscreen Broad Spectrum SPF 20 ELEGANT SHELL	Dicki Group	02.11.2021	18:57
Zerk	Female	62	ESSENTIAL BODY RELAXING ESSENCE	Pawlowski-Rippin	08.10.2021	06:15
Shanie	Male	47	Suboxone	Denesik, Zieme and Wiegand	21.05.2021	22:35
Phillip	Female	41	Whole Wheat	Hegmann, Streich and Lebsack	20.10.2021	17:08

13. Count the center wise number of slots available.

SQL Query - select center_id,center_name,count(slots.slot_id) as Slots from center join slots on center.center_id=slots.slot_centerID group by center_id

The screenshot shows a PostgreSQL query editor interface. At the top, there's a header bar with a gear icon, the text '201901419_SRS/postgres@PostgreSQL 14', and tabs for 'Query Editor' and 'Query History'. Below the header is the SQL query:

```
1 select center_id,center_name,count(slots.slot_id) as Slots from center join slots
2 on center.center_id=slots.slot_centerID group by center_id
```

Below the query, there are four tabs: 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected and displays a table with the following data:

	center_id [PK] integer	center_name character varying (50)	slots bigint
29	99	Denesik, Zieme and Wiegand	2
30	88	Buckridge-Deckow	1
31	41	Jerde Inc	1
32	40	Ruecker Inc	1
33	46	Krajcik Group	2
34	32	Hackett-Huels	1
35	53	Grady, Schaefer and Heller	4
36	7	O'Connor Group	1
37	9	Heller-Will	1
38	100	Hegmann, Streich and Lebsack	1
39	48	Langosh Group	1
40	26	Koepf Group	2
41	85	Okuneva LLC	1
42	78	Bogisich-MacGyver	2
43	24	Hill-Cassin	1
44	19	Cole, Towne and Kshlerin	1
45	81	Stark, Farrell and Glover	4
46	61	Upton and Sons	4

14. Provide the center details which have slots between 8.00 AM to 12 PM.

SQL Query

```
select * from center where center_id in  
(select slots.slot_centerid  
from slots where (cast(SUBSTRING(slots.slot_startTime,1,2) AS int)>=8  
and cast(SUBSTRING(slots.slot_startTime,1,2) AS int)<12))
```

The screenshot shows a PostgreSQL query editor interface. At the top, there's a toolbar with icons for file operations and a dropdown menu showing '201901419_SRS/postgres@PostgreSQL 14'. Below the toolbar, tabs for 'Query Editor' and 'Query History' are visible. The main area contains the SQL query and its execution results.

SQL Query:

```
1 select * from center where center_id in  
2 (select slots.slot_centerid  
3 from slots where (cast(SUBSTRING(slots.slot_startTime,1,2) AS int)>=8  
4 and cast(SUBSTRING(slots.slot_startTime,1,2) AS int)<12))  
5  
6
```

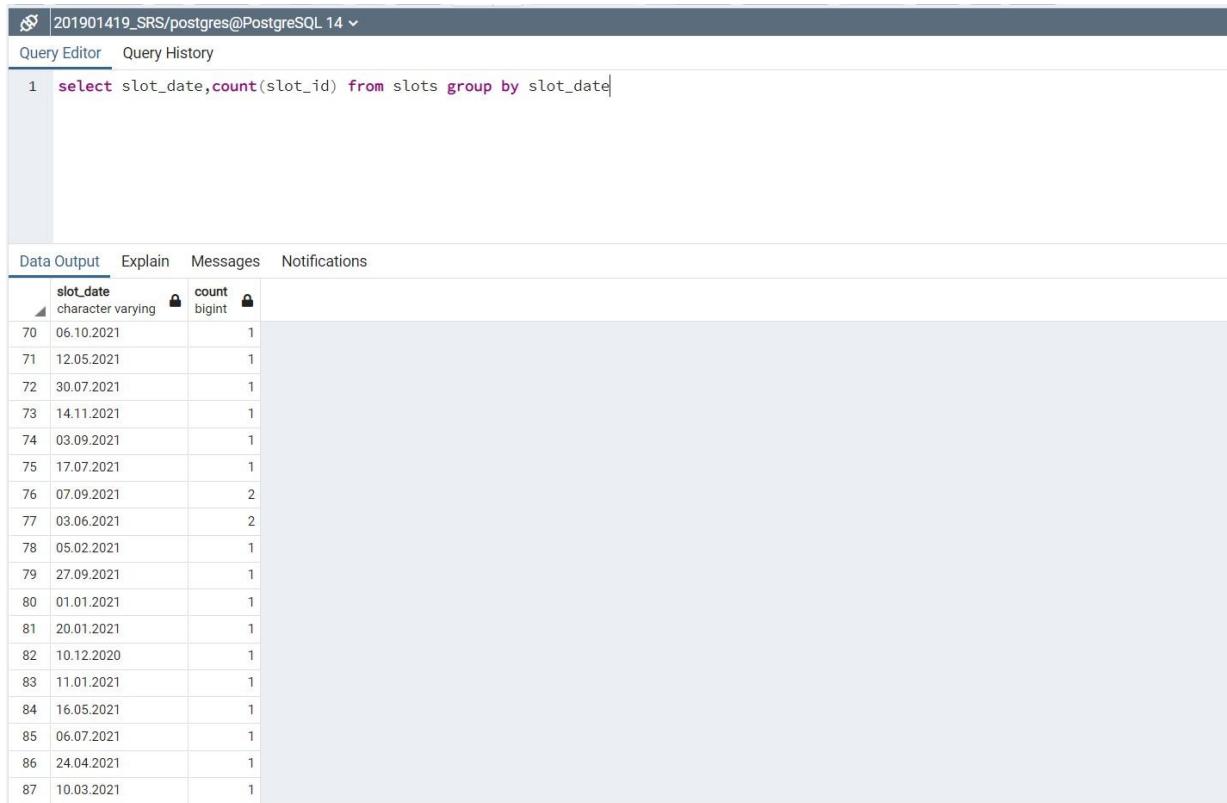
Data Output:

	center_id [PK] integer	center_name character varying (50)	center_address character varying (200)	center_city character varying (30)	center_vaccinestock integer
1	6	Gleason and Sons	42, Nagma Apartments, Chandpole	Ranapur	564
2	13	Dickens-Bernier	63, Aarushi Apartments, Marathahalli	Dhasal	842
3	19	Cole, Towne and Kshlerin	54, Eddie Society, Yeshwanthpura	Amarda	905
4	21	Ullrich-Rohan	73, Madhavi Heights, Model Town	Karen	550
5	22	Littel Group	68, Alex Apartments, Malad	Attungal	91
6	30	Hoeger Inc	55, Dhiraj Apartments, Andheri	Sindhia	7
7	39	Ortiz Group	33, Nayan Villas, RakhiGunj	Amola	7041
8	53	Grady, Schaefer and Heller	86, Deccan Gymkhana,	Banjaguri	68
9	61	Upton and Sons	64, Yeshwanthpura,	Chakardharpore	59928
10	84	Wolf Group	57, Heer Villas, JuliePur	Doli	9512
11	85	Okuneva LLC	71, Chhaya Villas, Chandpole	Jalia	690
12	92	Christiansen Inc	90, Dhiraj Villas, Malad	Ponnampet	83
13	96	Powlowski-Rippin	86, HimanshuPur,	Melwara	8600
14	98	Bruen, Steuber and Carroll	58, Bhaagyasree Apartments, Churchgate	Ustikolpenta	65

15. Count the date wise available slots.

SQL Query

```
select slot_date,count(slot_id) from slots group by slot_date
```



The screenshot shows a PostgreSQL query editor interface. The top bar displays the connection information: 201901419_SRS/postgres@PostgreSQL 14. Below the bar are tabs for 'Query Editor' and 'Query History'. The main area contains the SQL query:

```
1 select slot_date, count(slot_id) from slots group by slot_date
```

Below the query, there are four tabs: 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected and shows a table with two columns: 'slot_date' and 'count'. The data consists of 87 rows, each containing a date and a count of 1. The dates range from 06.10.2021 down to 10.03.2021.

	slot_date	count
70	06.10.2021	1
71	12.05.2021	1
72	30.07.2021	1
73	14.11.2021	1
74	03.09.2021	1
75	17.07.2021	1
76	07.09.2021	2
77	03.06.2021	2
78	05.02.2021	1
79	27.09.2021	1
80	01.01.2021	1
81	20.01.2021	1
82	10.12.2020	1
83	11.01.2021	1
84	16.05.2021	1
85	06.07.2021	1
86	24.04.2021	1
87	10.03.2021	1

16. Find the Male users who are vaccinated afternoon.

SQL Query

```
create or replace view afternoon_details as
```

```
select user_id from users join slots
```

```
on slots.slot_id=users.user_slotid
```

```
where (cast(SUBSTRING(slots.slot_startTime,1,2) AS int)>=12)
```

```
select * from users natural join afternoon_details
```

```
where users.user_gender='Male'
```

201901419_SRS/postgres@PostgreSQL 14 ▾

Query Editor Query History

```
1 create or replace view afternoon_details as
2 select user_id from users join slots
3 on slots.slot_id=users.user_slotid
4 where (cast(SUBSTRING(slots.slot_startTime,1,2) AS int)>=12 and (users.user_vaccine is not null))
5
6 select * from users natural join afternoon_details
7 where users.user_gender='Male'
```

Data Output Explain Messages Notifications

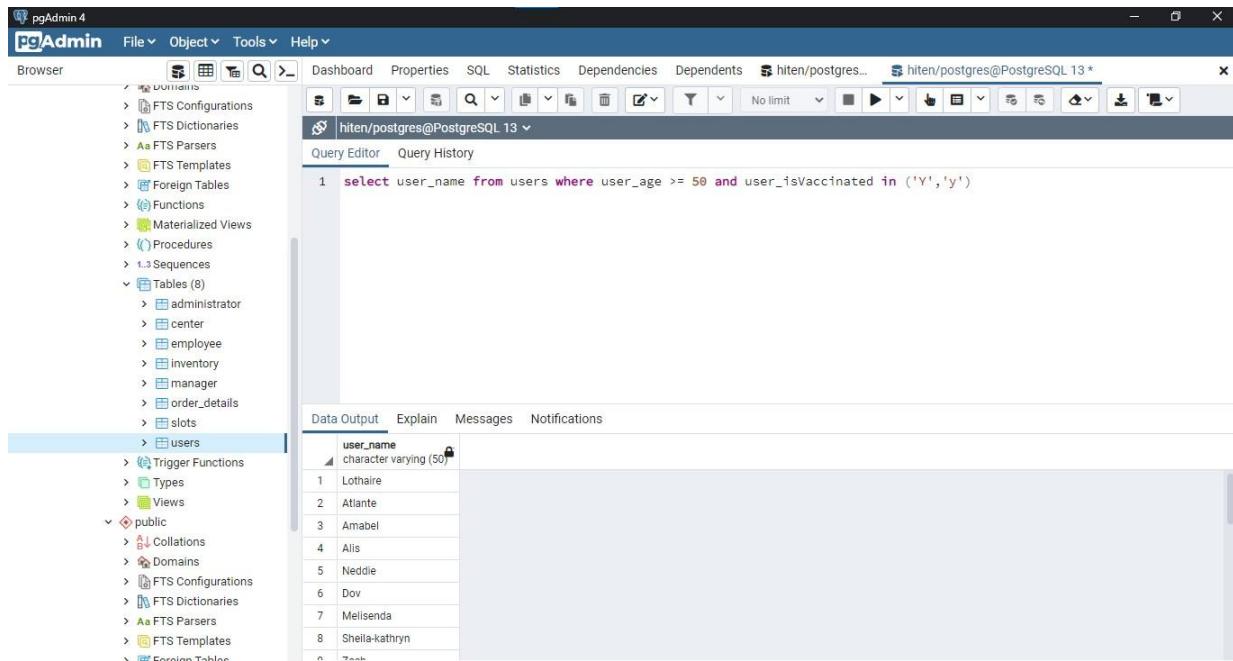
	user_id	user_centerid	user_slotid	user_name	user_gender	user_age	user_address	user_city	user_vaccine	use
1	1	54	23	Lothaire	Male	74	33, Charlie Society, Zeesha...	Sujnipur	Peter Pan	Y
2	2	25	32	Phillida	Male	43	82, Jayshree Nagar,	Alandi	Hormodendrum	Y
3	29	13	93	Claudie	Male	60	27, Kunti Heights, Aundh	Sumdah	Thyme	Y
4	31	78	57	Randie	Male	32	71, Basanti Heights, Harm...	Machora	Aprodine	Y
5	37	99	7	Bobbe	Male	66	28, Vikhroli,	Solda	NBE Citrus	Y
6	49	87	33	Charla	Male	32	58, FaisalPur,	Carchonde	Oxycodone Hydrochloride	Y
7	56	100	67	Candide	Male	43	96, Ram Apartments, Moni...	Bichaula	Ciprofloxacin	Y
8	63	90	68	Peggie	Male	23	13, Megha Society, Omar C...	Gosalpur	Zenchen	Y
9	68	17	60	Merle	Male	75	94, TrishanaPur,	Kurandi	Amlodipine and Benazepril...	Y
10	86	81	48	Hi	Male	22	33, John Apartments, Had...	Ingoitjala	Allergy	Y
11	99	2	36	Shanie	Male	47	19, Chitra Apartments, Aun...	Mysore	Suboxone	Y

✓ Successfully run. Total query runtime: 61 msec. 11 rows affected.

17. Find the people who are vaccinated and have an age above 50 years.

SQL Query

```
select user_name from users where user_age >= 50 and user_isVaccinated in ('Y','y')
```



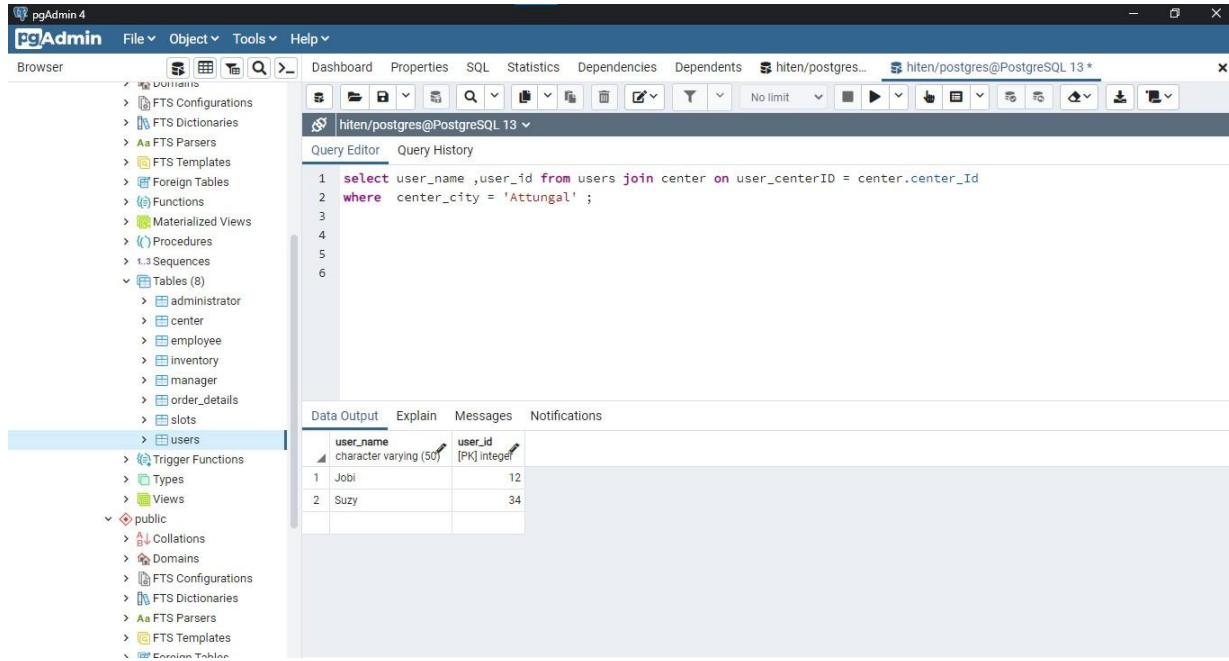
The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects, including Schemas, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (8), and a selected 'users' table. The main area contains a query editor with the SQL command: 'select user_name from users where user_age >= 50 and user_isVaccinated in ('Y','y')'. Below the query editor is a data output pane showing the results of the query:

user_name
Lothaire
Atalante
Amabel
Alis
Neddie
Dov
Melisenda
Sheila-kathryn
Zank

18.Find the user name and user id who have taken vaccines from ‘Attungal’.

SQL Query

```
select user_name ,user_id from users join center on user_centerID = center.center_Id  
where center_city = 'Attungal' ;
```



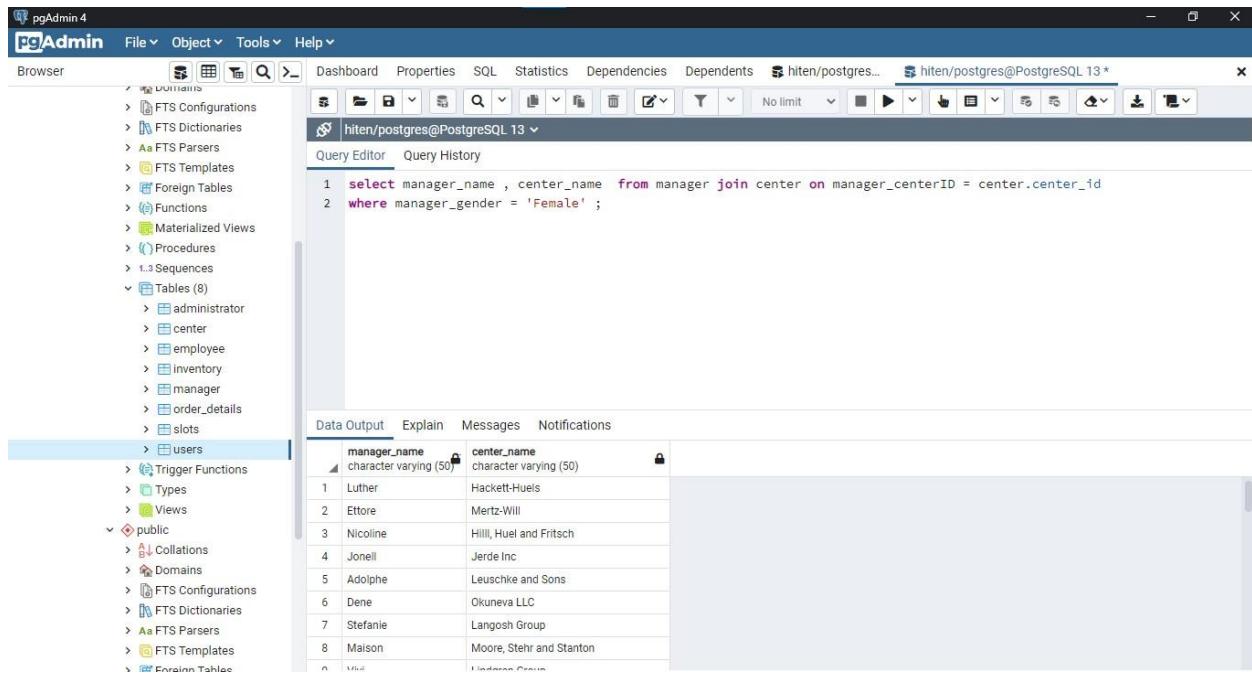
The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects like domains, FTS configurations, and tables. The 'Tables' section is expanded, showing tables such as administrator, center, employee, inventory, manager, order_details, slots, and users. The 'users' table is currently selected. The main area is the 'Query Editor' where the SQL query is typed. Below the editor is the 'Data Output' tab, which displays the results of the query. The results are presented in a table with two rows:

user_name	user_id
Jobi	12
Suzy	34

19.Find the manager name and center name where the Female manager controls the center.

SQL Query

```
select manager_name , center_name from manager join center on manager_centerID = center.center_id  
where manager_gender = 'Female' ;
```



The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects like Dictionaries, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Functions, Materialized Views, Procedures, Sequences, and Tables (with 8 entries). The 'users' table is currently selected. The main area is the 'Query Editor' pane, containing the SQL query:

```
1 select manager_name , center_name  from manager join center on manager_centerID = center.center_id  
2 where manager_gender = 'Female' ;
```

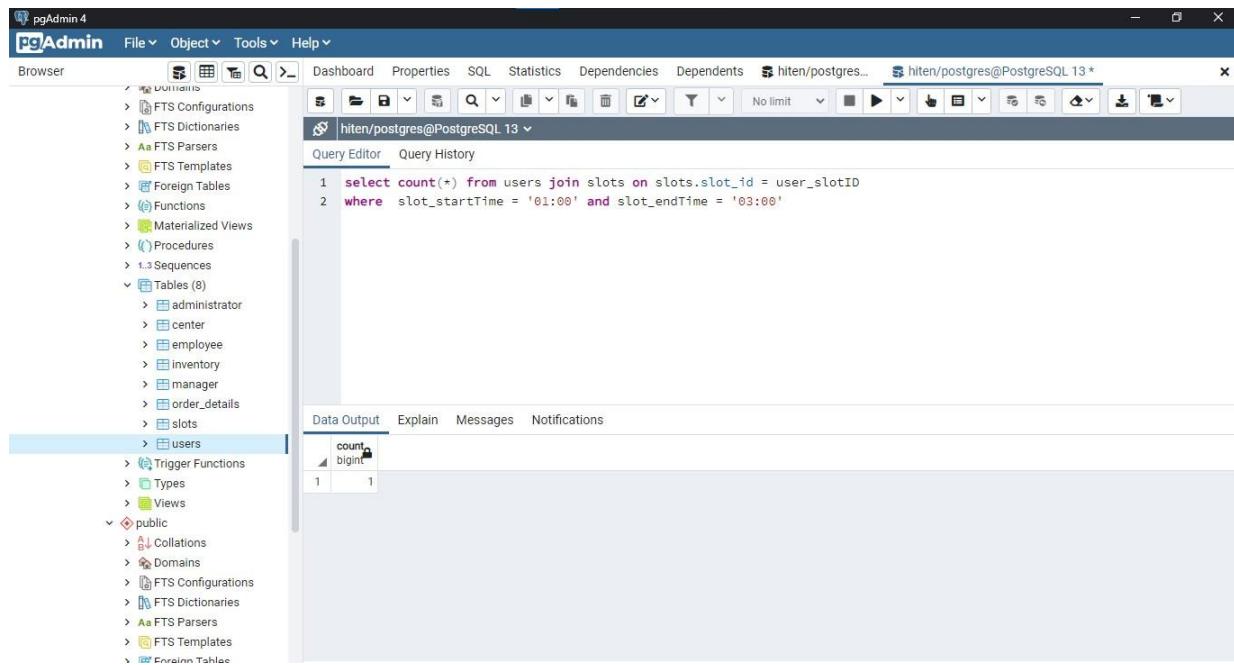
Below the query editor is the 'Data Output' tab, which displays the results of the query:

	manager_name	center_name
1	Luther	Hackett-Huels
2	Ettore	Mertz-Will
3	Nicoline	Hill, Huel and Fritsch
4	Jonell	Jerde Inc
5	Adolphe	Leuschke and Sons
6	Dene	Okuneva LLC
7	Stefanie	Langosh Group
8	Maison	Moore, Stehr and Stanton

20.Count the number of people who have taken vaccines between 1:00 to 3:00.

SQL Query

```
select count(*) from users join slots on slots.slot_id = user_slotID  
where slot_startTime = '01:00' and slot_endTime = '03:00'
```



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like administrator, center, employee, inventory, manager, order_details, slots, and users. The main area shows the SQL query in the Query Editor:

```
1 select count(*) from users join slots on slots.slot_id = user_slotID  
2 where slot_startTime = '01:00' and slot_endTime = '03:00'
```

The Data Output tab shows the result of the query:

count	bigint
1	1

21. Give the user details along with the employee from whom they get vaccinated.

SQL Query

```
select user_name , employee_name from users join employee on user_centerID = employee_centerID  
where user_isVaccinated in('Y','y')
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with the 'users' table selected. The main area contains a query editor window with the following SQL code:

```
1 select user_name , employee_name from users join employee on user_centerID = employee_centerID  
2 where user_isVaccinated in('Y','y')  
3
```

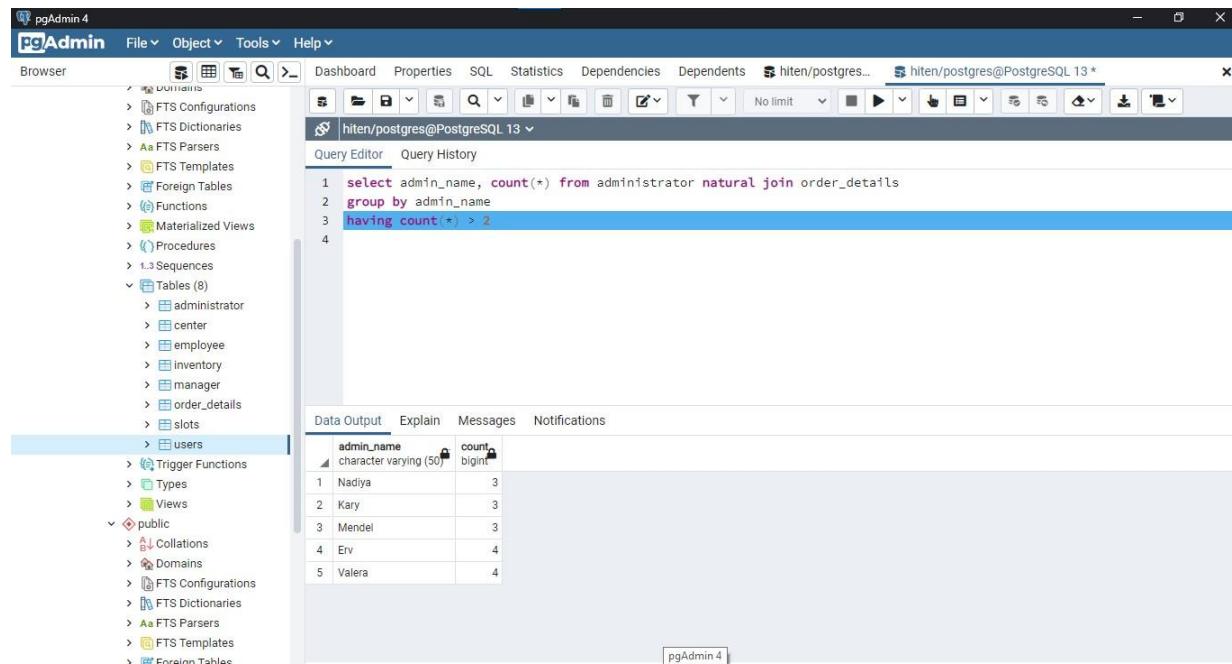
Below the query editor is a data output grid showing the results of the query:

	user_name	employee_name
1	Gennifer	Hildegarde
2	Jerri	Sheena
3	Nichol	Sheena
4	Melisenda	Sheena
5	Neddie	Fern
6	Cristionna	Star
7	Concettina	Star
8	Gerty	Dav
		S2_T9_Lab7b_Rel&DDL_Final_30-Oct-2021 - Google Docs

22. Find the name of the admin who has ordered vaccines more than two times.

SQL Query

```
select admin_name, count(*) from administrator natural join order_details  
group by admin_name  
having count(*) > 2
```



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

```
1 select admin_name, count(*) from administrator natural join order_details  
2 group by admin_name  
3 having count(*) > 2
```

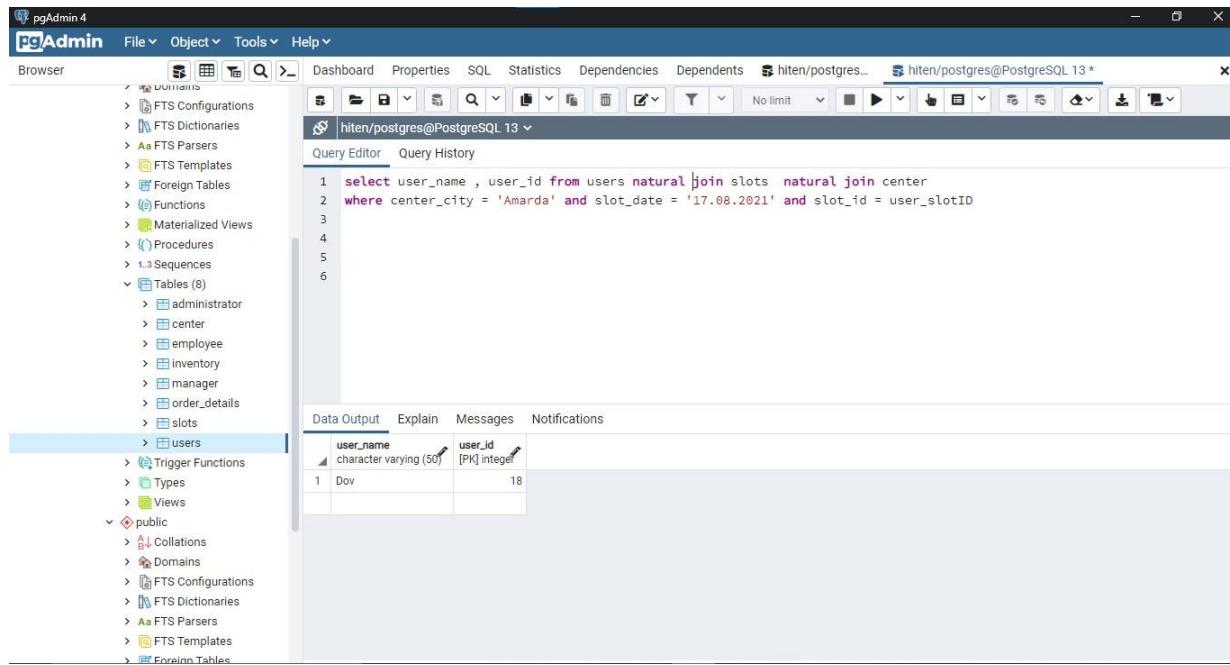
The results table shows the following data:

admin_name	count
Nadiya	3
Kary	3
Mendel	3
Erv	4
Valera	4

23. Find the details of all users who have taken vaccines at Amarda center on 17th august 2021.

SQL Query

```
select user_name , user_id from users natural join slots natural join center  
where center_city = 'Amarda' and slot_date = '17.08.2021' and slot_id = user_slotID
```



The screenshot shows the pgAdmin 4 interface. The left sidebar (Browser) displays database objects like FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (8), administrator, center, employee, inventory, manager, order_details, slots, and users. The 'users' table is currently selected. The top bar shows the connection information: hiten/postgres@PostgreSQL 13*. The main area is the Query Editor, which contains the following SQL code:

```
1 select user_name , user_id from users natural join slots natural join center  
2 where center_city = 'Amarda' and slot_date = '17.08.2021' and slot_id = user_slotID  
3  
4  
5  
6
```

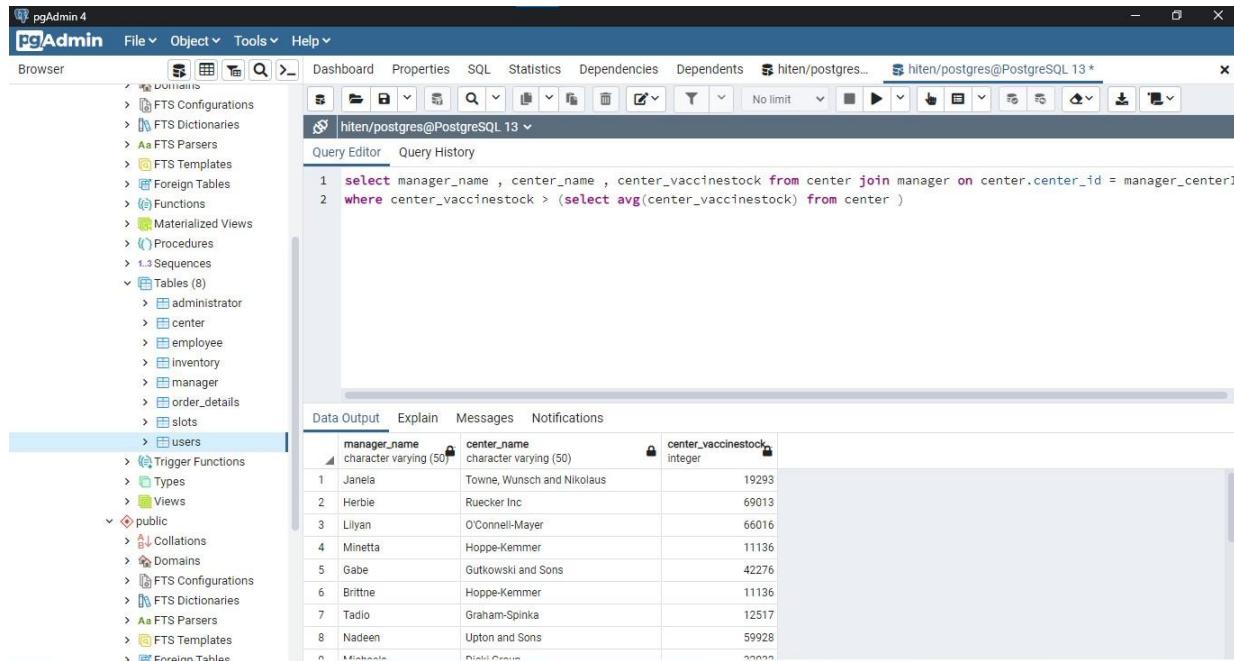
Below the Query Editor, there are tabs for Data Output, Explain, Messages, and Notifications. The Data Output tab shows the results of the query:

	user_name	user_id
1	Dov	18

24. Find the manager name and center name where the center's vaccine stock is greater than the average vaccine stock.

SQL Query

```
select manager_name , center_name , center_vaccinestock from center join manager on center.center_id = manager_centerID  
where center_vaccinestock > (select avg(center_vaccinestock) from center )
```



The screenshot shows the pgAdmin 4 interface with a query editor window displaying the results of the SQL query. The browser pane on the left lists various database objects like tables, functions, and types. The main window shows the query and its results.

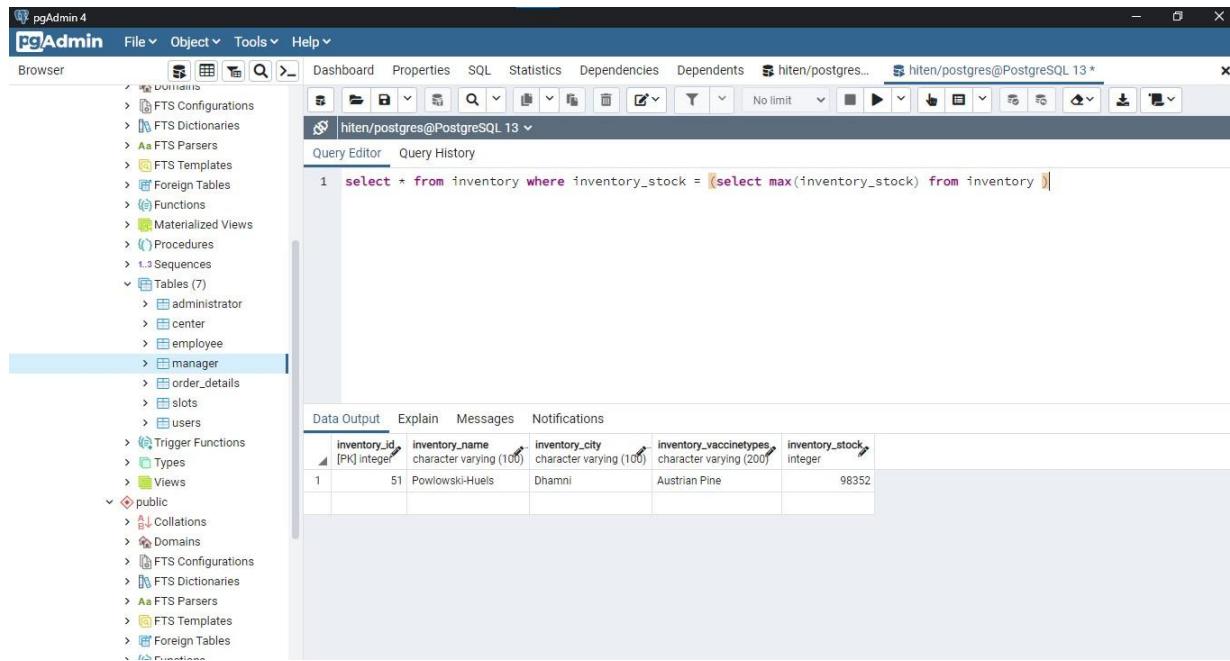
```
1 select manager_name , center_name , center_vaccinestock from center join manager on center.center_id = manager_centerID  
2 where center_vaccinestock > (select avg(center_vaccinestock) from center )
```

	manager_name	center_name	center_vaccinestock
1	Janelle	Towne, Wunsch and Nikolaus	19293
2	Herbie	Ruecker Inc	69013
3	Lilyan	O'Connell-Mayer	66016
4	Minetta	Hoppe-Kemmer	11136
5	Gabe	Gutkowski and Sons	42276
6	Brittni	Hoppe-Kemmer	11136
7	Tazio	Graham-Spinka	12517
8	Nadeen	Upton and Sons	59928
n	Mikaela	Gadel-Cronin	59928

25.Find the inventory details which have the maximum vaccine stock.

SQL Query

```
select * from inventory where inventory_stock = (select max(inventory_stock) from inventory )
```



The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects under the 'Browser' tab, including 'Schemas', 'Tables (7)', and 'public'. The 'manager' table is selected. The main area contains a 'Query Editor' tab with the following SQL query:

```
1 select * from inventory where inventory_stock = (select max(inventory_stock) from inventory )
```

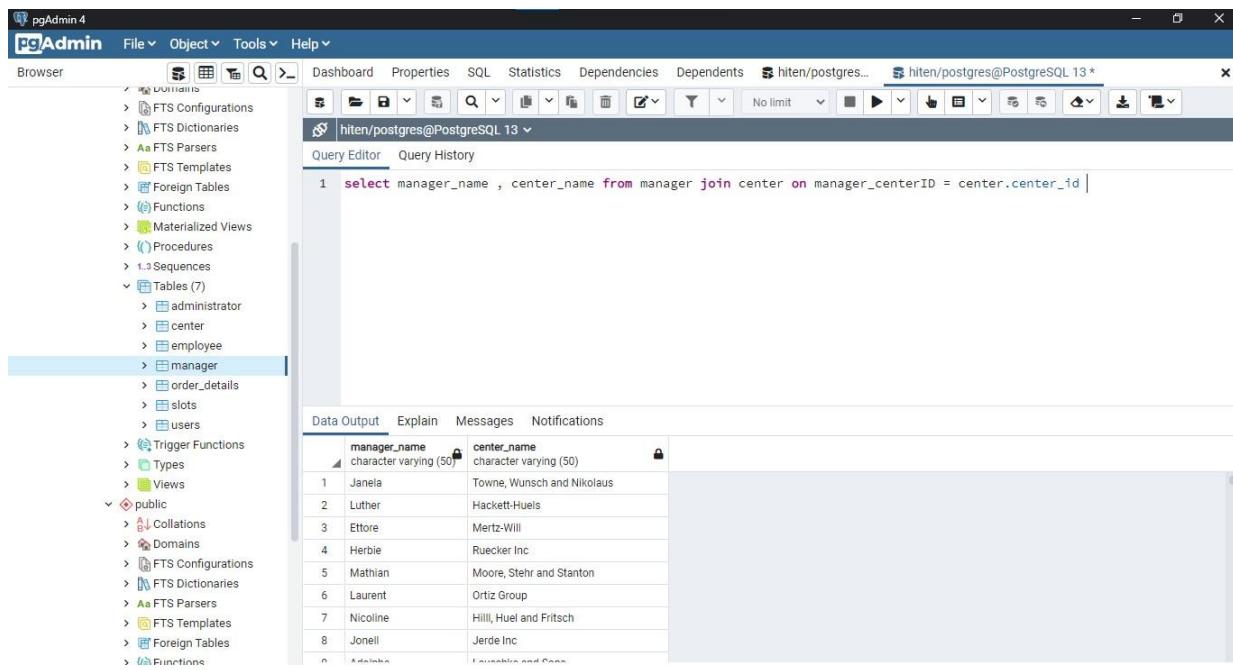
The results are displayed in a table titled 'Data Output' with the following data:

	inventory_id	inventory_name	inventory_city	inventory_vaccinatypes	inventory_stock
1	51	Powłowski-Huels	Dhamni	Austrian Pine	98352

26.List all the center name along with it's manager name.

SQL Query

```
select manager_name , center_name from manager join center on manager_centerID = center.center_id
```



The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects under the 'public' schema, including tables like administrator, center, employee, manager, and others. The 'manager' table is currently selected. On the right, the 'Query Editor' pane contains the SQL query: 'select manager_name , center_name from manager join center on manager_centerID = center.center_id'. Below the query, the 'Data Output' tab is active, showing the results of the query:

	manager_name	center_name
1	Janel	Towne, Wunsch and Niklaus
2	Luther	Hackett-Huels
3	Ettore	Mertz-Will
4	Herbie	Ruecker Inc
5	Mathian	Moore, Stehr and Stanton
6	Laurent	Ortiz Group
7	Nicoline	Hilll, Huel and Fritsch
8	Jonell	Jerde Inc
n	Audrina	Kronshke and Sons

27. Print the details of the slot which has the highest vaccine stock.

SQL Query

```
select * from slots where slot_numVaccines = ( select max(slot_numVaccines) from slots)
```

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects under the 'Tables (7)' section, with 'slots' selected. The main area is the 'Query Editor' window, which contains the following SQL query:

```
1 select * from slots where slot_numVaccines = ( select max(slot_numVaccines) from slots)
```

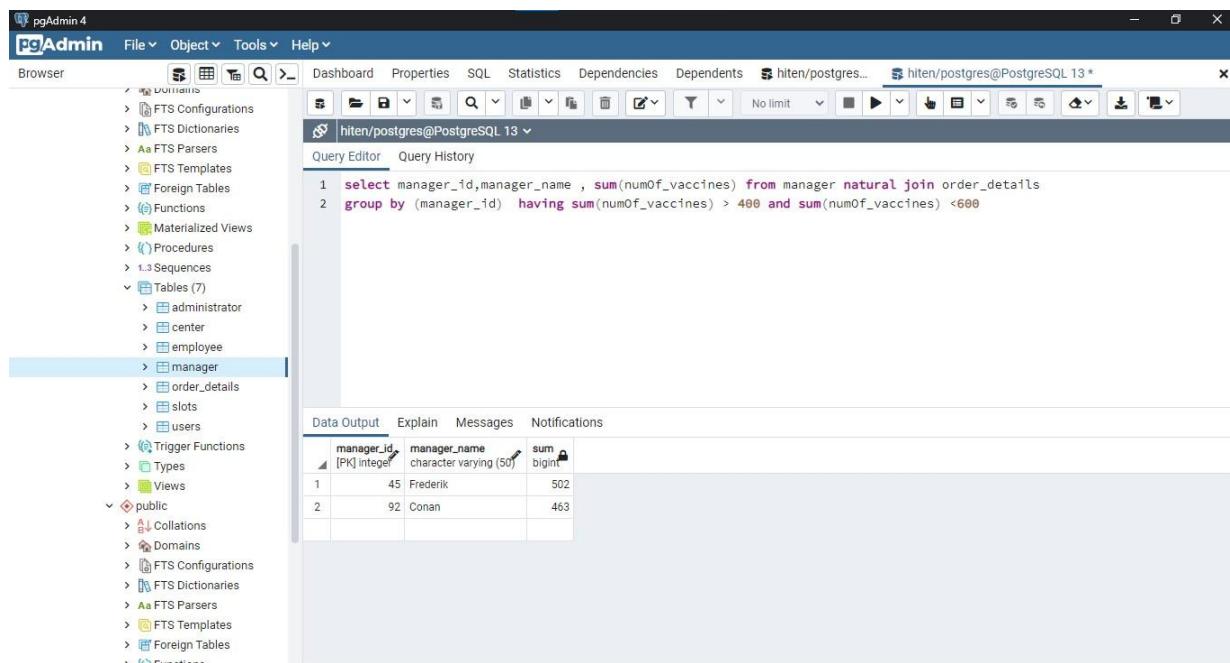
Below the query, the 'Data Output' tab is active, showing the results of the query:

slot_id	slot_centerid	slot_starttime	slot_endtime	slot_numvaccines	slot_date
1	42	85 10:41	12:41	98817	29.09.2021

28.Find the manager's name and who ordered the total number of vaccines between 400 and 600.

SQL Query

```
select manager_id,manager_name , sum(numOf_vaccines) from manager natural join order_details  
group by (manager_id) having sum(numOf_vaccines) > 400 and sum(numOf_vaccines) <600
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The browser pane on the left lists database objects: domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (7), administrator, center, employee, manager, order_details, slots, users, Trigger Functions, Types, Views, and public. The manager table is selected. The query editor contains the following SQL code:

```
1 select manager_id,manager_name , sum(numOf_vaccines) from manager natural join order_details  
2 group by (manager_id) having sum(numOf_vaccines) > 400 and sum(numOf_vaccines) <600
```

The results pane displays the output of the query:

	manager_id	manager_name	sum
1	45	Frederik	502
2	92	Conan	463

29. Find the total number of female employees.

SQL Query

```
select count(*) from employee where employee_gender = 'Female'
```

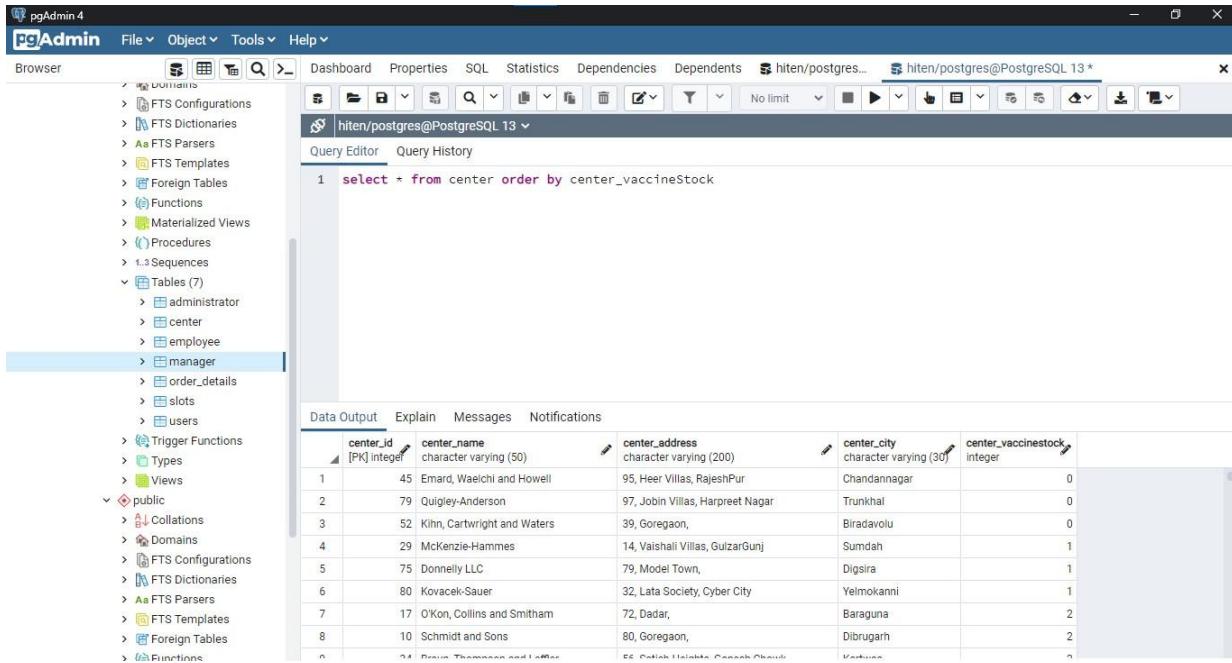
The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects: Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (7), and public. The 'Tables' node is expanded, showing administrator, center, employee, manager, order_details, slots, and users. The 'public' node is also expanded, showing Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, and Functions. The right pane is the 'Query Editor' with the title 'hiten/postgres@PostgreSQL 13 *'. It contains a single query: 'select count(*) from employee where employee_gender = 'Female''. Below the query, the 'Data Output' tab is selected, showing a result table with one row: 'count' (type: bigint) with value 51.

count	bigint
1	51

30.Sort the centers in ascending order of available vaccines stock.

SQL Query

```
select * from center order by center_vaccineStock
```



The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects like tables, functions, and types. The 'Tables' section is expanded, and the 'center' table is selected. The main area is the 'Query Editor' where the SQL query is entered:

```
1 select * from center order by center_vaccineStock
```

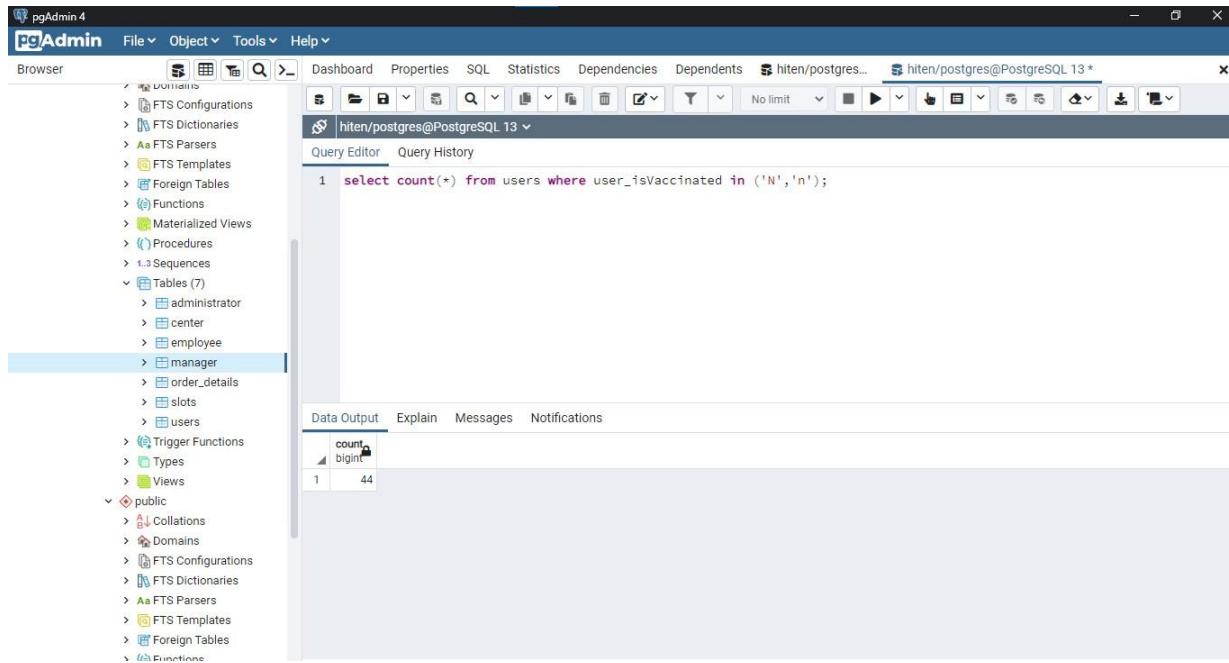
Below the query editor is the 'Data Output' tab, which displays the results of the query. The results are a table with the following data:

	center_id	center_name	center_address	center_city	center_vaccineStock
1	45	Emard, Waelchi and Howell	95, Heer Villas, RajeshPur	Chandannagar	0
2	79	Quigley-Anderson	97, Jobin Villas, Harpreet Nagar	Trunkhai	0
3	52	Kihn, Cartwright and Waters	39, Goregaon,	Biradavolu	0
4	29	McKenzie-Hammes	14, Vaishali Villas, GuzarGunj	Sumdah	1
5	75	Donnelly LLC	79, Model Town,	Digsira	1
6	80	Kovacek-Sauer	32, Lata Society, Cyber City	Yelmokanni	1
7	17	O'Kon, Collins and Smitham	72, Dadar,	Baraguna	2
8	10	Schmidt and Sons	80, Goregaon,	Dibrugrah	2
9	24	Orsi, Thompson and T. Allen	55, Carlton Heights, Cypress Charles	Vandikana	0

31.Count the total number of unvaccinated users.

SQL Query

```
select count(*) from users where user_isVaccinated in ('N','n');
```



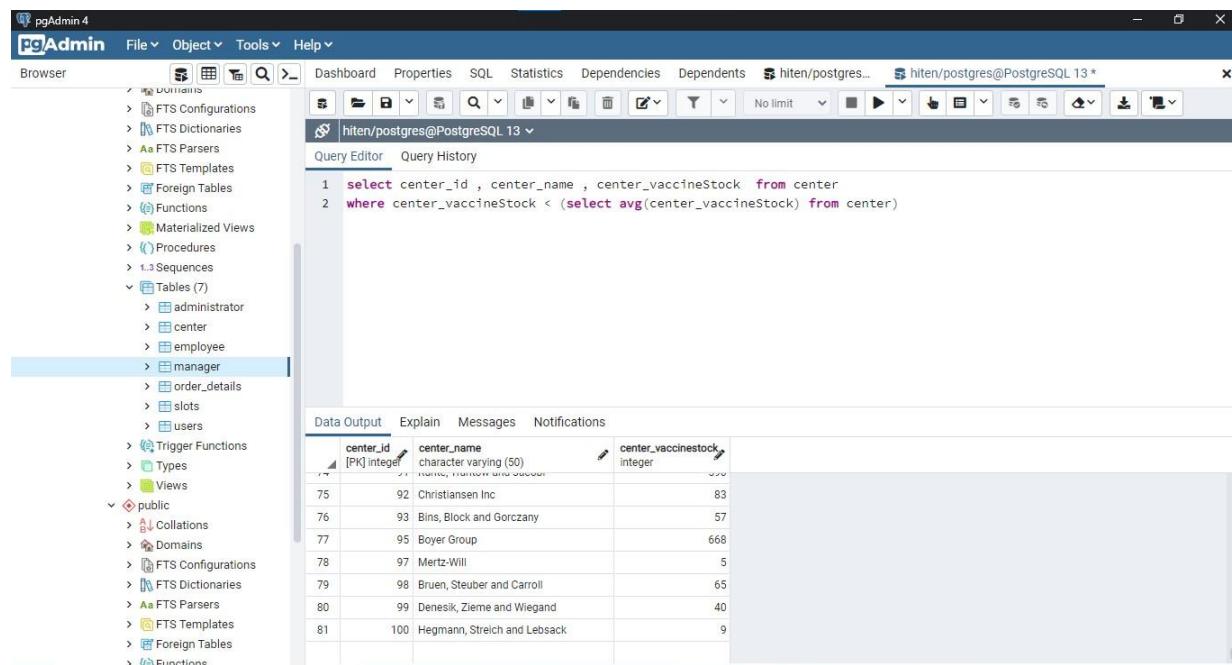
The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, which lists various database objects: Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (7), and Views. The 'Tables' node is expanded, showing administrator, center, employee, manager, order_details, slots, and users. The 'users' table is selected. The right pane is the 'Query Editor' with the query: `select count(*) from users where user_isVaccinated in ('N','n');`. Below the query, the 'Data Output' tab is active, showing a single row with 'count' and 'bigint' columns, both containing the value '44'. Other tabs include Explain, Messages, and Notifications.

count	bigint
1	44

32.List all the center name and center id which has vaccine stock less than all center's average vaccine stock.

SQL Query

```
select center_id , center_name , center_vaccineStock from center  
where center_vaccineStock < (select avg(center_vaccineStock) from center)
```



The screenshot shows the pgAdmin 4 interface with a query editor window. The browser pane on the left lists various database objects like FTS Configurations, FTS Dictionaries, and tables such as administrator, center, employee, manager, order_details, slots, and users. The central area contains the SQL query:

```
1 select center_id , center_name , center_vaccineStock from center  
2 where center_vaccineStock < (select avg(center_vaccineStock) from center)
```

The results pane displays the output of the query:

center_id	center_name	center_vaccineStock
75	Christiansen Inc	83
76	Bins, Block and Gorczany	57
77	Boyer Group	668
78	Mertz-Will	5
79	Bruen, Steuber and Carroll	65
80	Denesik, Zieme and Wiegand	40
81	Hegmann, Streich and Lebsack	9

33. Give the slotID of the user with user_id = 49

SQL query:

```
select user_slotID from users where user_id = 49
```

The screenshot shows the pgAdmin 4 interface. On the left is the Browser pane, which lists various database objects like Schemas, Tables, and Functions. The central area is the Query Editor, containing the SQL query: `select user_slotID from users where user_id = 49`. Below the query editor is the Data Output pane, which displays the result of the query: a single row with user_slotID set to 33. At the bottom right of the interface, there is a green success message: "Successfully run. Total query runtime: 57 msec. 1 rows affected."

user_slotID	integer
1	33

34. List the slots in the ascending order of their stock.

SQL query:

```
select * from slots order by slot_numvaccines
```

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

- Browser:** Shows the database structure:
 - Schemas (5): SV_DB, mtb, proj
 - proj: Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Table, Functions, Materialized Views, Procedures, Sequences, Tables (8): administrator, center, employee, inventory, manager, order_details, slots, users, Trigger Functions, Types, Views
 - sv_db: public, sv_db
 - postgres
 - Login/Group Roles
 - Tablespaces
- Query Editor:** Contains the SQL query: `select * from slots order by slot_numvaccines`
- Data Output:** A table showing the results of the query. The columns are: slot_id [PK] integer, slot_centerid integer, slot_starttime character varying, slot_endtime character varying, slot_numvaccines integer, slot_date character varying. The data is as follows:

slot_id	slot_centerid	slot_starttime	slot_endtime	slot_numvaccines	slot_date
1	76	93 07:15	09:15	0	07.09.2021
2	79	41 13:18	15:18	0	20.11.2020
3	37	58 00:36	02:36	1	12.06.2021
4	21	92 16:20	18:20	1	11.12.2020
5	48	81 22:47	00:47	1	14.11.2021
6	67	100 13:06	15:06	2	20.02.2021
7	11	24 20:10	22:10	3	15.10.2021
8	20	84 14:19	16:19	4	15.07.2021
9	34	2 18:33	20:33	4	11.01.2021
10	61	64 01:00	03:00	5	15.08.2021
11	1	77 00:00	02:00	5	20.07.2021
12	30	84 10:29	12:29	5	06.07.2021
13	53	53 08:52	10:52	6	22.08.2021
..	-	- - - - -

35. Give the user details along with the center name from where they get vaccinated

SQL query:

```
select user_id,user_name,center_id,center_name
from users join center on users.user_centerid = center.center_id
where users.user_isVaccinated = 'Y'
```

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects, including Schemas (5), pro (with Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences), and Tables (8) such as administrator, center, employee, inventory, manager, order_details, slots, and users. The 'Tables (8)' section is expanded. On the right, the 'Query Editor' pane shows the SQL query:

```
1 select user_id,user_name,employee_id,employee_name
2 from users join employee on users.user_centerid = employee.employee_centerID
3 where users.user_isVaccinated = 'Y'
```

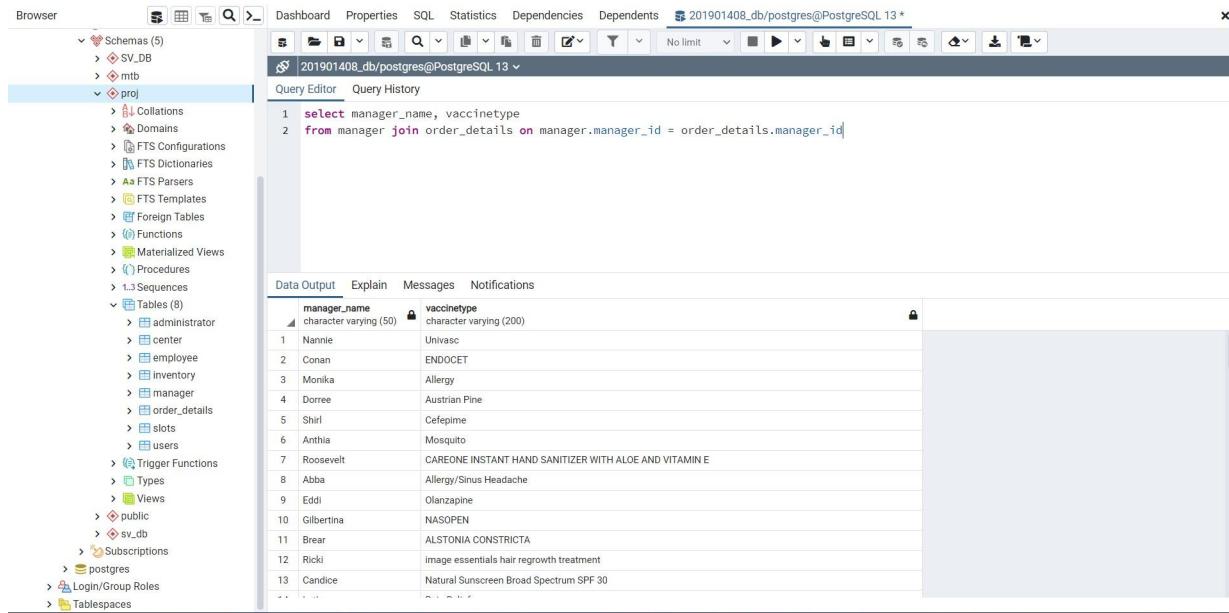
The 'Data Output' pane below the query editor displays the results of the query:

	user_id	user_name	employee_id	employee_name
1	52	Gennifer	4	Hildegarde
2	71	Jeri	7	Sheena
3	59	Nichol	7	Sheena
4	19	Melisenda	7	Sheena
5	17	Neddie	8	Fern
6	43	Cristionna	9	Star
7	41	Concettina	9	Star
8	82	Gerty	11	Dav
9	67	Gall	12	Belle
10	69	Allx	18	Hastings
11	71	Jeri	21	Raddy
12	59	Nichol	21	Raddy
13	19	Melisenda	21	Raddy
...

36. Give the manager's name along with the vaccine types they ordered.

SQL query:

```
select manager_name, vaccinetype  
from manager join order_details on manager.manager_id = order_details.manager_id
```



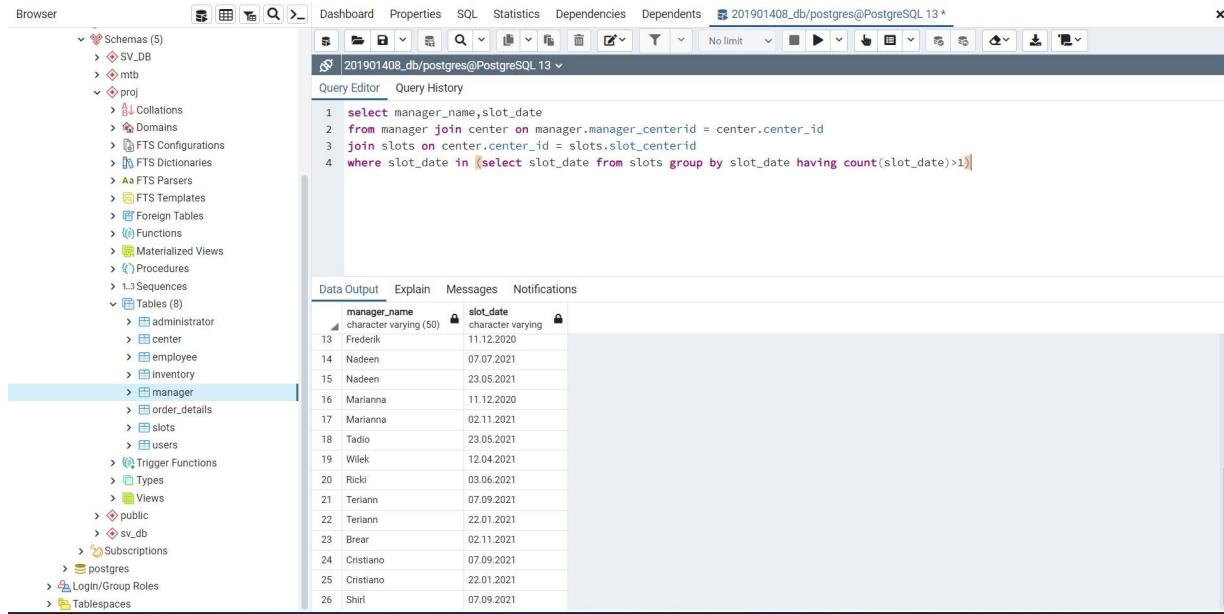
The screenshot shows the pgAdmin 4 interface. On the left is the Object Browser tree, which includes Schemas (5), Tables (8), and other database objects. The main area is the Query Editor, displaying the SQL query and its results. The results table has columns: manager_name and vaccinetype. The data is as follows:

manager_name	vaccinetype
Nannie	Univasc
Conan	ENDOCET
Monika	Allergy
Dorree	Austrian Pine
Shirl	Cefepime
Anthia	Mosquito
Roosevelt	CAREONE INSTANT HAND SANITIZER WITH ALOE AND VITAMIN E
Abba	Allergy/Sinus Headache
Eddi	Olanzapine
Gilbertina	NASOPEN
Brear	ALSTONIA CONSTRICTA
Ricki	image essentials hair regrowth treatment
Candice	Natural Sunscreen Broad Spectrum SPF 30

37. Give information about the manager who worked on the same day.

SQL query:

```
select manager_name,slot_date
from manager join center on manager.manager_centerid = center.center_id
join slots on center.center_id = slots.slot_centerid
where slot_date in (select slot_date from slots group by slot_date having count(slot_date)>1)
```



The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays the database schema, including tables like 'manager', 'center', 'slots', and 'order_details'. The 'Query Editor' pane contains the SQL query provided above. The 'Data Output' pane shows the results of the query, which lists managers and their working days. The table has two columns: 'manager_name' and 'slot_date'.

manager_name	slot_date
Frederik	11.12.2020
Nadeen	07.07.2021
Nadeen	23.05.2021
Marianna	11.12.2020
Marianna	02.11.2021
Tadio	23.05.2021
Wilek	12.04.2021
Ricki	03.06.2021
Teriann	07.09.2021
Teriann	22.01.2021
Brear	02.11.2021
Cristiano	07.09.2021
Cristiano	22.01.2021
Shirl	07.09.2021

38. Give the total number of vaccines at the particular center along with the center name.

SQL query:

```
select center_id,center_name,sum(slot_numvaccines) as total_vaccine
from center join slots on center.center_id = slots.slot_centerid
group by center_id,center_name
```

The screenshot shows the pgAdmin 4 interface. On the left is the Browser pane, which lists various database objects like Schemas, Tables, and Views. The Tables section shows eight tables: administrator, center, employee, inventory, manager, order_details, slots, and users. The center table is currently selected. On the right is the Query Editor pane, which contains the SQL query provided above. Below the query is the Data Output pane, which displays the results of the query. The results are as follows:

	center_id	center_name	total_vaccine
1	87	Bradtke, Bartell and Kreiger	9317
2	54	Moore, Stehr and Stanton	74785
3	71	Swaniewski, Willms and Huel	3725
4	4	Cassin-Hackett	81
5	96	Powlowski-Rippin	89230
6	63	Lindgren Group	49
7	90	Kautzer-Zieme	343
8	35	Nader-Koepf	83818
9	45	Emard, Waelchi and Howell	124
10	6	Gleason and Sons	19244
11	84	Wolf Group	9974
12	20	Orsi, O'Conor, Hahn and Kiehn	17

39. Give the center address where the particular user was vaccinated.

SQL query:

```
select center_address, user_name  
from users join center on center.center_id = users.user_centerid  
where user_isvaccinated = 'Y'
```

The screenshot shows the pgAdmin 4 interface. On the left is the Browser pane, which displays the database schema with various objects like Schemas, Tables, and Views. The Tables section is expanded, showing eight tables: administrator, center, employee, inventory, manager, order_details, slots, and users. The center table is selected. On the right is the Query Editor pane, which contains the SQL query:

```
1 select center_address, user_name  
2 from users join center on center.center_id = users.user_centerid  
3 where user_isvaccinated = 'Y'
```

Below the query editor is the Data Output pane, which displays the results of the query. The results are as follows:

	center_address	user_name
46	52 Monica Place	Husain
47	3928 Surrey Park	Gerty
48	70 Donald Lane	Liliane
49	80563 Warner Junction	Hi
50	280 Grover Point	Teodorico
51	7 Oakridge Junction	Milt
52	4 Gulseth Lane	Evelina
53	60 Tony Junction	Ninetta
54	5 Helena Alley	Zerk
55	84207 School Drive	Shanie
56	06273 Doe Crossing Avenue	Phillip

40. Give the total number of vaccines ordered by a particular admin along with the admin name.

SQL query:

```
select administrator.admin_id,admin_name,sum(numOf_vaccines) as total_vaccine  
from administrator join order_details on administrator.admin_id = order_details.admin_id  
group by administrator.admin_id,admin_name
```

The screenshot shows the pgAdmin 4 interface. On the left is the 'Browser' panel displaying database schema information, including tables like administrator, center, employee, inventory, manager, order_details, slots, and users. The main area is the 'Query Editor' where the SQL query is typed. Below the editor is the 'Data Output' tab showing the results of the query. The results table has columns: admin_id, admin_name, and total_vaccine. The data is as follows:

admin_id	admin_name	total_vaccine
51	Ariella	66007
52	Tamiko	808
53	Lulita	3
54	Aron	141
55	Vasily	156
56	Aubine	25
57	Yolonda	65339
58	Sutherlan	58481
59	Gwendolen	382
60	Emmy	3

Functions:

1. Function returns all the necessary user details, for the provided user id.

Function

```
CREATE OR REPLACE function "proj"."user_details"(userID int)
```

```
RETURNS TABLE (user_id int, user_name character varying(50),user_vaccine character varying(200),
    user_isVaccinated character(1),center_name character varying(50),
    slot_date character varying,slot_starttime character varying, slot_endtime character varying)
```

```
LANGUAGE 'plpgsql'
```

```
AS $BODY$
```

```
BEGIN
```

```
RETURN                               QUERY          (select
users.user_id,users.user_name,users.user_vaccine,users.user_isvaccinated,center.center_name,slots.slot_date,slots.slot_st
rtTime,
```

```
            slots.slot_endTime
```

```
from users join center on users.user_centerID = center.center_id
```

```
join slots on slots.slot_id = users.user_slotID where users.user_id = userID);
```

```
END;
```

```
$BODY$;
```

```
select * from user_details(71);
```

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

- Browser:** On the left, the database structure is shown under the 'proj' schema, including tables like 'administrator', 'center', 'employee', 'inventory', 'manager', 'order_details', 'slots', 'users', and views like 'Trigger Functions', 'Types', 'Views'.
- Query Editor:** The main area contains the SQL code for the function 'user_details'. The code includes a RETURN TABLE clause with columns for user_id, user_name, user_vaccine, user_isvaccinated, center_name, slot_date, slot_starttime, and slot_endtime. It also includes a SELECT statement that joins the users and center tables, and a JOIN clause that links users to slots based on user_id and user_slotID.
- Data Output:** Below the query editor, the results of the query are displayed in a table. The table has columns: user_id, user_name, user_vaccine, user_isvaccinated, center_name, slot_date, slot_starttime, and slot_endtime. There is one row returned, with values: user_id=71, user_name='Jeri', user_vaccine='Sulfasalazine', user_isvaccinated='Y', center_name='Wolf Group', slot_date='06.07.2021', slot_starttime='10:29', and slot_endtime='12:29'.
- Notifications:** A green notification bar at the bottom right indicates: "Successfully run. Total query runtime: 55 msec. 1 rows affected."

2. Function for providing the employees details working under the given manager id.

Function

```
CREATE OR REPLACE function "proj"."employee_details"(managerID int)
RETURNS TABLE (employee_id int, employee_name character varying(50),employee_centerid int,employee_gender
character varying(6))
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
RETURN QUERY
(select employee.employee_id,employee.employee_name,employee.employee_centerid,employee.employee_gender
from manager join center on manager.manager_centerid = center.center_id
join employee on center.center_id = employee.employee_centerid
where manager_id = managerID);
END;
$BODY$;
```

```
select * from employee_details(4);
```

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

- Browser:** Shows the database structure:
 - SV.DB
 - mtb
 - proj
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions (1)
 - Materialized Views
 - Procedures
 - Sequences
 - Tables (8)
 - administrator
 - center
 - employee
 - inventory
 - manager
 - order_details
 - slots
 - users
 - Trigger Functions
 - Types
 - Views
 - public
 - sv_db
 - Subscriptions
 - postres
 - Login/Group Roles
 - Tablespaces
- Query Editor:** Displays the SQL code for the function creation.
- Data Output:** Shows the results of the query `select * from employee_details(4);`, which returns three rows of data:

employee_id	employee_name	employee_centerid	employee_gender
70	Krisy	40	Female
74	Karry	40	Male
84	Alikee	40	Female

Triggers:

1. Trigger function and trigger call for inserting new tuple in inventory.

Trigger Function

```
CREATE OR REPLACE FUNCTION "proj"."Inventory_Update"()
RETURNS trigger
LANGUAGE 'plpgsql'
AS
$BODY$ DECLARE
i_ID numeric;
BEGIN
SELECT inventory.inventory_id INTO i_ID FROM inventory
WHERE inventory.inventory_id=NEW.inventory_id;
IF(i_ID=NEW.inventory_id) THEN
RAISE NOTICE 'Inventory_ID already exists. Violates the primary key constraints.';
RETURN null;
ELSE
RAISE NOTICE 'Data inserted successfully.';
RETURN new;
END IF;
END
$BODY$
```

Trigger Call

```
CREATE TRIGGER "Inventory_Trigger"
BEFORE INSERT
ON "proj".inventory
FOR EACH ROW
EXECUTE PROCEDURE "proj"."Inventory_Update"();
```

SQL Query

```
INSERT INTO inventory values (50,'MiddleCare','Delhi','CovidShield',1568);
```

Violation case

The screenshot shows the pgAdmin interface with the 'Browser' tab selected. The tree view on the left lists database objects under '201901408_db'. The 'Tables' node is expanded, showing 'center', 'employee', 'inventory', 'manager', 'order_details', 'slot', and 'users'. A 'Trigger Functions (1)' node is also present. The 'Query Editor' tab is active, displaying a SQL script. The script attempts to insert a new row into the 'inventory' table:

```

4 AS $BODYS
5 DECLARE
6   i_ID numeric;
7 BEGIN
8   SELECT inventory.inventory_id INTO i_ID FROM inventory
9   WHERE inventory.inventory_id=NEW.inventory_id;
10  IF(i_ID=NEW.inventory_id) THEN
11    RAISE NOTICE 'Inventory_ID already exists. Violates the primary key constraints.';
12    RETURN null;
13  ELSE
14    RAISE NOTICE 'Data inserted successfully.';
15    RETURN new;
16  END IF;
17
$BODY$
```

The script then creates a trigger named 'Inventory_Trigger' to handle the insertion:

```

20 CREATE TRIGGER "Inventory_Trigger"
21 BEFORE INSERT
22 ON "proj".inventory
23 FOR EACH ROW
24 EXECUTE PROCEDURE "proj"."Inventory_Update"();
```

Finally, it performs an 'INSERT INTO inventory values' statement:

```

25
26 INSERT INTO inventory values (50,'MiddleCare','Delhi','CovidShield',1568);
```

The 'Data Output' section shows the error message: 'NOTICE: Inventory_ID already exists. Violates the primary key constraints.' followed by 'INSERT 0 0'. The 'Messages' section shows 'Query returned successfully in 55 msec.'

Successful Case:

This screenshot is nearly identical to the previous one, showing the pgAdmin interface with the 'Browser' tab selected. The tree view and the 'Tables' node with its sub-tables are the same. The 'Query Editor' tab contains the same SQL script as the previous screenshot.

The 'Data Output' section shows the success message: 'NOTICE: Data inserted successfully.' followed by 'INSERT 0 1'. The 'Messages' section shows 'Query returned successfully in 70 msec.'

A green checkmark icon in the bottom right corner of the 'Messages' area indicates that the query was executed successfully.

2. Trigger function for inserting new employees.

Trigger Function

```
CREATE OR REPLACE function "proj".Employee_Update()
RETURNS trigger
LANGUAGE 'plpgsql'
AS
$BODY$ DECLARE
c_id numeric;
BEGIN
SELECT center.center_id INTO c_id FROM center
where center.center_id=NEW.employee_centerID;
IF(c_id=NEW.employee_centerID) THEN
RAISE NOTICE 'Data Inserted Successfully';
RETURN NEW;
ELSE
RAISE NOTICE 'Center is not available. Violates Foreign key constraints';
RETURN NULL;
END IF;
END
$BODY$
```

Trigger call

```
CREATE TRIGGER "Employee_Trigger"
BEFORE INSERT
ON "proj".employee
FOR EACH ROW
EXECUTE PROCEDURE "proj".Employee_Update();
```

```
insert into employee values(125,1,'Jenish','Male');
```

Unsuccessful case (violation case):

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

- Browser:** Shows the database structure with the `employee` table selected.
- Query Editor:** Contains the following SQL code:


```

14 RAISE NOTICE 'Center is not available. Violates Foreign key constraints';
15 RETURN NULL;
16 END IF;
17 END
18 $BODY$
```
- Output:** Displays the error message: "NOTICE: Center is not available. Violates Foreign key constraints" and "INSERT 0 0".
- Message Bar:** Shows "Query returned successfully in 51 msec."

Successful Case:

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

- Browser:** Shows the database structure with the `employee` table selected.
- Query Editor:** Contains the same SQL code as the previous screenshot:


```

14 RAISE NOTICE 'Center is not available. Violates Foreign key constraints';
15 RETURN NULL;
16 END IF;
17 END
18 $BODY$
```
- Output:** Displays the success message: "NOTICE: Data Inserted Successfully" and "INSERT 0 1".
- Message Bar:** Shows "Query returned successfully in 49 msec." with a green checkmark icon.

Section 7

Front End development Code

Backend Handling Program:

```
Vaccination > views.py
views.py

 1  from django.http import HttpResponseRedirect
 2  from django.shortcuts import redirect, render
 3  import psycopg2
 4
 5  # Setting up the connection to the Local database
 6  connection = psycopg2.connect(host="localhost",database="201901419_SRS",user="postgres",password="admin",options='--lc_ctype="en_US.UTF-8"')
 7  cursor = connection.cursor()
 8
 9
10 # Method for Home page
11 def Home(request):
12     return render(request,'Vaccination/index.html')
13
```

```
15 # Method for Inventory handling
16 def addInventory(request):
17     try:
18         if request.method=="POST":
19             if request.POST.get('i_insert'):
20                 # Handling post request for insertion
21                 i_ID = int(request.POST.get('i_ID'))
22                 Name = ""+request.POST.get('i_Name')+"""
23                 City = ""+request.POST.get('i_City')+"""
24                 VaccineType = ""+request.POST.get('i_type')+"""
25                 Stock = """+request.POST.get('i_stock')+"""
26
27                 querr = f'insert into inventory values({i_ID},{Name},{City},{VaccineType},{Stock})'
28                 cursor.execute(querr)
29                 connection.commit()
30
31                 querr = 'select * from inventory'
32                 cursor.execute(querr)
33                 connection.commit()
34                 tuples = cursor.fetchall()
35                 context = {'tuples':tuples}
36                 return render(request,'Vaccination/print_inventory.html',context)
37             elif request.POST.get('i_search'):
38                 # Handling post request for searching
39                 querr = 'select * from inventory'
40                 cursor.execute(querr)
41                 connection.commit()
42                 tuples = cursor.fetchall()
43                 context = {'tuples':tuples}
44                 return render(request,'Vaccination/print_inventory.html',context)
```

```

45     elif request.POST.get('i_delete'):
46         # Handling post request for deletion
47         print('Delete func')
48         iid = request.POST.get('i_delete')
49         querr = f'delete from inventory where inventory_id={iid}'
50         cursor.execute(querr)
51         connection.commit()
52         print('Deleted successfully')
53         querr = 'select * from inventory'
54         cursor.execute(querr)
55         connection.commit()
56         tuples = cursor.fetchall()
57         context = {'tuples':tuples}
58         return render(request,'Vaccination/print_inventory.html',context)
59
60     except:
61         return HttpResponse('<p>Some Error Occured</p>')
62
63
64     return render(request,'Vaccination/inventory.html')
65
66

```

```

66     # Method for User handling
67     def addUser(request):
68         try:
69             if request.method=="POST":
70                 if request.POST.get('u_insert'):
71                     # Handling post request for insertion
72                     u_ID = int(request.POST.get('u_ID'))
73                     u_CID = int(request.POST.get('u_Center_Id'))
74                     u_SID = int(request.POST.get('u_Slot_ID'))
75                     u_Name = """+request.POST.get('u_Name')+"""
76                     u_Gender = """+request.POST.get('u_Gender')+"""
77                     u_Age = int(request.POST.get('u_Age'))
78                     u_Add = """+request.POST.get('u_Address')+"""
79                     u_City = """+request.POST.get('u_City')+"""
80                     u_VaccineType = """+request.POST.get('u_Vaccine')+"""
81                     u_IS = """+request.POST.get('u_Is_Vaccinated')+"""
82
83                     querr = f'insert into users values({u_ID},{u_CID},{u_SID},{u_Name},{u_Gender},{u_Age},{u_Add},{u_VaccineType},{u_IS})'
84                     cursor.execute(querr)
85                     connection.commit()
86
87                     querr = 'select * from users'
88                     cursor.execute(querr)
89                     connection.commit()
90                     tuples = cursor.fetchall()
91                     context = {'tuples':tuples}
92                     return render(request,'Vaccination/print_user.html',context)

```

```

93     elif request.POST.get('u_search'):
94         # Handling post request for searching
95         querr = 'select * from users'
96         cursor.execute(querr)
97         connection.commit()
98         tuples = cursor.fetchall()
99         context = {'tuples':tuples}
100        return render(request,'Vaccination/print_user.html',context)
101    elif request.POST.get('u_delete'):
102        # Handling post request for deletion
103        u_id = request.POST.get('u_delete')
104        querr = f'delete from users where user_id={u_id}'
105        cursor.execute(querr)
106        connection.commit()
107
108        querr = 'select * from users'
109        cursor.execute(querr)
110        connection.commit()
111        tuples = cursor.fetchall()
112        context = {'tuples':tuples}
113        return render(request,'Vaccination/print_user.html',context)
114    except:
115        return HttpResponse('<p> Some error occured</p>')
116
117    return render(request,'Vaccination/user.html')
118
119

```

```

120    # Method for Admin handling
121    def addAdmin(request):
122        try:
123            if request.method=="POST":
124                if request.POST.get('a_insert'):
125                    # Handling post request for insertion
126                    a_ID = int(request.POST.get('a_ID'))
127                    Name = """+request.POST.get('a_Name')+"""
128                    Gender = """+request.POST.get('a_Gender')+"""
129                    City = """+request.POST.get('a_City')+"""
130                    querr = f'insert into administrator values({a_ID},{Name},{Gender},{City})'
131                    cursor.execute(querr)
132                    connection.commit()
133
134                    querr = 'select * from administrator'
135                    cursor.execute(querr)
136                    connection.commit()
137                    tuples = cursor.fetchall()
138                    context = {'tuples':tuples}
139                    return render(request,'Vaccination/print_admin.html',context)
140            elif request.POST.get('a_search'):
141                # Handling post request for searching
142                querr = 'select * from administrator'
143                cursor.execute(querr)
144                connection.commit()
145                tuples = cursor.fetchall()
146                context = {'tuples':tuples}
147                return render(request,'Vaccination/print_admin.html',context)

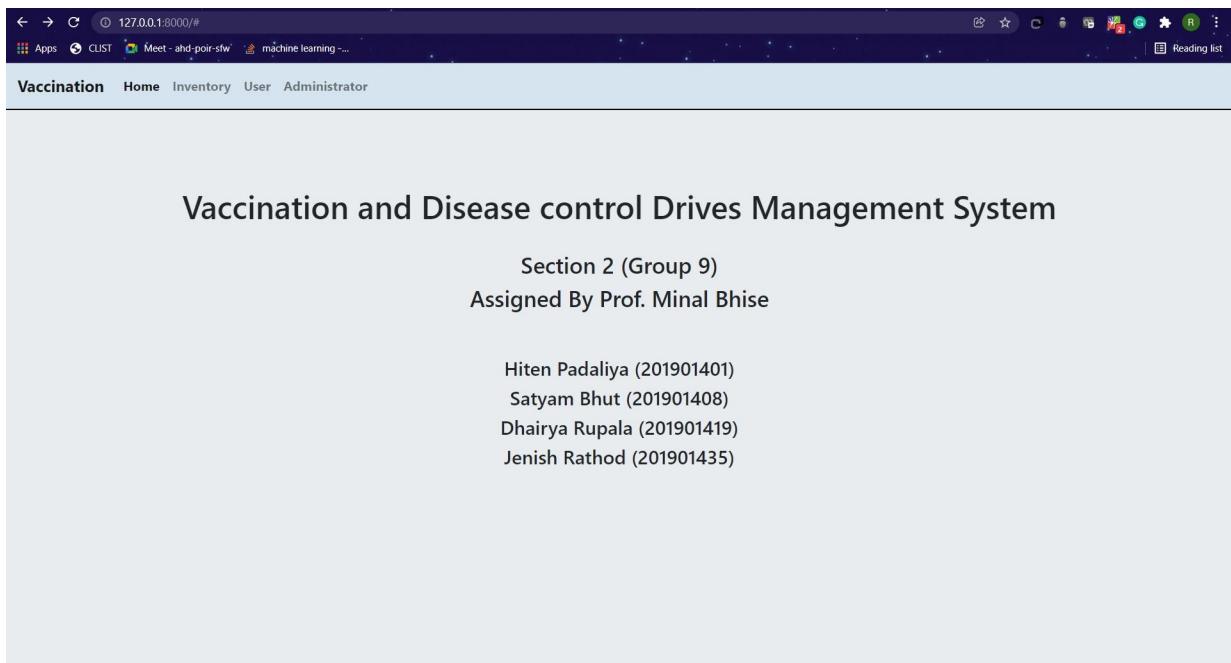
```

```

148     elif request.POST.get('a_delete'):
149         # Handling post request for deletion
150         a_id = request.POST.get('a_delete')
151         querr = f'delete from administrator where admin_id={a_id}'
152         cursor.execute(querr)
153         connection.commit()
154
155         querr = 'select * from administrator'
156         cursor.execute(querr)
157         connection.commit()
158         tuples = cursor.fetchall()
159         context = {'tuples':tuples}
160         return render(request,'Vaccination/print_admin.html',context)
161     except:
162         return HttpResponse('<p> Some error occurred</p>')
163
164     return render(request,'Vaccination/admin.html')
165

```

Home Page of the Website:



Admin Form of the Website:

The screenshot shows a web browser window with the URL `127.0.0.1:8000/administrator/`. The page title is "Admin". It contains a form with four input fields: "Id" (value: 12345), "Name" (value: Hiten Kaneriya), "Gender" (value: Male), and "City" (value: Rajkot). Below the form are two buttons: "Insert" and "Show All".

ID	Name	Gender	City
1	Tamiko	Male	Sujnipur
2	Gert	Female	Alandi
3	Dexter	Female	Indapurí
4	Tracie	Male	Dabheri
5	Bartel	Male	Singia
6	Millie	Female	Ranjapur
7	Eula	Male	Nainpura
8	Emmy	Male	Mewar
9	Angelo	Female	Vadamadurai

Showing the Admin Data:

The screenshot shows a web browser window with the URL `127.0.0.1:8000/administrator/`. The page title is "Admin Data". It displays a table with nine rows of data, each containing an ID, Name, Gender, and City, along with a "Delete" button.

ID	Name	Gender	City	
1	Tamiko	Male	Sujnipur	<button>Delete</button>
2	Gert	Female	Alandi	<button>Delete</button>
3	Dexter	Female	Indapurí	<button>Delete</button>
4	Tracie	Male	Dabheri	<button>Delete</button>
5	Bartel	Male	Singia	<button>Delete</button>
6	Millie	Female	Ranjapur	<button>Delete</button>
7	Eula	Male	Nainpura	<button>Delete</button>
8	Emmy	Male	Mewar	<button>Delete</button>
9	Angelo	Female	Vadamadurai	<button>Delete</button>

