Dhirubhai Ambani
Institute of Information and Communication Technology

Automobile Service Centre

Team ID: S3_T5

IT214: Database Management System
Assign By Prof. Minal Bhise

Mentor TA: Raj Shah

Prepared By:
201901410 – Deep Tank
201901432 – Nilesh Khimani
201901436 – Harsh Kacha
201901438 – Yash Mandaviya

Table of Contents

Section1: The final version of SRS

1.1 Introduction

1.1.1 Purpose

- One of the main reasons for creating a database system 'Automobile service centre'is to make an efficient system that covers all aspects related to automobile serviceson both user and admin side. The main goal of this project is to develop a databasefor a system that can be used to schedule and pay for automobile services.
- Customers, workers, planned services, and payments will all be tracked by the system. The technology will make it simple for employees to be assigned a specifictask, as well as for consumers to schedule service and pay for it. And as far as the admin is concerned it will help them to keep eye on each and every aspect of the centre.

1.1.2 Intended Audience and Reading Suggestions

- This SRS document will be useful for the service centre's owners, developers, projectmanagers, users, testers, documentation writers, etc.
- The system is intended for owners of the centre, employees (users at one end) and customers (users on the other end). Access to certain functions may differ from oneuser to the next. (For example, owners and managers have access to employee information, but employees do not.)

1.1.3 Product Scope

The programme aims to make the process of scheduling and paying for vehicle maintenanceas simple as possible. It includes the following features:

- Contains a mechanism for tracking each service that a client has requested.
- Maintains a record of every equipment in transportation, storage, processing, andusage.
- Makes physical inventory transportation, processing, and tracking more efficient.
- Assist audit efforts by ensuring inventory correctness.
- Keep track of the personnel details and their clearance to issue different equipmentand what has already been issued.
- Customers can get service of emergency breakdown like pickup and drop facilities according to extra charges.

1.1.4 Description

Requirements

System Requirements:

- It is required that the system we have designed don't crash all of a sudden, that's why
  It is important that the flow of the system follows a hierarchical order, we just don't want that
  Manager of the service centre gets a notification from authorized people that the inventory of
  The service centre has lesser things than it should have even before the service centres'
  Inventory is open for the use of parts.

Security Requirements:

- The software designed to two-way use, but there is a cache here, all the data privileges and Access to the data of the service center should be unreachable for employee working there and customers of the service center.
- And customer to customer private details should be hidden from each other and only accessed by the service center manager or authorized people or the customer itself.

Interview Requirements:

- As we are slowly improving the quality of the software we need some user suggestions as system is specially designed for the user to use so how the software improve, suggestions from user , budget used , systems failure , success , and much more data can be easily gained from the user as customers are using the non-perfect system (for tests only)
- That's why it is very important that as software developer we hear the needs and Suggestions from the user it will be very helpful in building good software to use.

Software Quality Requirements:

- Taking the process of building database further the software (front end + back end) part should be as it is user friendly and , it is easy for user to keep track of all of its needs and appointments that's why building a good software with good quality is crucial.
- The availability of the software should be available on vast level , it should save time of customer.
- As software has been built for the user the correctness of the same is important , user don't want that after updating data the storage still has its older (corrupted) data.
- As the whole software is being maintained by the administration of the service center , the software must be accurate and maintainable.

1.1.5 Relations

I. **Customers:** Information on the service centre's customers. customer_id, name, andcontact information will be included.

II. **Admin:** Information about employee and feedback from the employee as well asusers.

III. **Workers:** Information on the employees. This will include characteristics such asemployee id, employee name, and contact information.

IV. **Centre**: This relation will include attributes such as center_id, city, address, contact,timing, employee_cap.

V. **Services:** This relation will include attributes such as service id, customer id, employeeid (id allocated to employee), vehicle type, amount, status.

VI. **Available Services:** This table will include service_name, service_type, and cost asattributes. (A list of services offered)

VII. **Schedule:** Service id, pickup date, drop date, pickup time, drop time, pickup position,drop position, and information of employee and customer will all be fields in this table(name, contact details)

VIII. **Payment:** This table will include payment id, service id, customer id, amount,payment method, and ext_time information (exact time and date of payment).

IX. **Vehicle**: It will include information on vehicles such as vehicle_type, manufacturer,vehicle_name, and engine_type.

X. **Emergency Service:** It will include all requests which are made by user to get an immediate appointment. It will include services to be done immediately with extra charges.

XI. **Billing:** This will provide soft copy as well as physical copy of the bill to thecustomer.

XII. **Inventory:** This will keep track of automobile parts in a particular centre.

XIII. **Offers:** This will give information about offers that can be applicable with theparticular services.

1.1.6 Workflow of the Automobile service centre

- If we take the workflow of the automobile service center as simplest aspossible than the workflow can be described as following:
  - Customer visits the service centre with the problem in his/her vehicle.
  - In service centre, employees address the problem
  - Customer gets the repaired vehicle

❖ All above points can be described as follow:

- **Customer request service**:
  - very first step is that if the customer is visiting the service center first time then that customer will be provided unique customer_id and thenhis/her data will be added to the customer table.
  - If customer has visited the service center in past then the service center will have that customers data and hence it will be easier for thesevice center to help the customer.
  - Then customer requests for a service and then service centre will seeif the service is available or not in the services table, and if the serviceis available then they will check for the slot is available or not.
  - Then center will let the customer know, if the slot is available then center wil assign that slot to that perticular customer, if the slot is notavailable then service center will suggest the customer to take any other available slot.

- **Assignment of employee to the service ordered:**
  - After receiving the service request from the customer, cutomer will begetting service_id and one mechanic/employee would be assigned forthe service.

- **Scheduling the service ordered:**
  - After receiving the service requests from the customer, we will beadding the service with service_id in schedule table in which it willcontain details like pickup_date, drop_date with customer_id.

- **Updating the inventory dataset:**
  - After the service completion, inventory dataset would be updated aswe might have used some inventory items as per service requirements.

- **Generating the bill:**
  - After the service completion, according to the service done as per customer's request and applying applicable offers, payment_receipt(bill) has to be generated. So, we calculate the finalpayable and generate the bill.

- **Feedback from the customer:**
  - After service completion, customer will be asked for service feedbackabout the service and mechanic's / employee's work.

1.2 Document the requirements collection

1.2.1 Background Readings and References

- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.428.8720&rep=rep1&type=pdf
- https://vertabelo.com/blog/automobile-repair-shop-data-model/

Working and Flow:

i. **Customers and Contacts:**
- Customer relation will record all the customers who visited the service center atleast one time.
- It will contain several details of the customers like, customer_name (first_name,last_name), mobile number, email_address, address.
- Generally, Contacts relation will have the information about the communicationbetween the employee of the center and the customer itself.

ii. **Working Centers and Employees:**
- This working_center table will have the list of all the working centers. We can uniquely identify the cities by city_name and postal_codes of the cities. So, thistable will have attributes like, city_name, postal_codes.
- It contains the record of the employees of the center such as, employee_name,employee_id, contact, address, joining_date, city etc.

iii. **Vehicles:**
- This table will have the details of the vehicle which had visits to the center.
- It will consist several attributes like, vin (unique number to vehicle), plate_number,manufacture_year, vehicle_model, engine_type etc.

iv. **Visits:**
- This will keep record or history of the visits made by the customer to the servicecenter.
- Visits will have attributes like, service center_name, center_id, visit_date,visit_time, details of the vehicle like vehicle_model, number_plate, vehicle_service_detail, invoice, payment done by the cutomer.

v. **Services and offers:**
- This will contain the records of services that a particular center can provide to thecustomers as well as the prices for the same.
- Each service can be distinguished by unique service_id. It will have attributeslike, service_name, service price, time needed for service to be done.
- Offers relation will have the offers that can be offered to the customer for aparticular service with some minimum requirements to be fulfilled by the customer.

Basic Requirements for the system from the Background Readings
- Needs to maintain the information about working_centers, employees working in the center and details of the customers who had visits to the center and who havescheduled their visits to the center.
- System should maintain the data of all the customers who got service for theirvehicles from the particular service center.
- System should maintain the catalog of the services that can be provided to thecustomers along with offers that customer may apply with service.
- System has to maintain the scheduled services along with the emrgency_services that all the services need to be on time given to the customerfor the service.

Features that System can provide
- System has various service centers listed in the system. So, customers don't have to waste their time and they can visit any near working center and they canget service for their vehicle.
- To increase the efficiency and work-flow of the system, system has relations andsub-relations (references - primary key & foreign key).
- Because of this nesting of the queries can be avoided and system performancecan be increased.

1.2.2 Mock Interview Summary

Interview 1:

System: Automobile Service Centre

Project Reference: ACX3-173

Interviewee: 1) Mr. Vijay Hirapara (**Role Play**) **Designation**: Manager

Interviewer: 1) Harsh Kacha (**Role Play**)

Designation: Business Development Executive

2) Nilesh Khimani (**Role Play**)

Designation: Developer

Date: 08/10/2021 **Time:** 16:30

Duration: 1 hour **Place:** Google meet

Purpose of Interview:

- To identify the problems, requirements and suggestion given by manager ofthe centre.

Documents to be brought:

- Annual budget of the service centre.
- Analysis provided by developer and overview of usage of online site andapplication.
- Blueprint of the network that connected all the computers of the servicecentre.

Summary of the interview:

- Manager wants that the arrangements to be done such that queuing delay decreases at service centre of vehicles for which the appointment is given tothe customers.
- The notification that services centre get from customer on regular andemergency services should be in less than 3 minutes after successful registration.
- Manager wants to make system efficient to schedule the appointmentbooked by customers on different basis. Ex: By vehicle_type, proposed_visit_time, etc.
- Service Centre wants us to minimize the waiting time and of thevehicle arranging at centre and minimizing servicing time of thevehicles.
- The system should divide proposed areas in a way that every customergets the nearest service centre possible and customer in emergency should get the quickest response from the nearest service centre.

Interview 2:

System: Automobile Service Centre

Project Reference: AHN0-982

Interviewee: 1) Yash Mandaviya (**Role Play**) **Designation**: Customer

Interviewer: 1) Harsh Kacha (**Role Play**)

Designation: Business Development Executive

2) Nilesh Khimani (**Role Play**)

Designation: Developer

Date: 08/10/2021 **Time:** 17:30

Duration: 45 minutes **Place:** Google meet

Purpose of Interview:

- To identify if the current system is providing best services to the customersand get suggestion from the customers.

Documents to be brought:

- Customers' service center identification card.
- Customers' driving license.

Summary of the interview:

- Customer wants to keep the record of all of his/her bills in the mobileapplication of previous visits.
- After booking an appointment on the application or website, customer wants to get an acknowledgement to verify that the service center has gotand approved the appointment.
- Customer wants that service center must have a real-time tracking system,in which the service center has the location access of all of its registered customer to help them in emergency situations like accident.
- In regular (pre-booked appointment) visits at the service center, the customer wants to inspect his/her vehicle after all the repairing work is doneand if he is satisfied then after he will pay the bills.

Interview 3:

System: Automobile Service Centre

Project Reference: EJD5-068

Interviewee: 1) Ankit Rathod (**Role Play**) **Designation**: Stack holder

Interviewer: 1) Harsh Kacha (**Role Play**)

Designation: Business Development Executive

2) Nilesh Khimani (**Role Play**)

Designation: Developer

Date: 08/10/2021 **Time:** 18:30

Duration: 45 minutes **Place:** Google meet

Purpose of Interview (Meeting):

- Discussion on adding certain feature in system.

Documents to be brought by interviewer:

- Survey results.
- Potential budget for change in system.
- Chart on expected rise in profit.

Documents to be brought by interviewee:

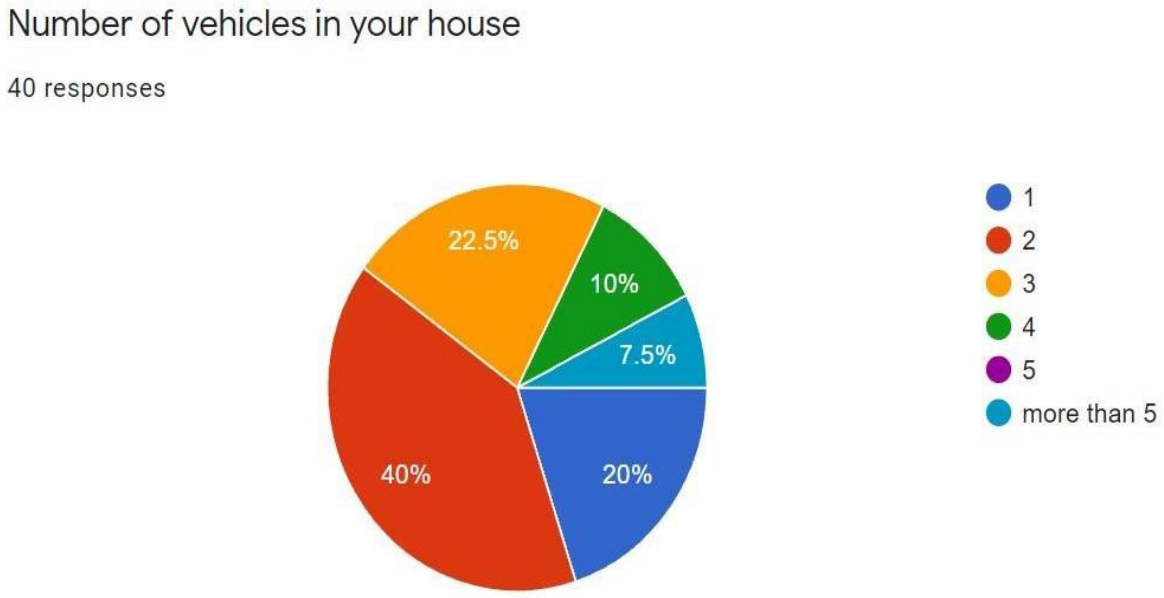
- Analysis on effects of changes made in history on system.

Summary of the Meeting:

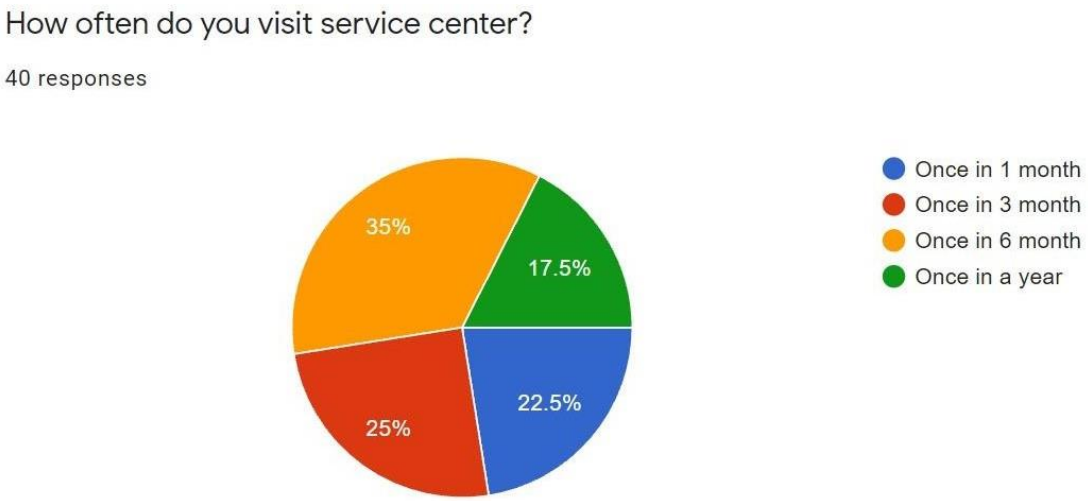
- The funds provider comes in the picture when an attempt to improve insystem is supposed for greater good of the service center.
- If any new application (which at first glance looks beneficial) is proposed tothe system then all financial support given by stock holders Is important.
- in meeting the effects of this new application is broadly discussedbetween both parties and then limitations are being decided.
- The preposition is rejected by Business development executive Ifcertain conditions are not reached after applying the changes.

1.2.3 Questionnaire:

1) **Number of vehicles in your house**

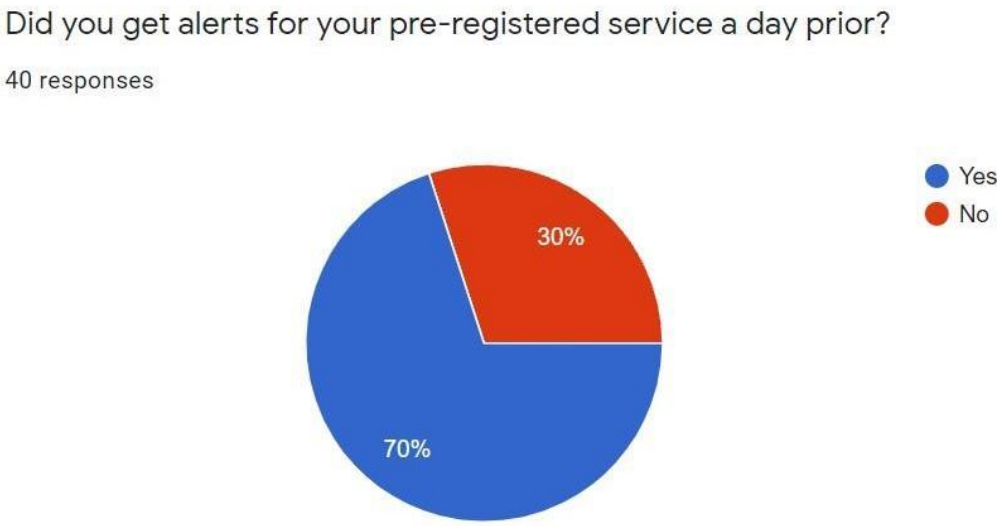


2) How often do you visit service center?

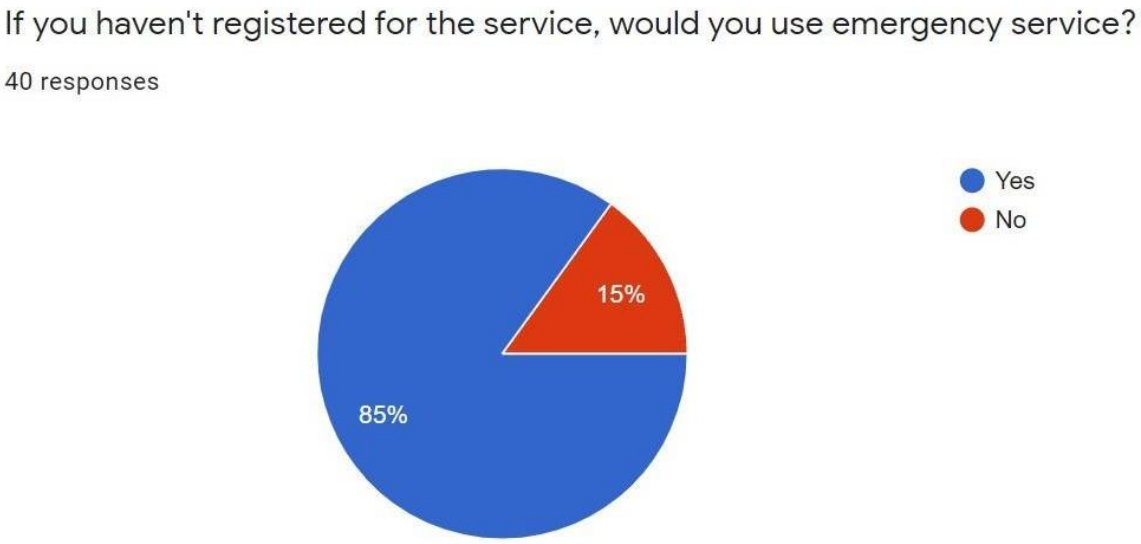


3) Did you get the notifications for the first 3 free services for yourvehicle?

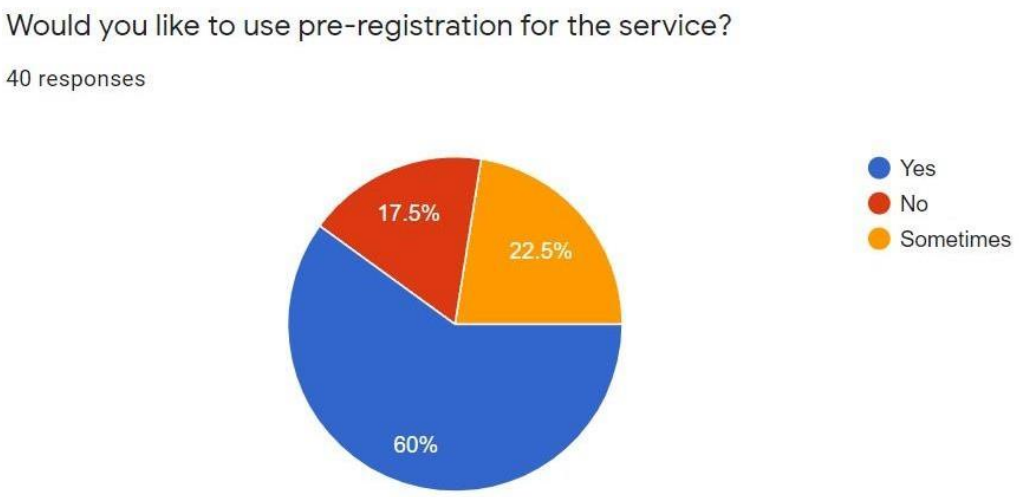
4) **Would you like to use pre-registration for the service?**



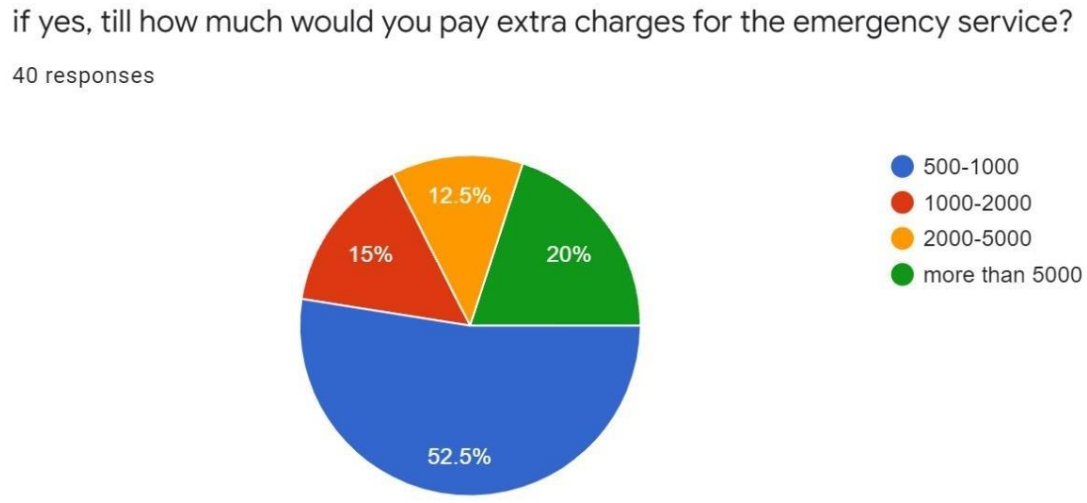
5) **Did you get alerts for your pre-registered service a day prior?**



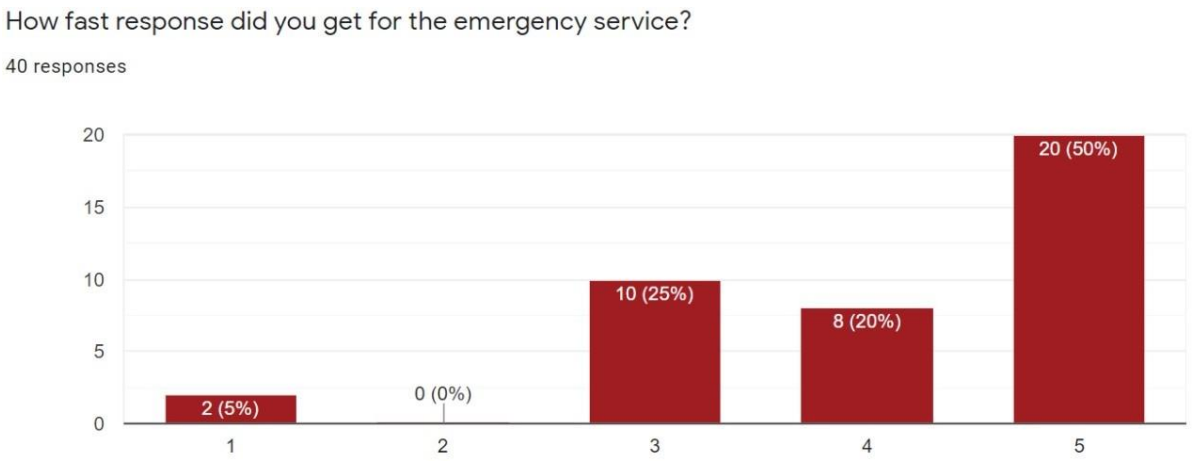
6) If you haven't registered for the service, would you use emergencyservice?



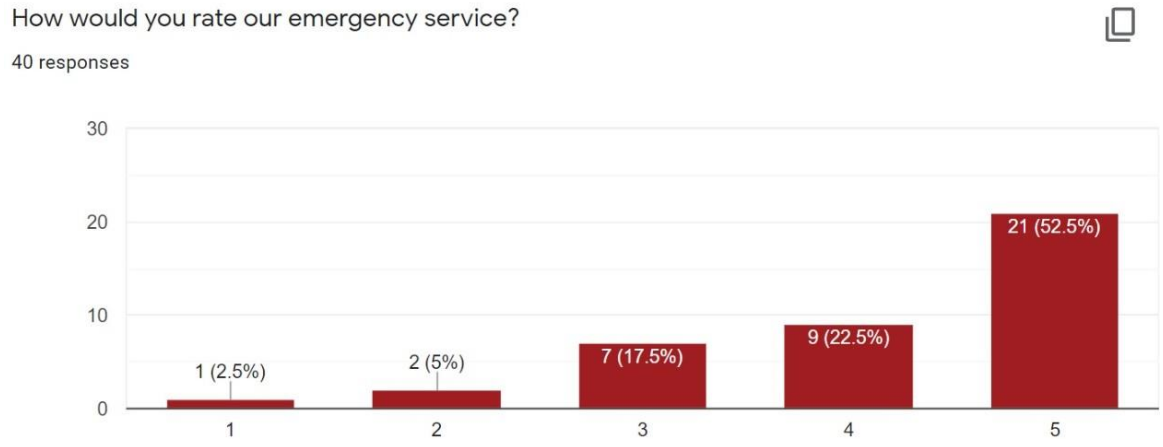
7) if yes, till how much would you pay extra charges for theemergency service?



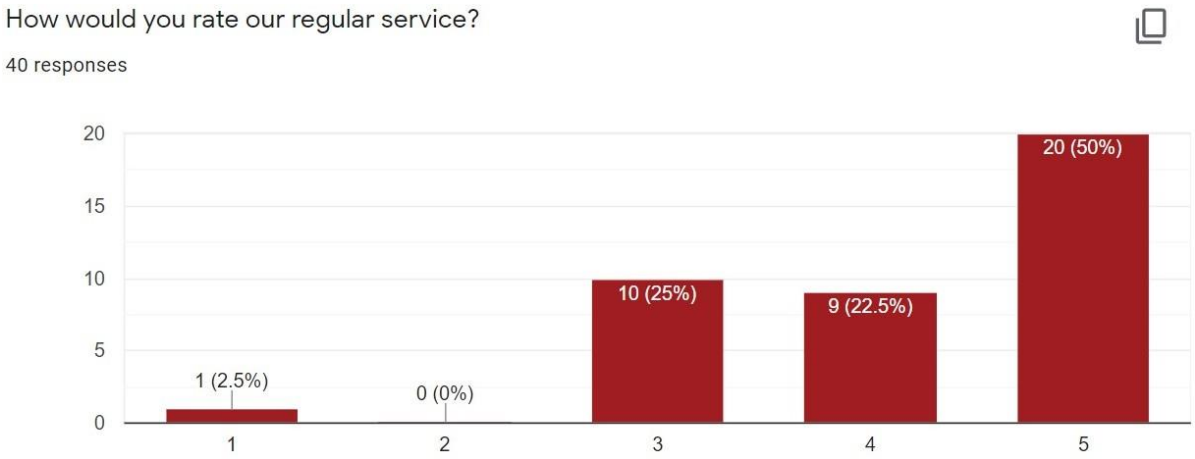
8) How fast response did you get for the emergency service?



9) How would you rate our emergency service?



10) How would you rate our regular service?



1.2.4 Observation:

- First of all, we observed the number of vehicles in customer's house and found outthat most of the customers are having 2 vehicles.
- Then we can see that around 35% users visit service center once in a six month andthere are few customers around 18% who visits service center once in a year.
- More than 2/3 users did get the notifications for the first three services for theirvehicle.
- Around 60% users would like to pre-registration for the service.
- Most of the users (70%) did get alerts for their pre-registered service a day prior.
- In the case of emergency services, the majority voted that they will use emergencyservices and half of the users would like to give extra 500-1000 INR for it.
- Half of the users said that they got very quick response from the service center foremergency services.
- Almost 53% rated our emergency services as very good and 50% users rated ourregular services as very good.

1.3 Fact Finding Chart

- Process of collecting data from examining documents, interviewing, questionnaire,observations and research.

Objective	Techniques	Subject	Time Commitment
To get the most basic knowledge ofthe system	Background readings/observation	Report of the service centre, IEEE docs, research papers by developers	2 hours
To find out the need/expectation and review of the customers	Interview	Customer	45 minutes
How the data is inserted and management of the data in systemand crucial delays on update of data	Interview and Background Readings	Developer, Current system operators	1 hour
To understand theflow of system andsetbacks where system might fail	Observation and using the current system	Employees, SRS documentwriter	1 hour
Execution of suggestion provided by customers	Interview	Department Head and Stack-holder	2 hours
Setting priority on the differentphases	Interview	Manager	1 hour

1.4 List of requirements

- It should keep track of service centre information such as center id, address, city,number of employees, contact information, and so on.
- It should keep track of employee information such as employee id, name, pay, joiningdate, and so on for employees working in a service centre.
- Customer information such as customer id, customer name, order date, and so onshould be kept track of.
- It should keep track of information about the service center's various services, as wellas any applicable discounts.
- It should keep track of order_id, order_date, service_id, service_price, paymentdetails, and other information about the customer's service order.

- We should create roles like CEO, manager, employee, etc, and provide them the privileges decided by the administrator to restrict misuse of the system and increasing system security.
- System should keep track of the inventory dataset. After completion of every service on the center, inventory dataset should be updated accordingly.
- Easy to use and simple mechanism of the system.
- It should include entity integrity and referential integrity to uniquely identify each employee, customer, services etc., and to prevent the system from providing wrong information.
- Feedbacks form the customers as well as from the employees should be taken on a regular basis and feedbacks should be addressed properly.

1.5 User Classes and Characteristic

- **Developers:** Developers are the people who are going to build the whole system. If there will be any problems regarding the database, they are the once who is going to solve the issues. In this case, they are going to access particular dataset in which problems may have occurred.

- **Customers:** Customers will use the system to book a service for their vehicles. They will have access to all the which are related to them such as, available services, Emergency Services, schedule, service history, pricing etc.

- **Employees / Mechanics:** Employees are the once who will be working on the services. They will have access to customer dataset, schedules, payments and inventory dataset.

- **Owner of the Company:** He is the administrator of the system who will be having access to all the privileges. This is the owner who controls the work-flow of the centers. He is the person who will be guiding to center managers according to the feedbacks from the customers as well as employees or mechanics.

- **Center Owner:** He is the person who owns the center in a particular city. He guides the manager and takes the necessary actions according to the manager feedbacks for the particular center that he owns. He will have the access to all the privileges ofthe particular center like, employees, inventory, payments etc.

- **Center Manager:** Manager will have the access to all the database that employees have and he will have the access to employee's database too. He is the person whowill be managing the services, payments, inventory for a particular center.

1.6 Operating Environment

- **Hardware, Software and Connectivity Requirements:**
  - Servers to store all the databases
  - Basic need is computers to operate the whole system.
  - Internet Connectivity

- **External Interface requirements:**
  - Very large place too accommodates all types of vehicles at all the service centers.
  - All the automobile parts needed for the services for all types of vehicles.
  - Toolkits for employees
  - Towing service that needed to pick up the vehicle.

1.7 Product Function

- **Insert_center:** This function will include attributes like center_id, city, address,contact as an input and insert it into the center relation.

- **Insert_employee:** This function will include attributes like employee_id, name, contact, address, salary, center_id.

- **Register_customer:** This function will include attributes like customer_id, name, contact and add them to the customers relation.

- **Insert_vehicle_details:** This function takes attributes of vehicle as input and adds them to the vehicle relation.

- **Delete_center:** This function is used to remove center from the center relation.

- **Delete_employee:** This function is used to remove an employee from the employee's relation.

- **Customer_info:** This is a simple function which is used for viewing the profile of a customer.

- **Book_service:** This function is used to book a service by a customer. The function will take details of customer, vehicle, type of service, pickup date as input.

- **Schedule_service:** This function takes details from the book_service function and after some calculations, it will give an estimated time of completion.

- **Notify_for_service:** This function takes the date of a completed service and generates regular service notification.

- **Apply_offer:** This function applies the offer related to the service and returns the final price.

- **Generate_bill:** This function generates the final bill of services requested by a customer.

- **Update_status:** This function is used to update the status of the services.

- **Get_status:** This function is used to check the status of a service request.

- **Get_feedback:** This function is used to get feedback from the customer and store itin the database.

1.8 Privileges

- **Here is the list of all functions and users who can access those modules.**

  i. **Register & Sign Up:**
  - Manager
  - Employee
  - Customer

  ii. **Issued service/parts:**
  - Employee
  - Customer

  iii. **Search:**
  - **Manager:** can see all the tasks which are assigned to employees andalso see the available materials for servicing.
  - **Employee:** can see the list of services that they are supposed to doand also check the parts that they need for assigned service.
  - **Customer:** can only see the services that they can get from servicecentre.

  iv. **equipment issue / return / upgrade:**
  - Manager

  v. **add/remove equipment**:
  - Employee

1.9 Assumption

- Users of this application are expected to have the appropriate hardware and softwareto execute it.
- It is also expected that data in the database is updated to the proper value at all times.
- Users will have alternate resources available in the case of an unexpected problem.

1.10 Business Constrains

- Details regarding the services provided, as well as their availability and status, shouldbe updated on a regular basis.
- If there is a delay in services or components are unavailable, replacements should be requested, and refunds/discounts should be given accordingly.
- The implementation of new database system and replacement of existing database system must be done in a timely manner so that existing services scheduled by the customers are not affected in any way possible.
- Implementation of a new database system can require an upgrade in existing hardware and software. The budget allocated for the upgrade can directly affect the design of the database system.
- The database system can require maintenance and upgrade in future times. The company needs to take into account the costs required for maintenance and upgrade while allocating a budget for the development of the entire database system.

Section2: Noun Analysis

2.1 Final Problem Description

Introduction

Purpose

- One of the main reasons for creating a database system 'Automobile service centre' is to make an efficient system that covers all aspects related to automobile services on both user and admin side. The main goal of this project is to develop a database for a system that can be used to schedule and pay for automobile services.
- Customers, workers, planned services, and payments will all be tracked by the system. The technology will make it simple for employees to be assigned a specific task, as well as for consumers to schedule service and pay for it. And as far as the admin is concerned it will help them to keep eye on each and every aspect of the centre.

Intended Audience and Reading Suggestions

- This SRS document will be useful for the service centre's owners, developers, project managers, users, testers, documentation writers, etc.
- The system is intended for owners of the centre, employees (users at one end) and customers (users on the other end). Access to certain functions may differ from one user to the next. (For example, owners and managers have access to employee information, but employees do not.)

Product Scope

The programme aims to make the process of scheduling and paying for vehicle maintenance as simple as possible. It includes the following features:

- Contains a mechanism for tracking each service that a client has requested.
- Maintains a record of every equipment in transportation, storage, processing, and usage.
- Makes physical inventory transportation, processing, and tracking more efficient.
- Assist audit efforts by ensuring inventory correctness.
- Keep track of the personnel details and their clearance to issue different equipment and what has already been issued.
- Customers can get service of emergency breakdown like pickup and drop facilities according to extra charges.

Description

Requirements

- The technology will be used to make reservations and pay for automobile services. This technology may be used by customers from various location and various types of vehicles. For follow-up, each customer will be allocated to an agent.
- A pickup and drop option will be available, as well as client feedback after the completion of the service. Regular service notifications, emergency breakdown service, and other services will be available through the system.
- There will be two sides in the system. Customers will have one side of the system, while centre workers will have the other. Customers will have access to choices such as scheduling and payment, while staff will have pickup/drop-off location options.

Relations

I. **Customers:** Information on the service centre's customers. customer_id, name, and contact information will be included.

II. **Admin:** Information about employee and feedback from the employee as well as users.

III. **Workers:** Information on the employees. This will include characteristics such as employee id, employee name, and contact information.

IV. **Centre**: This relation will include attributes such as center_id, city, address, contact, timing, employee_cap.

V. **Services:** This relation will include attributes such as service id, customer id, employee id (id allocated to employee), vehicle type, amount, status.

VI. **Available Services:** This table will include service_name, service_type, and cost as attributes. (A list of services offered)

VII. **Schedule:** Service id, pickup date, drop date, pickup time, drop time, pickup position, drop position, and information of employee and customer will all be fields in this table (name, contact details)

VIII. **Payment:** This table will include payment id, service id, customer id, amount, payment method, and ext_time information (exact time and date of payment).

IX. **Vehicle**: It will include information on vehicles such as vehicle_type, manufacturer, vehicle_name, and engine_type.

X. **Emergency Service:** It will include all requests which are made by user to get an immediate appointment. It will include services to be done immediately with extra charges.

XI. **Billing:** This will provide soft copy as well as physical copy of the bill to the customer.

XII. **Inventory:** This will keep track of automobile parts in a particular centre.

XIII. **Offers:** This will give information about offers that can be applicable with the particular services.

Workflow of the Automobile service centre:

- If we take the workflow of the automobile service center as simplest as possible than the workflow can be described as following:
  - Customer visits the service centre with the problem in his/her vehicle.
  - In service centre, employees address the problem
  - Customer gets the repaired vehicle

- All above points can be described as follow:

  - **Customer request service:**
    - very first step is that if the customer is visiting the service center first time then that customer will be provided unique customer_id and then his/her data will be added to the customer table.
    - If customer has visited the service center in past then the service center will have that customers data and hence it will be easier for the sevice center to help the customer.
    - Then customer requests for a service and then service centre will see if the service is available or not in the services table, and if the service is available then they will check for the slot is available or not.

- Then center will let the customer know, if the slot is available then center wil assign that slot to that perticular customer, if the slot is not available then service center will suggest the customer to take any other available slot.

- **Assignment of employee to the service ordered:**
  - After receiving the service request from the customer, cutomer will be getting service_id and one mechanic/employee would be assigned for the service.

- **Scheduling the service ordered:**
  - After receiving the service requests from the customer, we will be adding the service with service_id in schedule table in which it will contain details like pickup_date, drop_date with customer_id.

- **Updating the inventory dataset:**
  - After the service completion, inventory dataset would be updated as we might have used some inventory items as per service requirements.

- **Generating the bill:**
  - After the service completion, according to the service done as per customer's request and applying applicable offers, payment_receipt(bill) has to be generated. So, we calculate the final payable and generate the bill.

- **Feedback from the customer:**
  - After service completion, customer will be asked for service feedback about the service and mechanic's / employee's work.

2.2 Noun & Verb Analysis

Noun	Verb
Main reason	Service
database system ‘automobile service centre	Centre
efficient system	Make
Automobile service	Side
Admin side	Project
Main goal	Develop
Automobile services	Can
Customers	Be
Specific task	Used
service centre’s	Schedule
Srs	Pay
Schedule service	Will
Project manager	Task

Documentation writers	Well
Access	Help
Certain functions	Keep
Employee information	Eye
Vehicle maintenance	Document
programme aims	End
Contains	Access
Maintains	Differ
Makes	Have
Physical inventory transportation	Do
Assist	Scope
Audit	Programme
Efforts	Record
Inventory correctness	Inventory
Keep	Process
Personal details	Assist
Different equipment	Audit
Emergency breakdown	Track
Drop facilities	Issue
Extra charges	Got
Requirements	Like
Automobile services	Drop
Various location	Follow
Various types	Get
Drop option	Up
Client feedback	Option
Regular	While
Service notification	Staff
Emergency breakdown	Off
Customers	Name
Centre workers	Contact
Pickup/drop-off location options	Include

Information	Id
Service centre’s customers	Address
Contact information	Type
Employee name	Cost
Employee id	List
Service id	Amount
Customer id	Date
Employee id	Time
Vehicle type	Position
Pickup date	Exact
Drop date	Provide
Pickup time	Copy
Drop time	Bill
Pickup position	Give
Drop position	Take
Contact details	Request
Payment id	Step
Payment method	Past
Ext_time information	See
Exact time	Check
Immediate appointment	Slot
Extra charges	Let
Soft copy	Know
Physical copy	assign
Automobile parts	suggest
Particular services	Would
Service centre	Contain
Unique customer_id	Calculate
His/her data	Generate
Customer table	Work
Service table	Reasons
Centre will	Is

Particular customer	Covers
available slot	Services
Assignment	Functions
Service request	Aims
Scheduling	Includes
Schedule table	Features
Updating	Contains
Inventory dataset	Has
Service completion	Maintains
Inventory dataset	Makes
Inventory items	Details
Service requirements	Charges
Generating	Types
Service completion	Sides
Feedback	Attributes
Customer’s request	Fields
Service feedback	Requests
Mechanic’s / employee’s work	Parts
	Offers
	Visit
	Gets
	Points
	Item
	Related
	Used
	Track
	planned
	Assigned
	Concerned
	Intended
	Issued
	Requested

	Allocated
	Included
	Cost
	Offered
	Made
	Described
	Repaired
	Provided
	Added
	visited
	Let
	Ordered
	Updated
	Generated
	Asked
	Related
	Used
	Planned
	Tracked
	Concerned
	Assigned
	Been
	Included
	Made
	Generated
	Asked
	Creating
	Reading
	Tracking
	Scheduling
	Paying
	Following

	Processing
	Ensuring
	According
	Timing
	Receiving
	Visiting
	Getting
	Adding

Candidate entity set	**Candidate attribute set**	**Candidate relationship set**
Customer	Costomer_id, customer_name, contact_number, address	Pays, books, gives, assigns
Employee	Emplyee_id, employee_name, contact_number, address, salary	Joins, assigns
purchase	Purchase_id, part_id, units, cost, purhcase_date	
Inventory	Part_id, center_id, part_name, stock, retal_price	
offers	Offer_id, discount_percentage, max_discount, start_date, end_date	
center	Center_id, city, address, contact	

manager	Manager_id, manager_name, center_id, contact	
feedback	feedback_id, feedback	gives
Vehicle	Vehicle_id, vehicle_type, manufacturer, vehicle_name, engine_type	
schedule	Pickup_date, drop_date, pickup_time, drop_time, pickup_location, drop_location	schedules
available_services	Service_id, service_name, service_type, service_price, service_duration, offer_id	
Order	Order_id, service_id, employee_id, vehicle_type, amount, status	Assigns, schedules
Payment	payment_id , order_id, amount, payment_method, timestamp	pays
Emergency_service	Order_id, payment_id, service_id, employee_id, vehicle_type, amount, status, extra_charges	Assigns, schedules

2.3 Rejected Verbs	Reason
Generating	General
Receiving	General
Records	Irrelevant
Requesting	General
Schedule	Duplicate
Taken	Irrelevant
Taken	irrelevant
Task	general
Generated jobs	General
Add	General
Adding	General
Alerts	Irrelevant
Apply	Vague
Applying	Vague
Document	General
Drop	Irrelevant
Employs	Irrelevant
End	General
Finished	General
Follow	General
Following	General
Fulfil bill	irrelevant
Changed	General
Charges	Vague
Check	General
Completed	General
Consists	General
Contain	General
Options	General
Ordered	duplicate
Paying	Duplicate
Present	General
Process	General
Provide	General
Purchase	Duplicate

2.4 Rejected Nouns	Reason
Documentation writers	Irrelevant
Service duration	duplicate
Available_services	Duplicate

Workflows	Irrelevant
Admin dashboard	Irrelevant
Assignment	General
Service duration	Duplicate
Payment receipt	Irrelevant
Calculation	General
Access employee details	General
New employees	general
Customer	Duplicate
Emergency breakdown pickup	General
Pickup/drop duration	Duplicate
Employs	General
Employee’s professionalism	Irrelevant
New employee joins	General
Details	Vague
Particular task	General
User dashboard	Irrelevant
Schedule	Duplicate
Drop facilities	General
Feedback	Duplicate
Order_status	Duplicate
Automobile services	General
Slot availability	General
Employs	General
Calculation	General
Access employee details	General
New employees	General
Description requirements	Irrelevant
Options	General
Customer relation	General
Automobile	General
Regular service alerts	general
Access	Vague
Joining	General
Implements	General
Generating	General
Updation	General
Drawbacks	General
Spare parts	Irrelevant
Srs	General
audit	General
Assist	General
Different equipment	General
Certain functions	Irrelevant
Schedule service	General
Programme aims	general
Efficient system	Irrelevant
Keep	Irrelevant
Extra charges	General
Emergency service	General
Various location/ types	Irrelevant
Physical copy	General
Available slot	Duplicate

2.5 Identify Entity Types

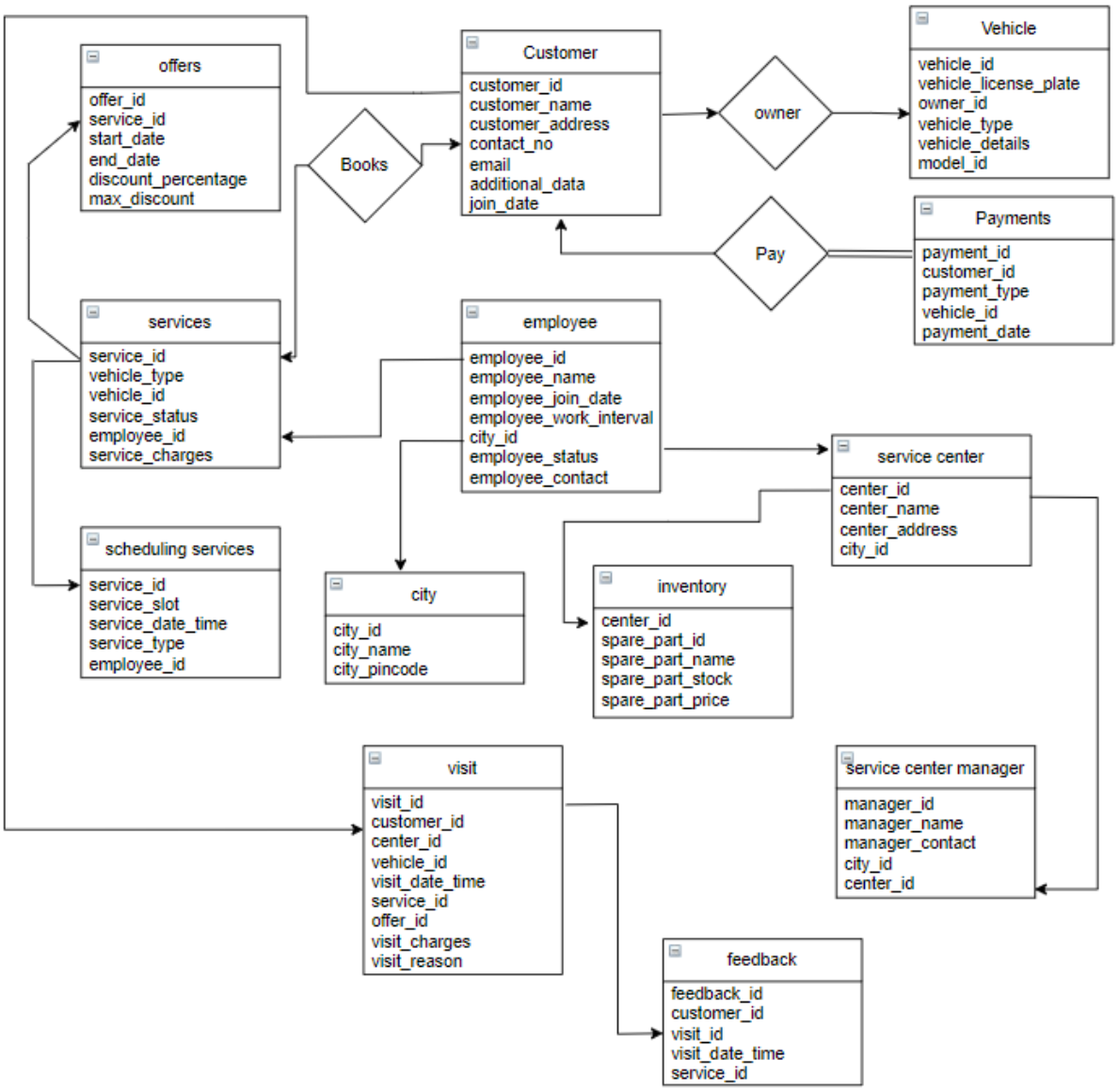
Entity Set	Entity Set type	Identifying Relation	Identifying Entity Set
Scheduling_service	Weak entity	Schedules	Order_service
payments	Weak entity	Pays	Customers

2.6 Identify Relationship Types

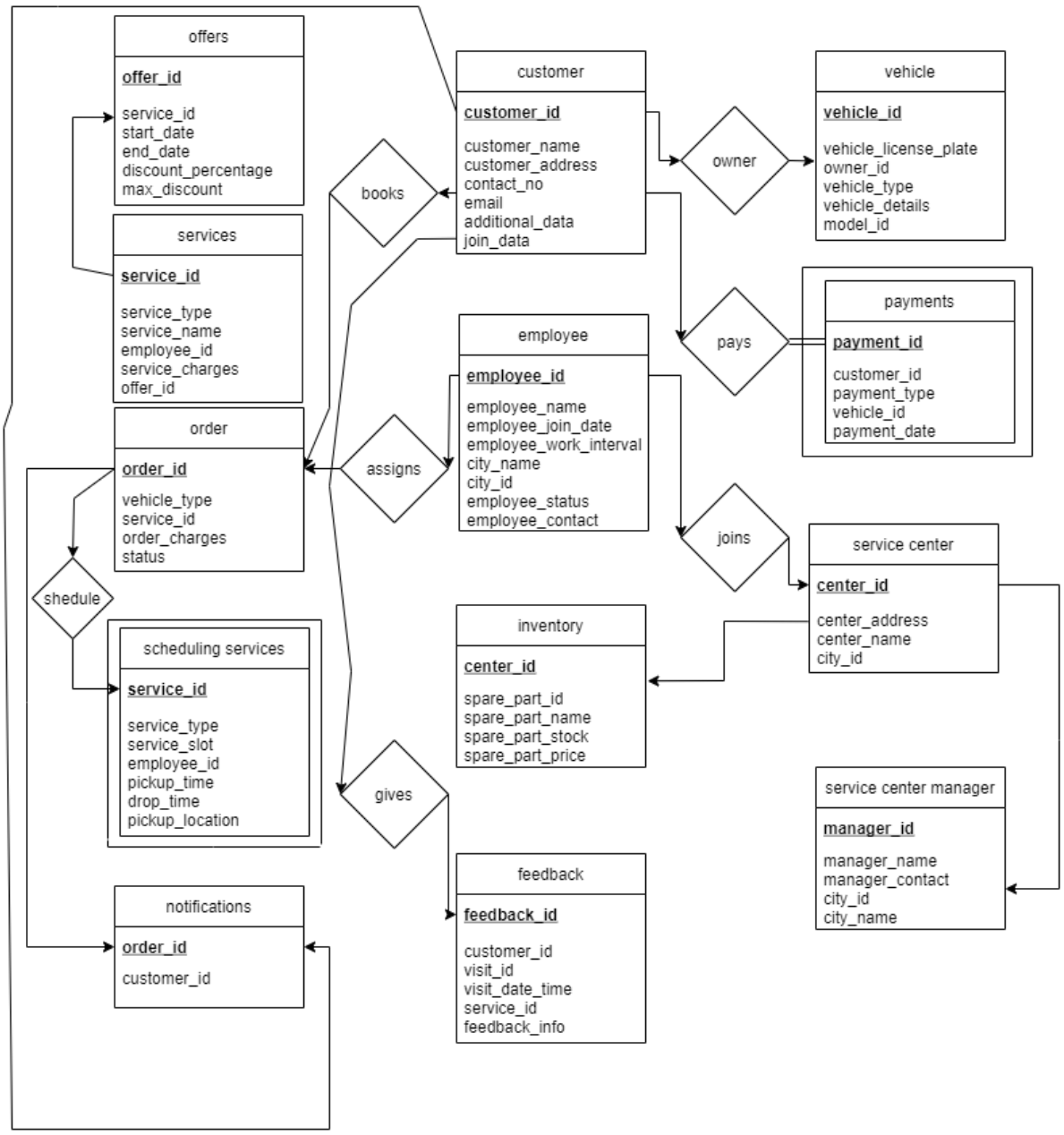
Relationship Name	Relationship Type
Pays	Ternary
Books	Binary
Gives	Binary
Assigns	Binary
Schedules	Binary
Joins	Binary
Owner	Binary

Section3: ER-Diagrams all versions

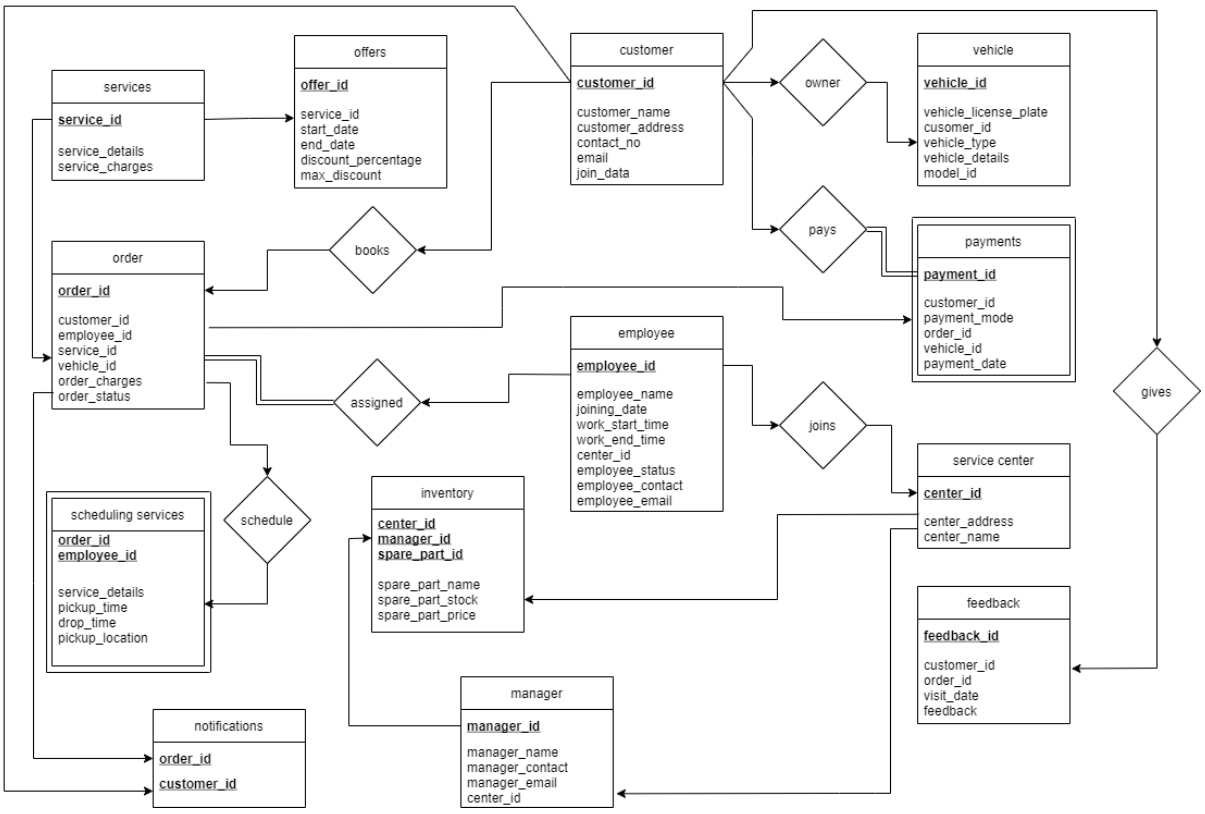
3.1 E-R Diagram (ERD) Version 1



3.2 E-R Diagram (ERD) Version 2

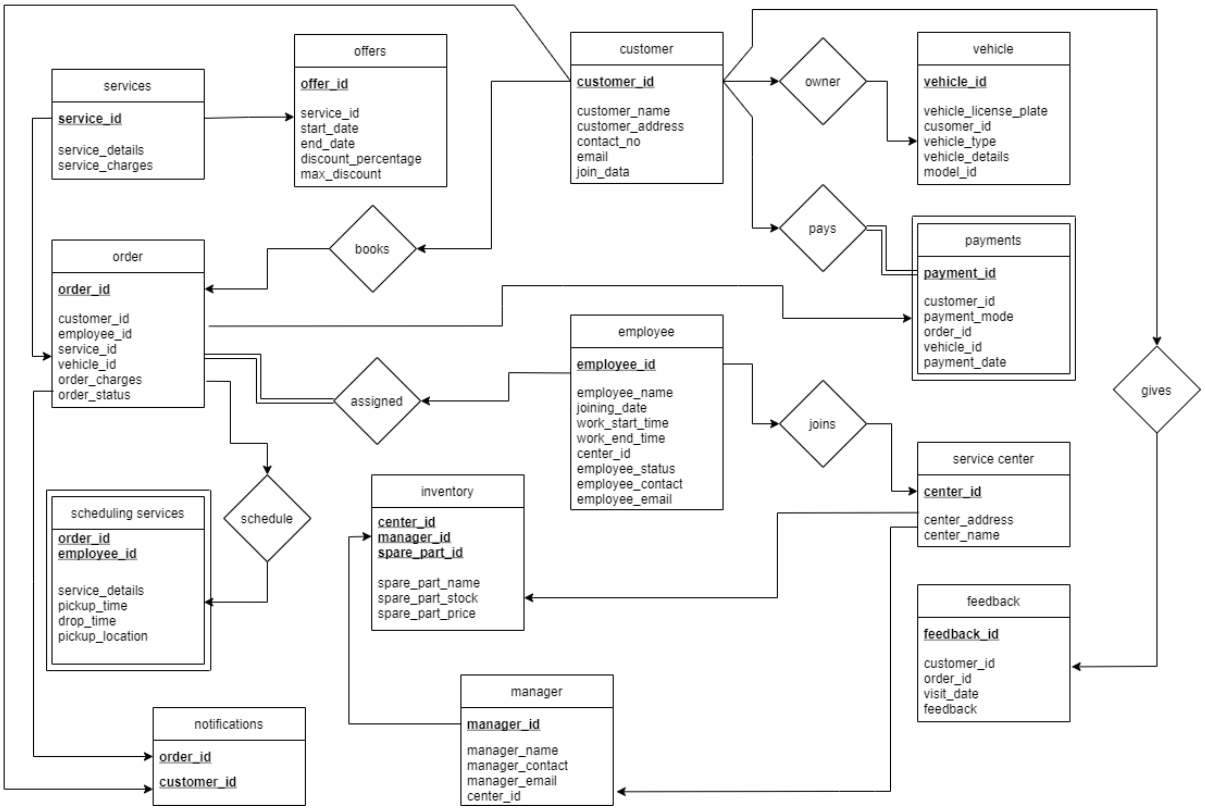


3.3 Final E-R Diagram (ERD)



Section4: Conversion of Final E-R Diagram to Relational Model

4.1 Final Version of E-R Diagram

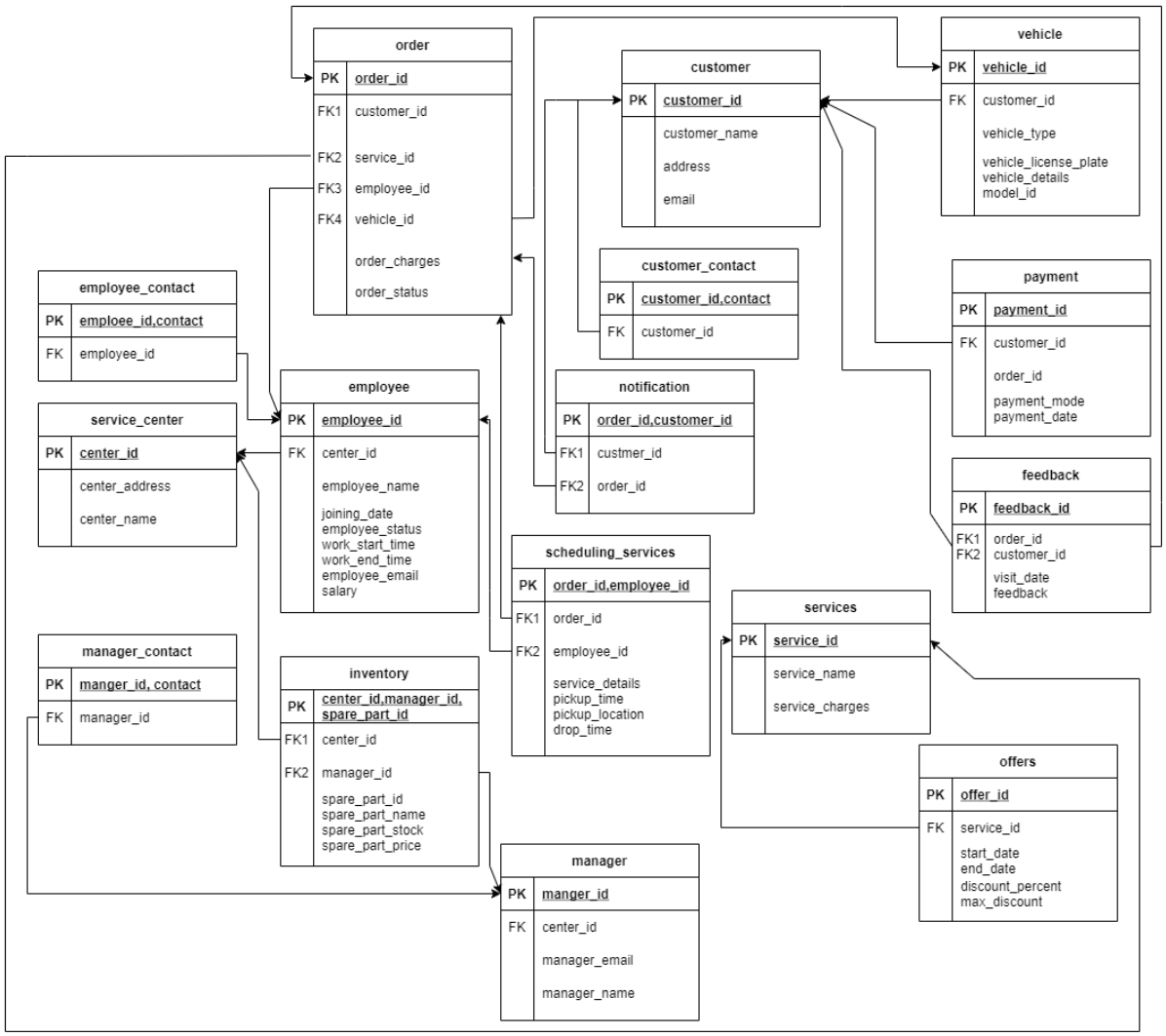


4.2 Mapping ER Model to Relation Model:

- Customer (**customer_id**, customer_name, customer_address,contact_no, email)
- Employee (**employee_id**, employee_name, joining_date, work_start_time, work_end_time, center_id, employee_status,employee_contact, employee_email)
- Services (**service_id**, service_details, service_charges)
- Offers (**offer_id**, service_id, start_date, end_date,discount_percentage, max_discount)
- Order (**order_id**, customer_id, employee_id, service_id, vehicle_id,order_charges, order_status)
- Scheduling services (**order_id**, **employee_id**, service_details, pickup_time, drop_time, pickup_location)
- Notifications (**order_id**, **customer_id**)

- Vehicle (**vehicle_id**, vehicle_license_plate, customer_id,vehicle_type, vehicle_details, model_id)
- Payments (**payment_id**, customer_id, payment_mode,order_id, vehicle_id, payment_date)
- Service center (**center_id**, center_address, center_name)
- Manager (**manager_id**, manager_name, manager_contact, manager_email, center_id)
- Inventory (**center_id**, **manager_id**, spare_part_id, spare_art_name, spare_part_stock, spare_part_price)
- Feedback (**feedback_id**, customer_id, order_id, visit_date,feedback)

4.3 Relational Model



Section5: Normalization and Schema Refinement

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

- Customer (**customer_id**, customer_name, customer_address, contact_no, email)
- Employee (**employee_id**, employee_name, joining_date, work_start_time, work_end_time, center_id, employee_status, employee_contact, employee_email)
- Services (**service_id**, service_details, service_charges)
- Offers (**offer_id**, service_id, start_date, end_date, discount_percentage, max_discount)
- Order (**order_id**, customer_id, employee_id, service_id, vehicle_id, order_charges, order_status)
- Scheduling services (**order_id**, **employee_id**, service_details, pickup_time, drop_time, pickup_location)
- Notifications (**order_id**, **customer_id**)
- Vehicle (**vehicle_id**, vehicle_license_plate, customer_id, vehicle_type, vehicle_details, model_id)
- Payments (**payment_id**, customer_id, payment_mode, order_id, vehicle_id, payment_date)
- Service center (**center_id**, center_address, center_name
- Manager (**manager_id**, manager_name, manager_contact, manager_email, center_id)
- Inventory (**center_id**, **manager_id**, spare_part_id, spare_art_name, spare_part_stock, spare_part_price)
- Feedback (**feedback_id**, customer_id, order_id, visit_date, feedback)

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

- Customer (**customer_id**, customer_name, customer_address, contact_no, email)
  - Primary key: customer_id
  - Foreign key: N/A
  - Functional dependencies: customer_id → customer_name, customer_id → customer_address, customer_id → contact_no, customer_id → email

- Employee (**employee_id**, employee_name, joining_date, work_start_time, work_end_time, center_id, employee_status, employee_contact, employee_email)
  - Primary key: employee_id
  - Foreign key: centre_id
  - Functional dependencies: employee_id → centre_id, employee_id → employee_name, employee_id → joining_date, employee_id → work_start_time, employee_id → work_end_time, employee_id → employee_status, employee_id → employee_contact, employee_id → employee_email

- Services (**service_id**, service_name, service_charges)
  - Primary key: service_id
  - Foreign key: N/A
  - Functional dependencies: service_id → service_name, service_id → service_charges, service_id → offer_id,

- Offers (**offer_id**, service_id, start_date, end_date, discount_percentage, max_discount)
  - Primary key: offer_id
  - Foreign key: service_id
  - Functional dependencies: offer_id → service_id, offer_id → start_date, offer_id →end_date, offer_id → discount_percetage, offer_id → max_discount

- Order (**order_id**, customer_id, employee_id, service_id, vehicle_id, order_charges, order_status)
  - Primary key: order_id
  - Foreign key: customer_id, employee_id, service_id, vehicle_id
  - Functional dependencies: order_id → employee_id, order_id → customer_id, order_id → service_id, order_id → vehicle_id, order_id →order_chardes, order_id → order_status

- Scheduling services (**order_id**, **employee_id**, service_details, pickup_time, drop_time, pickup_location)
  - Primary key: (order_id, employee_id)
  - Foreign key:
  - Functional dependencies: (order_id, employee_id) → service_details, (order_id, employee_id) → pickup_time, (order_id, employee_id) → drop_time, (order_id, employee_id) → pickup_location

- Notifications (**order_id**, **customer_id**)
  - Primary key: (order_id, customer_id)
  - Foreign key: customer_id
  - Functional dependencies: (order_id, customer_id) → customer_id

- Vehicle (**vehicle_id**, vehicle_license_plate, customer_id, vehicle_type, vehicle_details, model_id)
  - Primary key: vehicle_id
  - Foreign key: customer_id
  - Functional dependencies: vehicle_id → customer_id, vehicle_id → vehicle_license_plate, vehicle_id → vehicle_type, vehicle_id → vehicle_details, vehicle_id → model_id

- Payments (**payment_id**, customer_id, payment_mode, order_id, vehicle_id, payment_date)
  - Primary key: payment_id
  - Foreign key: customer_id, order_id, vehicle_id
  - Functional dependencies: payment_id → customer_id, payment_id → order_id, payment_id → vehicle_id, payment_id → payment_mode, payment_id → payment_date

- Service center (**center_id**, center_address, center_name)
  - Primary key: centre_id
  - Foreign key: N/A
  - Functional dependencies: centre_id → centre_address, centre_id → centre_name

- Manager (**manager_id**, manager_name, manager_contact, manager_email, center_id)
  - Primary key: manager_id
  - Foreign key: centre_id
  - Functional dependencies: manager_id → centr_id, manager_id → manager_name, manager_id → manager_contact, manager_id → manager_email

- Inventory (**center_id**, **manager_id**, **spare_part_id**, spare_part_name, spare_part_stock, spare_part_price)
  - Primary key: (centre_id ,manager_id, spare_part_id)
  - Foreign key: (centre_id ,manager_id)
  - Functional dependencies: (centre_id, manager_id, spare_part_id) → spare_part_name, (centre_id ,manager_id, spare_part_id) → spare_part_stock, (centre_id ,manager_id, spare_part_id) → spare_part_price

- Feedback (**feedback_id**, customer_id, order_id, visit_date, feedback)
  - Primary key: feedback_id
  - Foreign key: customer_id, order_id
  - Functional dependencies: feedback_id → customer_id, feedback_id → order_id, feedback_id → visit_date, feedback_id → feedback

5.3 Investigate every schema for the following:

❖ **List of redundancies existing for every schema which is part of the database (document it)**

Relation	Redundant Attribute	Reason
Payment	Customer_id	Payment relation has order_id as foriegn key, we can easily find customer_id using order_id. We will remove customer_id from the payment relation.

❖ **List of updates, delete, and insert anomalies for every schema (document it)**

Relation	Attribute causing Anomaly - Anomaly type	Reason
customer	contact - Delete	contact leads to deletion anomaly because it being a multivalued attribute, deletion of one contact number leads to deletion of the whole customer tuple.
employee	Contact - Delete	contact leads to deletion anomaly because it being a multivalued attribute, deletion of one contact number leads to deletion of the whole employee tuple.
manager	Contact - Delete	contact leads to deletion anomaly

5.4 Normalize the database up to 1NF (scalar values)

1NF: A relational schema R is in its 1st NF if the domains of all attributes of R are atomic.

- **Customer**
  - The attributes of this relation are: **customer_id**, customer_name, customer_address, contact_no, email
  - contact can be a multivalued attribute. Thus, it does not follow atomicity and the relation is not in 1NF form.
  - To make this relation in 1NF form, we will create a relation customer_contact (customer_id, contact) and remove contact attribute from the customer relation.
  - Note: We will take the address as a single input string which makes it indivisible.

- **Employee**
  - The attributes of this relation are: **employee_id**, employee_name, joining_date, work_start_time, work_end_time, center_id, employee_status, employee_contact, employee_email
  - contact can be a multivalued attribute. Thus, it does not follow atomicity and the relation is not in 1NF form.

- To make this relation in 1NF form, we will create a relation employee_contact (employee_id, contact) and remove contact attribute from the customer relation.
- Note: We will take the address as a single input string which makes it indivisible.

- **Services**
  - The attributes of this relation are: **service_id**, service_details, service_charges
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.

- **Offer**
  - The attributes of this relation are: **offer_id**, service_id, start_date, end_date, discount_percentage, max_discount
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.

- **Order**
  - The attributes of this relation are: **order_id**, customer_id, employee_id, service_id, vehicle_id, order_charges, order_status
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.

- **Scheduling Services**
  - The attributes of this relation are: **order_id**, **employee_id**, service_details, pickup_time, drop_time, pickup_location
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.

- **Notification**
  - The attributes of this relation are: **order_id**, **customer_id**
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.

- **Vehicle**
  - The attributes of this relation are: **vehicle_id**, vehicle_license_plate, customer_id, vehicle_type, vehicle_details, model_id
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.

- **Payment**
  - The attributes of this relation are: **payment_id**, customer_id, payment_mode, order_id, vehicle_id, payment_date
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.

- **Service Centre**
  - The attributes of this relation are: **center_id**, center_address, center_name
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.
  - Note: We will take the address as a single input string which makes it indivisible

- **Manager**
  - The attributes of this relation are: **manager_id**, manager_name, manager_contact, manager_email, center_id
  - contact can be a multivalued attribute. Thus, it does not follow atomicity and the relation is not in 1NF form.

- To make this relation in 1NF form, we will create a relation manager_contact (manager_id, contact) and remove contact attribute from the customer relation.

- **Inventory**
  - The attributes of this relation are: **center_id**, **manager_id**, spare_part_id, spare_art_name, spare_part_stock, spare_part_price
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.

- **Feedback**
  - The attributes of this relation are: **feedback_id**, customer_id, order_id, visit_date, feedback
  - There are no composite or multivalued attributes, hence they are all indivisible. As a result, all attributes are atomic, and the relationship is in 1NF form.

5.5 Normalize the database further to 2NF/3NF/BCNF (Remove Partial Dependencies)

- **Customer:**
  - Updated FD: customer_id → customer_name, customer_id → customer_address, customer_id →email
  - All non-primary attributes are fully functionally dependent on customer_id and thus this relation is in 2NF form.
  - No transitive dependency exists in the customer relation hence this relation is in 3NF.
  - Already in BCNF

- **Employee:**
  - Updated FD: employee_id → centre_id, employee_id → employee_name, employee_id → joining_date, employee_id → work_start_time, employee_id → work_end_time, employee_id → employee_status, employee_id → employee_email
  - All non-primary attributes are fully functionally dependent on employee_id and thus this relation is in 2NF form.
  - No transitive dependency exists in employee relation hence this relation is in 3NF.
  - Already in BCNF

- **Services:**
  - Updated FD: service_id → service_details, service_id → service_charges, service_id → offer_id
  - All non-primary attributes are fully functionally dependent on service_id and thus this relation is in 2NF form.
  - No transitive dependency exists in the services relation hence this relation is in 3NF.
  - Already in BCNF

- **Offers:**
  - Updated FD: offer_id → service_id, offer_id → start_date, offer_id →end_date, offer_id → discount_percetage, offer_id → max_discount
  - All non-primary attributes are fully functionally dependent on offer_id and thus this relation is in 2NF form.
  - No transitive dependency exists in offers relation hence this relation is in 3NF.
  - Already in BCNF

- **Order:**
  - Updated FD: order_id → employee_id, order_id → customer_id, order_id → service_id, order_id → vehicle_id, order_id →order_chardes, order_id → order_status
  - All non-primary attributes are fully functionally dependent on order_id and thus this relation is in 2NF form.
  - No transitive dependency exists in order relation hence this relation is in 3NF.
  - Already in BCNF

- **Scheduling services:**
  - Updated FD: (order_id, employee_id) → service_details, (order_id, employee_id) → pickup_time, (order_id, employee_id) → drop_time, (order_id, employee_id) → pickup_location
  - All non-primary attributes are fully functionally dependent on order_id and thus this relation is in 2NF form.
  - No transitive dependency exists in the scheduling_services relation hence this relation is in 3NF.
  - Already in BCNF

- **Notifications:**
  - Updated FD: (order_id, customer_id) → customer_id
  - All non-primary attributes are fully functionally dependent on order_id and thus this relation is in 2NF form.
  - No transitive dependency exists in the Notifications relation hence this relation is in 3NF.
  - Already in BCNF

- **Vehicle:**
  - Updated FD: vehicle_id → customer_id, vehicle_id → vehicle_license_plate, vehicle_id → vehicle_type, vehicle_id → vehicle_details, vehicle_id → model_id
  - All non-primary attributes are fully functionally dependent on vehicle_id and thus this relation is in 2NF form.
  - No transitive dependency exists in the vehicle relation hence this relation is in 3NF.
  - Already in BCNF

- **Payments**:
  - Updated FD: payment_id → customer_id, payment_id → order_id, payment_id → vehicle_id, payment_id → payment_mode, payment_id → payment_date
  - All non-primary attributes are fully functionally dependent on payment_id and thus this relation is in 2NF form.
  - No transitive dependency exists in the payments relation hence this relation is in 3NF.
  - Already in BCNF

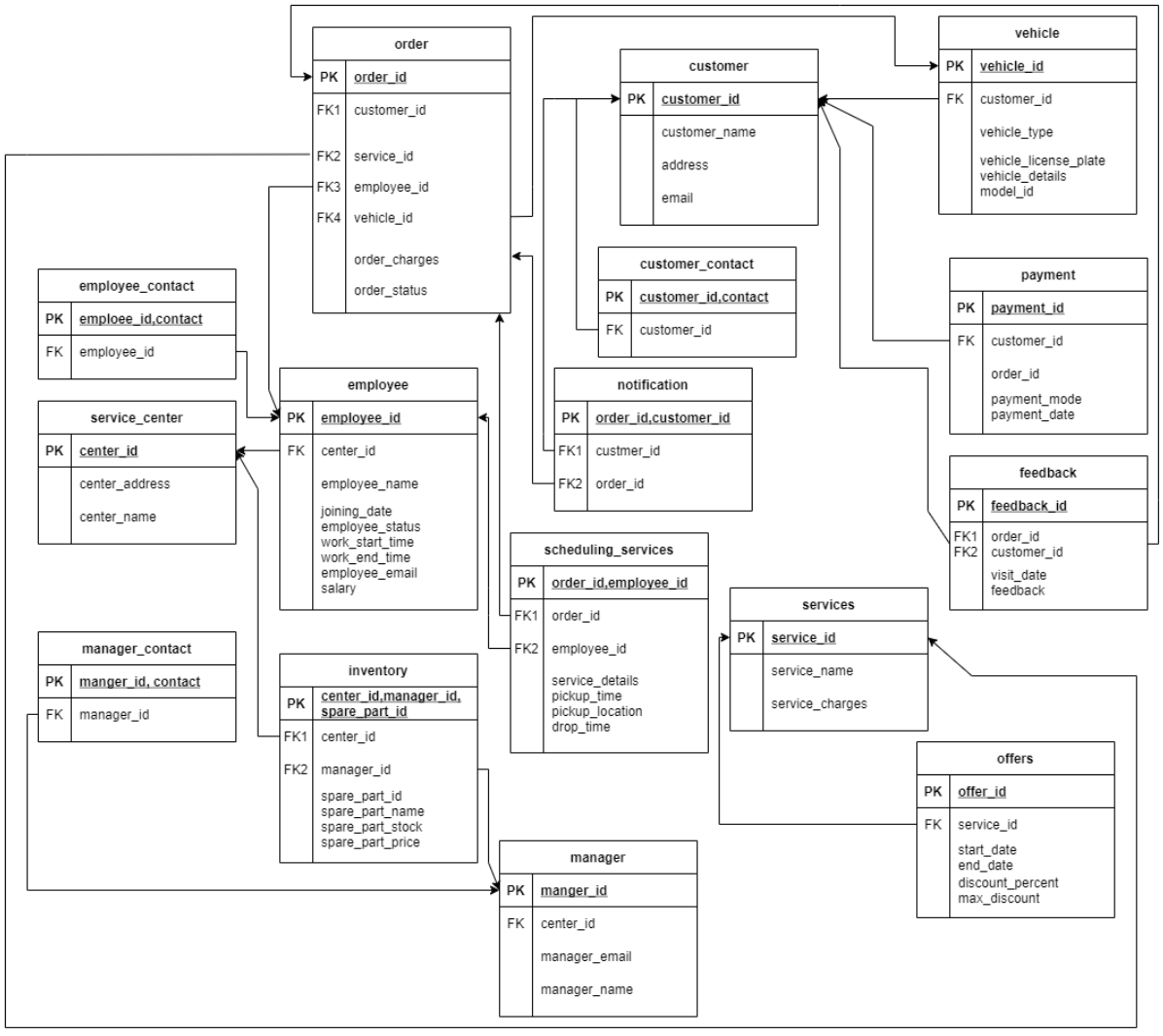
- **Service Centre**:
  - Updated FD: centre_id → centre_address, centre_id → centre_name
  - All non-primary attributes are fully functionally dependent on center_id and thus this relation is in 2NF form.
  - No transitive dependency exists in the Service Center relation hence this relation is in 3NF.
  - Already in BCNF

- **Manager:**
  - Updated FD: manager_id → centr_id, manager_id → manager_name, manager_id → manager_email
  - All non-primary attributes are fully functionally dependent on manager_id and thus this relation is in 2NF form.
  - No transitive dependency exists in the manager relation hence this relation is in 3NF.
  - Already in BCNF

- **Inventory:**
  - Updated FD: (centre_id ,manager_id) → spart_id, (centre_id ,manager_id) → spare_part_name, (centre_id ,manager_id) → spare_part_stock, (centre_id ,manager_id) → spare_part_price
  - All non-primary attributes are fully functionally dependent on (part_id, center_id) and thus this relation is in 2NF form.
  - No transitive dependency exists in the inventory relation hence this relation is in 3NF.
  - Already in BCNF

- **Feedback:**
  - Updated FD: feedback_id → customer_id, feedback_id → order_id, feedback_id → visit_date, feedback_id → feedback
  - All non-primary attributes are fully functionally dependent on feedback_id and thus this relation is in 2NF form.
  - No transitive dependency exists in the feedback relation hence this relation is in 3NF.
  - Already in BCNF

5.6 Final Relational Model



5.7 Final Relational Schema

- Customer (**customer_id**, customer_name, customer_address, email)
- Employee (**employee_id**, employee_name, joining_date, work_start_time, work_end_time, center_id, employee_status, employee_email)
- Services (**service_id**, service_name, service_charges)
- Offers (**offer_id**, service_id, start_date, end_date, discount_percentage, max_discount)
- Order (**order_id**, customer_id, employee_id, service_id, vehicle_id, order_charges, order_status)
- Scheduling services (**order_id**, **employee_id**, service_name, pickup_time, drop_time, pickup_location)
- Notifications (**order_id**, **customer_id**)
- Vehicle (**vehicle_id**, vehicle_license_plate, customer_id, vehicle_type, vehicle_details, model_id)
- Payments (**payment_id**, customer_id, payment_mode, order_id, vehicle_id, payment_date)
- Service center (**center_id**, center_address, center_name)
- Manager (**manager_id**, manager_name, manager_email, center_id)
- Inventory (**center_id**, **manager_id**, spare_part_id, spare_part_stock, spare_part_price)
- Feedback (**feedback_id**, customer_id, order_id, visit_date, feedback)
- Customer_contact (**customer_id**, contact)
- Manager_contact (**manager_id**, contact)
- Employee_contact (**employee_id**, contact)

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

6.1 Final DDL Script

- **Customer:**

```
create table if not exists as_db.customer
(
customer_id bigint not null,
customer_name character varying(100),
customer_address character varying(100),
email character varying(100),
Primary key(customer_id)
);
```

- **Vehicles:**

```
create table if not exists as_db.vehicles
(
vehicle_id character varying(100) not null,
customer_id character varying(100),
vehicle_liscence_plate bigint,
vehicle_type bigint,
model_id character varying(100),
PRIMARY KEY(vehicle_id),
FOREIGN KEY (customer_id) REFERENCES as_db.customer (customer_id) ON UPDATE
CASCADE ON DELETE CASCADE
);
```

- **Services:**

```
create table if not exists as_db.services
(
service_id character varying(100) not null,
service_charges integer,
service_name character varying(100),
PRIMARY KEY(service_id)
);
```

- **Offes:**

```
create table if not exists as_db.offers
(
offer_id character varying(100) not null,
service_id character varying(100),
start_date date,
end_date date,
discount_percentage bigint,
max_discount bigint,
PRIMARY KEY(offer_id),
FOREIGN KEY(service_id) REFERENCES as_db.services (service_id) ON UPDATE
CASCADE ON DELETE CASCADE
);
```

- **Service_centre:**

```
create table if not exists as_db.service_center
(
center_id character varying(100) not null,
center_address character varying(100),
center_name character varying(100),
PRIMARY KEY(center_id)
);
```

- **Employee:**

```
create table if not exists as_db.employee
(
employee_id character varying(100) not null,
employee_name character varying(100),
joining_date date,
employee_email character varying(100),
work_start_time time,
work_end_time time,
center_id character varying(100),
employee_status integer,
employee_salary integer,
PRIMARY KEY(employee_id),
FOREIGN KEY(center_id) REFERENCES as_db.service_center (center_id) ON UPDATE
CASCADE ON DELETE CASCADE
);
```

- **Orders:**

```
create table if not exists as_db.orders
(
order_id character varying(100)  not null,
customer_id character varying(100) ,
employee_id character varying(100) ,
service_id character varying(100) ,
vehicle_id character varying(100) ,
order_charges integer,
order_status integer,
PRIMARY KEY(order_id),

FOREIGN KEY(service_id) REFERENCES as_db.services (service_id) ON
UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(customer_id) REFERENCES as_db.customer (customer_id) ON
UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(employee_id) REFERENCES as_db.employee (employee_id) ON
UPDATE CASCADE ON DELETE CASCADE
);
```

- **Payments:**

```
create table if not exists as_db.payments
(
payment_id character varying(100) not null,
customer_id character varying(100),
payment_mode integer,
order_id character varying(100),
vehicle_id character varying(100),
payment_date date,
PRIMARY KEY(payment_id),

FOREIGN KEY(order_id) REFERENCES as_db.orders (order_id) ON
UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(customer_id) REFERENCES as_db.customer (customer_id) ON
UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(vehicle_id) REFERENCES as_db.vehicles (vehicle_id) ON
UPDATE CASCADE ON DELETE CASCADE
);
```

- **Manager:**

```
create table if not exists as_db.manager
(
manager_id character varying(100) not null,
manager_name character varying(100),
manager_email character varying(100),
center_id character varying(100),
PRIMARY KEY(manager_id),
FOREIGN KEY(center_id) REFERENCES as_db.service_center (center_id) ON UPDATE
CASCADE ON DELETE CASCADE
);
```

- **Feedback:**

```
create table if not exists as_db.feedbacks
(
feedback_id character varying(100) not null,
customer_id character varying(100),
order_id character varying(100),
visit_date date,
feedback text,
PRIMARY KEY(feedback_id),
FOREIGN KEY(order_id) REFERENCES as_db.orders (order_id) ON
UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(customer_id) REFERENCES as_db.customer (customer_id) ON
UPDATE CASCADE ON DELETE CASCADE
);
```

- **Scheduling_sevices:**

```
create table if not exists as_db.scheduling_services
(
order_id character varying(100) not null,
employee_id character varying(100) not null,
pick_up_time time,
drop_time time,
pick_up_location character varying(100),
service_name character varying(100),
PRIMARY KEY(order_id, employee_id),
FOREIGN KEY(employee_id) REFERENCES as_db.employee (employee_id)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(order_id) REFERENCES as_db.orders (order_id)
ON UPDATE CASCADE ON DELETE CASCADE
);
```

- **Notifications:**

```
create table if not exists as_db.notifications
(
order_id character varying(100) not null,
customer_id character varying(100) not null,
PRIMARY KEY(order_id, customer_id), FOREIGN KEY(order_id) REFERENCES as_db.orders
(order_id)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(customer_id) REFERENCES as_db.customer (customer_id)
ON UPDATE CASCADE ON DELETE CASCADE
);
```

- **Inventory:**

```
create table if not exists as_db.inventory
(
center_id character varying(100) not null,
```

```
manager_id character varying(100) not null,
spare_part_id character varying(100),
spare_part_stock bigint,
spare_part_price bigint,
PRIMARY KEY(center_id, manager_id,spare_part_id),
FOREIGN KEY(center_id) REFERENCES as_db.service_center (center_id)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY(manager_id) REFERENCES as_db.manager (manager_id)
ON UPDATE CASCADE ON DELETE CASCADE
);
```

- **Customer_contact:**

```
create table if not exists as_db.customer_contact
(
customer_id character varying(100) not null,
contact character varying(100) not null,
primary key(customer_id,contact),
FOREIGN KEY (customer_id) REFERENCES as_db.customer (customer_id) ON UPDATE
CASCADE ON DELETE CASCADE
);
```

- **Employee_contact:**

```
create table if not exists as_db.employee_contact
(
employee_id bigint not null,
contact bigint not null,
primary key(employee_id,contact),
FOREIGN KEY (employee_id) REFERENCES as_db.employee (employee_id) ON UPDATE
CASCADE ON DELETE CASCADE
);
```

- **Manager_contact:**

```
create table if not exists as_db.manager_contact
(
manager_id character varying(100) not null,
contact character varying(100) not null,
primary key(manager_id,contact),
FOREIGN KEY (manager_id) REFERENCES as_db.manager (manager_id) ON UPDATE
CASCADE ON DELETE CASCADE
);
```

6.2 Primary Key Dependency (key constraints)

- Customer table - customer_id
- Vehicle table - vehicle_id
- Services table - service_id
- Offers table - offer_id
- Service center - center_id
- Employee table - employee_id
- Order table - order_id
- Payments table - payment_id
- Manager table - manager_id
- Feedback table - feedback_id
- Scheduling services table - (order_id, employee_id)

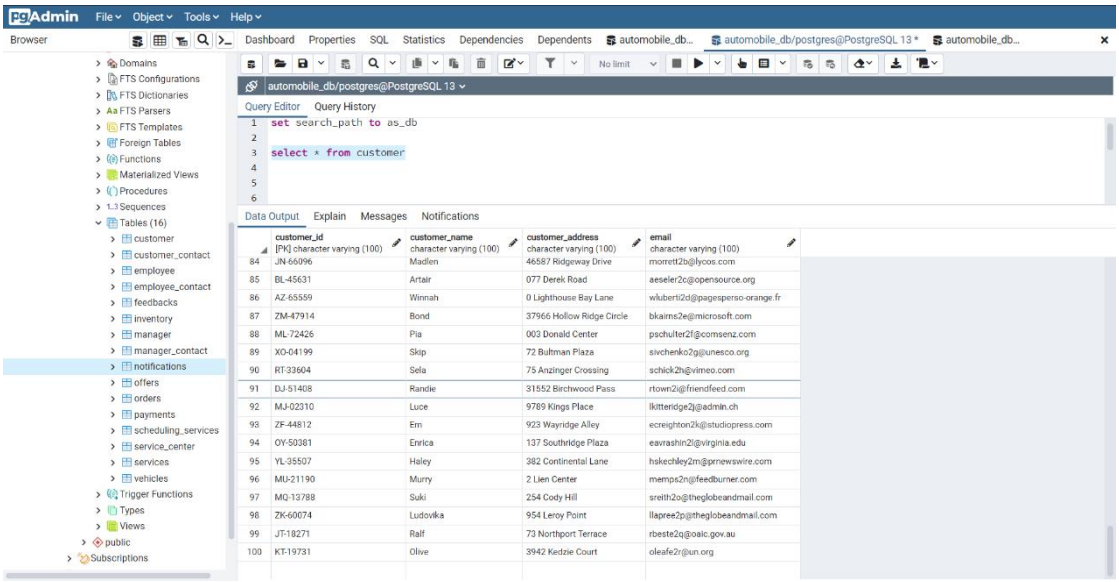
- Notifications table - (order_id, customer_id)
- Inventory table - (center_id, manager_id)
- Customer_contact – customer_id
- Manager_contact – manager_id
- Employee_id – employee_id

6.3 Foreign Key Dependencies (referential integrity constraints):

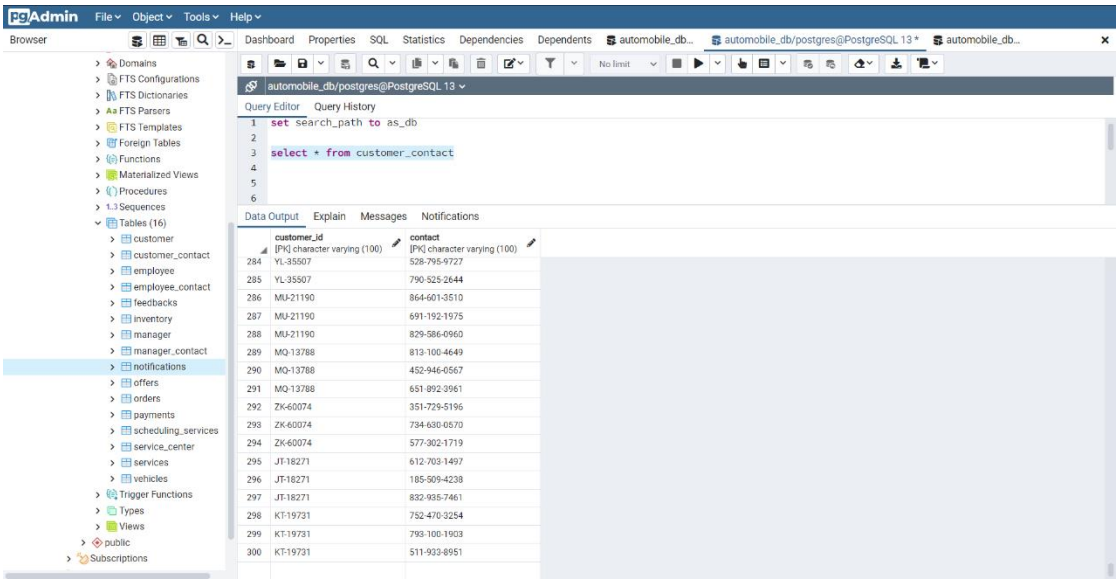
- **customer_id** references to **vehicle table**, **payment table**, **feedback table**, **order table** and **notifications table**.
- **service_id** references to **offers table** and **order table.**
- **order_id** references to **scheduling services table**, **payments table** and **notifications table**.
- **employee_id** references to **order table** and **scheduling services table**.
- **center_id** references to **employee table**, **manager table** and **inventory table**.
- **manager_id** references to **inventory table**.

6.4 Snapshot of all tables with data

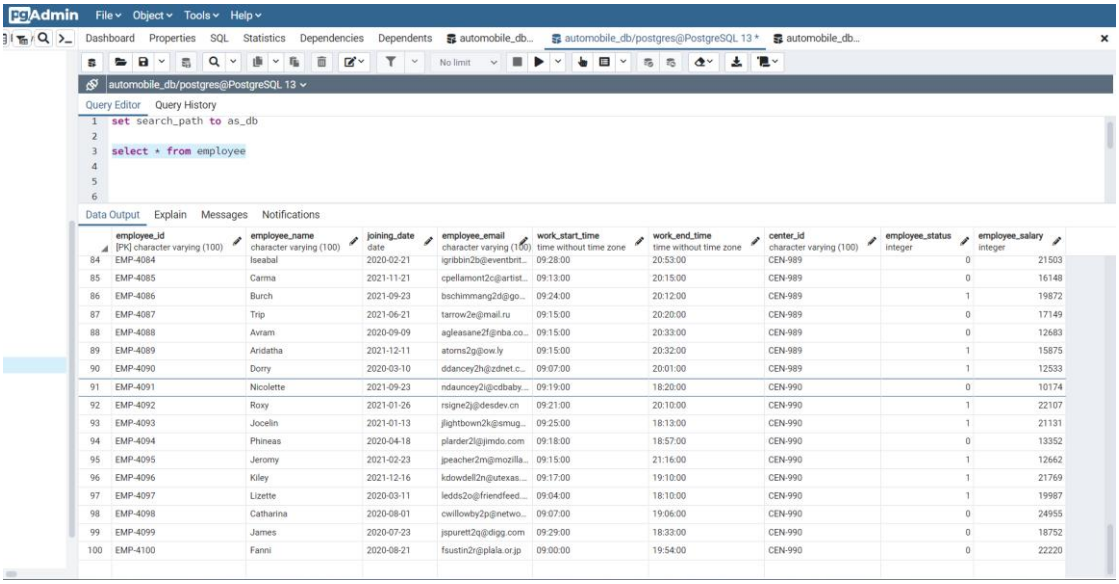
i. Customer



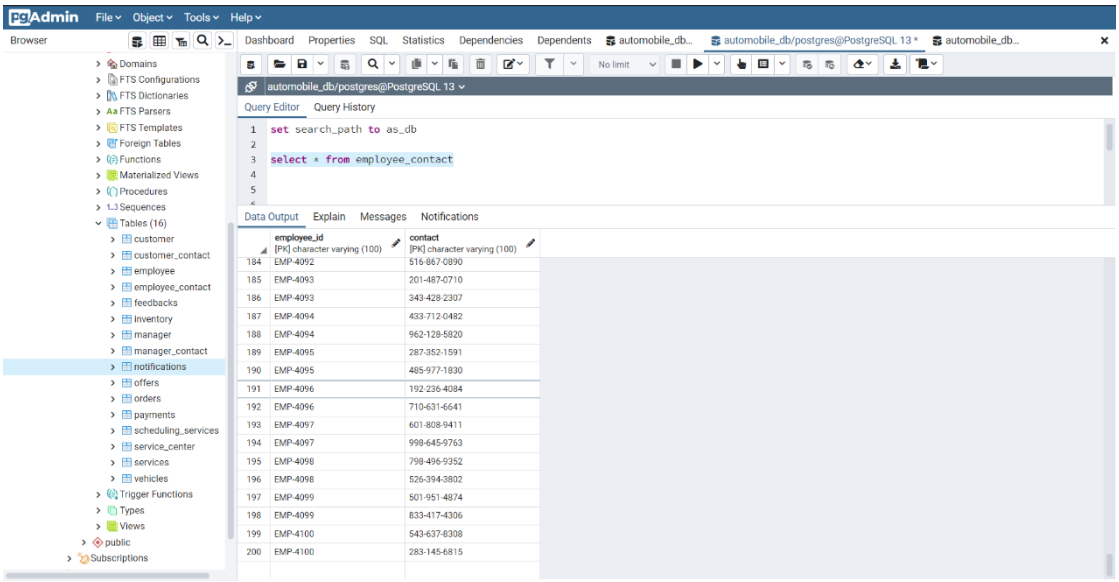
ii. Customer_contact



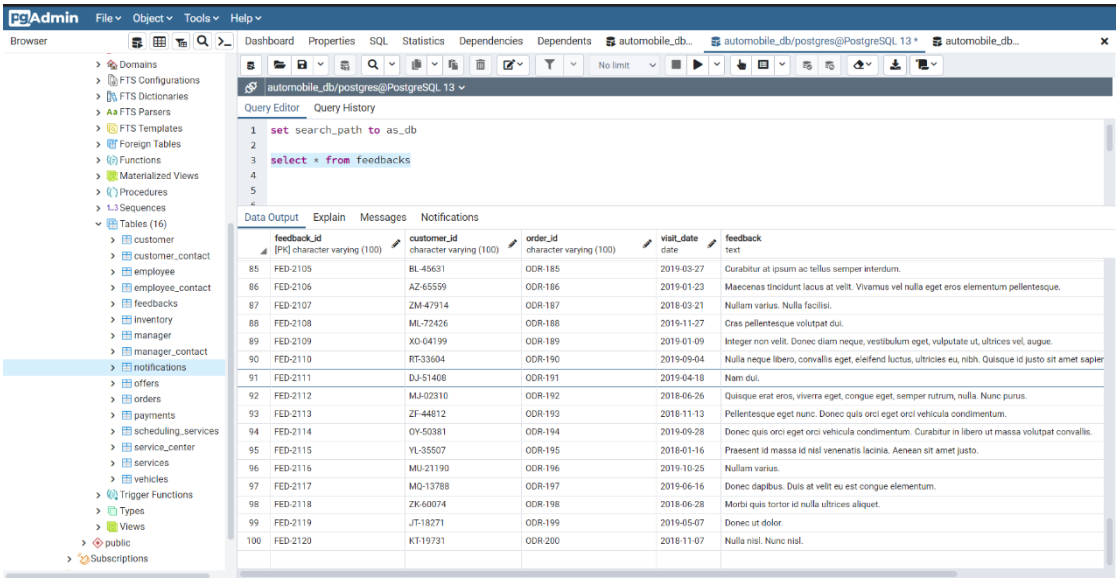
iii. Employee



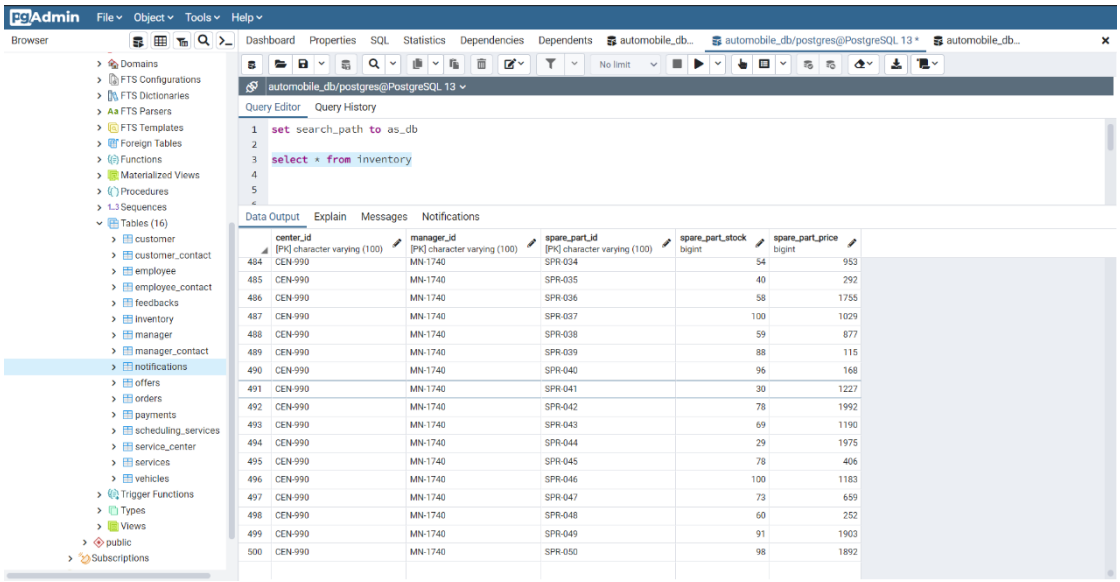
iv. Employee_contact



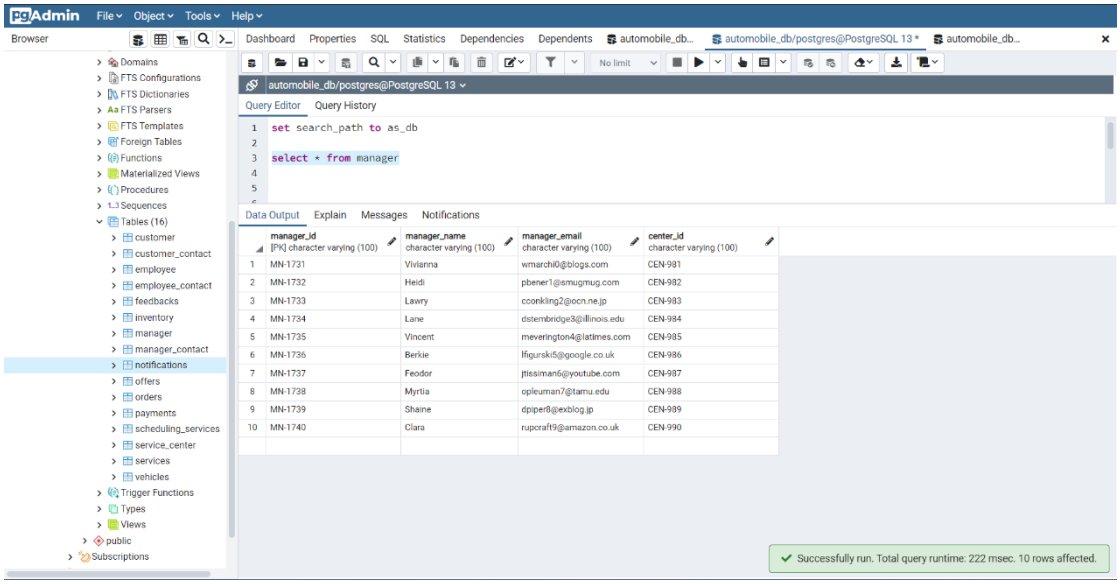
v. Feedback



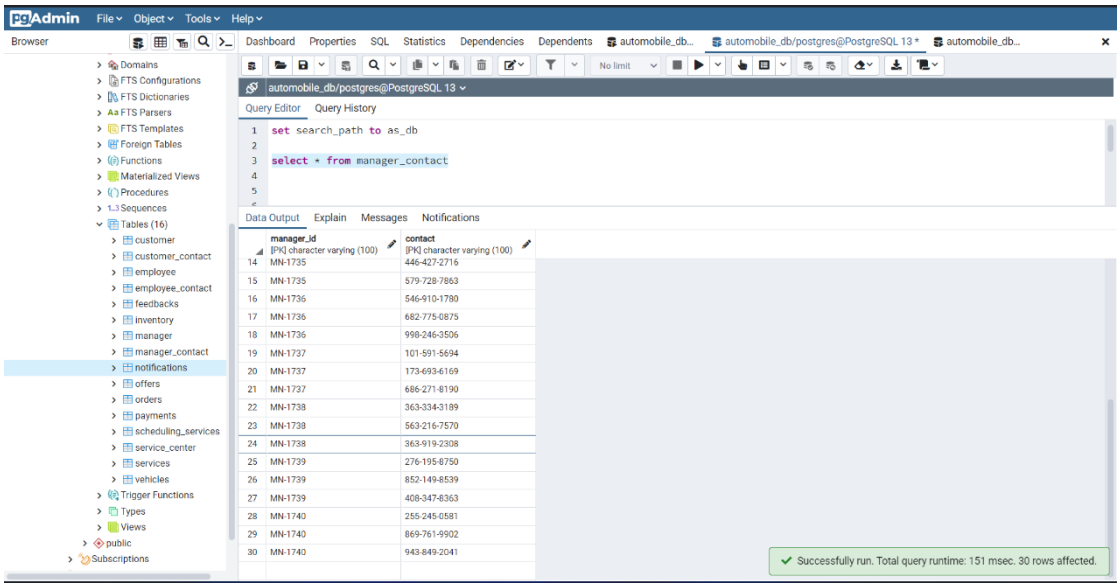
vi. Inventory



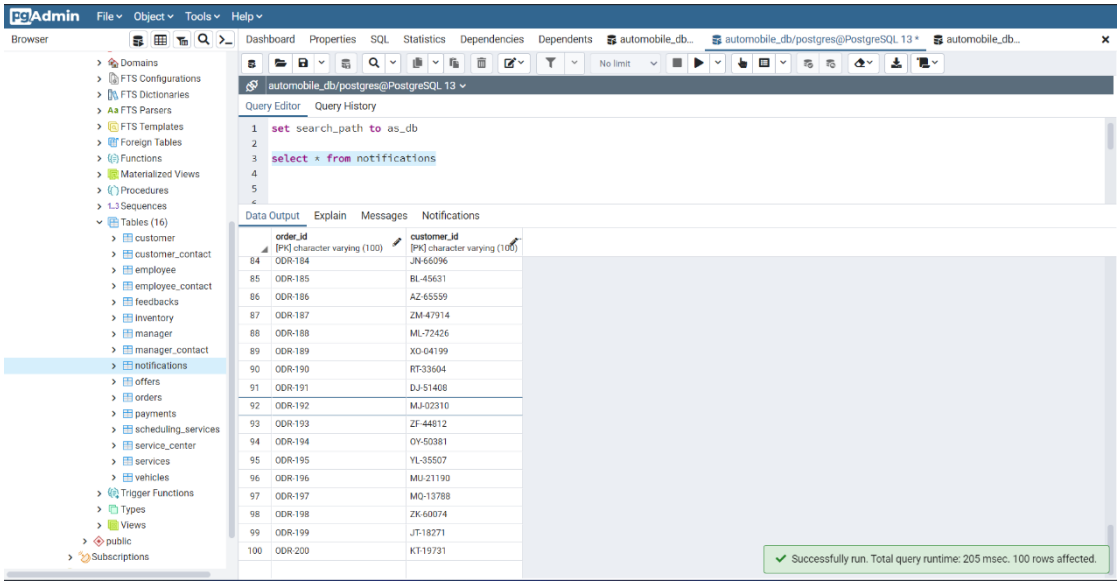
vii. Manager



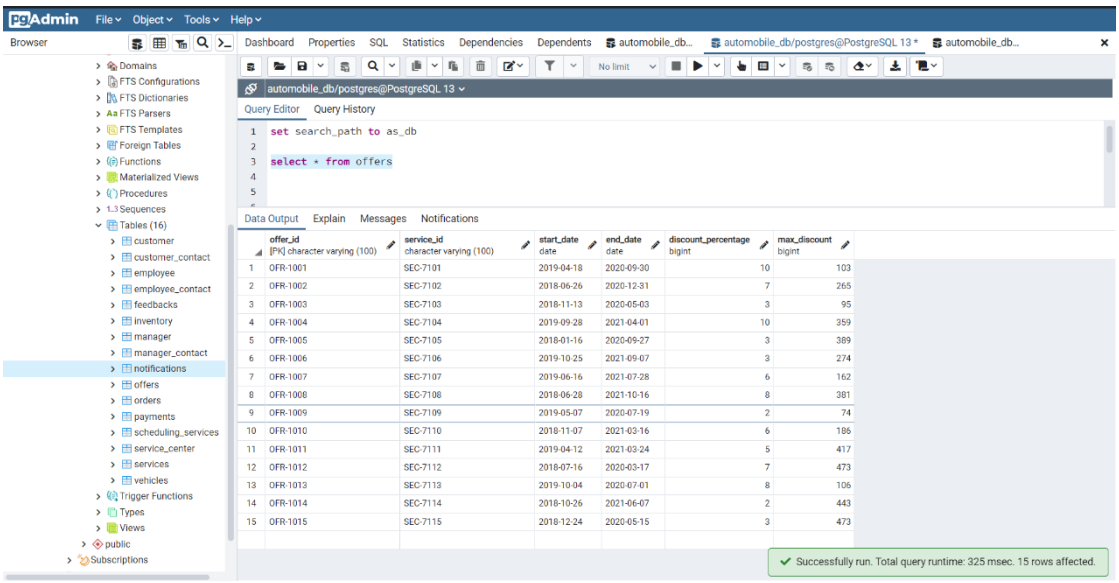
viii. Manager_contact



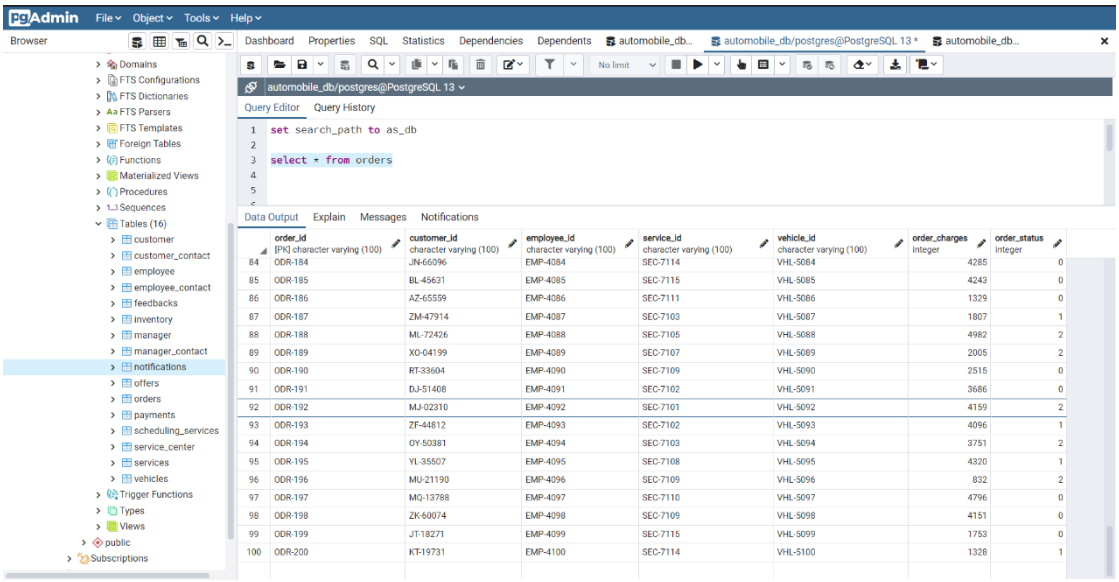
ix. Notification



x. Offers

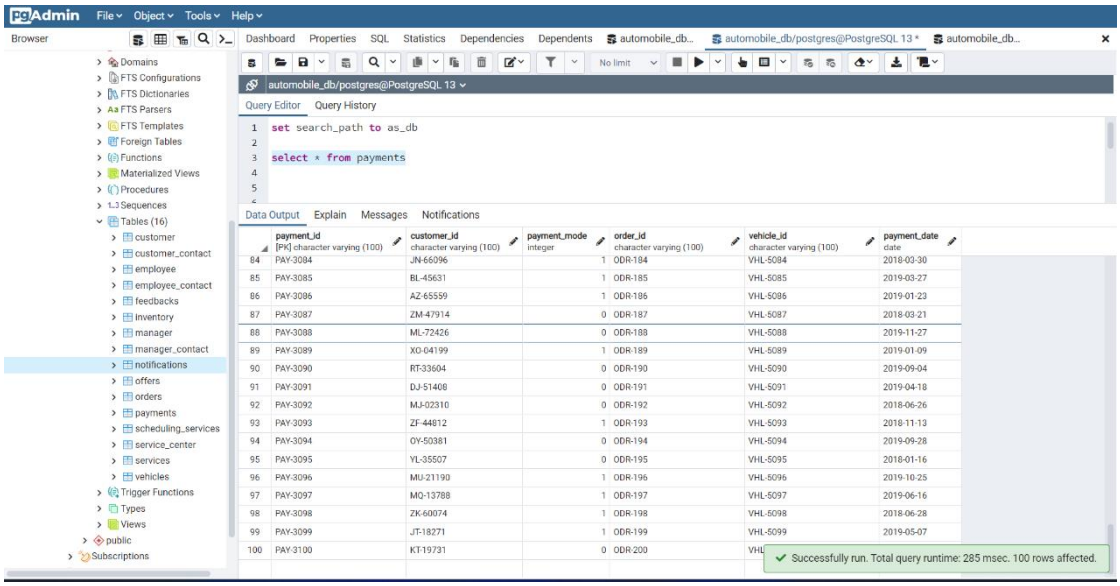


xi. Orders



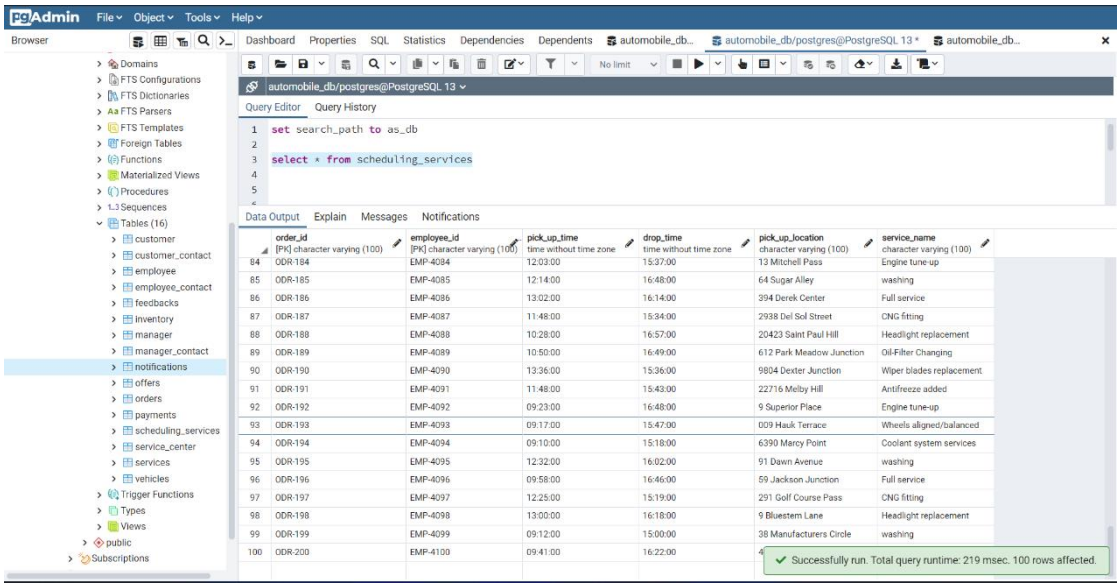
- **order_status:**
  - 0- pending
  - 1- in progress
  - 2- completed

xii. Payment

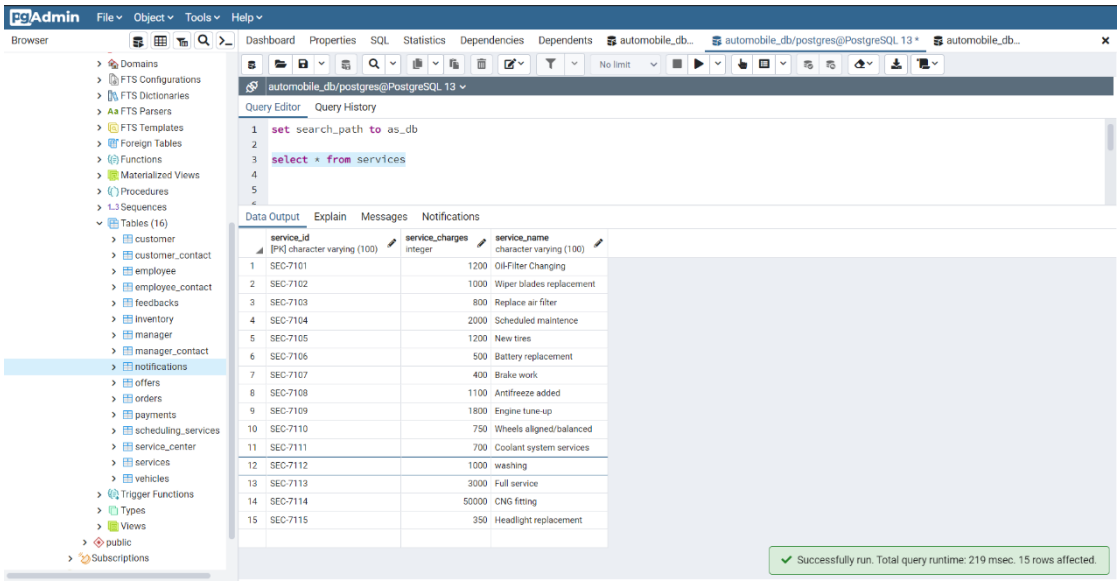


- **Payment_mode:**
  0- cash
  1- online

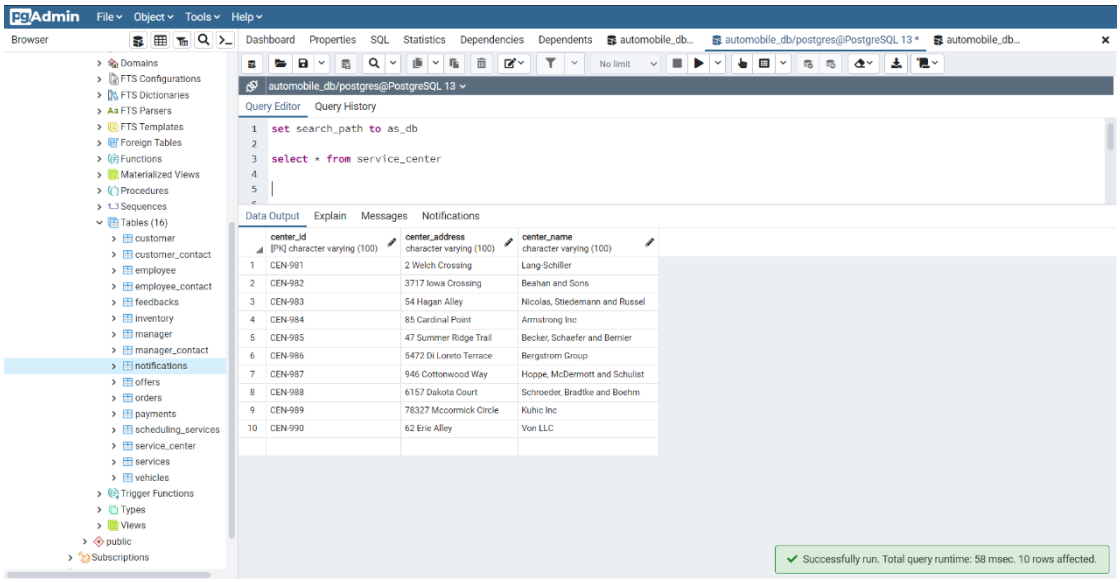
xiii. Scheduling_service



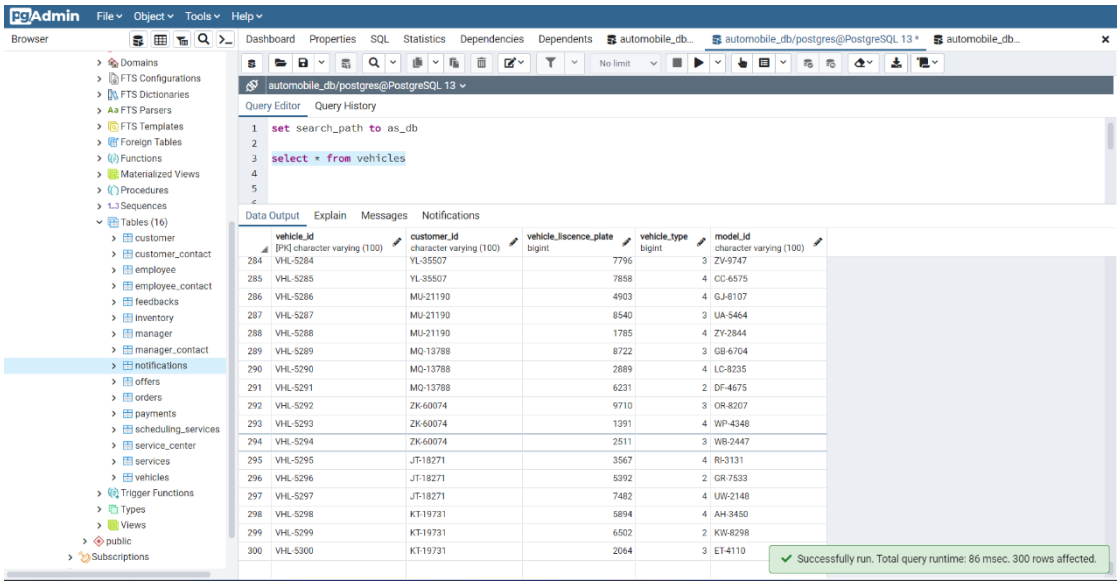
xiv. Services



xv. Service_centre



xvi. Vehicles

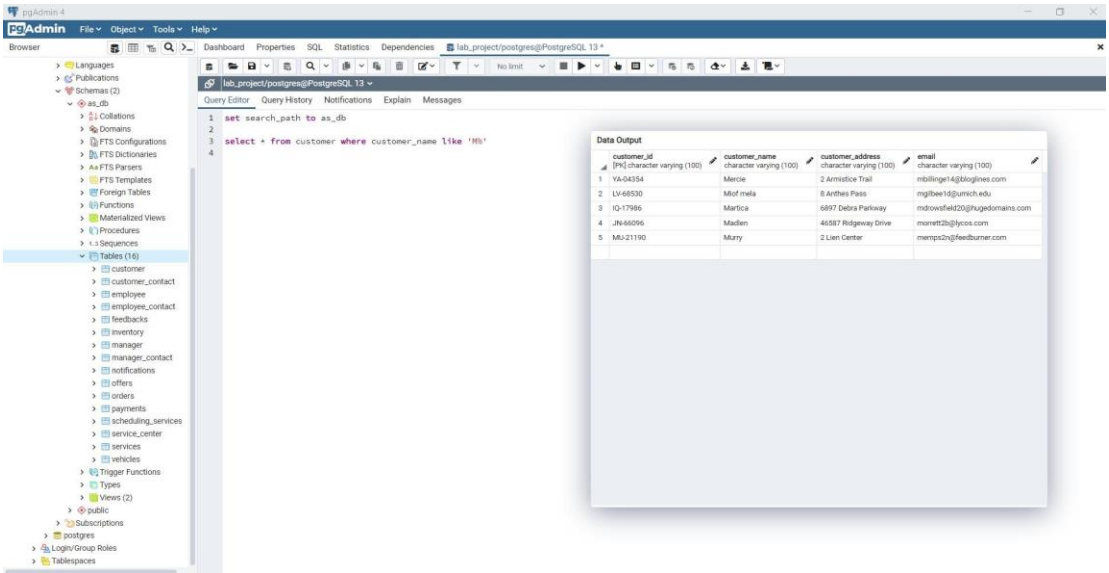


6.5 40 SQL Queries with Snapshots of output of each query

6.5.1 Running Simple Queries

1. Show the data of all customers whose name starts from M.

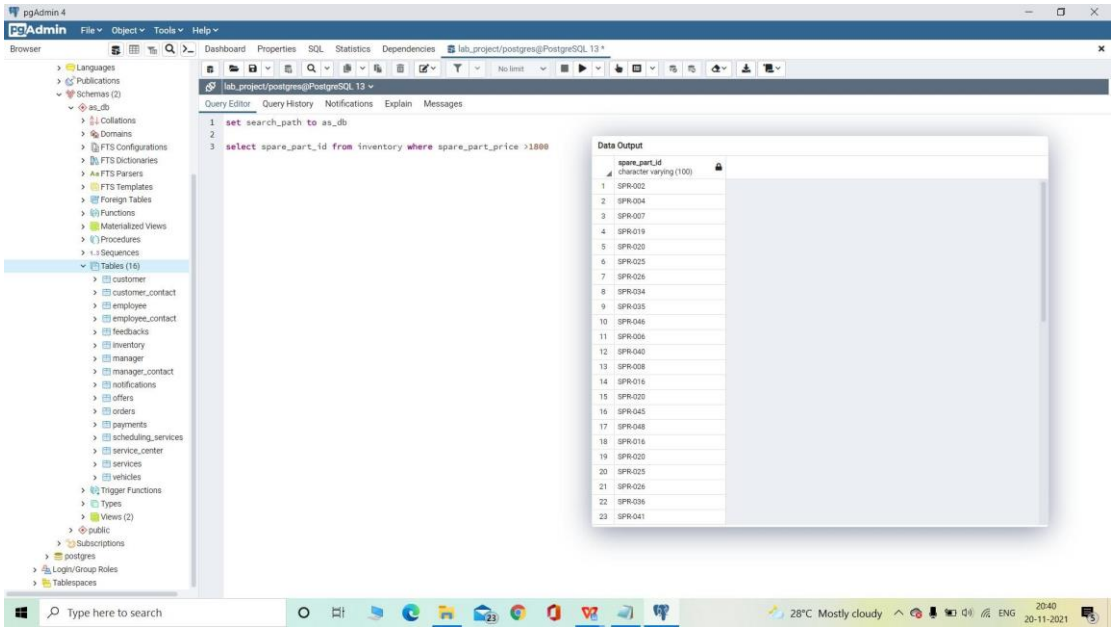
- **Query Command:** select * from customer where customer_name like 'M%'



- Tuples: 5

2. Select all spare parts with price greater than 1800 in inventory

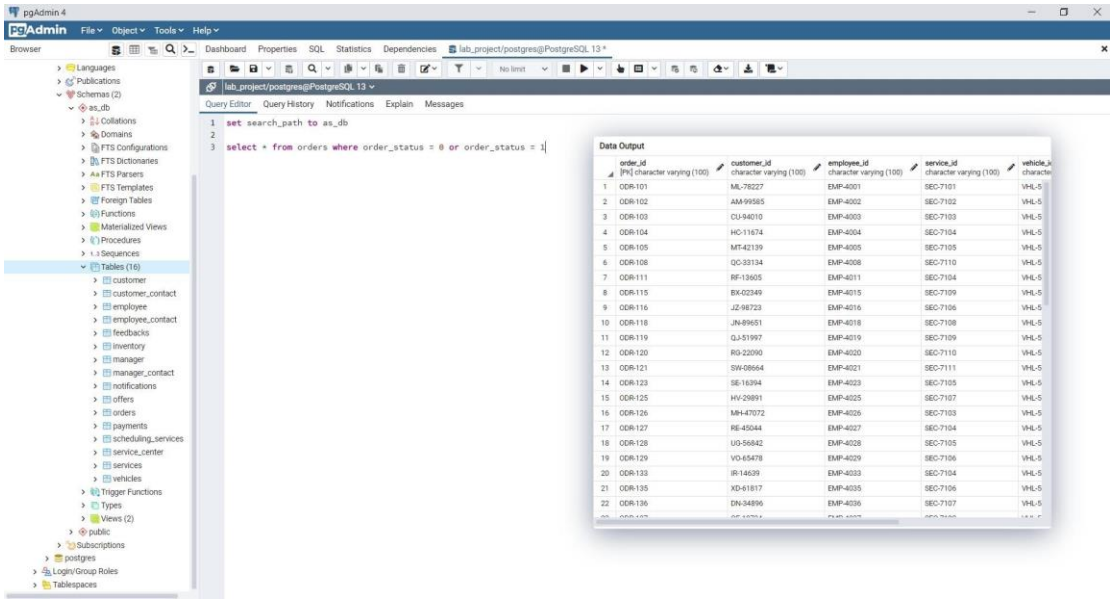
- **Query Command:** select spare_part_id from inventory where spare_part_price >1800



- Tuples: 55

3. Show data of uncompleted orders.

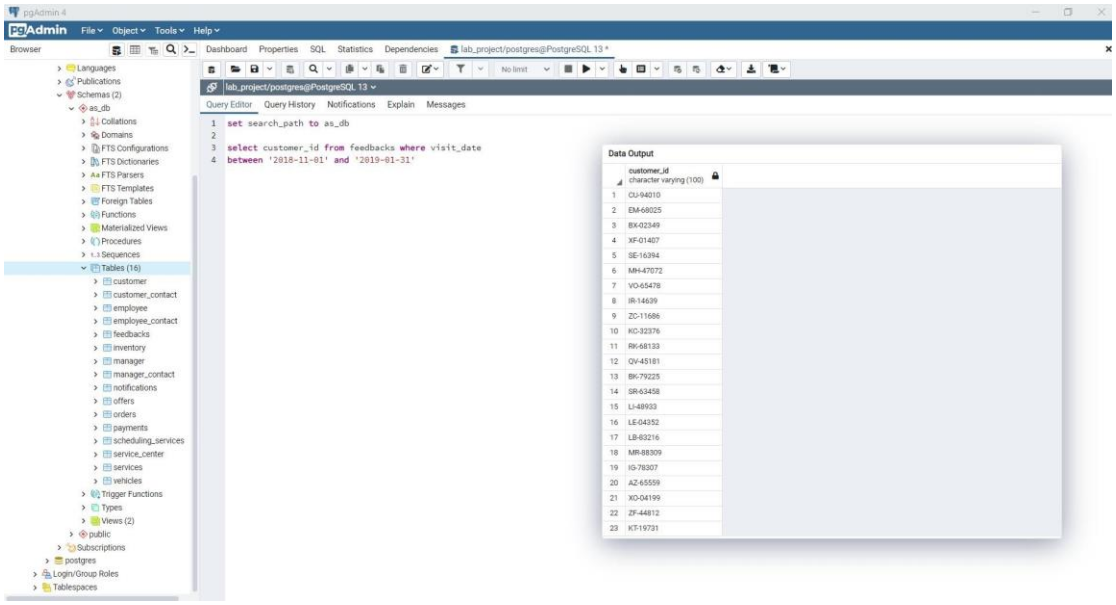
- **Query command:** select * from orders where order_status = 0 or order_status = 1



- Tuples: 58

4. Show the customer_id of all customers who have visited respective service centre between November 2018 to January 2019.

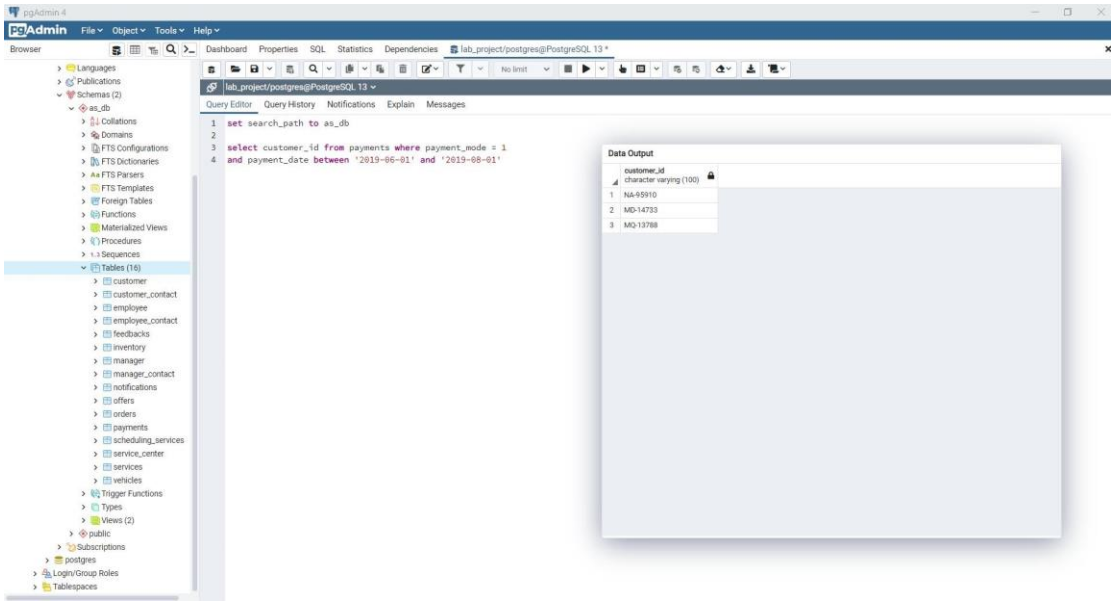
- **Query command:** select customer_id from feedbacks where visit_date between '2018-11-01' and '2019-01-31'



- Tuples: 23

5. List out all customers who has done online transactions during June and July of 2019 for the service given by service centre.

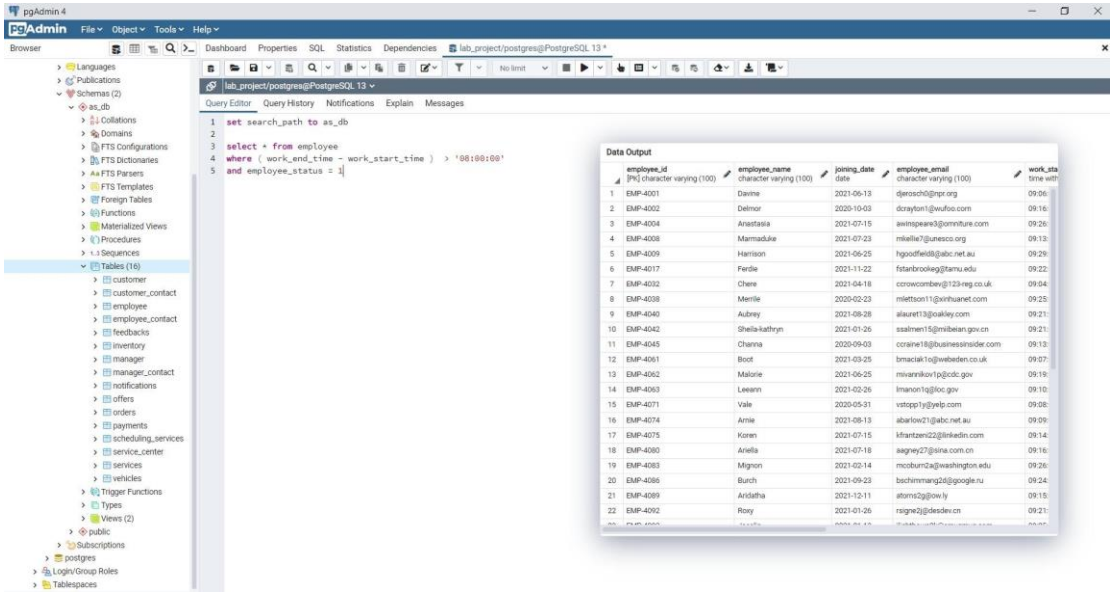
- **Query Command:** select customer_id from payments where payment_mode = 1 and payment_date between '2019-06-01' and '2019-08-01'



- Tuples: 3

6. Show data of all employees who are active and working more than 8 hours a day.

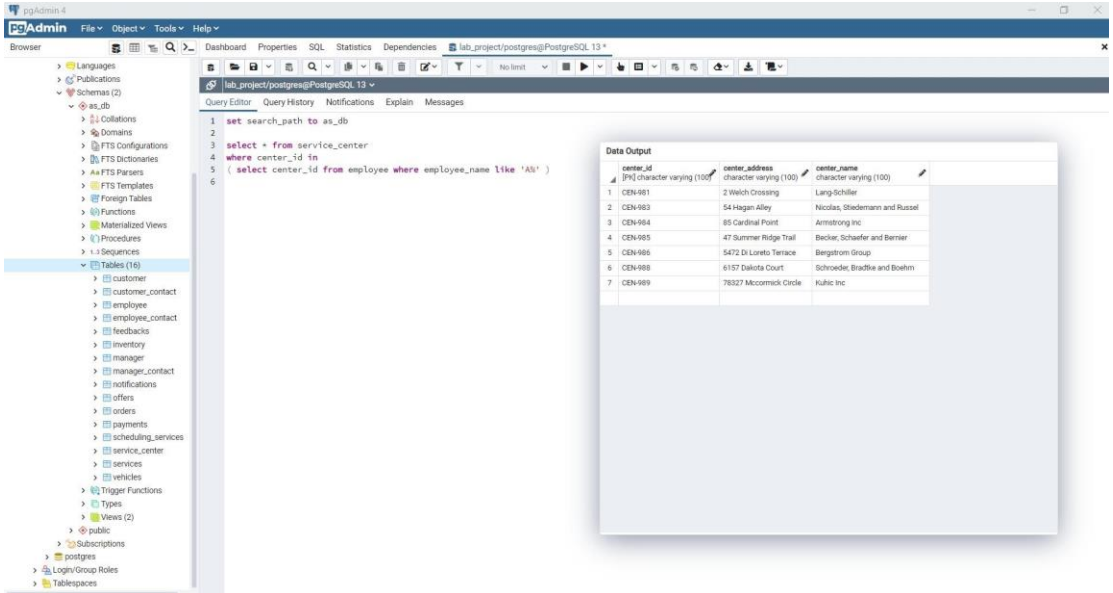
- **Query command:** select * from employee where (work_end_time work_start_time) > '08:00:00' and employee_status = 1



- Tuples: 40

7. Show the details of the service centre where at least one employee works with name starting with 'A'.

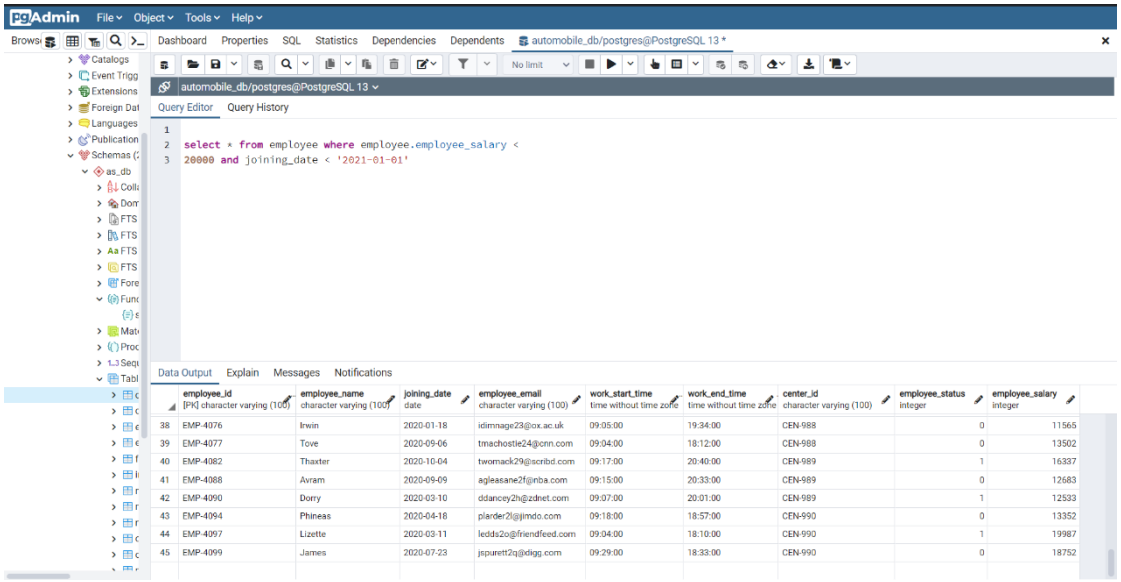
- **Query command:** select * from service_center where center_id in (select center_id from employee where employee_name like 'A%')



- Tuples: 7

8. Show details of all employees who have joined before 1st Jan 2021 and has salary less than 20000.

- **Query Command:** select * from employee where employee.employee_salary < 20000 and joining_date < '2021-01-01'

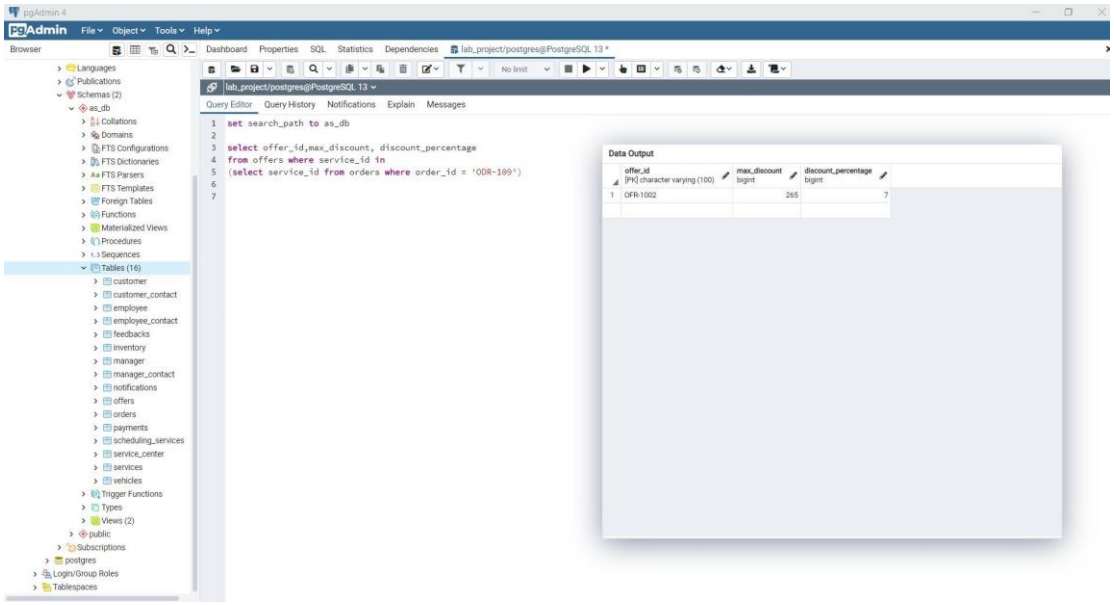


	employee_id [PK] character varying (100)	employee_name character varying (100)	joining_date date	employee_email character varying (100)	work_start_time time without time zone	work_end_time time without time zone	center_id character varying (100)	employee_status integer	employee_salary integer
38	EMP-4076	Irwin	2020-01-18	idimnage23@ox.ac.uk	09:05:00	19:34:00	CEN-988	0	11565
39	EMP-4077	Tove	2020-09-06	tmachostle24@cnn.com	09:04:00	18:12:00	CEN-988	0	13502
40	EMP-4082	Thaxter	2020-10-04	twomack29@scribd.com	09:17:00	20:40:00	CEN-989	1	16337
41	EMP-4088	Avram	2020-09-09	agleasane2f@inba.com	09:15:00	20:33:00	CEN-989	0	12683
42	EMP-4090	Dorry	2020-03-10	ddancey2h@zdnet.com	09:07:00	20:01:00	CEN-989	1	12533
43	EMP-4094	Phineas	2020-04-18	plarder2l@jimdo.com	09:18:00	18:57:00	CEN-990	0	13352
44	EMP-4097	Lizette	2020-03-11	ledds2o@friendfeed.com	09:04:00	18:10:00	CEN-990	1	19987
45	EMP-4099	James	2020-07-23	jspurett2q@digg.com	09:29:00	18:33:00	CEN-990	0	18752

- Tuples: 48

9. Show all offer related information for order_id ODR-109.

- **Query Command:** select offer_id,max_discount, discount_percentage from offers where service_id in (select service_id from orders where order_id = 'ODR-109')

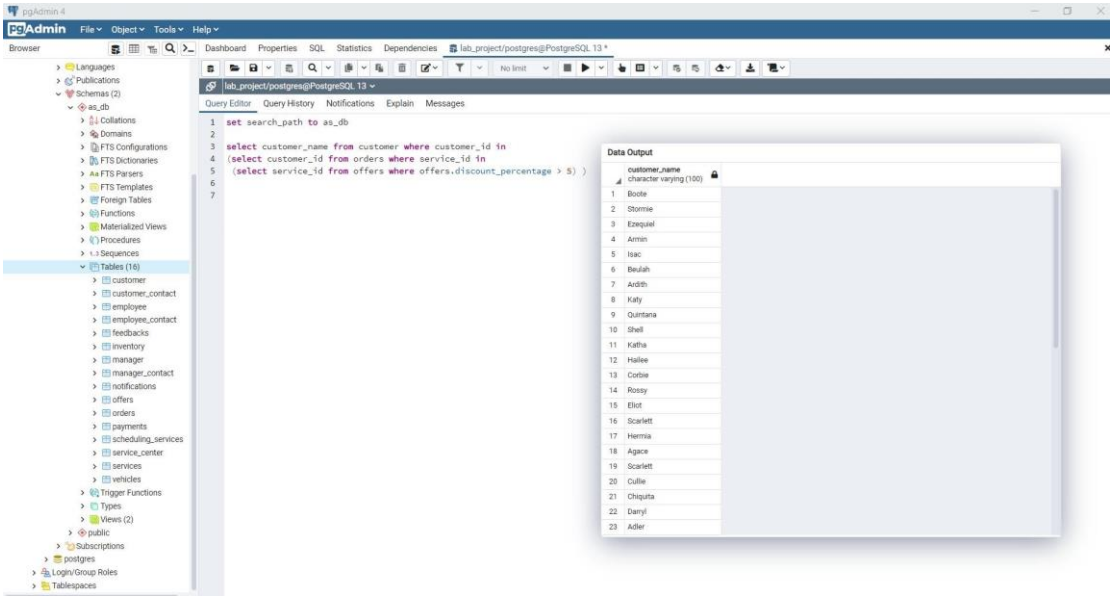


	offer_id [PK] character varying (100)	max_discount bigint	discount_percentage bigint
1	OFR-1002	265	7

- Tuples: 1

10. List out all customers who have got atleast 5% discount on their most recent order.

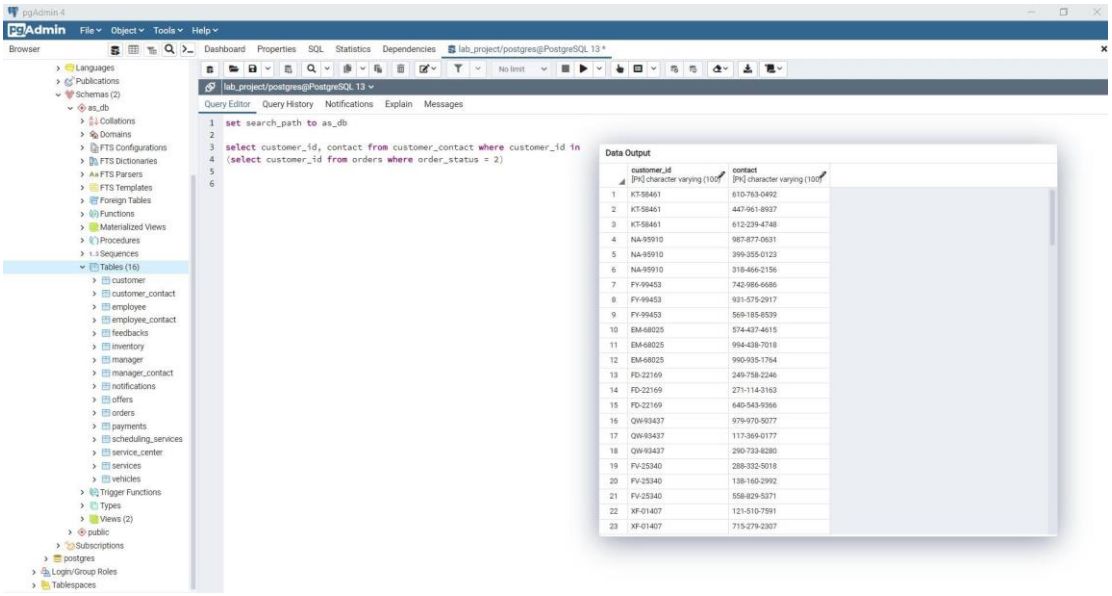
- **Query Command:** select customer_name from customer where customer_id in
  (select customer_id from orders where service_id in (select service_id from offers where offers.discount_percentage > 5))



- Tuples: 49

11. List out customers whose most recent order has been completed.

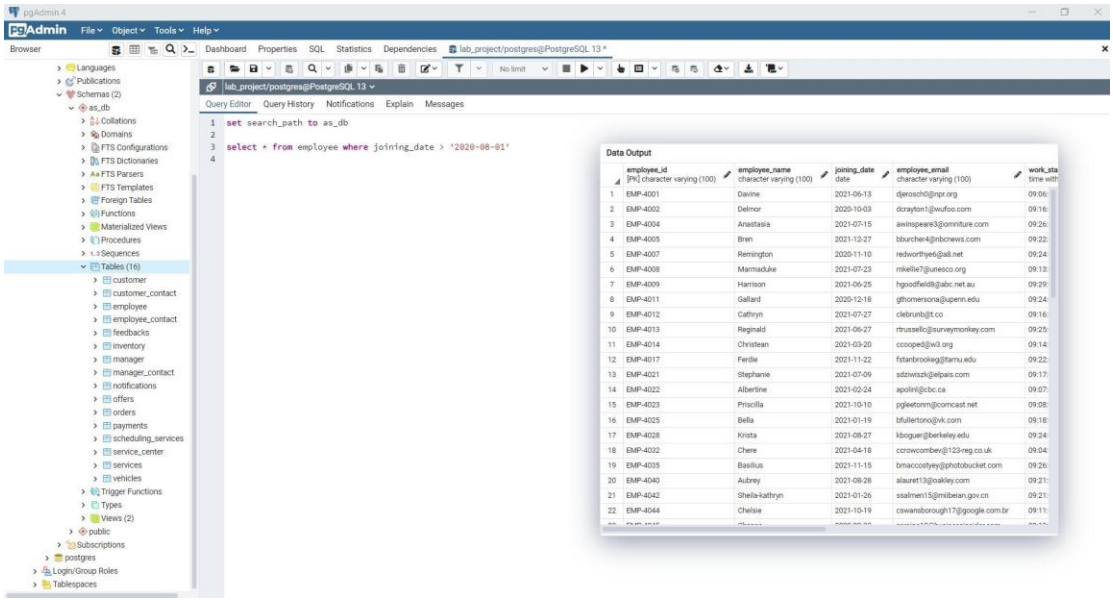
- **Query Command :** select customer_id, contact from customer_contact where customer_id in(select customer_id from orders where order_status = 2)



- Tuples : 132

12. Show details of all employees who have joined the service center since august 2020.

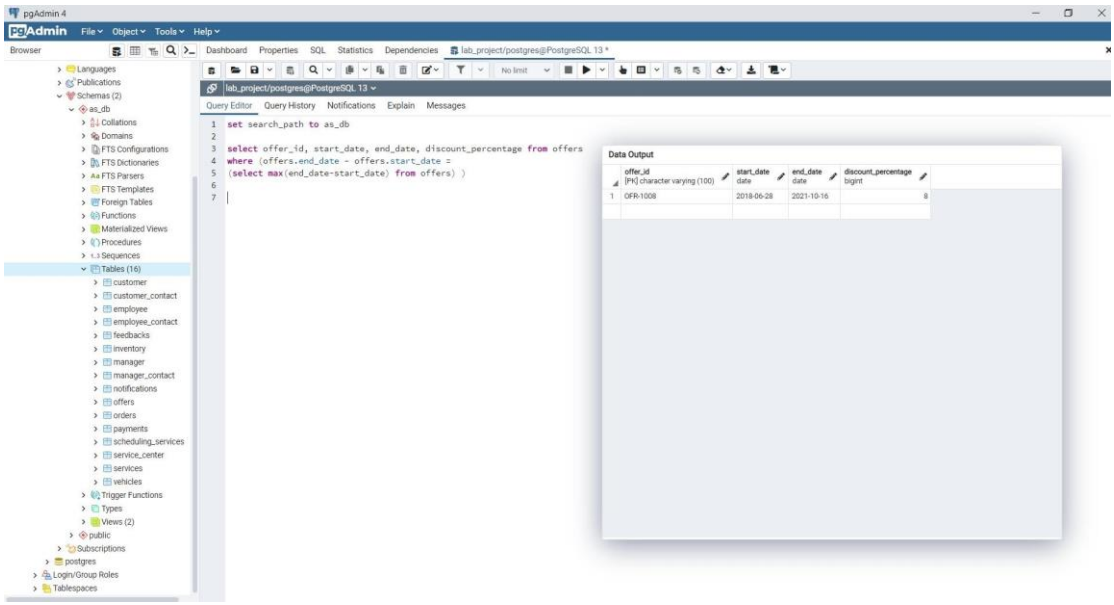
- **Query Command:** select * from employee where joining_date > '2020-08-01'



- Tuples: 66

13. Find discount percentage of the offer that is active for maximum duration.

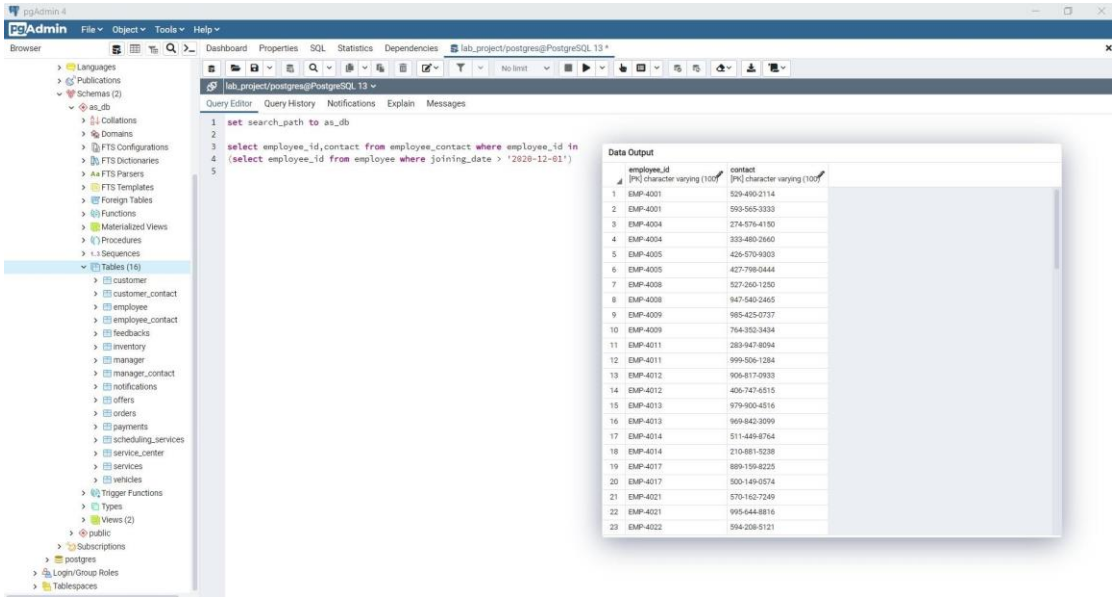
- **Query Command:** select offer_id, start_date, end_date, discount_percentage from offers where (offers.end_date - offers.start_date = (select max(end_datestart_date) from offers))



- Tuples: 1

14. Show contact numbers of all employee who have joined since December 2020.

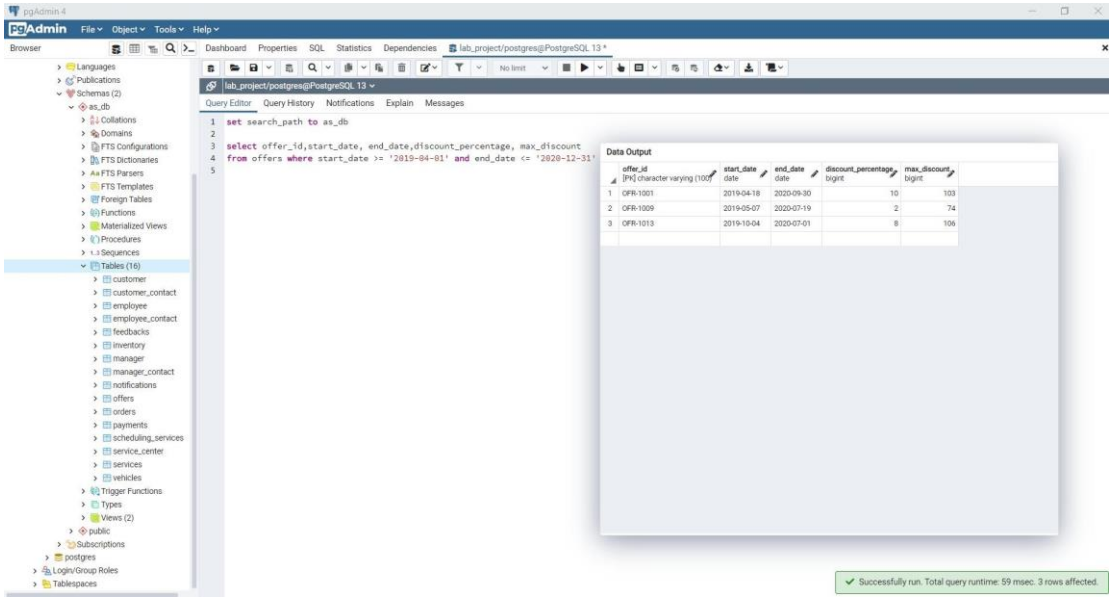
- **Query Command:** select employee_id,contact from employee_contact where employee_id in (select employee_id from employee where joining_date > '2020-12-01')



- Tuples: 94

15. Find all the offers which are active between 2019-04-01 and 2020-12-31

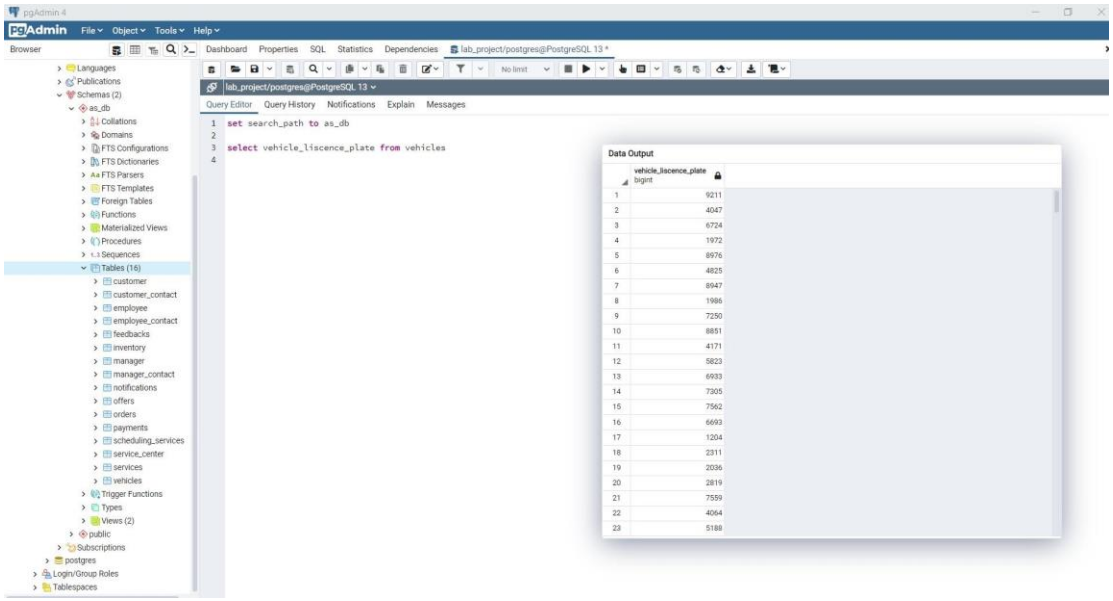
- **Query Command:** select offer_id,start_date, end_date,discount_percentage, max_discount from offers where start_date >= '2019-04-01' and end_date <= '2020-12-31'



- Tuples: 3

16. Show the license plate numbers of all vehicles ever visited the centre.

- **Query Command:** select vehicle_liscence_plate from vehicles

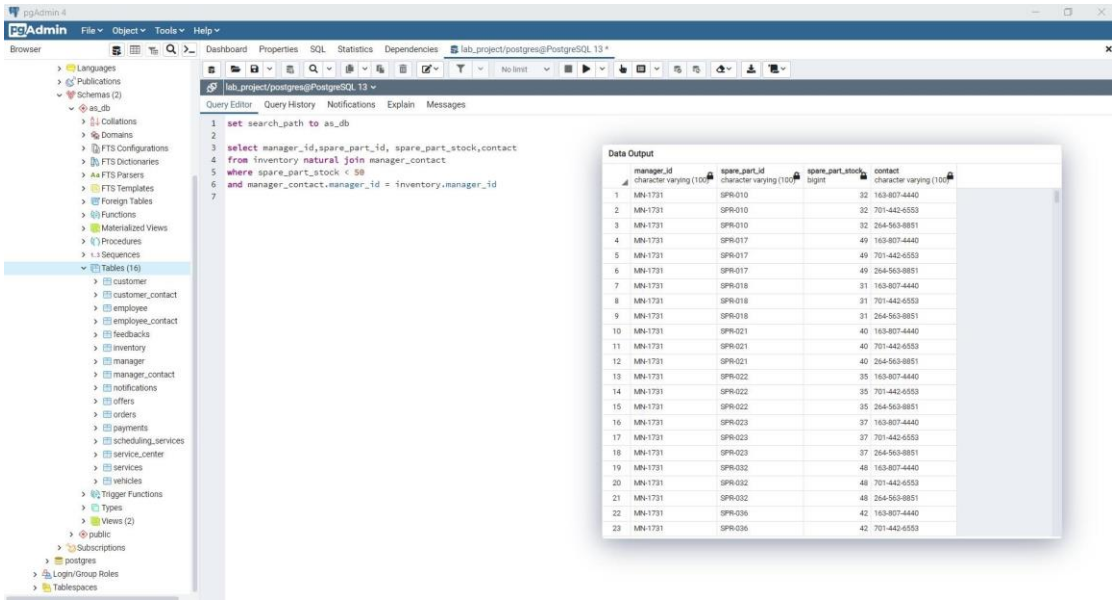


- Tuples: 300

6.5.2 Running Complex Queries.

1. Service center decides to never let the stock of any spare part to go beyond 50 , so this is the query to find all spare parts with stock less than 50 and then shows contact of the manager of the center to notify to order more parts.

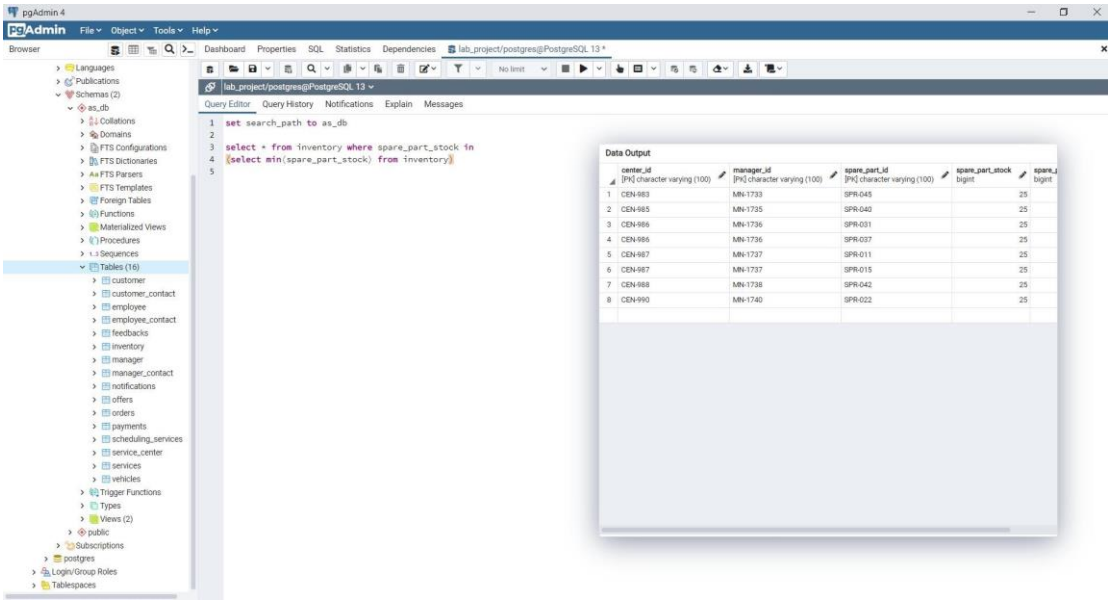
- **Query Command:** select manager_id,spare_part_id, spare_part_stock,contact from inventory natural join manager_contact where spare_part_stock < 50 and manager_contact.manager_id = inventory.manager_id



- Tuples: 480

2. Show the details of the spare part which is in lowest quantity in inventory.

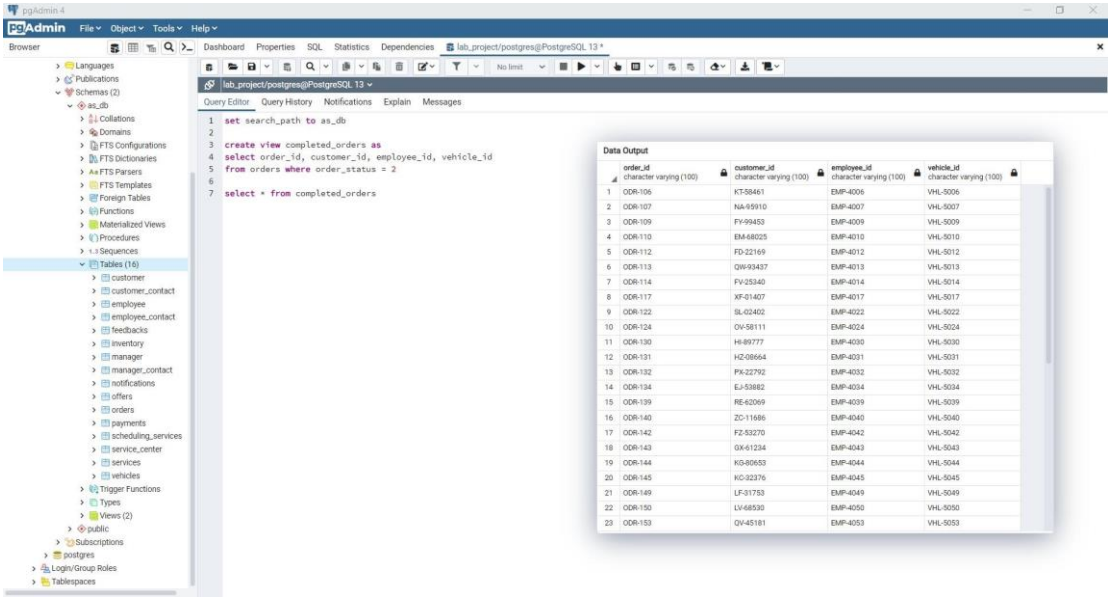
- **Query Command:** select * from inventory where spare_part_stock in (select min(spare_part_stock) from inventory)



- Tuples:8

3. Show all details for completed orders.

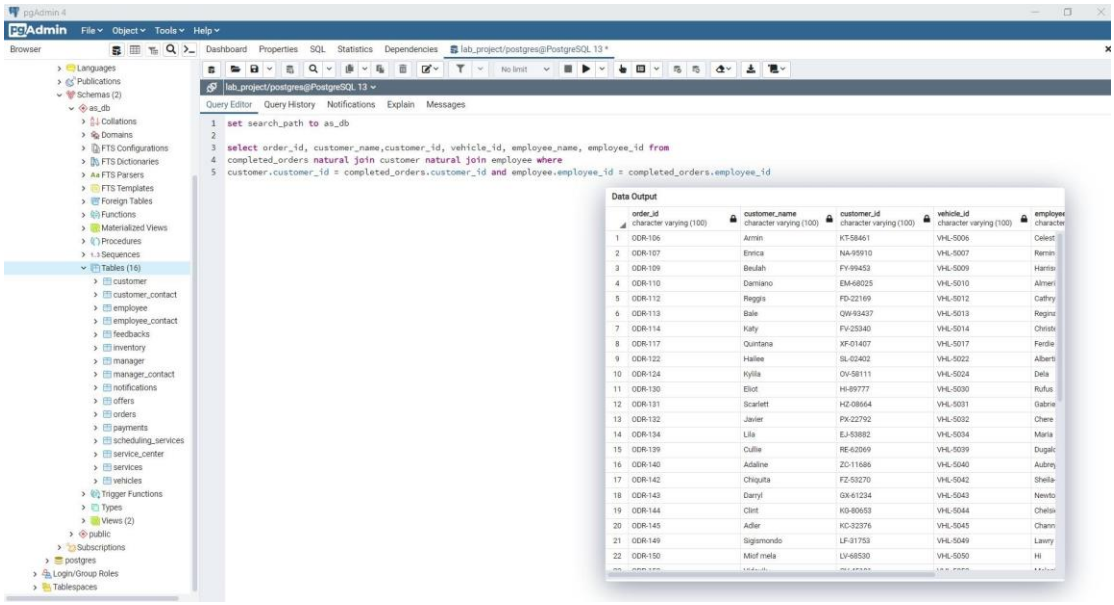
- **Query Command:** create view completed_orders as select order_id, customer_id, employee_id, vehicle_id from orders where order_status = 2 Select * from completed_orders



- Tuples: 44

4. Give all types of information related to every types of orders.

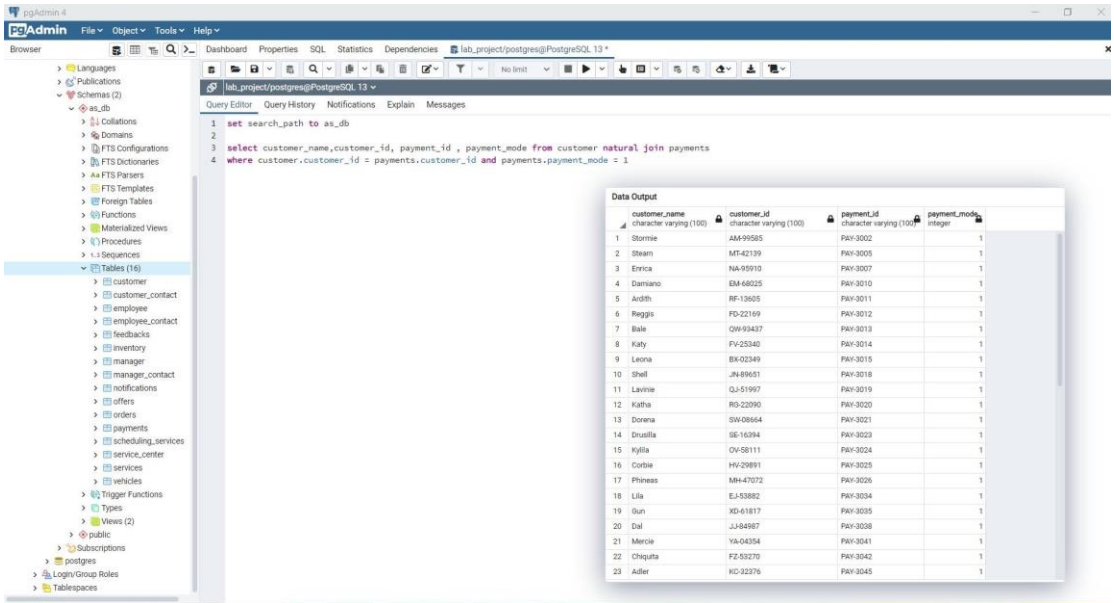
- **Query Command:** select order_id, customer_name,customer_id, vehicle_id, employee_name, employee_id from completed_orders natural join customer natural join employee where customer.customer_id=completed_orders.customer_id and employee.employee_id = completed_orders.employee_id



- Tuples: 44

5. Showing information of customers who have paid charges through online.

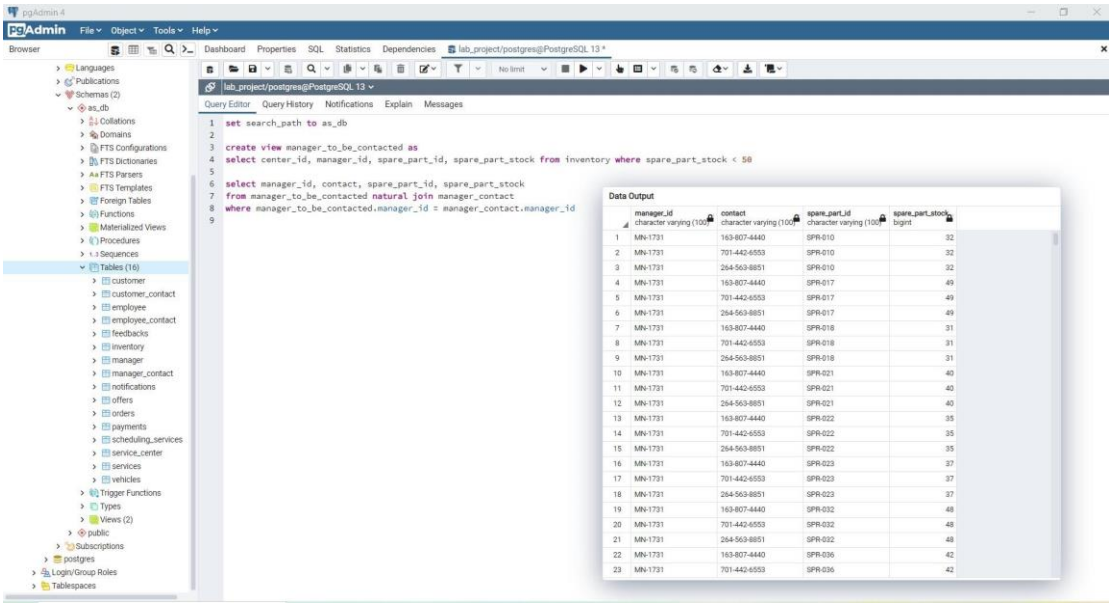
- Query command: select customer_name,customer_id, payment_id , payment_mode from customer natural join payments where
  customer.customer_id = payments.customer_id and payments.payment_mode = 1



- Tuples: 51

6. Giving manager notification if stock goes down for certain spare parts.

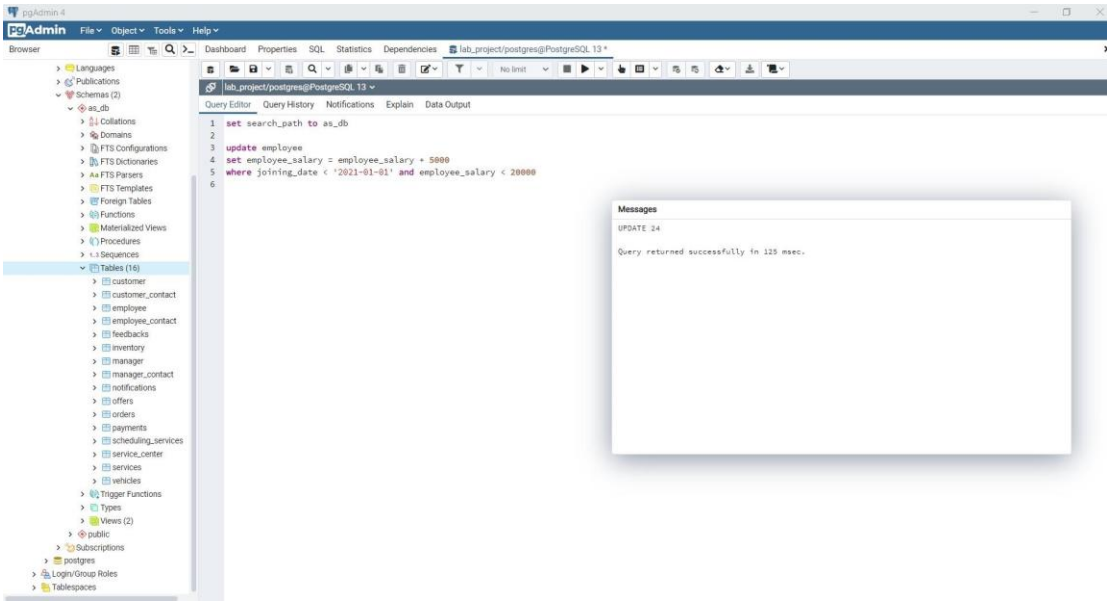
- **Query command:** create view manager_to_be_contacted as select center_id, manager_id, spare_part_id, spare_part_stock from inventory where spare_part_stock < 50 select manager_id, contact, spare_part_id, spare_part_stock from manager_to_be_contacted natural join manager_contact where manager_to_be_contacted.manager_id = manager_contact.manager_id



- Tuples: 480

7. Updating salaries of employees who are active and joined center before 1st Jan' 21

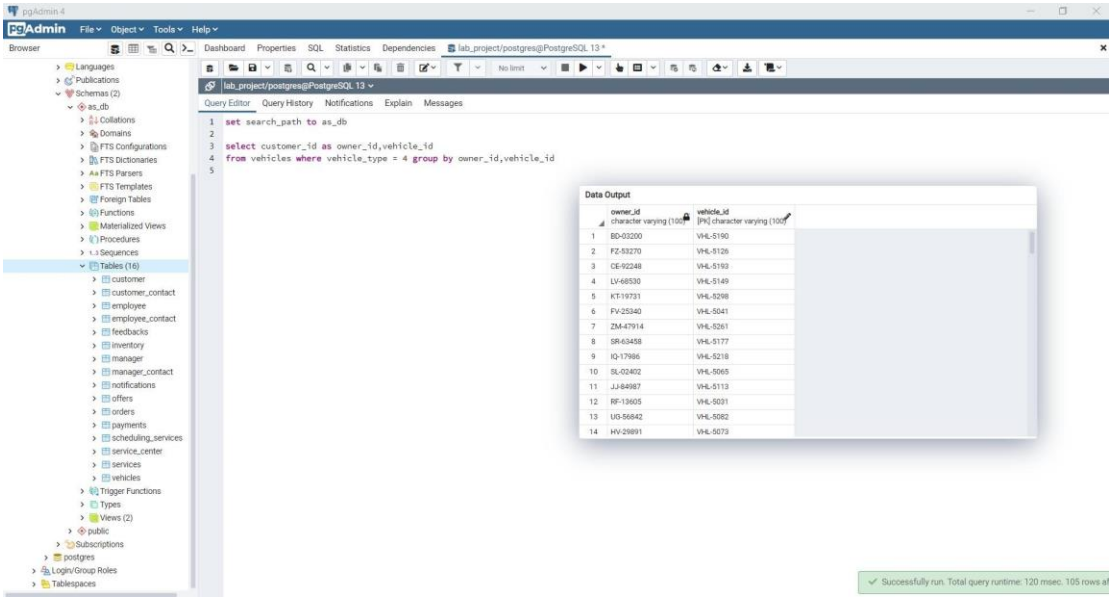
- **Query Command:** update employee set employee_salary = employee_salary + 5000 where joining_date < '2021-01-01' and employee_salary < 20000



- Tuples: 24

8. Extracting information about customers who owns atleast one 4 wheelers.

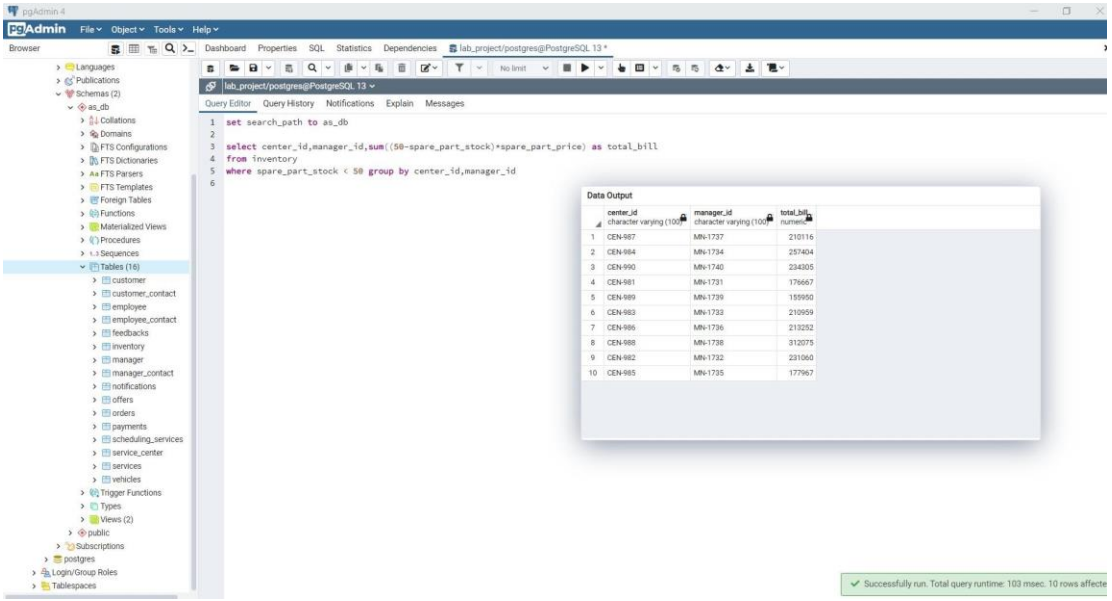
- **Query command:** select customer_id as owner_id,vehicle_id from vehicles where vehicle_type = 4 group by owner_id,vehicle_id



- Tuples: 105

9. In case stock of any spare part goes less than 50 then calculating bill to order them all at once.

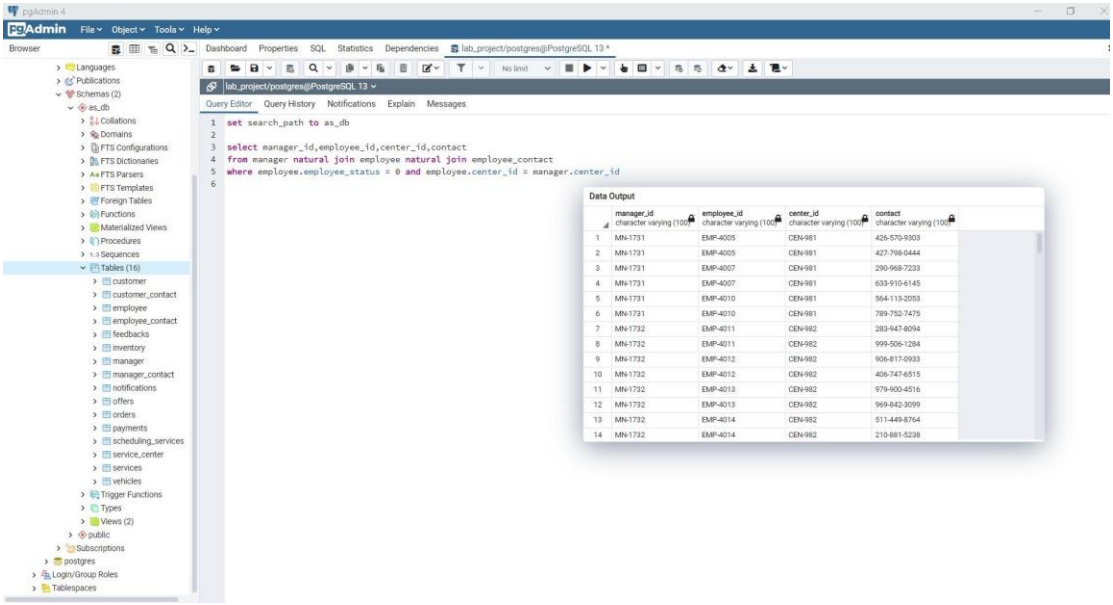
- **Query Command:** select center_id,manager_id,sum((50spare_part_stock)*spare_part_price) as total_bill from inventory where spare_part_stock < 50 group by center_id,manager_id



- Tuples: 10

10. Extracting information about employees who are inactive at the moment.

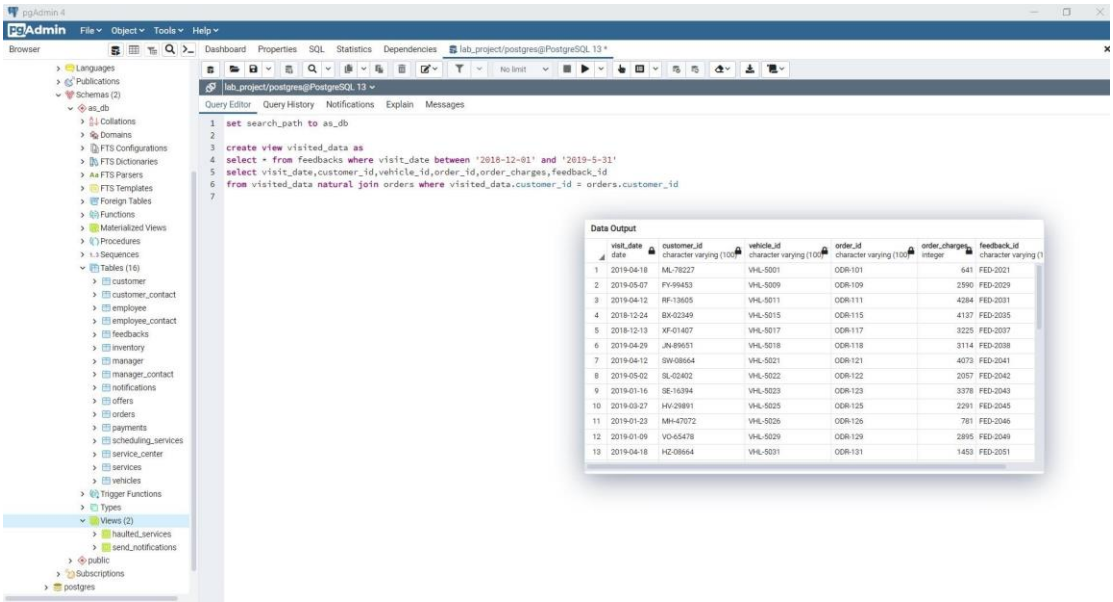
- **Query Command:** select manager_id,employee_id,center_id,contact from manager natural join employee natural join employee_contact where employee.employee_status = 0 and employee.center_id = manager.center_id



- Tuples: 116

11. Showing all information of the customers who have visited the center between December 2018 and May 2019

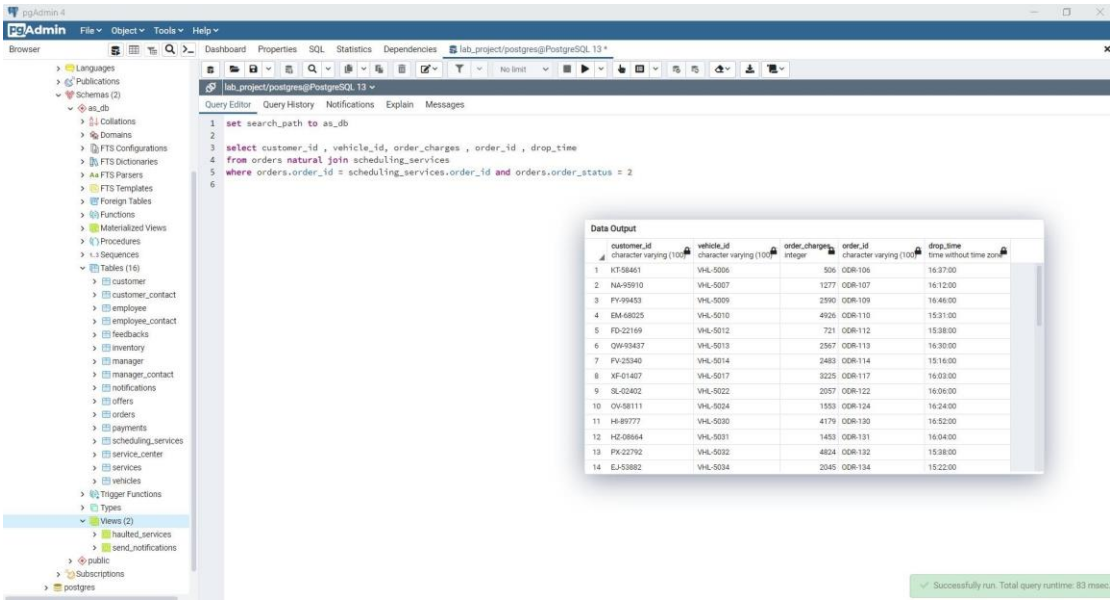
- **Query Command :** create view visited_data as select * from feedbacks where visit_date between '2018-12-01' and '2019-5-31' select visit_date, customer_id,vehicle_id,order_id,order_charges,feedback_id from visited_data natural join orders where visited_data.customer_id = orders.customer_id



- Tuples: 38

12. Especially for sending notifications to customer for completed services.

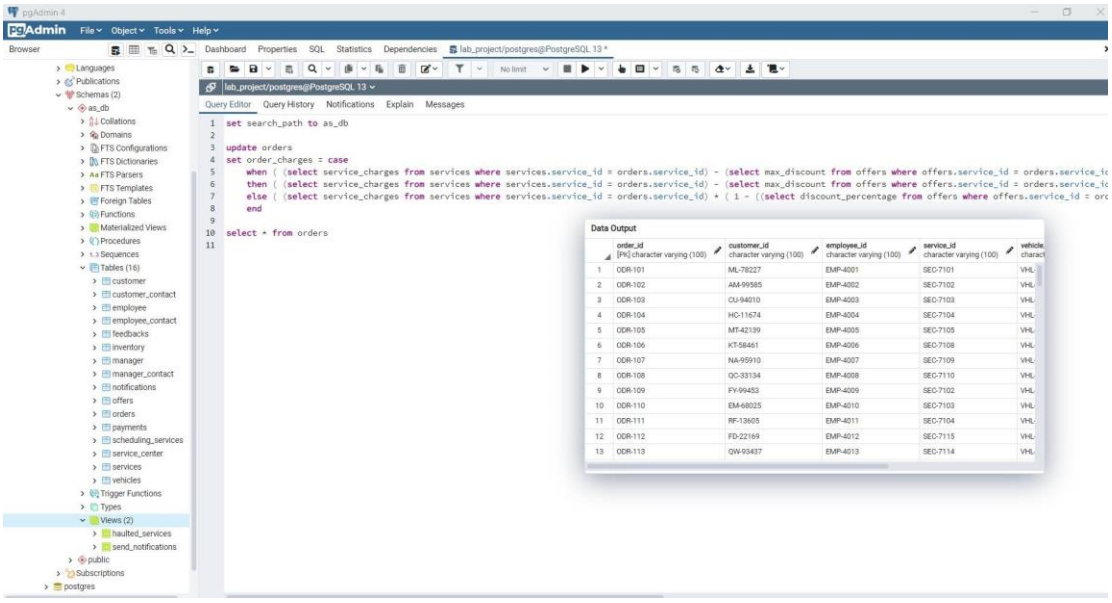
- **Query command:** select customer_id , vehicle_id, order_charges , order_id , drop_time from orders natural join scheduling_services where orders.order_id = scheduling_services.order_id and orders.order_status = 2



- Tuple: 83

13. Calculating the order_charges applying according offers.

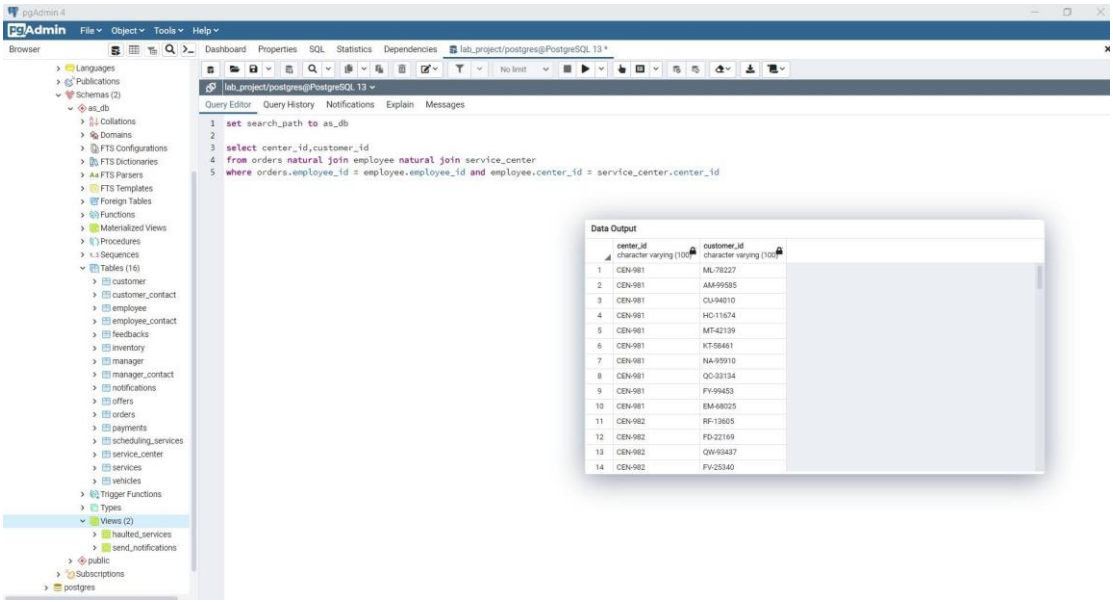
- **Query Command:** update orders set order_charges = case when ((select service_charges from services where services.service_id = orders.service_id) - (select max_discount from offers where offers.service_id = orders.service_id)) > (select service_charges from services where services.service_id = orders.service_id)*(1 - ((select discount_percentage from offers where offers.service_id = orders.service_id)/100))
  then ((select service_charges from services where services.service_id = orders.service_id) - (select max_discount from offers where offers.service_id = orders.service_id)) else ((select service_charges from services where services.service_id = orders.service_id) * (1 - ((select discount_percentage from offers where offers.service_id = orders.service_id)/100))) end
  select * from orders



- Tuples: 102

14. To see which customer is connected with which center.

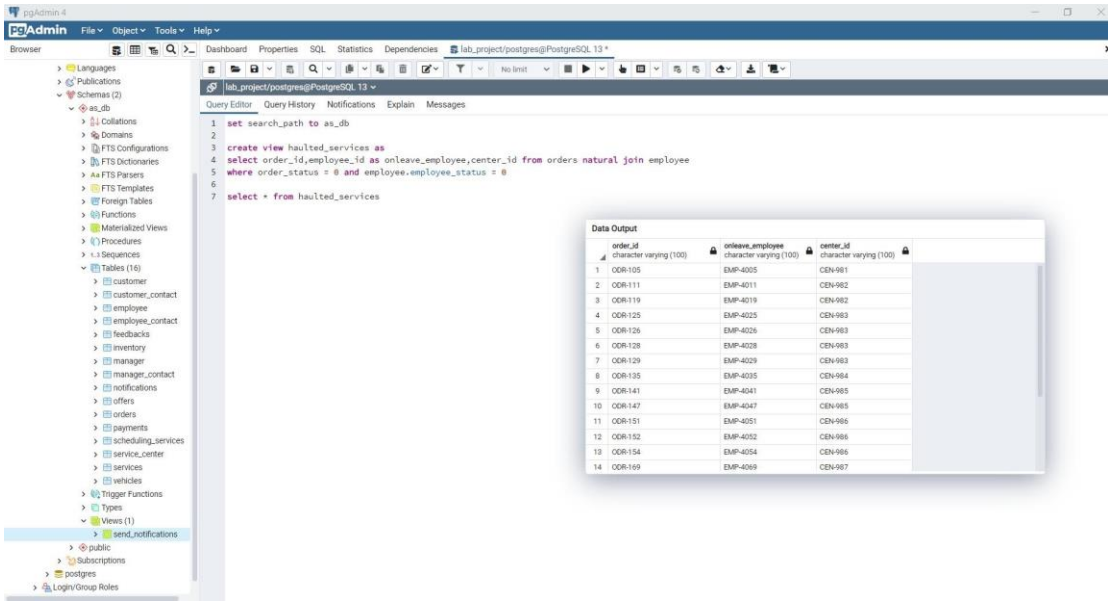
- **Query command:** select center_id,customer_id from orders natural join employee natural join service_center where orders.employee_id = employee.employee_id and employee.center_id = service_center.center_id



- Tuples: 102

15. Show the information of the orders which are pending and employee working on them is currently inctive.

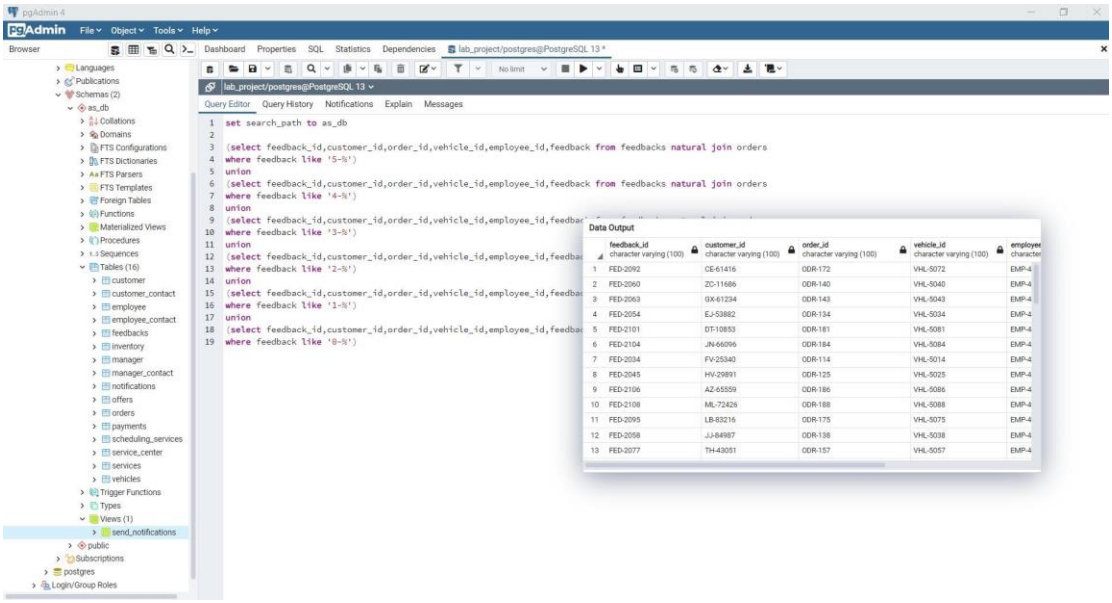
- **Query Command:** create view haulted_services as select order_id,employee_id as onleave_employee,center_id from orders natural join employee where order_status = 0 and employee.employee_status = 0 select * from haulted_services



- Tuple: 21

16. To get the Information of the customers who are not satisfied with the service given.

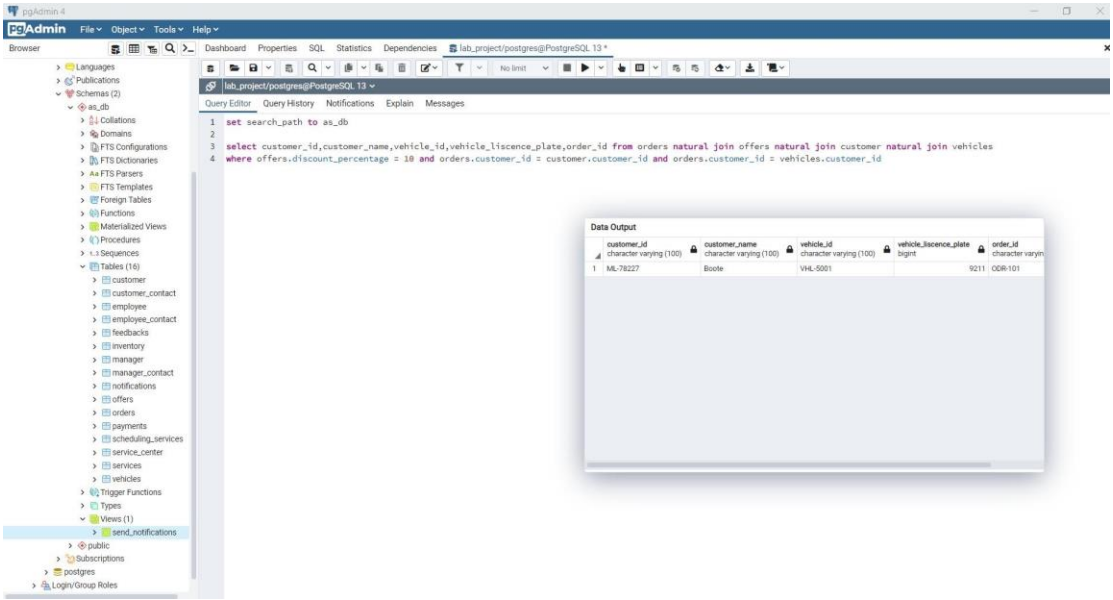
- **Query Command:** (select feedback_id,customer_id,order_id,vehicle_id,employee_id,feedback from feedbacks natural join orders where feedback like '5-%') union (select feedback_id, customer_id, order_id, vehicle_id, employee_id, feedback from feedbacks natural join orders where feedback like '4-%') union (select feedback_id, customer_id, order_id, vehicle_id, employee_id, feedback from feedbacks natural join orders where feedback like '3-%') union (select feedback_id, customer_id, order_id, vehicle_id, employee_id, feedback from feedbacks natural join orders where feedback like '2-%') union (select feedback_id, customer_id, order_id, vehicle_id, employee_id, feedback from feedbacks natural join orders where feedback like '1-%')union (select feedback_id,customer_id,order_id,vehicle_id,employee_id,feedback from feedbacks natural join orders where feedback like '0-%')



- Tuples: 59

17. Getting information of the customer who got 10% off in his most recent service.

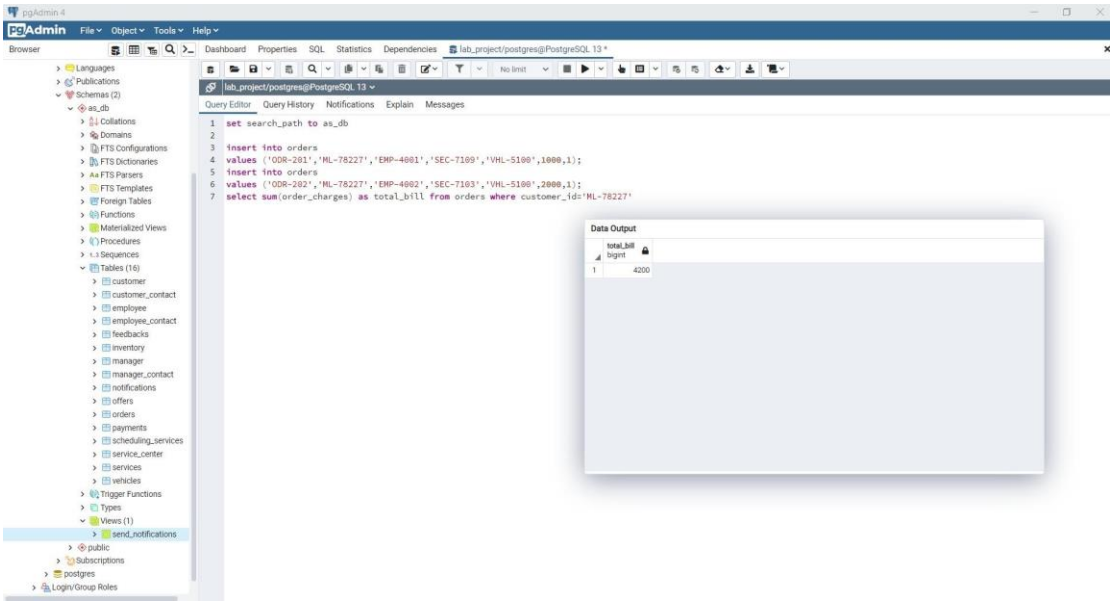
- **Query Command:** select customer_id, customer_name, vehicle_id, vehicle_liscence_plate, order_id from orders natural join offers natural join customer natural join vehicles where offers.discount_percentage = 10 and orders.customer_id = customer.customer_id and orders.customer_id = vehicles.customer_id



- Tuple: 1

18. Adding few more orders for same customer then calculating total bill.

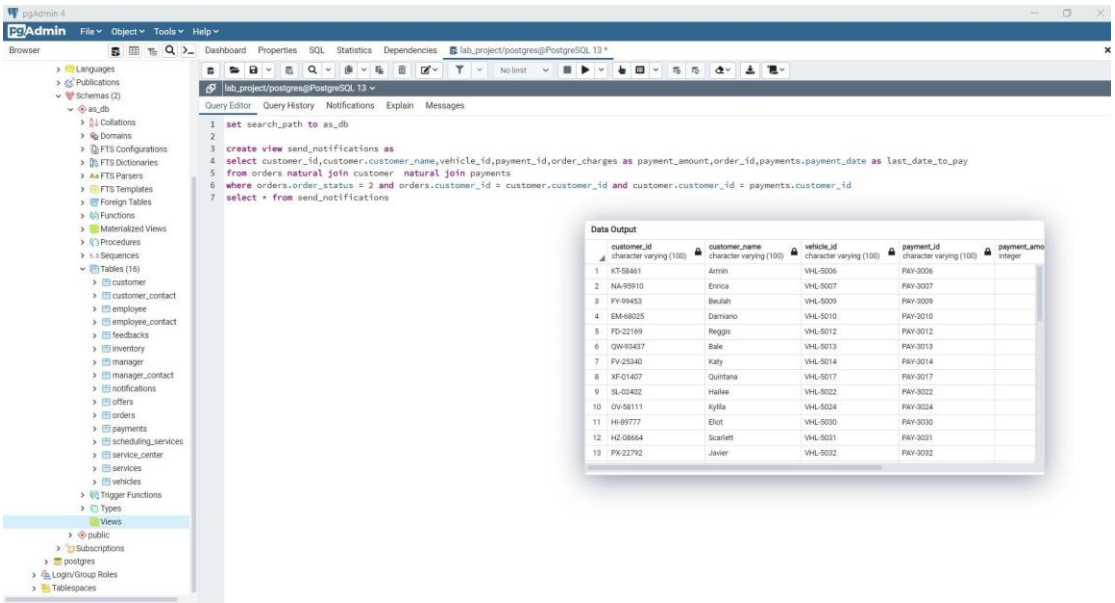
- **Query Command:** insert into orders values ('ODR-201','ML-78227','EMP-4001','SEC-7109','VHL-5100',1000,1); insert into orders values ('ODR-202','ML-78227','EMP-4002','SEC-7103','VHL-5100',2000,1); select sum(order_charges) as total_bill from orders where customer_id='ML-78227'



- Tuple: 1

19. Sending notifications to the customers whose service is completed.

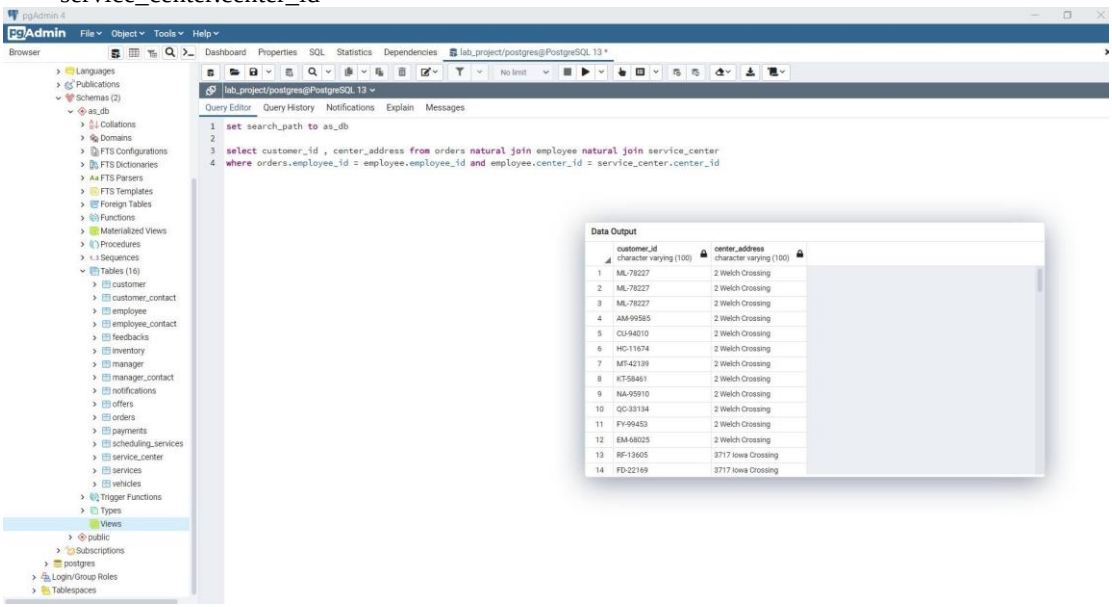
- **Query Command:** create view send_notifications as select customer_id, customer.customer_name, vehicle_id, payment_id, order_charges as payment_amount,order_id,payments.payment_date as last_date_to_pay from orders natural join customer natural join payments where orders.order_status = 2 and orders.customer_id = customer.customer_id and customer.customer_id = payments.customer_id select * from send_notifications



- Tuples: 44

20. Finding centre_address of every customer.

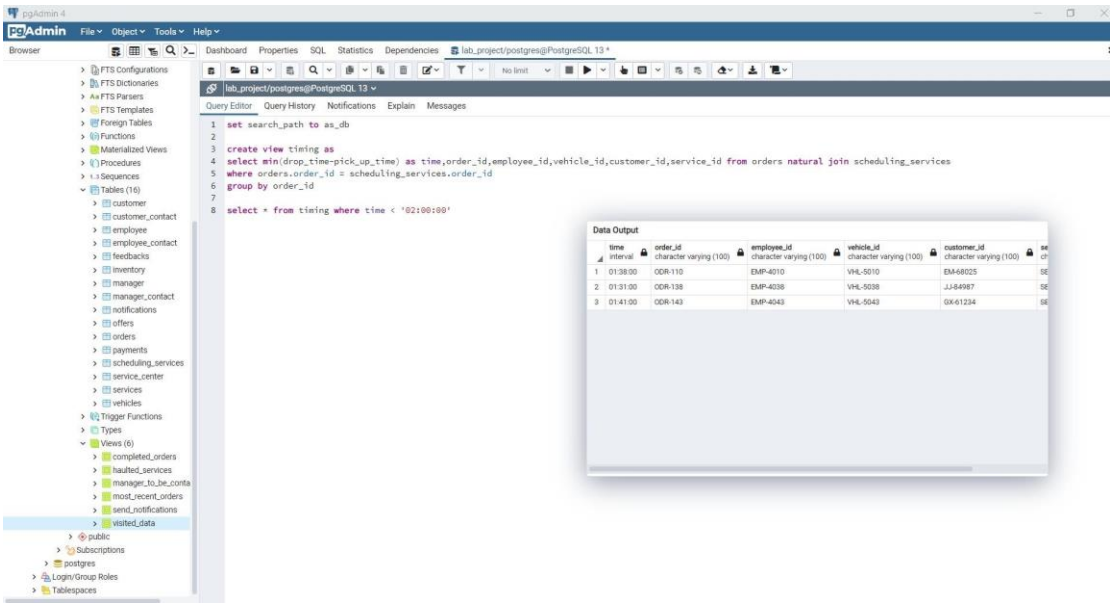
- **Query Command:** select customer_id , center_address from orders natural join employee natural join service_center where orders.employee_id = employee.employee_id and employee.center_id = service_center.center_id



- Tuples: 102

21. Finding orders which are done in 2 hours.

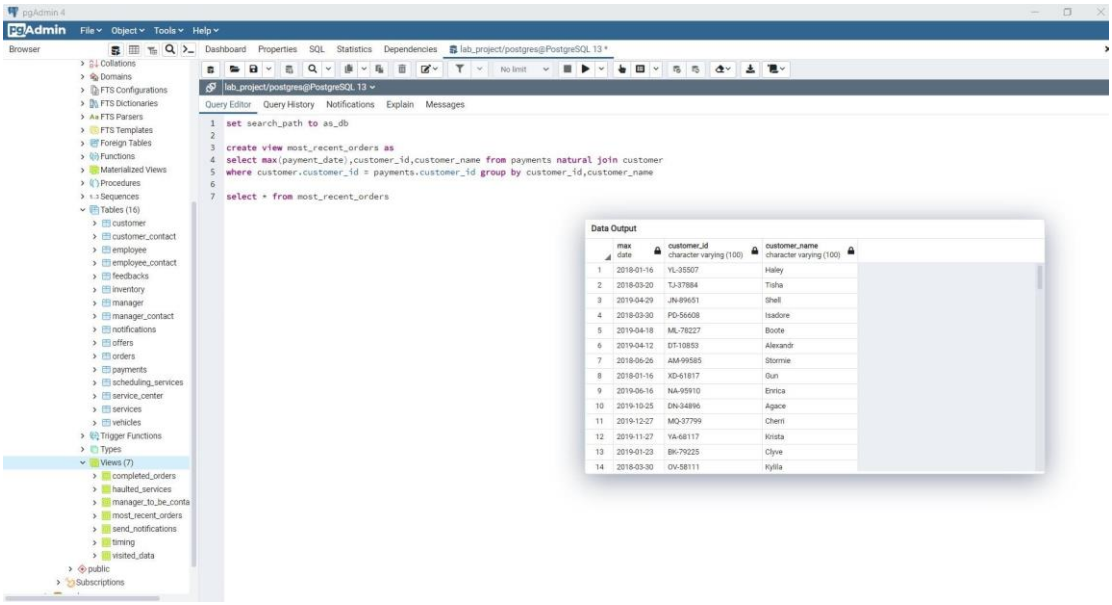
- **Query Command:** create view timing as select min(drop_time-pick_up_time) as time,order_id, employee_id, vehicle_id, customer_id, service_id from orders natural join scheduling_services where orders.order_id = scheduling_services.order_id
  group by order_id select * from timing where time < '02:00:00'



- Tuples: 3

22. Finding information of the most recent orders.

- **Query Command:** create view most_recent_orders as select max(payment_date), customer_id, customer_name from payments natural join customer where customer.customer_id = payments.customer_id group by customer_id,customer_name select * from most_recent_orders



- Tuple: 100

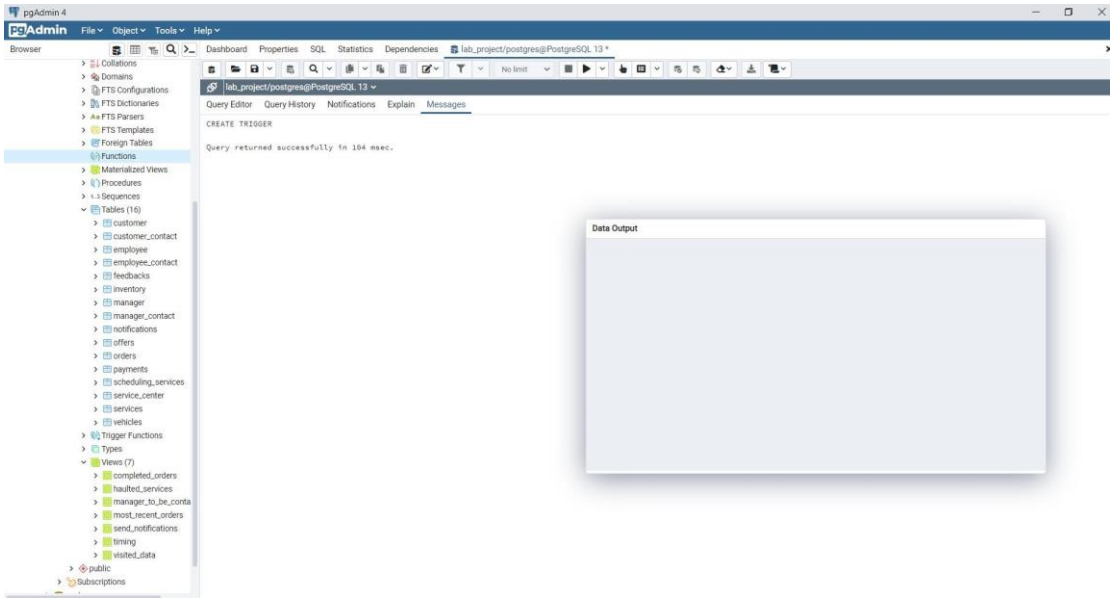
6.5.3 Trigger Function

1. We have created a trigger function to trigger the user defined error message in case of any illegal entry of tuple in table. (ex, adding tuple with already existing primary key)

- **Trigger function command:**

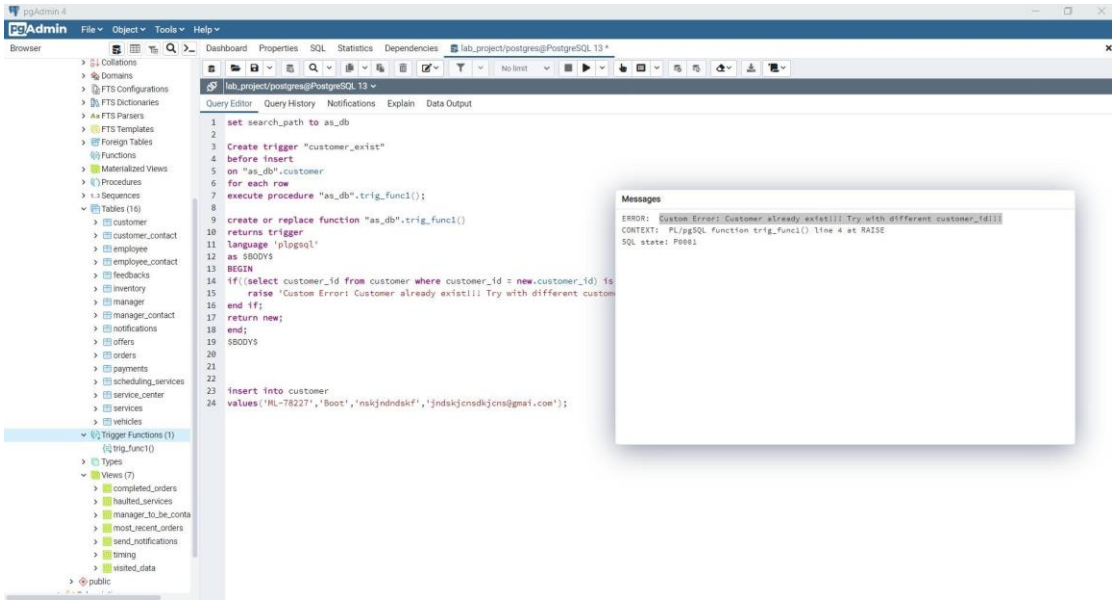
```
Create trigger "customer_exist" before
insert on "as_db".customer for each row
execute procedure "as_db".trig_func1();

create or replace function
"as_db".trig_func1() returns trigger
language 'plpgsql'
as $BODY$
BEGIN
if((select customer_id from customer
where customer_id = new.customer_id)
is not null) then raise 'Custom Error:
Customer already exist!!! Try with
different customer_id!!!' ;
end if;
return new;
end;
$BODY$
```



- Now we try to add tuple in customer table with existing customer_id then it will throw an user defined error “Custom Error: Customer already exist!!! Try with different customer_id!!! “ adding new tuple with existing customer_id insert into customer

values('ML-78227','Boot','nskjndndskf','jndskjcnsdkjcns@gmai.com');



6.5.4 Function

1. We have created a function which takes an input in text form which is order_id , and the function will return its status.

- **Function command:**
  CREATE OR REPLACE FUNCTION "as_db"."status"(temp_order_id text)

  RETURNS integer

  LANGUAGE 'plpgsql' AS
  $BODY$

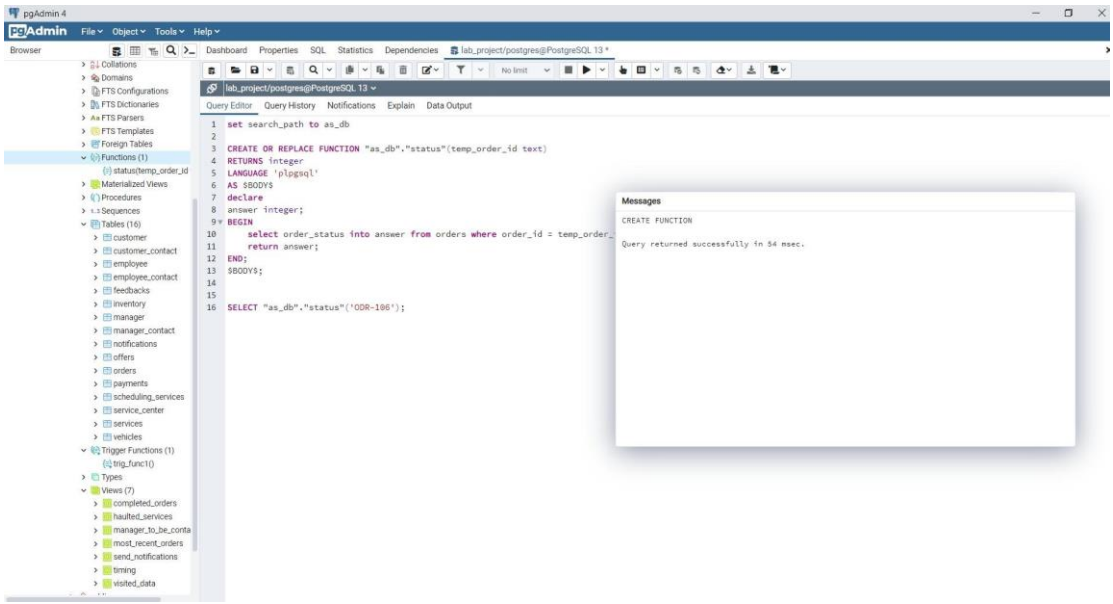
  declare answer
  integer;

  BEGIN

  select order_status
  into answer from
  orders where order_id
  = temp_order_id;
  return answer;

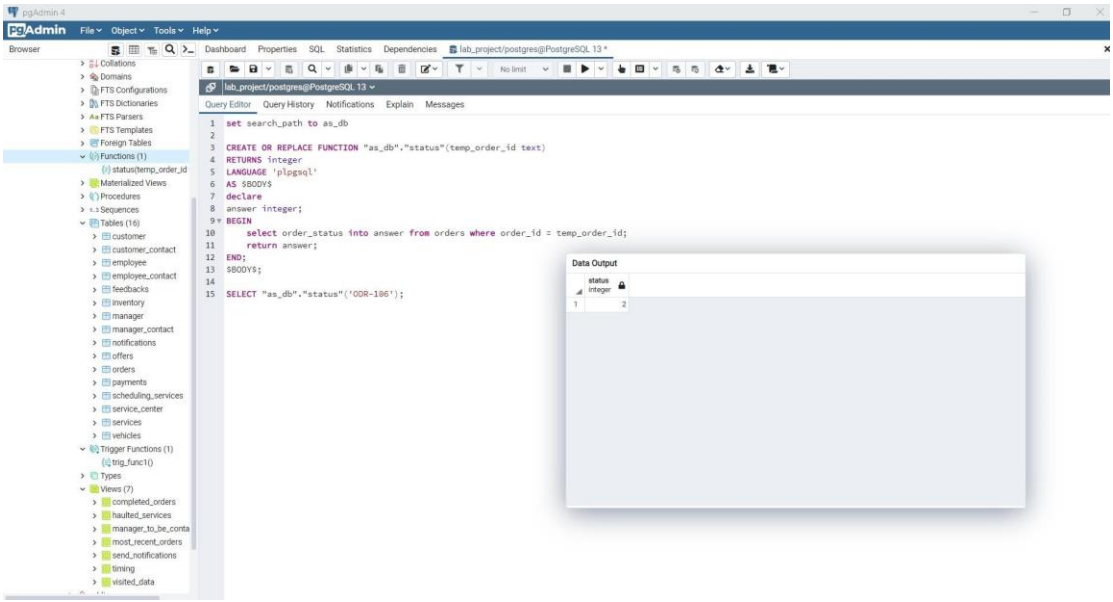
  END;

  $BODY$;



- now passing value(order_id) in function to check its status from database.

SELECT "as_db"."status"('ODR-106');



- Output is single digit 2 which shows that order with order_id ODR-106 is completed.

Section7: Project Code with output screenshots

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

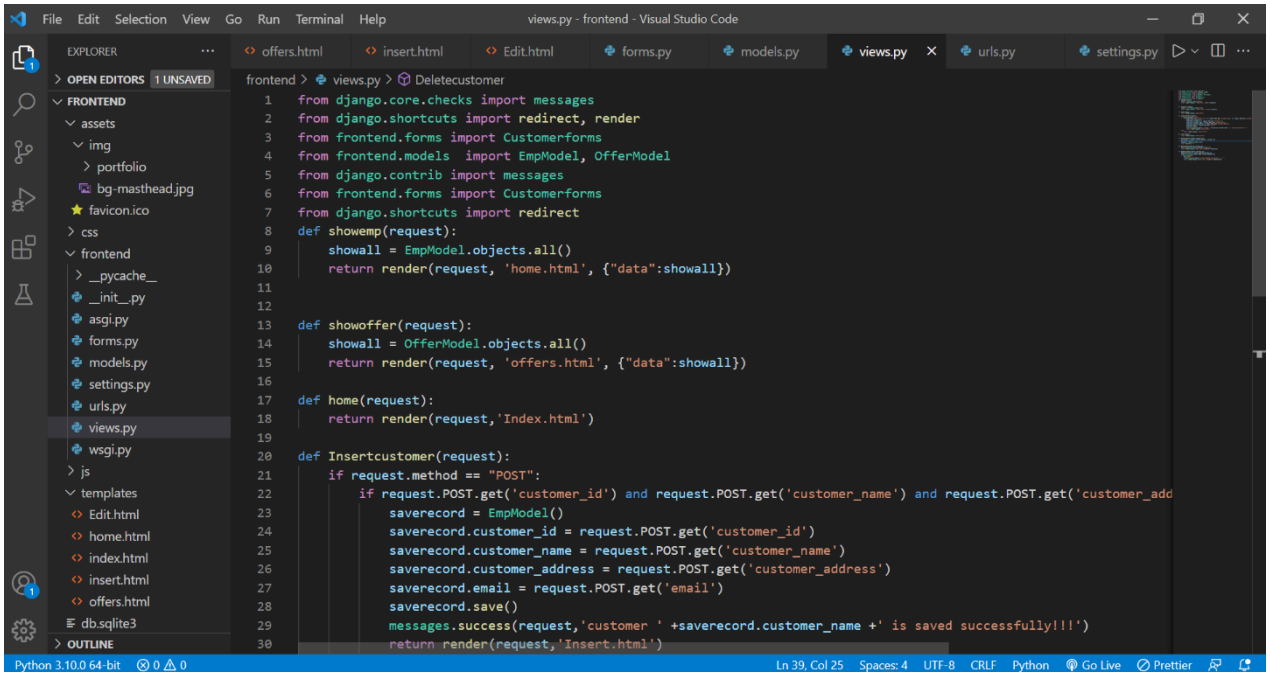
7.1 Codes File Link

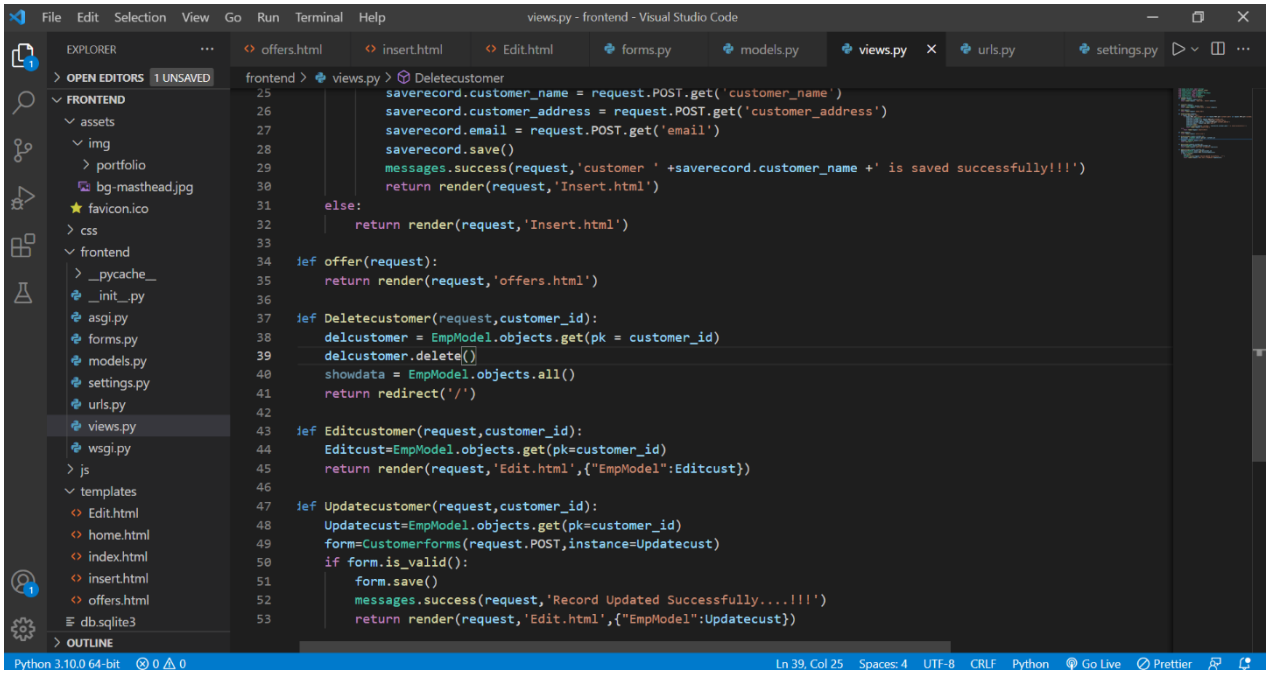
https://drive.google.com/file/d/1wY4IsmmKXTsVPb8jM4bSwT25tQaos5TH/view?usp=sharing

7.2 Snapshot of Code:

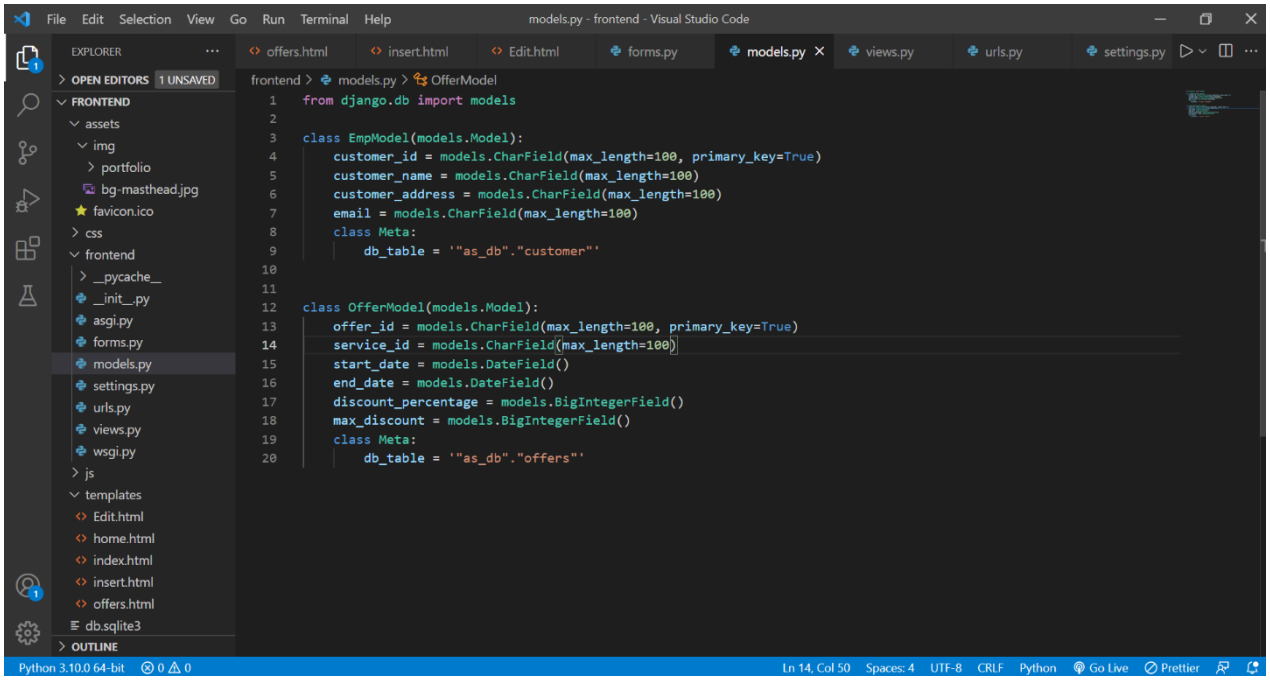
❖ views.py

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

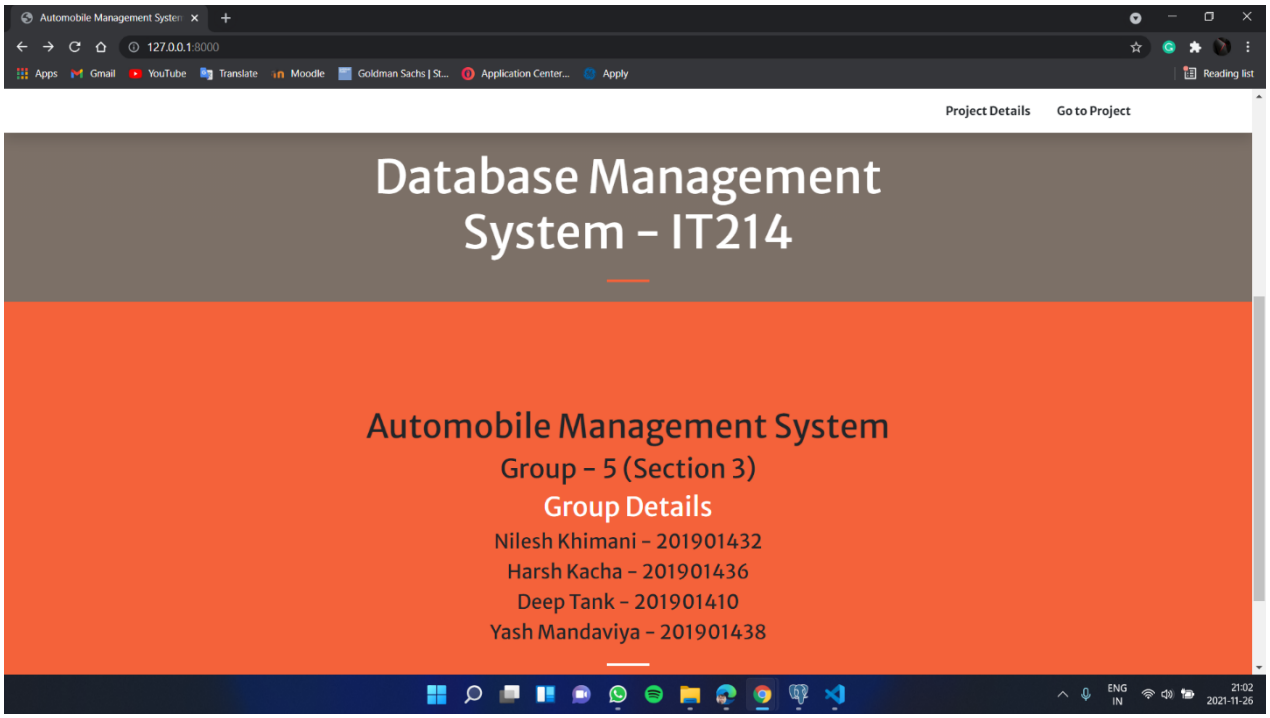



File Edit Selection View Go Run Terminal Help
views.py - frontend - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
FRONTEND
assets
img
portfolio
bg-masthead.jpg
favicon.ico
css
frontend
__pycache__
__init__.py
asgi.py
forms.py
models.py
settings.py
urls.py
views.py
wsgi.py
js
templates
Edit.html
home.html
index.html
insert.html
offers.html
db.sqlite3
OUTLINE
offers.html
insert.html
Edit.html
forms.py
models.py
views.py
urls.py
settings.py
frontend > views.py > Deletecustomer
saverecord.customer_name = request.POST.get('customer_name')
saverecord.customer_address = request.POST.get('customer_address')
saverecord.email = request.POST.get('email')
saverecord.save()
messages.success(request,'customer ' +saverecord.customer_name +' is saved successfully!!!')
return render(request,'Insert.html')
else:
return render(request,'Insert.html')
def offer(request):
return render(request,'offers.html')
def Deletecustomer(request,customer_id):
delcustomer = EmpModel.objects.get(pk = customer_id)
delcustomer.delete()
showdata = EmpModel.objects.all()
return redirect('/')
def Editcustomer(request,customer_id):
Editcust=EmpModel.objects.get(pk=customer_id)
return render(request,'Edit.html',{"EmpModel":Editcust})
def Updatecustomer(request,customer_id):
Updatecust=EmpModel.objects.get(pk=customer_id)
form=Customerforms(request.POST,instance=Updatecust)
if form.is_valid():
form.save()
messages.success(request,'Record Updated Successfully....!!!')
return render(request,'Edit.html',{"EmpModel":Updatecust})
Python 3.10.0 64-bit
Ln 39, Col 25
Spaces: 4
UTF-8
CRLF
Python
Go Live
Prettier

❖ models.py

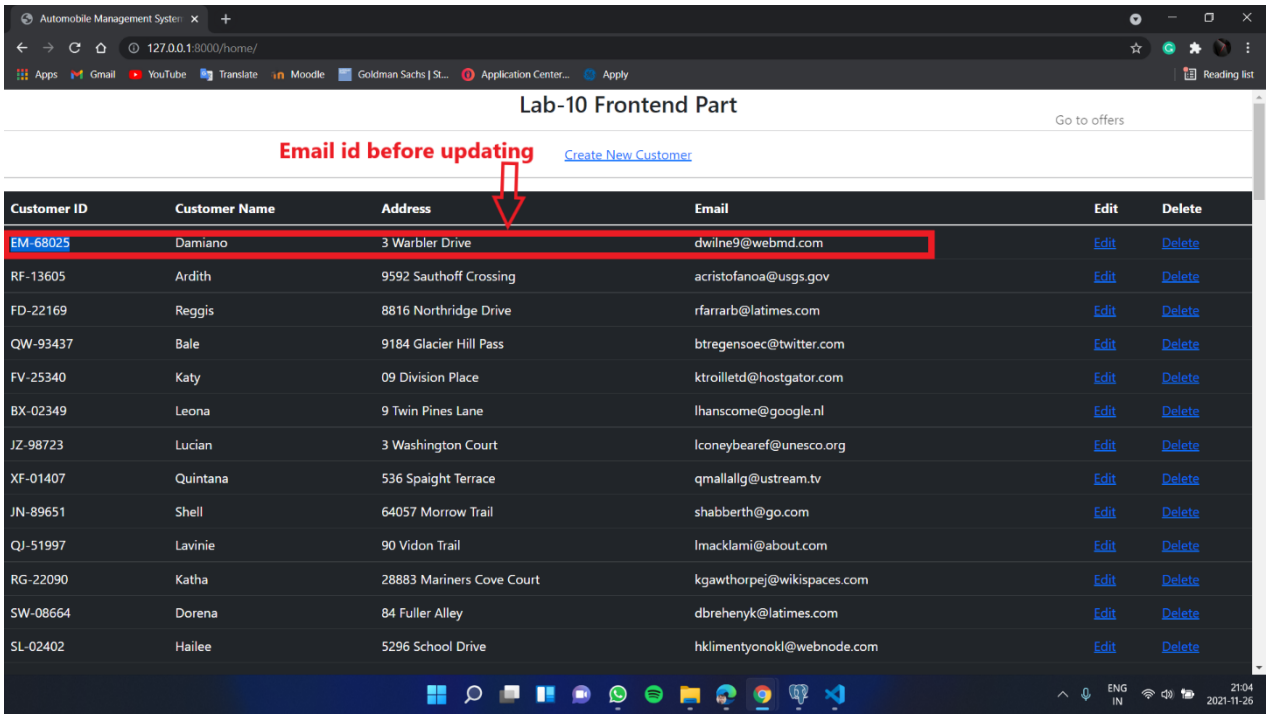

File Edit Selection View Go Run Terminal Help
models.py - frontend - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
FRONTEND
assets
img
portfolio
bg-masthead.jpg
favicon.ico
css
frontend
__pycache__
__init__.py
asgi.py
forms.py
models.py
settings.py
urls.py
views.py
wsgi.py
js
templates
Edit.html
home.html
index.html
insert.html
offers.html
db.sqlite3
OUTLINE
offers.html
insert.html
Edit.html
forms.py
models.py
views.py
urls.py
settings.py
frontend > models.py > OfferModel
from django.db import models
class EmpModel(models.Model):
customer_id = models.CharField(max_length=100, primary_key=True)
customer_name = models.CharField(max_length=100)
customer_address = models.CharField(max_length=100)
email = models.CharField(max_length=100)
class Meta:
db_table = '"as_db"."customer"'
class OfferModel(models.Model):
offer_id = models.CharField(max_length=100, primary_key=True)
service_id = models.CharField(max_length=100)
start_date = models.DateField()
end_date = models.DateField()
discount_percentage = models.BigIntegerField()
max_discount = models.BigIntegerField()
class Meta:
db_table = '"as_db"."offers"'
Python 3.10.0 64-bit
Ln 14, Col 50
Spaces: 4
UTF-8
CRLF
Python
Go Live
Prettier

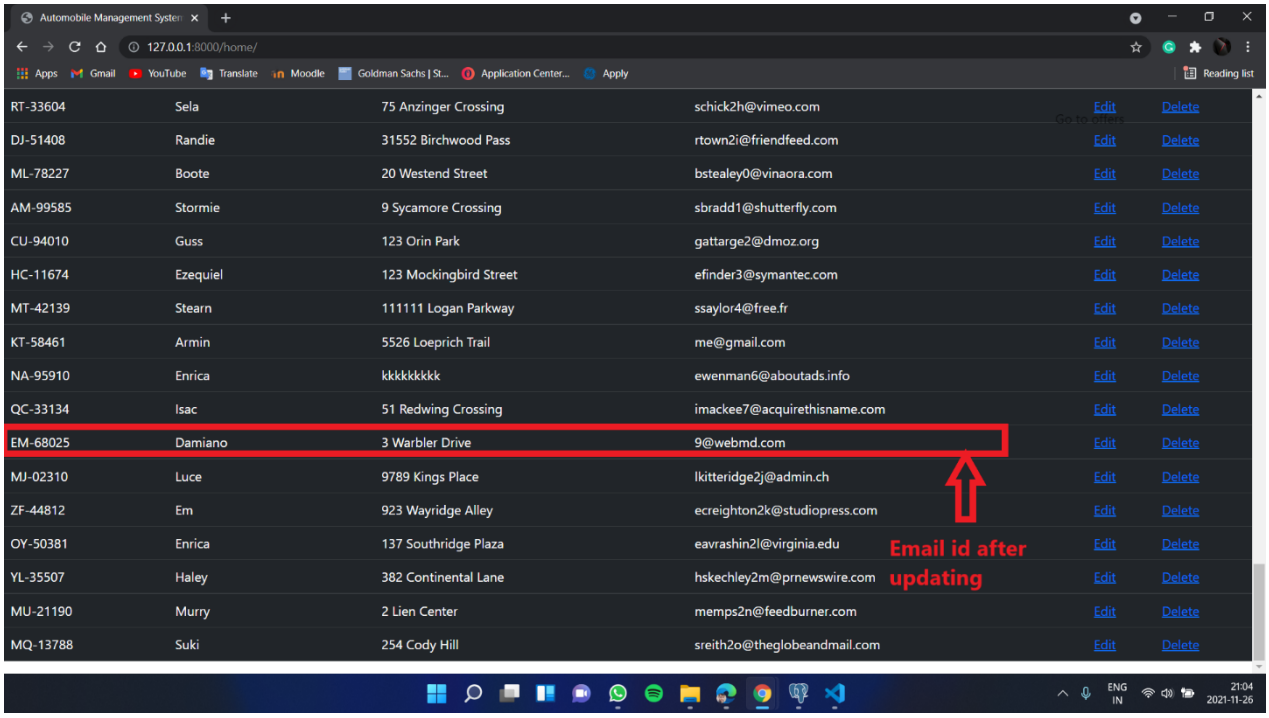
7.3 Snapshot of Webpage:



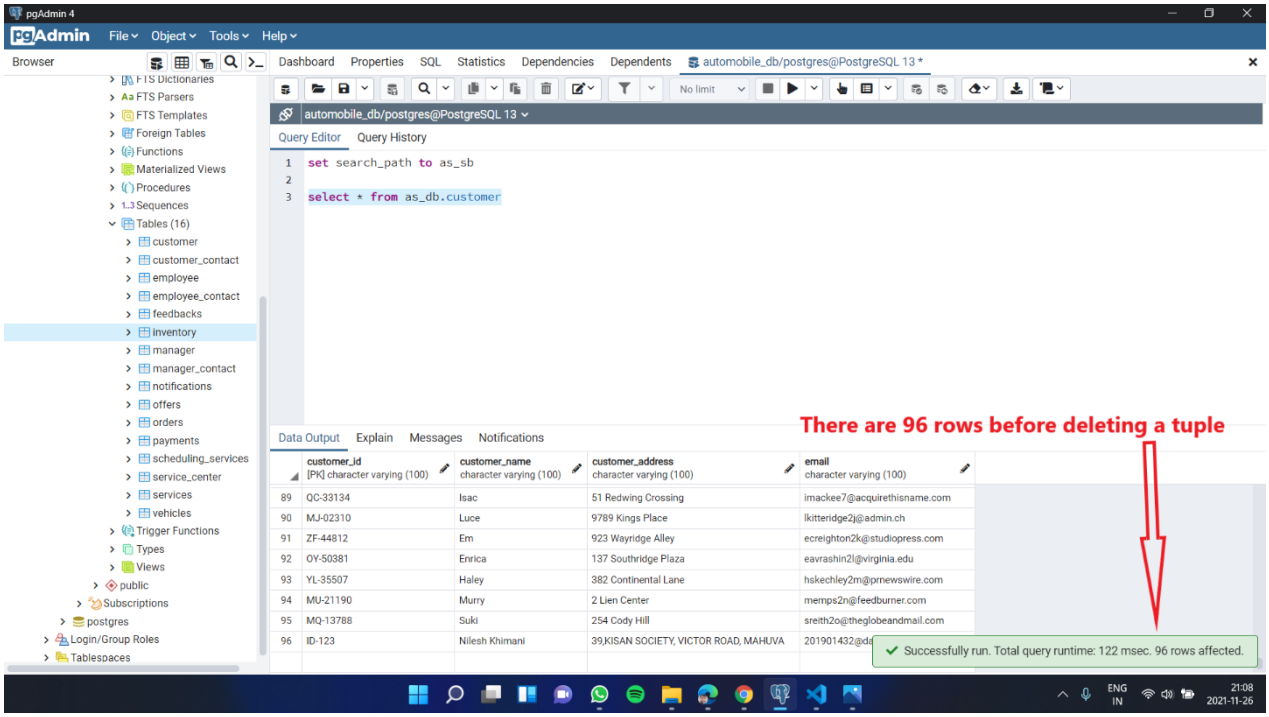
7.4 Snapshot of Queries and Outputs:

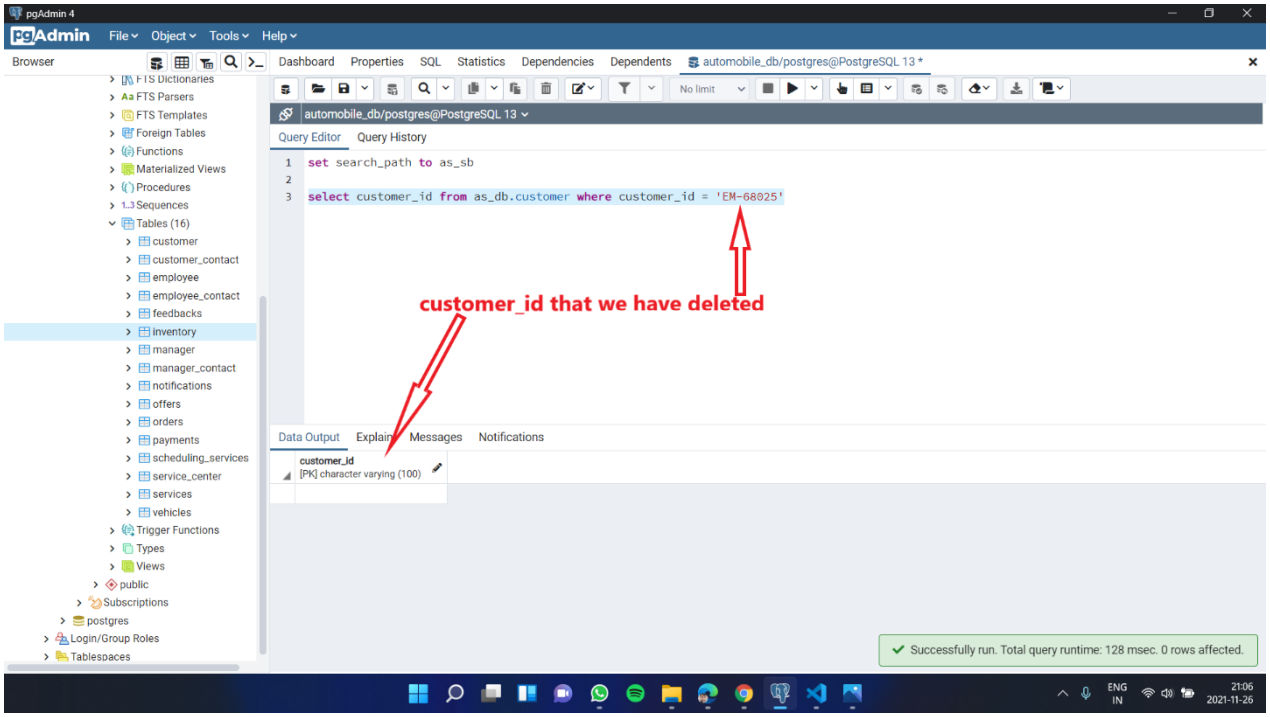
- **Update**

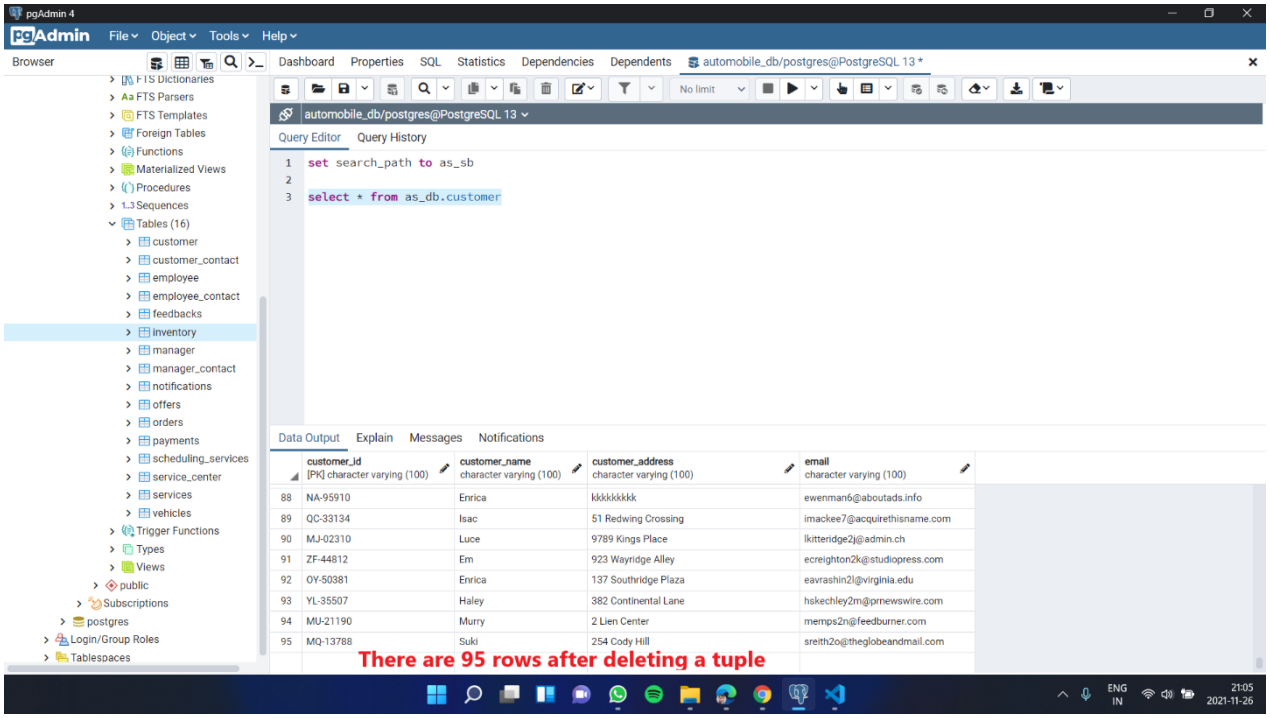



Automobile Management Syster
127.0.0.1:8000/home/
Apps Gmail YouTube Translate Moodle Goldman Sachs | St... Application Center... Apply
Reading list
RT-33604 Sela 75 Anzinger Crossing schick2h@vimeo.com Edit Delete
DJ-51408 Randie 31552 Birchwood Pass rtown2i@friendfeed.com Edit Delete
ML-78227 Boote 20 Westend Street bstealey0@vinaora.com Edit Delete
AM-99585 Stormie 9 Sycamore Crossing sbradd1@shutterfly.com Edit Delete
CU-94010 Guss 123 Orin Park gattarge2@dmoz.org Edit Delete
HC-11674 Ezequiel 123 Mockingbird Street efinder3@symantec.com Edit Delete
MT-42139 Stearn 111111 Logan Parkway ssaylor4@free.fr Edit Delete
KT-58461 Armin 5526 Loeprich Trail me@gmail.com Edit Delete
NA-95910 Enrica kkkkkkkkk ewenman6@aboutads.info Edit Delete
QC-33134 Isac 51 Redwing Crossing imackee7@acquirethisname.com Edit Delete
EM-68025 Damiano 3 Warbler Drive 9@webmd.com Edit Delete
MJ-02310 Luce 9789 Kings Place lkitteridge2j@admin.ch Edit Delete
ZF-44812 Em 923 Wayridge Alley ecreighton2k@studiopress.com Edit Delete
OY-50381 Enrica 137 Southridge Plaza eavrashin2l@virginia.edu Edit Delete
YL-35507 Haley 382 Continental Lane hskechley2m@prnewswire.com Edit Delete
MU-21190 Murry 2 Lien Center memps2n@feedburner.com Edit Delete
MQ-13788 Suki 254 Cody Hill sreith2o@theglobeandmail.com Edit Delete
Email id after updating
ENG IN 21:04 2021-11-26

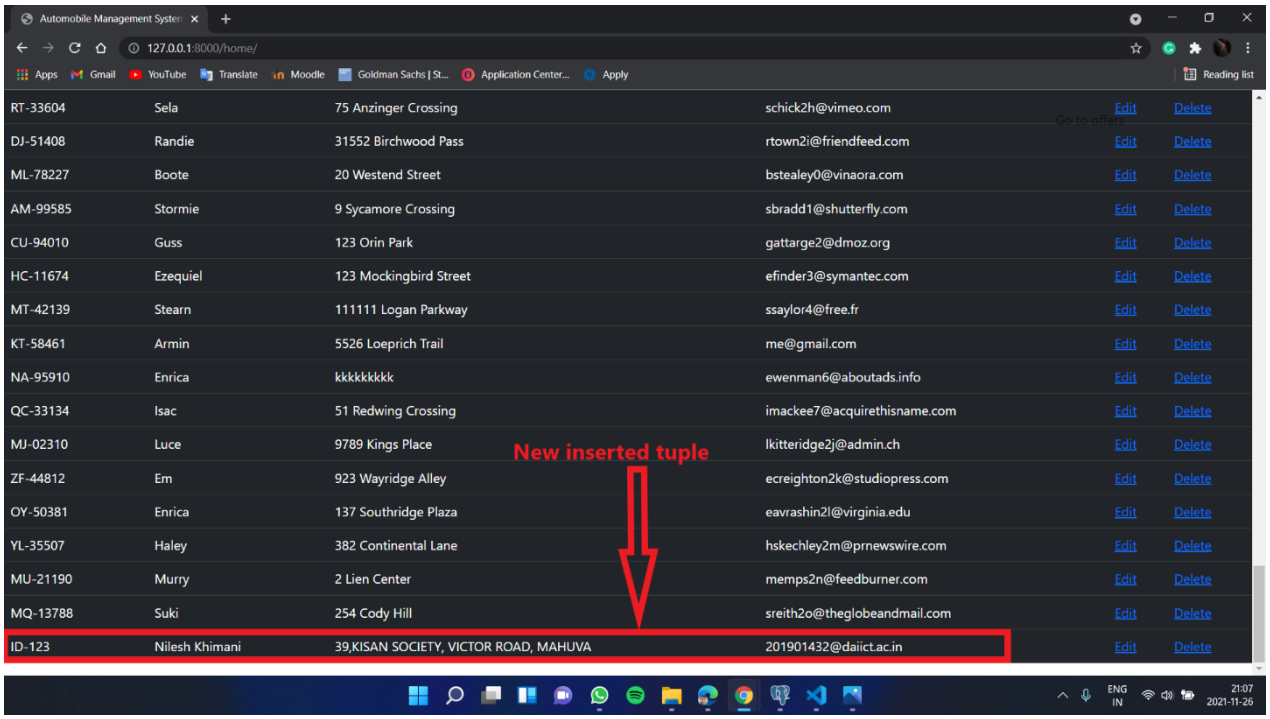
❖ Delete


pgAdmin 4
File Object Tools Help
Browser
Dashboard Properties SQL Statistics Dependencies Dependents automobile_db/postgres@PostgreSQL 13 *
automobile_db/postgres@PostgreSQL 13
Query Editor Query History
1 set search_path to as_sb
2
3 select * from as_db.customer
Data Output Explain Messages Notifications
There are 96 rows before deleting a tuple
customer_id [PK] character varying (100) customer_name character varying (100) customer_address character varying (100) email character varying (100)
89 QC-33134 Isac 51 Redwing Crossing imackee7@acquirethisname.com
90 MJ-02310 Luce 9789 Kings Place lkitteridge2j@admin.ch
91 ZF-44812 Em 923 Wayridge Alley ecreighton2k@studiopress.com
92 OY-50381 Enrica 137 Southridge Plaza eavrashin2l@virginia.edu
93 YL-35507 Haley 382 Continental Lane hskechley2m@prnewswire.com
94 MU-21190 Murry 2 Lien Center memps2n@feedburner.com
95 MQ-13788 Suki 254 Cody Hill sreith2o@theglobeandmail.com
96 ID-123 Nilesh Khimani 39,KISAN SOCIETY, VICTOR ROAD, MAHUVA 201901432@dd
Successfully run. Total query runtime: 122 msec. 96 rows affected.
ENG IN 21:08 2021-11-26


pgAdmin 4
File Object Tools Help
Browser
Dashboard Properties SQL Statistics Dependencies Dependents automobile_db/postgres@PostgreSQL 13 *
automobile_db/postgres@PostgreSQL 13
Query Editor Query History
1 set search_path to as_sb
2
3 select customer_id from as_db.customer where customer_id = 'EM-68025'
customer_id that we have deleted
Data Output Explain Messages Notifications
customer_id
[PK] character varying (100)
Successfully run. Total query runtime: 128 msec. 0 rows affected.


pgAdmin 4
File Object Tools Help
Browser
Dashboard Properties SQL Statistics Dependencies Dependents automobile_db/postgres@PostgreSQL 13 *
automobile_db/postgres@PostgreSQL 13
Query Editor Query History
1 set search_path to as_sb
2
3 select * from as_db.customer
Data Output Explain Messages Notifications
customer_id [PK] character varying (100)
customer_name character varying (100)
customer_address character varying (100)
email character varying (100)
88 NA-95910 Enrica kkkkkkkkk ewenman6@aboutads.info
89 QC-33134 Isac 51 Redwing Crossing imackee7@acquirethisname.com
90 MJ-02310 Luce 9789 Kings Place lkitteridge2j@admin.ch
91 ZF-44812 Em 923 Wayridge Alley ecreighton2k@studiopress.com
92 OY-50381 Enrica 137 Southridge Plaza eavrashin2l@virginia.edu
93 YL-35507 Haley 382 Continental Lane hskechley2m@prnewswire.com
94 MU-21190 Murry 2 Lien Center memps2n@feedburner.com
95 MQ-13788 Suki 254 Cody Hill sreith2o@theglobeandmail.com
There are 95 rows after deleting a tuple

❖ Insert



❖ Offer table