



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	4
1.1 Introduction	5
1.1.1 Purpose	5
1.1.2 Intended Audience and Reading Suggestions.....	5
1.1.3 Product Scope.....	5
1.1.4 Description	5
1.1.5 Relations	6
1.1.6 Workflow of the Automobile service centre	7
1.2 Document the requirements collection.....	8
1.2.1 Background Readings and References.....	8
1.2.2 Mock Interview Summary	9
1.2.3 Questionnaire:	11
1.2.4 Observation:.....	16
1.3 Fact Finding Chart.....	17
1.4 List of requirements	17
1.5 User Classes and Characteristic	18
1.6 Operating Environment	18
1.7 Product Function	19
1.8 Privileges	20
1.9 Assumption.....	20
1.10 Business Constrains.....	20
Section2: Noun Analysis	21
2.1 Final Problem Description.....	22
2.2 Noun & Verb Analysis	24
2.3 Rejected Verbs.....	31
2.4 Rejected Nouns	31
2.5 Identify Entity Types	33
2.6 Identify Relationship Types	33
Section3: ER-Diagrams all versions	34
3.1 E-R Diagram (ERD) Version 1	35
3.2 E-R Diagram (ERD) Version 2	36
3.3 Final E-R Diagram (ERD)	37
Section4: Conversion of Final E-R Diagram to Relational Model	38
4.1 Final Version of E-R Diagram.....	39
4.2 Mapping ER Model to Relation Model:.....	39
4.3 Relational Model.....	40

Section5: Normalization and Schema Refinement	41
5.1 List all the Relations & Schemas with all details (Original Design of Database)	42
5.2 Identify and list all types of dependencies (PK, FK, Functional Dependencies) for each relation..	42
5.3 Investigate every schema for the following:	44
5.4 Normalize the database up to 1NF (scalar values).....	44
5.5 Normalize the database further to 2NF/3NF/BCNF (Remove Partial Dependencies)	46
5.6 Final Relational Model.....	48
5.7 Final Relational Schema.....	49
Section6: SQL: Final DDL Scripts, Insert statements, 40 SQL Queries with Snapshots of the output of each query	50
6.1 Final DDL Script	51
6.2 Primary Key Dependency (key constraints).....	54
6.3 Foreign Key Dependencies (referential integrity constraints):.....	55
6.4 Snapshot of all tables with data.....	55
6.5 40 SQL Queries with Snapshots of output of each query	63
6.5.1 Running Simple Queries.....	63
6.5.2 Running Complex Queries.....	71
6.5.3 Trigger Function.....	83
6.5.4 Function	85
Section7: Project Code with output screenshots	87
7.1 Codes File Link.....	88
7.2 Snapshot of Code:	88
7.3 Snapshot of Webpage:.....	90
7.4 Snapshot of Queries and Outputs:	90

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 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.

1.1.2 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.)

1.1.3 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.

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, 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.

1.1.6 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 customer's data and hence it will be easier for the service center to help the customer.
 - Then customer requests for a service and then service center 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 will assign that slot to that particular 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, customer 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.

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 a particular 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 of the centre.

Documents to be brought:

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

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 to the customers.
- The notification that services centre get from customer on regular and emergency services should be in less than 3 minutes after successful registration.
- Manager wants to make system efficient to schedule the appointment booked by customers on different basis. Ex: By vehicle_type, proposed_visit_time, etc.
- Service Centre wants us to minimize the waiting time and of the vehicle arranging at centre and minimizing servicing time of the vehicles.
- The system should divide proposed areas in a way that every customer gets 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 customers and 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 mobile application 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 got and 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 done and 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:

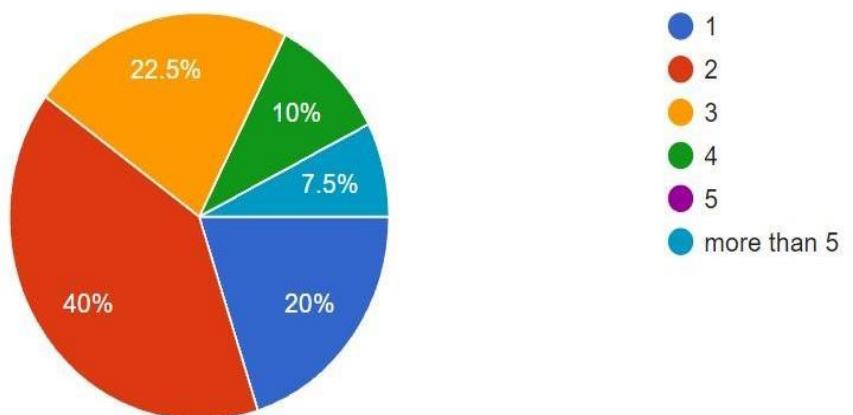
- The funds provider comes in the picture when an attempt to improve in system is supposed for greater good of the service center.
- If any new application (which at first glance looks beneficial) is proposed to the system then all financial support given by stock holders is important.
- In meeting the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

1.2.3 Questionnaire:

1) Number of vehicles in your house

Number of vehicles in your house

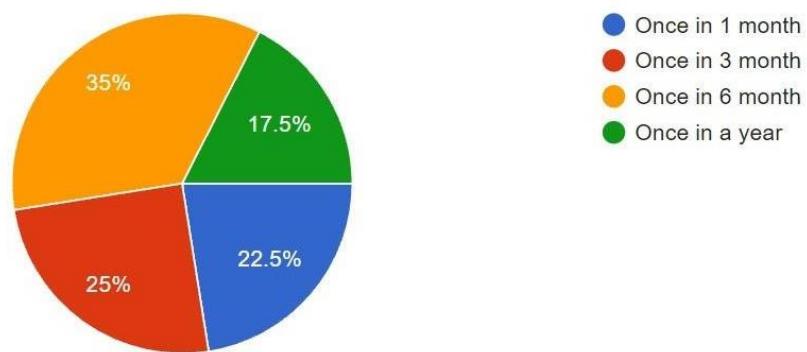
40 responses



2) How often do you visit service center?

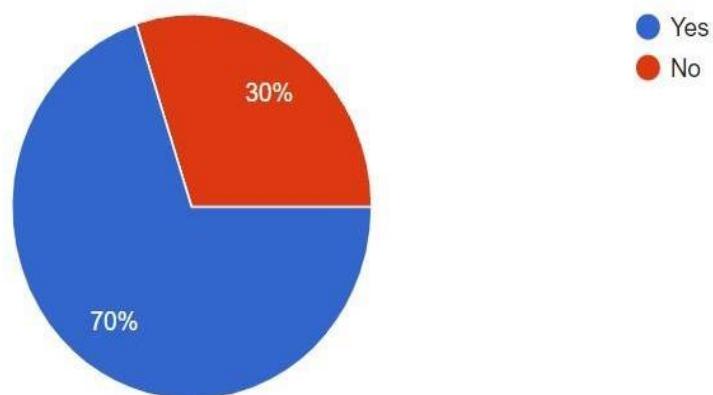
How often do you visit service center?

40 responses

**3) Did you get the notifications for the first 3 free services for your vehicle?**

Did you get the notifications for the first 3 free services for your vehicle?

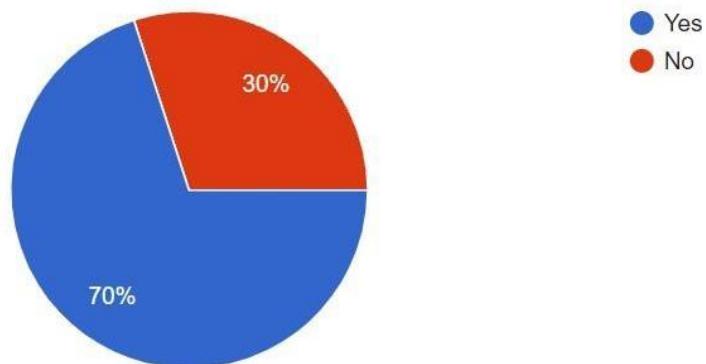
40 responses



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

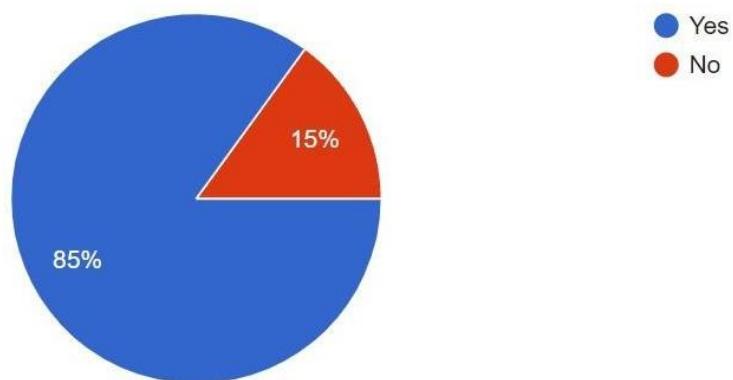
Did you get alerts for your pre-registered service a day prior?

40 responses

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

If you haven't registered for the service, would you use emergency service?

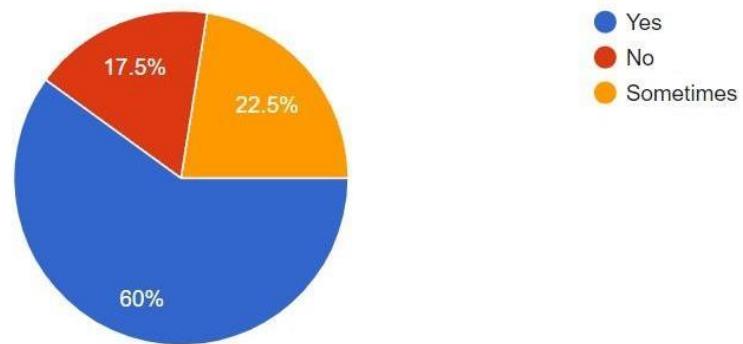
40 responses



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

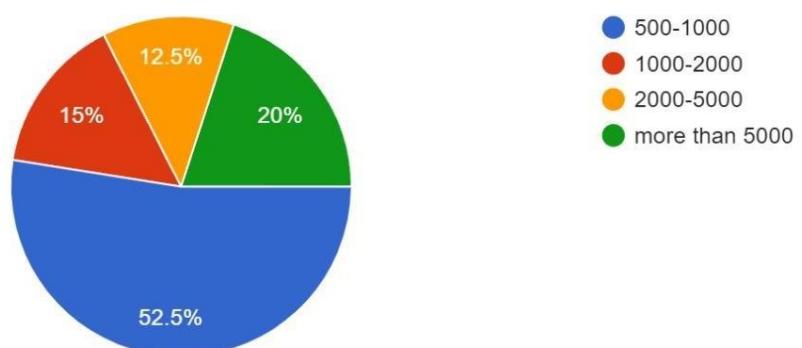
Would you like to use pre-registration for the service?

40 responses

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

if yes, till how much would you pay extra charges for the emergency service?

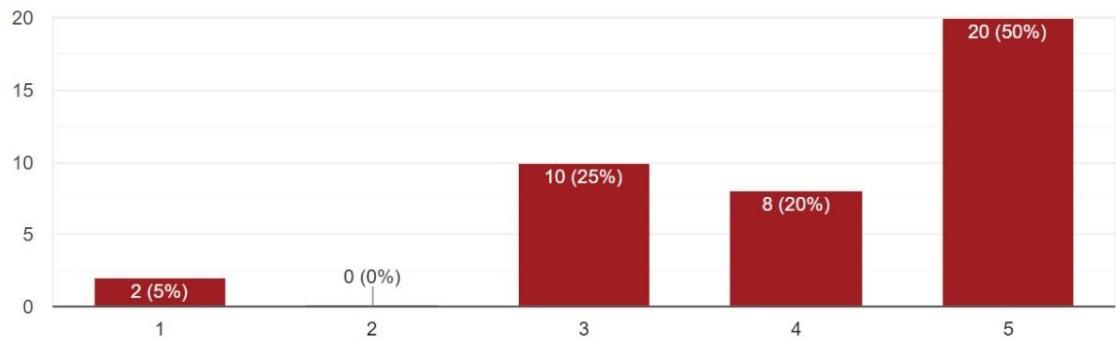
40 responses



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

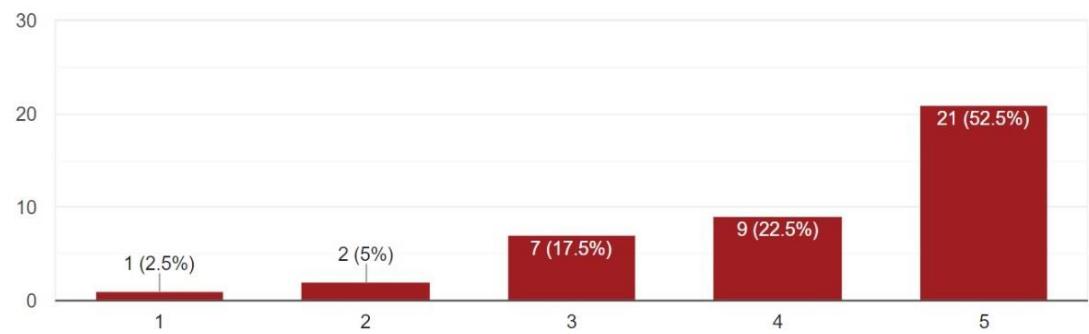
How fast response did you get for the emergency service?

40 responses

**9) How would you rate our emergency service?**

How would you rate our emergency service?

40 responses

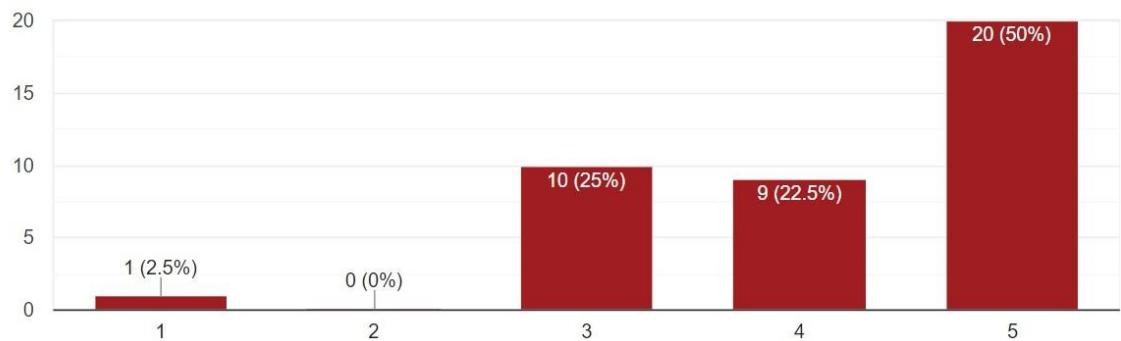


10) How would you rate our regular service?

How would you rate our regular service?



40 responses



1.2.4 Observation:

- First of all, we observed the number of vehicles in customer's house and found out that most of the customers are having 2 vehicles.
- Then we can see that around 35% users visit service center once in a six month and there 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 their vehicle.
- 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 emergency services 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 for emergency services.
- Almost 53% rated our emergency services as very good and 50% users rated our regular 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 of the 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 system and crucial delays on update of data	Interview and Background Readings	Developer, Current system operators	1 hour
To understand the flow of system and setbacks where system might fail	Observation and using the current system	Employees, SRS document writer	1 hour
Execution of suggestion provided by customers	Interview	Department Head and Stakeholder	2 hours
Setting priority on the different phases	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, joining date, and so on for employees working in a service centre.
- Customer information such as customer id, customer name, order date, and so on should be kept track of.
- It should keep track of information about the service center's various services, as well as any applicable discounts.
- It should keep track of order_id, order_date, service_id, service_price, payment_details, 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 from 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 ones who are 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 ones 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 of the 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 who will 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 it in 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 and also see the available materials for servicing.
 - **Employee:** can see the list of services that they are supposed to do and also check the parts that they need for assigned service.
 - **Customer:** can only see the services that they can get from service centre.
 - 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 software to 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 Constraints

- Details regarding the services provided, as well as their availability and status, should be 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 service 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 will assign that slot to that particular 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, customer 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	<u>Customer_id</u> , customer_name, contact_number, address	Pays, books, gives, assigns
Employee	<u>Employee_id</u> , employee_name, contact_number, address, salary	Joins, assigns
purchase	<u>Purchase_id</u> , part_id, units, cost, purhcase_date	
Inventory	<u>Part_id</u> , center_id, part_name, stock, retal_price	
offers	<u>Offer_id</u> , discount_percentage, max_discount, start_date, end_date	
center	<u>Center_id</u> , city, address, contact	

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

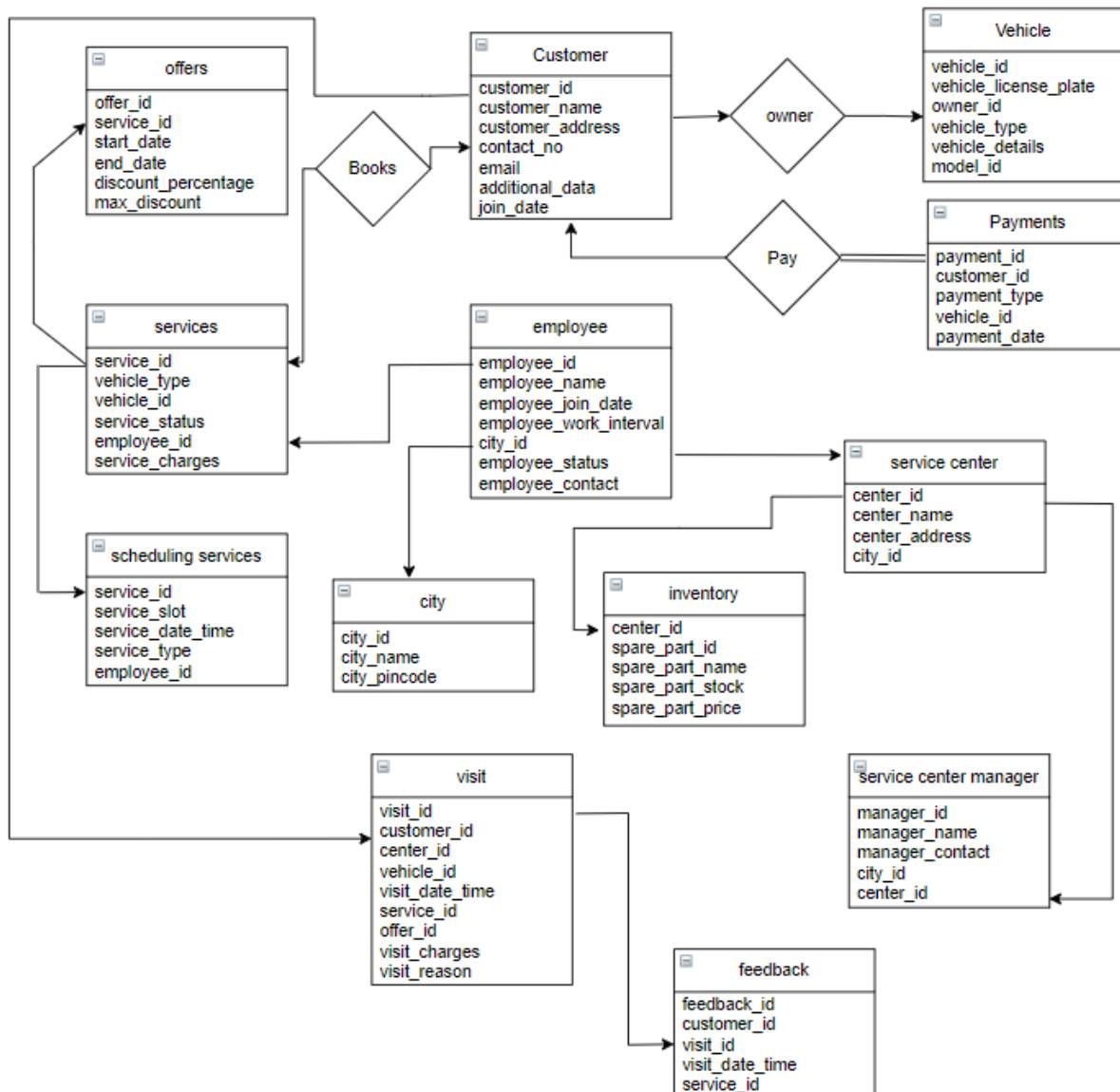
Entity Set	Entity Set type	Identifying Relation	Identifying Entity Set
Scheduling_service	Weak entity	Schedules	Order_service
payments	Weak entity	Pays	Customers

2.6 Identify Relationship Types

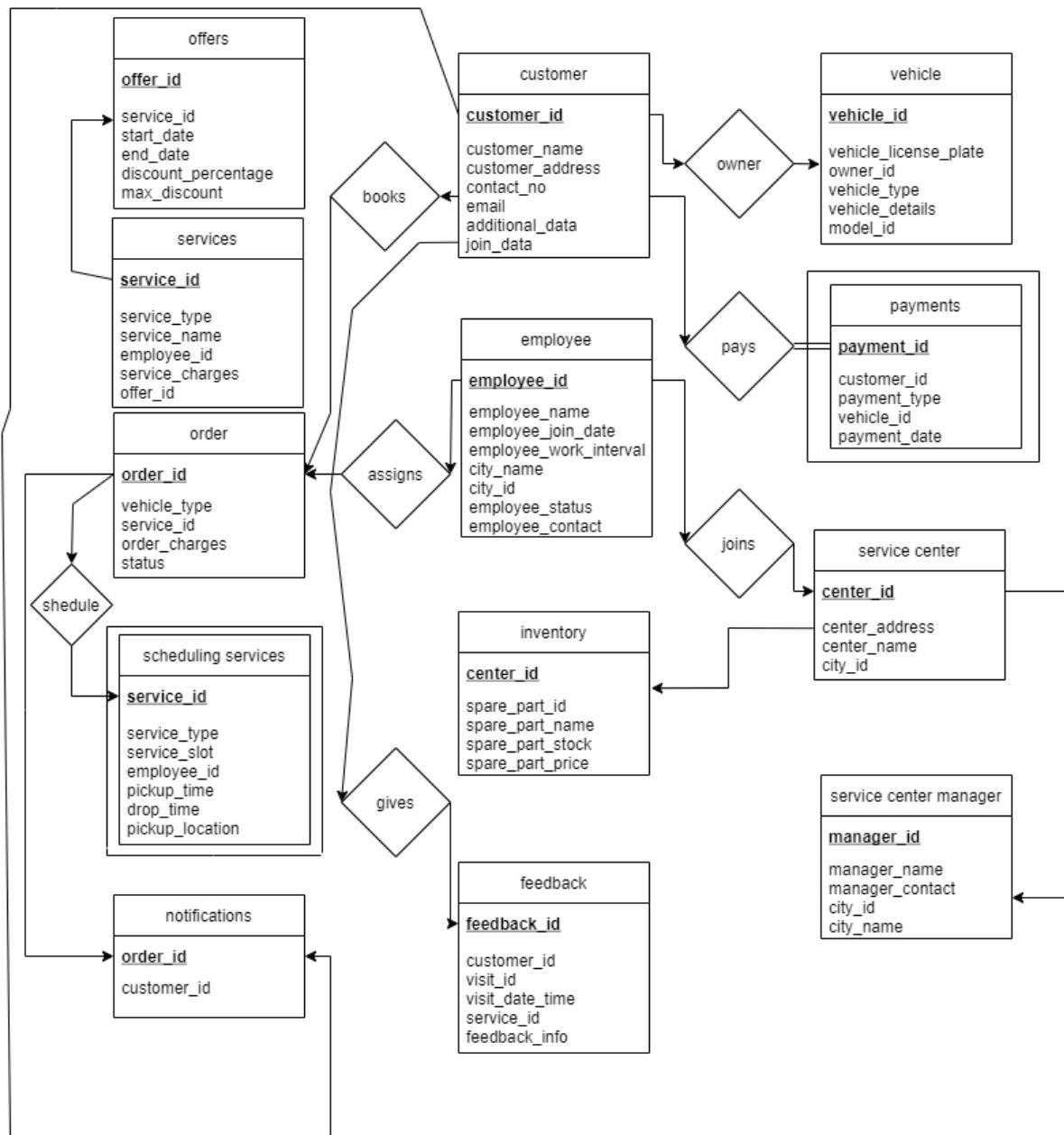
Relationship Name	Relationship Type
Pays	Ternary
Books	Binary
Gives	Binary
Assigns	Binary
Schedules	Binary
Joins	Binary
Owner	Binary

Section3: ER-Diagrams all versions

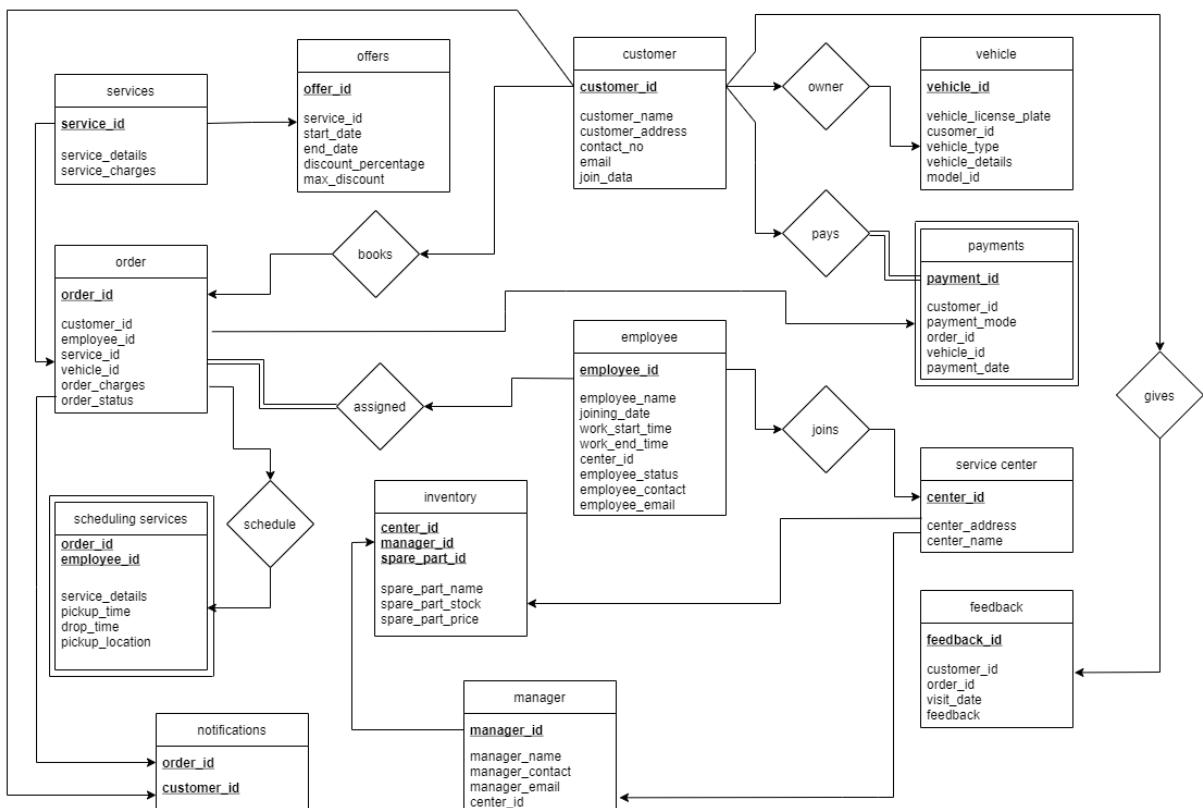
3.1 E-R Diagram (ERD) Version 1



3.2 E-R Diagram (ERD) Version 2

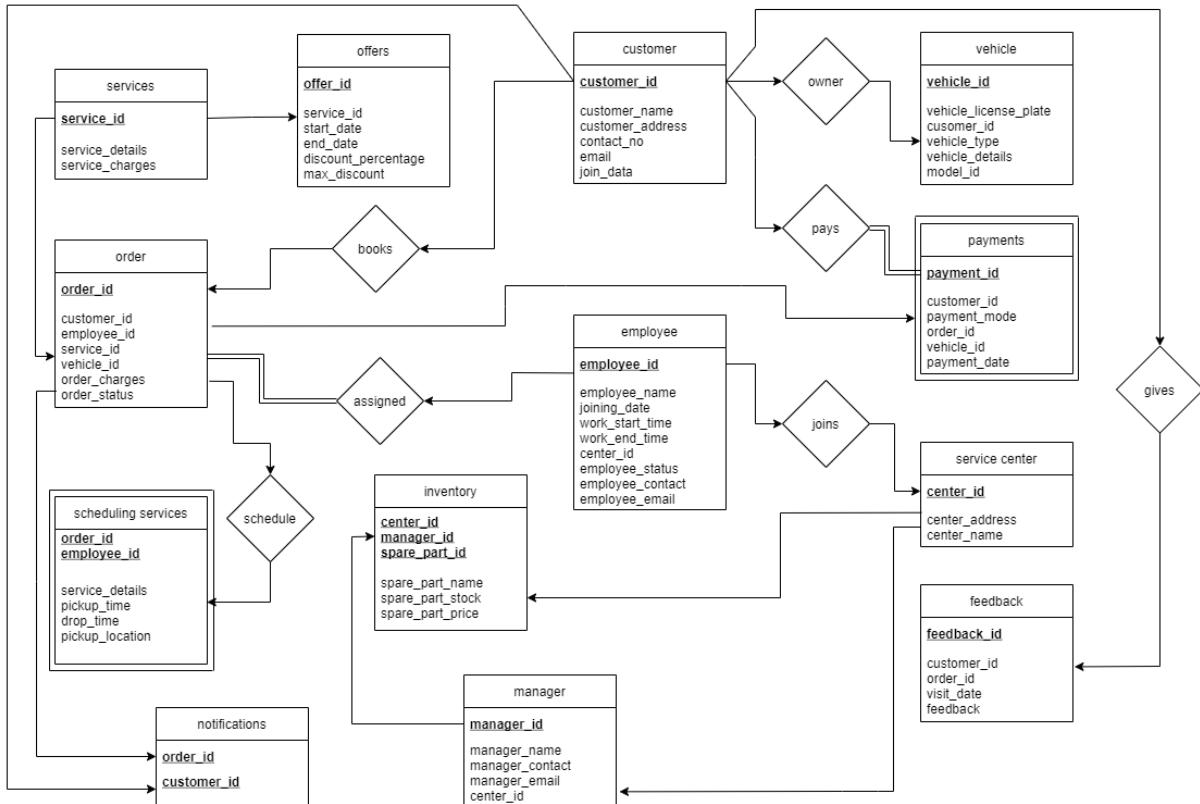


3.3 Final E-R Diagram (ERD)



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

4.1 Final Version of E-R Diagram

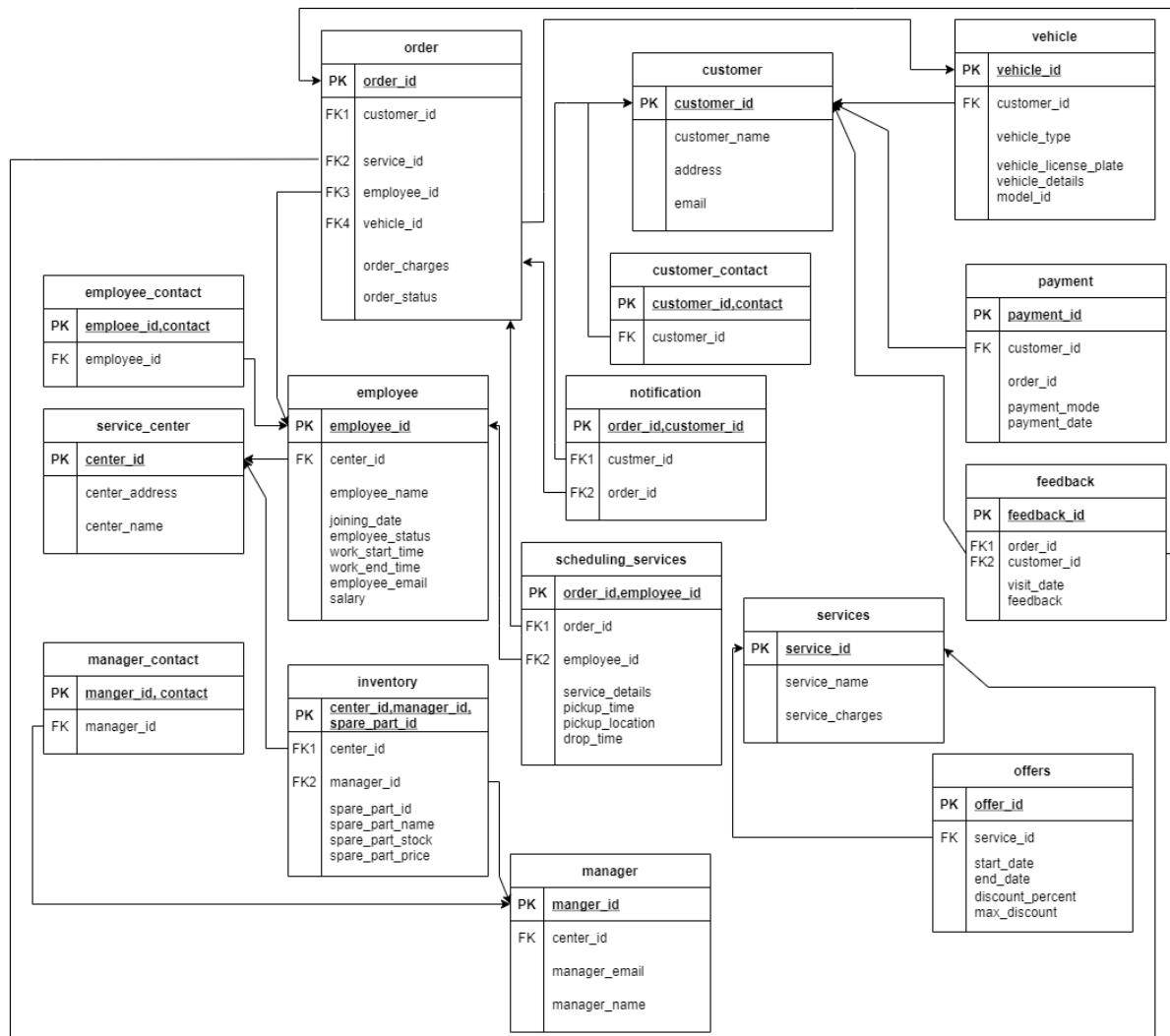


4.2 Mapping ER Model to Relation Model:

- 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_percentage, 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_charges, 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 → centre_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 foreign 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_percentage, 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: $\text{order_id} \rightarrow \text{employee_id}$, $\text{order_id} \rightarrow \text{customer_id}$, $\text{order_id} \rightarrow \text{service_id}$, $\text{order_id} \rightarrow \text{vehicle_id}$, $\text{order_id} \rightarrow \text{order_chardes}$, $\text{order_id} \rightarrow \text{order_status}$
 - 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: $(\text{order_id}, \text{employee_id}) \rightarrow \text{service_details}$, $(\text{order_id}, \text{employee_id}) \rightarrow \text{pickup_time}$, $(\text{order_id}, \text{employee_id}) \rightarrow \text{drop_time}$, $(\text{order_id}, \text{employee_id}) \rightarrow \text{pickup_location}$
 - 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 $\text{scheduling_services}$ relation hence this relation is in 3NF.
 - Already in BCNF

- **Notifications:**
 - Updated FD: $(\text{order_id}, \text{customer_id}) \rightarrow \text{customer_id}$
 - 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: $\text{vehicle_id} \rightarrow \text{customer_id}$, $\text{vehicle_id} \rightarrow \text{vehicle_license_plate}$, $\text{vehicle_id} \rightarrow \text{vehicle_type}$, $\text{vehicle_id} \rightarrow \text{vehicle_details}$, $\text{vehicle_id} \rightarrow \text{model_id}$
 - 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: $\text{payment_id} \rightarrow \text{customer_id}$, $\text{payment_id} \rightarrow \text{order_id}$, $\text{payment_id} \rightarrow \text{vehicle_id}$, $\text{payment_id} \rightarrow \text{payment_mode}$, $\text{payment_id} \rightarrow \text{payment_date}$
 - 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: $\text{centre_id} \rightarrow \text{centre_address}$, $\text{centre_id} \rightarrow \text{centre_name}$
 - 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: $\text{manager_id} \rightarrow \text{centr_id}$, $\text{manager_id} \rightarrow \text{manager_name}$, $\text{manager_id} \rightarrow \text{manager_email}$
 - 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

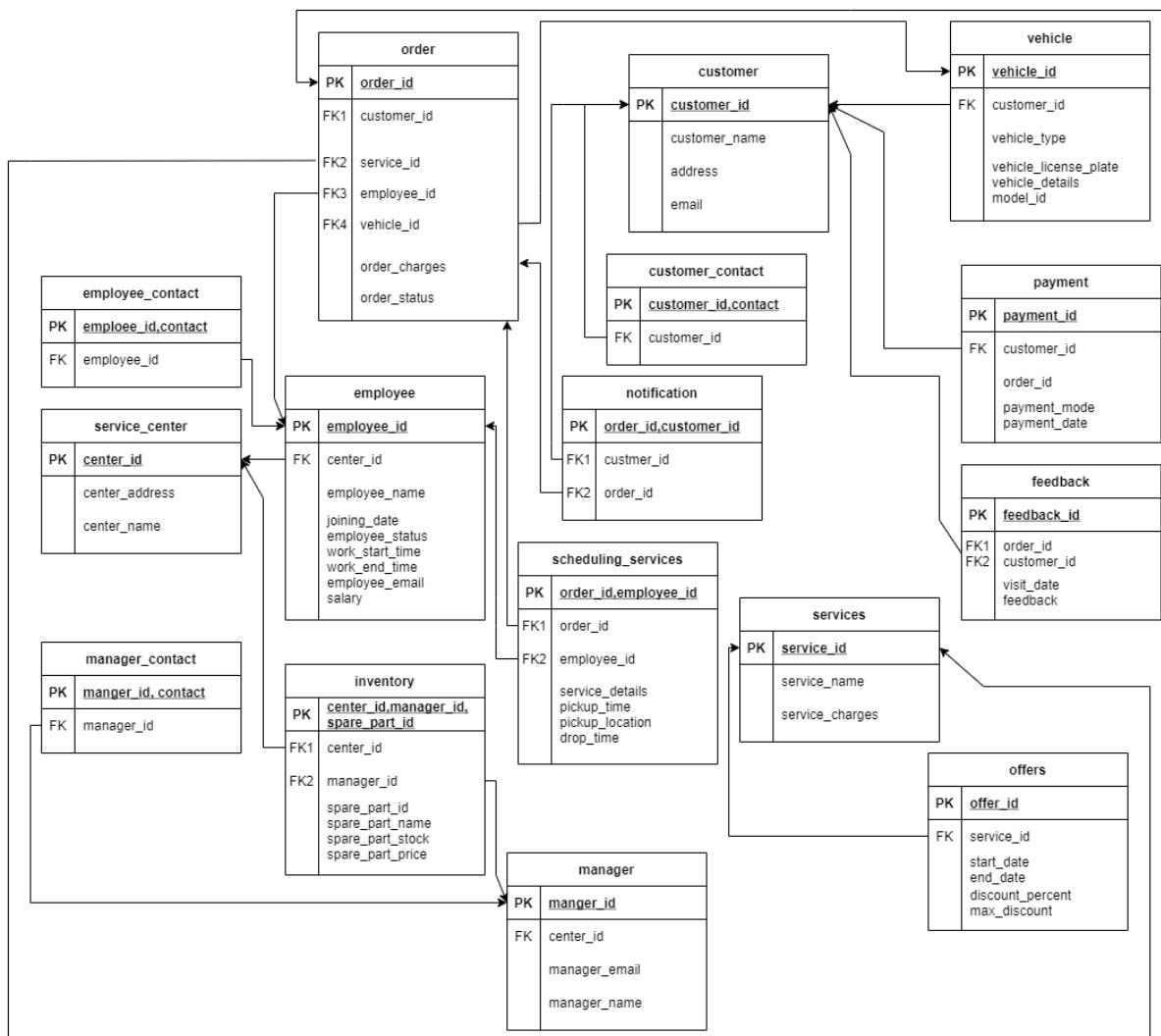
- Inventory:**

- Updated FD: $(\text{centre_id}, \text{manager_id}) \rightarrow \text{spare_part_id}$, $(\text{centre_id}, \text{manager_id}) \rightarrow \text{spare_part_name}$, $(\text{centre_id}, \text{manager_id}) \rightarrow \text{spare_part_stock}$, $(\text{centre_id}, \text{manager_id}) \rightarrow \text{spare_part_price}$
- All non-primary attributes are fully functionally dependent on $(\text{part_id}, \text{center_id})$ 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: $\text{feedback_id} \rightarrow \text{customer_id}$, $\text{feedback_id} \rightarrow \text{order_id}$, $\text{feedback_id} \rightarrow \text{visit_date}$, $\text{feedback_id} \rightarrow \text{feedback}$
- 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):

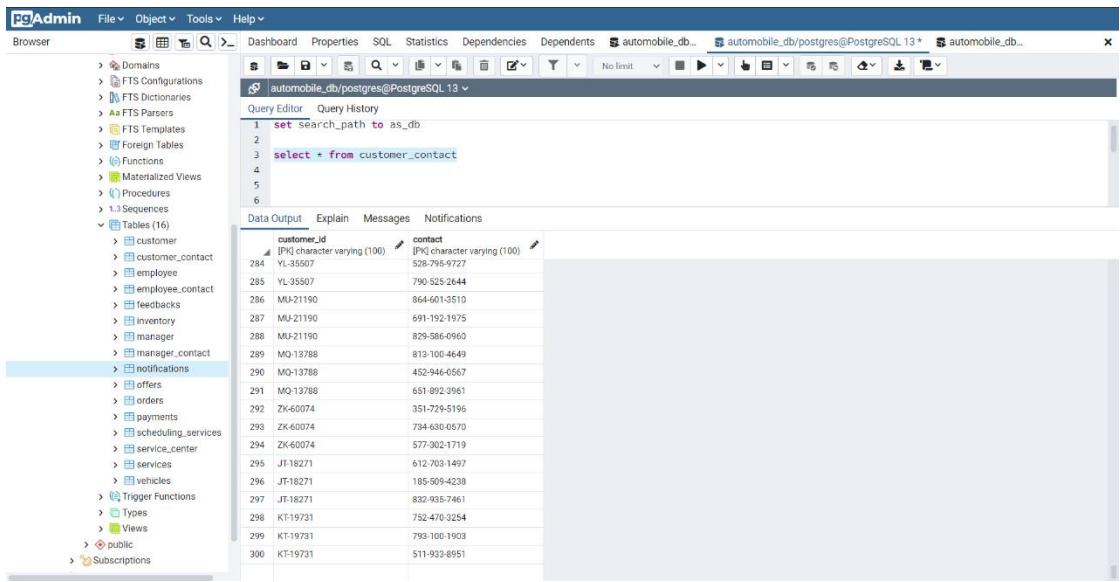
- **customer_id** references to **vehicle table, payment table, feedback table, order table** and **notifications table**.
- **service_id** references to **offers table** and **order table**.
- **order_id** references to **scheduling services table, payments table** and **notifications table**.
- **employee_id**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

customer_id	customer_name	customer_address	email
JN-65996	Madlen	46587 Ridgeway Drive	mroret2@lycos.com
BL-45631	Artair	077 Derek Road	aeseler2@opensource.org
AZ-65559	Winnah	0 Lighthouse Bay Lane	wlibert2d@pageperso-orange.fr
ZM-47914	Bond	37966 Hollow Ridge Circle	bkaire2@microsoft.com
ML-72426	Pia	003 Donald Center	pschulter2@comsenz.com
XO-04199	Skip	72 Bulman Plaza	svchenk2g@unesco.org
RT-33604	Sela	75 Anzinger Crossing	schick2h@vimeo.com
DJ-51408	Randie	31552 Birchwood Pass	rtown2@friendfeed.com
MJ-02310	Luce	9789 Kings Plaza	lkitteridge2j@admin.ch
ZF-44812	Em	923 Wayridge Alley	ecraigton2j@studiodpress.com
OY-50381	Enrica	137 Southridge Plaza	eavashri2j@virginia.edu
YL-35507	Haley	382 Continental Lane	hskechley2m@prnewswire.com
MU-21190	Murry	2 Lien Center	mmpo2n@feedburner.com
MQ-13788	Suki	254 Cody Hill	sreith2q@theglobemailmail.com
ZK-60074	Ludovika	954 Leroy Point	llapre2q@theglobemailmail.com
JT-19271	Ralf	73 Northport Terrace	rbeste2q@oalc.gov.au
KT-19731	Olive	3942 Kedzie Court	oleafe2r@un.org

ii. Customer_contact



The screenshot shows the pgAdmin interface with the 'automobile_db' database selected. The left sidebar lists various database objects like Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (16), Triggers, Types, Views, and Subscriptions. The 'Tables (16)' section is expanded, and the 'customer_contact' table is selected. The main pane displays the table's data output. The SQL query in the Query Editor is:

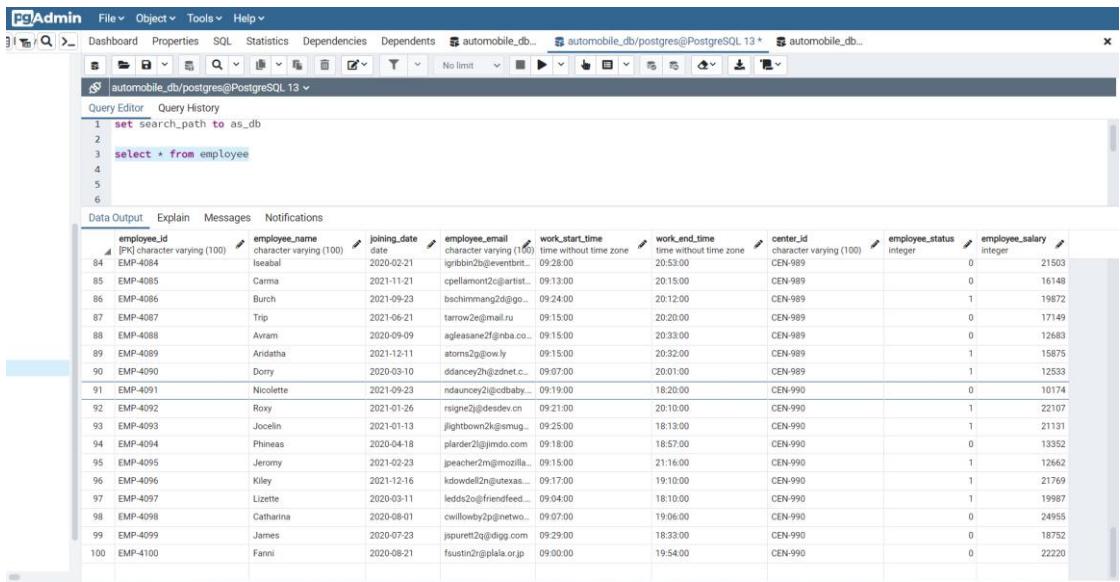
```

1 set search_path to as_db
2
3 select * from customer_contact
4
5
6
    
```

The Data Output pane shows the following data:

customer_id	contact
YL-35507	528-795-2644
MU-21190	864-601-3510
MU-21190	691-192-1975
MU-21190	829-586-6960
MQ-13788	813-100-4649
MQ-13788	452-946-0567
MQ-13788	651-992-3961
ZK-60074	351-729-5196
ZK-60074	734-630-0570
ZK-60074	577-302-1719
JT-18271	612-703-1497
JT-18271	185-509-4238
JT-18271	832-935-7461
KT-19731	752-470-5254
KT-19731	793-100-1903
KT-19731	511-933-6951

iii. Employee



The screenshot shows the pgAdmin interface with the 'automobile_db' database selected. The left sidebar lists various database objects like Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, Tables (16), Triggers, Types, Views, and Subscriptions. The 'Tables (16)' section is expanded, and the 'employee' table is selected. The main pane displays the table's data output. The SQL query in the Query Editor is:

```

1 set search_path to as_db
2
3 select * from employee
4
5
6
    
```

The Data Output pane shows the following data:

employee_id	employee_name	joining_date	employee_email	work_start_time	work_end_time	center_id	employee_status	employee_salary
EMP-4084	Iseabal	2020-02-21	igribbon2@eventbrite...	09:28:00	20:53:00	CEN-989	0	21503
EMP-4085	Carma	2021-11-21	cpellamont2@garitz...	09:13:00	20:15:00	CEN-989	0	16148
EMP-4086	Burch	2021-09-23	bschimmang2@go...go...	09:24:00	20:12:00	CEN-989	1	19872
EMP-4087	Trip	2021-06-21	tarrow2@mail.ru	09:15:00	20:20:00	CEN-989	0	17149
EMP-4088	Aviram	2020-09-09	aglesan2@nba.co...	09:15:00	20:33:00	CEN-989	0	12683
EMP-4089	Aridatha	2021-12-11	atoms2@cow.ly	09:15:00	20:32:00	CEN-989	1	15875
EMP-4090	Dory	2020-03-10	odarcey2@zmet.c...	09:07:00	20:01:00	CEN-989	1	12533
EMP-4091	Nicolette	2021-09-23	ndauncey2@coldbab...	09:19:00	18:20:00	CEN-990	0	10174
EMP-4092	Rox	2021-01-26	rsign2j@fesdev.cn	09:21:00	20:10:00	CEN-990	1	22107
EMP-4093	Jocelin	2021-01-13	jlightbow2k@smug...	09:25:00	18:13:00	CEN-990	1	21131
EMP-4094	Phineas	2020-04-18	plander2@jimdo.com	09:18:00	18:57:00	CEN-990	0	13352
EMP-4095	Jeremy	2021-02-23	jpeacher2m@mozilla...	09:15:00	21:16:00	CEN-990	1	12662
EMP-4096	Kiley	2021-12-16	kdownell2@utexas...	09:17:00	19:10:00	CEN-990	1	21769
EMP-4097	Lizette	2020-03-11	ledds2o@friendfeed...	09:04:00	18:10:00	CEN-990	1	19987
EMP-4098	Catharina	2020-08-01	cvillowy2p@netwro...	09:07:00	19:06:00	CEN-990	0	24955
EMP-4099	James	2020-07-23	jspurett2@idigg.com	09:29:00	18:33:00	CEN-990	0	18752
EMP-4100	Fanni	2020-08-21	fsustin2@plala.or.jp	09:00:00	19:54:00	CEN-990	0	22220

iv. Employee_contact

The screenshot shows the pgAdmin interface with the 'automobile_db' database selected. In the left sidebar, under 'Tables (16)', the 'employee_contact' table is highlighted. The main pane displays the table's data output. The query in the 'Query Editor' is:

```
1 set search_path to as_db
2
3 select * from employee_contact
4
5
```

The data output shows the following rows:

employee_id	contact
EMP-4092	516-867-0990
EMP-4093	201-487-0710
EMP-4093	343-428-2307
EMP-4094	433-712-0482
EMP-4094	962-128-5820
EMP-4095	287-952-1591
EMP-4095	485-977-1830
EMP-4096	192-236-4094
EMP-4096	710-631-6641
EMP-4097	601-808-9411
EMP-4097	998-645-9763
EMP-4098	798-456-9352
EMP-4098	526-394-5802
EMP-4099	501-951-4874
EMP-4099	833-417-4306
EMP-4100	543-637-6308
EMP-4100	283-145-6815

v. Feedback

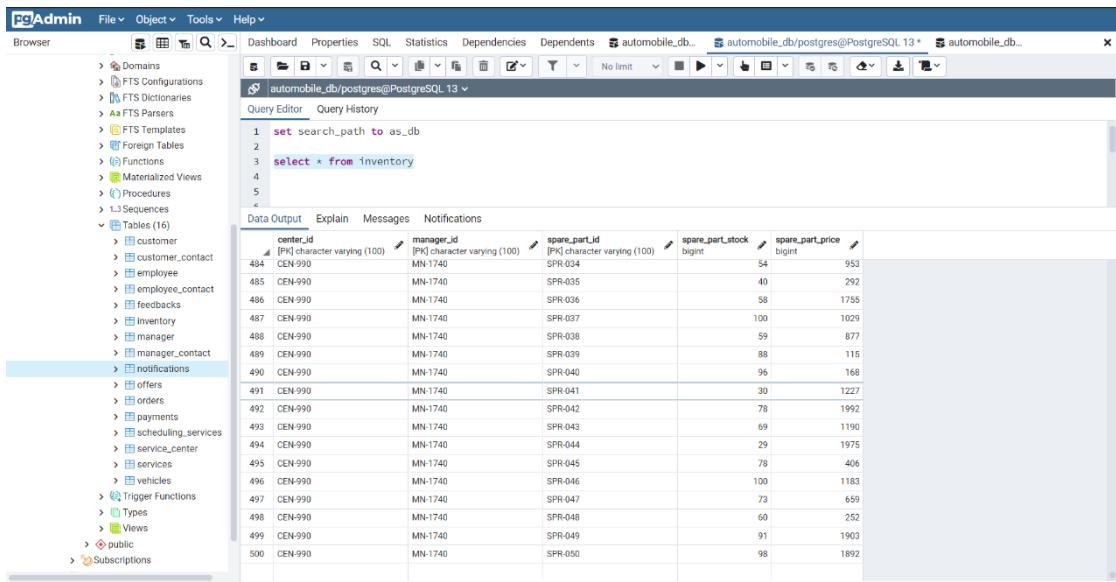
The screenshot shows the pgAdmin interface with the 'automobile_db' database selected. In the left sidebar, under 'Tables (16)', the 'feedbacks' table is highlighted. The main pane displays the table's data output. The query in the 'Query Editor' is:

```
1 set search_path to as_db
2
3 select * from feedbacks
4
5
```

The data output shows the following rows:

feedback_id	customer_id	order_id	visit_date	feedback
FED-2105	BL-45631	ODR-185	2019-03-27	Curabitur at ipsum ac tellus semper interdum.
FED-2106	AZ-65599	ODR-186	2019-01-23	Maecenas tincidunt laicus at velit. Vivamus vel nulla eget eros elementum pellentesque.
FED-2107	ZM-47914	ODR-187	2018-03-21	Nullam varius. Nulla facilisi.
FED-2108	ML-72426	ODR-188	2019-11-27	Grae pellentesque volutpat du.
FED-2109	XO-04199	ODR-189	2019-01-09	Integer non velit. Donec diam neque, vestibulum eget, vulputate ut, ultrices vel, augue.
FED-2110	RT-33504	ODR-190	2019-09-04	Nulla neque libero, convallis eget, eleifend luctus, ultricies eu, nibh. Quisque id justo sit amet sapien.
FED-2111	DJ-51408	ODR-191	2019-04-18	Nam dui.
FED-2112	MJ-02310	ODR-192	2018-06-26	Quisque erat eros, viverra eget, congue eget, semper rutrum, nulla. Nunc purus.
FED-2113	ZF-44612	ODR-193	2018-11-13	Pellentesque eget nunc. Donec quis orci eget orci vehicula condimentum.
FED-2114	OY-50381	ODR-194	2019-09-28	Donec quis orci eget orci vehicula condimentum. Curabitur in libero ut massa volutpat convallis.
FED-2115	YL-35507	ODR-195	2018-01-16	Præsent id massa id nisl venenatis facilia. Aenean sit amet justo.
FED-2116	MU-21190	ODR-196	2019-10-25	Nullam varius.
FED-2117	MQ-13788	ODR-197	2019-06-16	Donec dapibus. Duis at velit eu est congue elementum.
FED-2118	ZK-60074	ODR-198	2018-06-28	Morbi quis tortor id nulla ultrices aliquet.
FED-2119	JT-18271	ODR-199	2019-05-07	Donec ut dolor.
FED-2120	KT-19731	ODR-200	2018-11-07	Nulla nisl. Nunc nisl.

vi. Inventory



The screenshot shows the pgAdmin interface with the 'Query Editor' tab active. The query entered is:

```

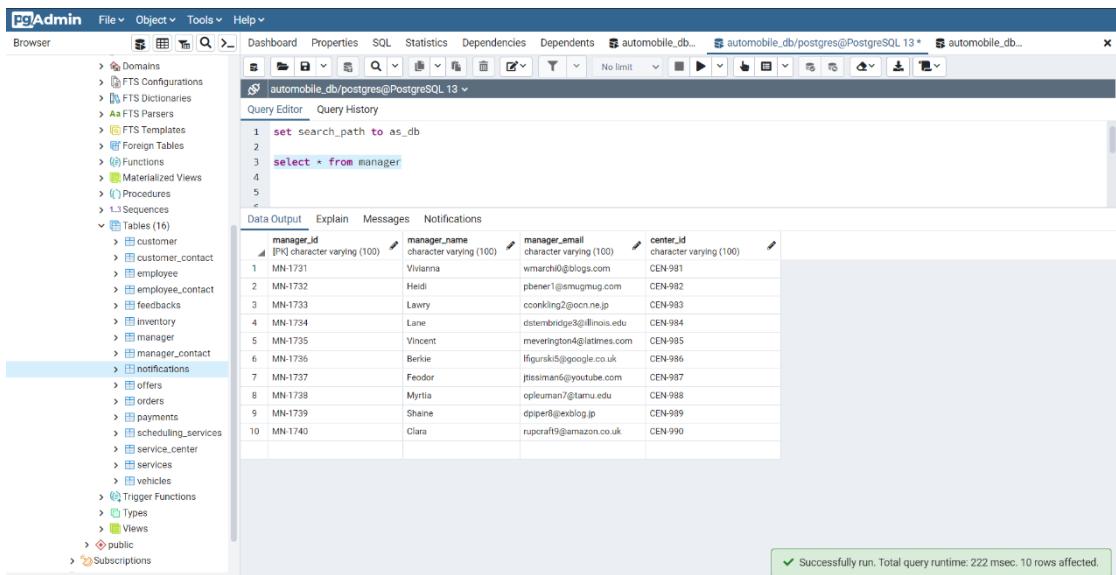
1 set search_path to as_db
2
3 select * from inventory
4
5

```

The results are displayed in a table titled 'Data Output' with the following columns: center_id, manager_id, spare_part_id, spare_part_stock, and spare_part_price. The data consists of 500 rows, each representing an inventory item with its center ID, manager ID, part ID, stock level, and price.

center_id	manager_id	spare_part_id	spare_part_stock	spare_part_price
484	CEN-990	SPR-034	54	953
485	CEN-990	SPR-035	40	292
486	CEN-990	SPR-036	58	1755
487	CEN-990	SPR-037	100	1029
488	CEN-990	SPR-038	59	877
489	CEN-990	SPR-039	88	115
490	CEN-990	SPR-040	96	168
491	CEN-990	SPR-041	30	1227
492	CEN-990	SPR-042	78	1992
493	CEN-990	SPR-043	69	1190
494	CEN-990	SPR-044	29	1975
495	CEN-990	SPR-045	78	406
496	CEN-990	SPR-046	100	1183
497	CEN-990	SPR-047	73	659
498	CEN-990	SPR-048	60	252
499	CEN-990	SPR-049	91	1903
500	CEN-990	SPR-050	98	1892

vii. Manager



The screenshot shows the pgAdmin interface with the 'Query Editor' tab active. The query entered is:

```

1 set search_path to as_db
2
3 select * from manager
4
5

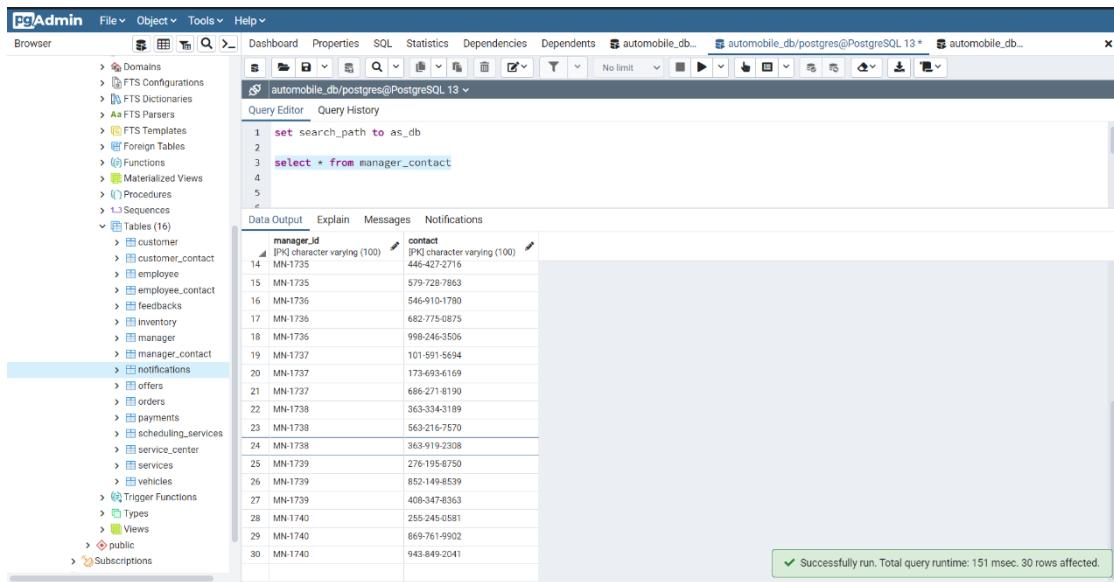
```

The results are displayed in a table titled 'Data Output' with the following columns: manager_id, manager_name, manager_email, and center_id. The data consists of 10 rows, each representing a manager with their ID, name, email, and assigned center ID.

manager_id	manager_name	manager_email	center_id
MN-1731	Viviana	wmarchi@blogs.com	CEN-981
MN-1732	Heidi	pbener1@smugmug.com	CEN-982
MN-1733	Lavry	cooking@ocn.ne.jp	CEN-983
MN-1734	Lane	dsterbridge@illinois.edu	CEN-984
MN-1735	Vincent	meverington@latimes.com	CEN-985
MN-1736	Berkie	lfigurski@google.co.uk	CEN-986
MN-1737	Feodor	jtsimian@youtube.com	CEN-987
MN-1738	Myrlia	opleuman@stan.edu	CEN-988
MN-1739	Shaine	dpiper8@exblog.jp	CEN-989
MN-1740	Clara	ruporat9@amazon.co.uk	CEN-990

✓ Successfully run. Total query runtime: 222 msec. 10 rows affected.

viii. Manager_contact



The screenshot shows the pgAdmin interface with the 'Query Editor' tab active. The query executed is:

```

1 set search_path to as_db
2
3 select * from manager_contact
4
5

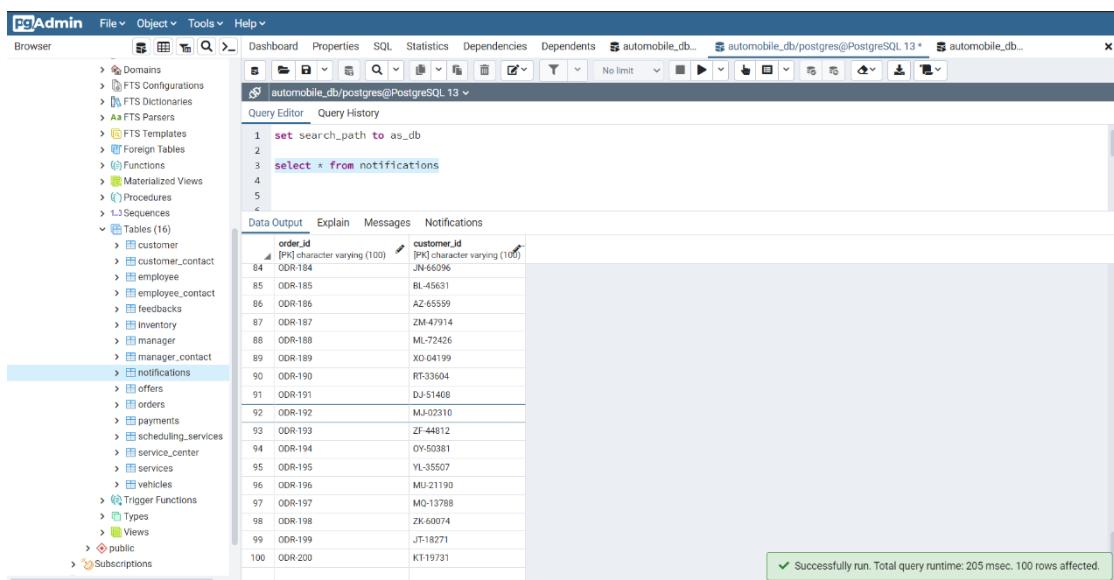
```

The results are displayed in a table titled 'Data Output' with three columns: 'manager_id', 'contact', and 'customer_id'. The data consists of 30 rows, each containing a unique identifier and a contact name. A green message at the bottom right indicates the query was successfully run with a total runtime of 151 msec and 30 rows affected.

manager_id	contact	customer_id
MN-1735	446-427-2716	
MN-1735	579-728-7863	
MN-1736	546-910-1780	
MN-1736	682-775-5875	
MN-1736	998-246-3506	
MN-1737	101-591-5694	
MN-1737	173-693-6169	
MN-1737	686-271-8190	
MN-1738	363-334-3189	
MN-1738	563-216-7570	
MN-1738	363-919-2308	
MN-1739	276-195-6750	
MN-1739	852-149-8539	
MN-1739	408-347-8363	
MN-1740	255-245-0581	
MN-1740	869-761-9902	
MN-1740	943-849-2041	

✓ Successfully run. Total query runtime: 151 msec. 30 rows affected.

ix. Notification



The screenshot shows the pgAdmin interface with the 'Query Editor' tab active. The query executed is:

```

1 set search_path to as_db
2
3 select * from notifications
4
5

```

The results are displayed in a table titled 'Data Output' with three columns: 'order_id', 'customer_id', and 'customer_name'. The data consists of 100 rows, each containing an order ID and a customer ID. A green message at the bottom right indicates the query was successfully run with a total runtime of 205 msec and 100 rows affected.

order_id	customer_id	customer_name
ODR-184	JN-66096	
ODR-185	BL-45631	
ODR-186	AZ-65559	
ODR-187	ZM-47914	
ODR-188	ML-72426	
ODR-189	XO-04199	
ODR-190	RT-33504	
ODR-191	DJ-51408	
ODR-192	MJ-02310	
ODR-193	ZF-44812	
ODR-194	OY-50381	
ODR-195	YL-35507	
ODR-196	MU-21190	
ODR-197	MO-13788	
ODR-198	ZK-60074	
ODR-199	JT-18271	
ODR-200	KT-19731	

✓ Successfully run. Total query runtime: 205 msec. 100 rows affected.

x. Offers

The screenshot shows the pgAdmin interface with the 'Browser' tab selected. The left sidebar lists various database objects under the 'Tables (16)' section, with 'notifications' currently highlighted. The main area displays the results of a SQL query:

```

1 set search_path to as_db
2
3 select * from offers
4
5
6
7
8
9
10
11
12
13
14
15

```

Data Output

offer_id	service_id	start_date	end_date	discount_percentage	max_discount
OFR-1001	SEC-7101	2019-04-18	2020-09-30	10	103
OFR-1002	SEC-7102	2018-06-26	2020-12-31	7	265
OFR-1003	SEC-7103	2018-11-13	2020-05-03	3	95
OFR-1004	SEC-7104	2019-09-28	2021-04-01	10	359
OFR-1005	SEC-7105	2018-01-16	2020-09-27	3	389
OFR-1006	SEC-7106	2019-10-25	2021-09-07	3	274
OFR-1007	SEC-7107	2019-06-16	2021-07-28	6	162
OFR-1008	SEC-7108	2018-06-28	2021-10-16	8	381
OFR-1009	SEC-7109	2019-05-07	2020-07-19	2	74
OFR-1010	SEC-7110	2018-11-07	2021-03-16	6	186
OFR-1011	SEC-7111	2019-04-12	2021-03-24	5	417
OFR-1012	SEC-7112	2018-07-16	2020-03-17	7	473
OFR-1013	SEC-7113	2019-10-04	2020-07-01	8	106
OFR-1014	SEC-7114	2018-10-26	2021-06-07	2	443
OFR-1015	SEC-7115	2018-12-24	2020-05-15	3	473

Successfully run. Total query runtime: 325 msec. 15 rows affected.

xi. Orders

The screenshot shows the pgAdmin interface with the 'Browser' tab selected. The left sidebar lists various database objects under the 'Tables (16)' section, with 'notifications' currently highlighted. The main area displays the results of a SQL query:

```

1 set search_path to as_db
2
3 select * from orders
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Data Output

order_id	customer_id	employee_id	service_id	vehicle_id	order_charges	order_status
ODR-184	JN-66096	EMP-4084	SEC-7114	VHL-5084	4285	0
ODR-185	BL-45631	EMP-4085	SEC-7115	VHL-5085	4243	0
ODR-186	AZ-65559	EMP-4086	SEC-7111	VHL-5086	1329	0
ODR-187	ZM-47914	EMP-4087	SEC-7103	VHL-5087	1807	1
ODR-188	ML-72426	EMP-4088	SEC-7105	VHL-5088	4982	2
ODR-189	XO-04199	EMP-4089	SEC-7107	VHL-5089	2005	2
ODR-190	RT-33604	EMP-4090	SEC-7109	VHL-5090	2515	0
ODR-191	DJ-51408	EMP-4091	SEC-7102	VHL-5091	3686	0
ODR-192	MJ-02310	EMP-4092	SEC-7101	VHL-5092	4159	2
ODR-193	ZF-44812	EMP-4093	SEC-7102	VHL-5093	4096	1
ODR-194	OY-50381	EMP-4094	SEC-7103	VHL-5094	3751	2
ODR-195	YL-35507	EMP-4095	SEC-7108	VHL-5095	4320	1
ODR-196	MU-21190	EMP-4096	SEC-7109	VHL-5096	832	2
ODR-197	MO-13788	EMP-4097	SEC-7110	VHL-5097	4796	0
ODR-198	ZK-60074	EMP-4098	SEC-7109	VHL-5098	4151	0
ODR-199	JT-18271	EMP-4099	SEC-7115	VHL-5099	1753	0
ODR-200	KT-19731	EMP-4100	SEC-7114	VHL-5100	1328	1

- **order_status:**

- 0- pending
- 1- in progress
- 2- completed

xii. Payment

The screenshot shows the pgAdmin interface with the 'automobile_db' database selected. In the left sidebar, under 'Tables (16)', the 'payments' table is highlighted. The 'Query Editor' tab contains the following SQL code:

```

1 set search_path to as_db
2
3 select * from payments
4
5

```

The 'Data Output' tab displays the results of the query, which are as follows:

payment_id	customer_id	payment_mode	order_id	vehicle_id	payment_date
PAY-3094	JN-65096	1	ODR-184	VHL-5084	2018-03-30
PAY-3085	BL-45631	1	ODR-185	VHL-5085	2019-03-27
PAY-3096	AZ-65559	1	ODR-186	VHL-5086	2019-01-23
PAY-3087	ZM-47914	0	ODR-187	VHL-5087	2018-03-21
PAY-3088	ML-72426	0	ODR-188	VHL-5088	2019-11-27
PAY-3089	XO-04199	1	ODR-189	VHL-5089	2019-01-09
PAY-3090	RT-33604	0	ODR-190	VHL-5090	2019-09-04
PAY-3091	DJ-51408	0	ODR-191	VHL-5091	2019-04-18
PAY-3092	MJ-02310	0	ODR-192	VHL-5092	2018-06-26
PAY-3093	ZF-44812	1	ODR-193	VHL-5093	2018-11-13
PAY-3094	OY-50381	0	ODR-194	VHL-5094	2019-09-28
PAY-3095	YL-35507	0	ODR-195	VHL-5095	2018-01-16
PAY-3096	MU-21190	1	ODR-196	VHL-5096	2019-10-25
PAY-3097	MO-13788	1	ODR-197	VHL-5097	2019-06-16
PAY-3098	ZK-60074	1	ODR-198	VHL-5098	2018-06-28
PAY-3099	JT-18271	1	ODR-199	VHL-5099	2019-05-07
PAY-3100	KT-19731	0	ODR-200	VHL-	

A green success message at the bottom right of the results pane states: "Successfully run. Total query runtime: 285 msec. 100 rows affected."

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

xiii. Scheduling_service

The screenshot shows the pgAdmin interface with the 'automobile_db' database selected. In the left sidebar, under 'Tables (16)', the 'scheduling_services' table is highlighted. The 'Query Editor' tab contains the following SQL code:

```

1 set search_path to as_db
2
3 select * from scheduling_services
4
5

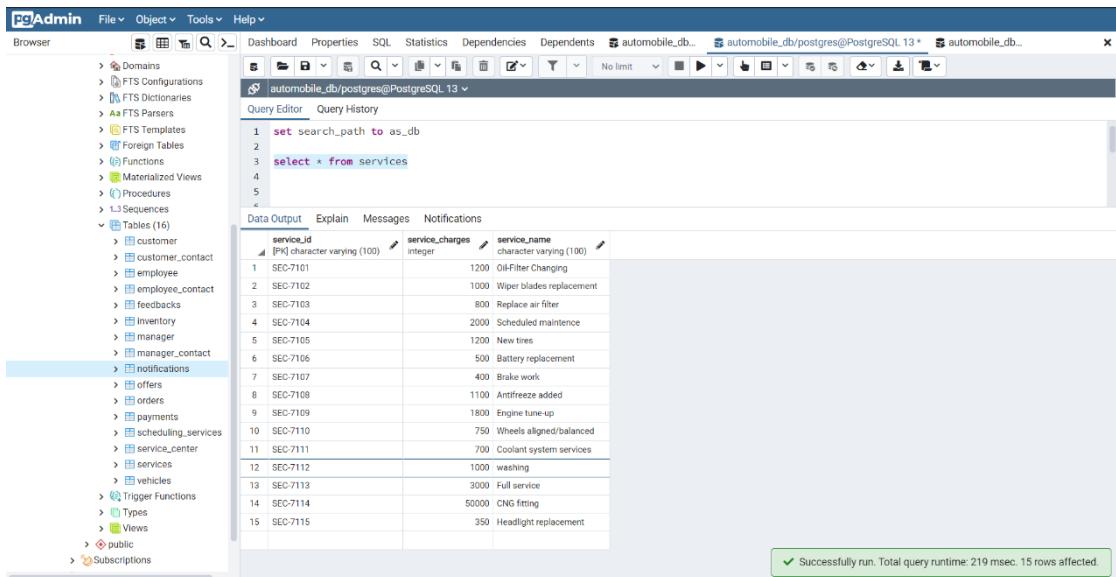
```

The 'Data Output' tab displays the results of the query, which are as follows:

order_id	employee_id	pick_up_time	drop_time	pick_up_location	service_name
ODR-184	EMP-4034	12:03:00	15:37:00	13 Mitchell Pass	Engine tune-up
ODR-185	EMP-4085	12:14:00	16:48:00	64 Sugar Alley	washing
ODR-186	EMP-4085	13:02:00	16:14:00	394 Derek Center	Full service
ODR-187	EMP-4087	11:48:00	15:34:00	2938 Del Sol Street	CNG fitting
ODR-188	EMP-4088	10:28:00	16:57:00	20423 Saint Paul Hill	Headlight replacement
ODR-189	EMP-4089	10:50:00	16:49:00	612 Park Meadow Junction	Oil-Filter Changing
ODR-190	EMP-4090	13:36:00	15:36:00	9804 Dexter Junction	Wiper blades replacement
ODR-191	EMP-4091	11:48:00	15:43:00	22716 Melby Hill	Antifreeze added
ODR-192	EMP-4092	09:33:00	16:48:00	9 Superior Place	Engine tune-up
ODR-193	EMP-4093	09:17:00	15:47:00	009 Haul Terrace	Wheels aligned/balanced
ODR-194	EMP-4094	09:10:00	15:18:00	6390 Marcy Point	Coolant system services
ODR-195	EMP-4095	12:32:00	16:02:00	91 Dawn Avenue	washing
ODR-196	EMP-4096	09:58:00	16:46:00	59 Jackson Junction	Full service
ODR-197	EMP-4097	12:25:00	15:19:00	291 Golf Course Pass	CNG fitting
ODR-198	EMP-4098	13:00:00	16:18:00	9 Bluestem Lane	Headlight replacement
ODR-199	EMP-4099	09:12:00	15:00:00	38 Manufacturers Circle	washing
ODR-200	EMP-4100	09:41:00	16:22:00		

A green success message at the bottom right of the results pane states: "Successfully run. Total query runtime: 219 msec. 100 rows affected."

xiv. Services

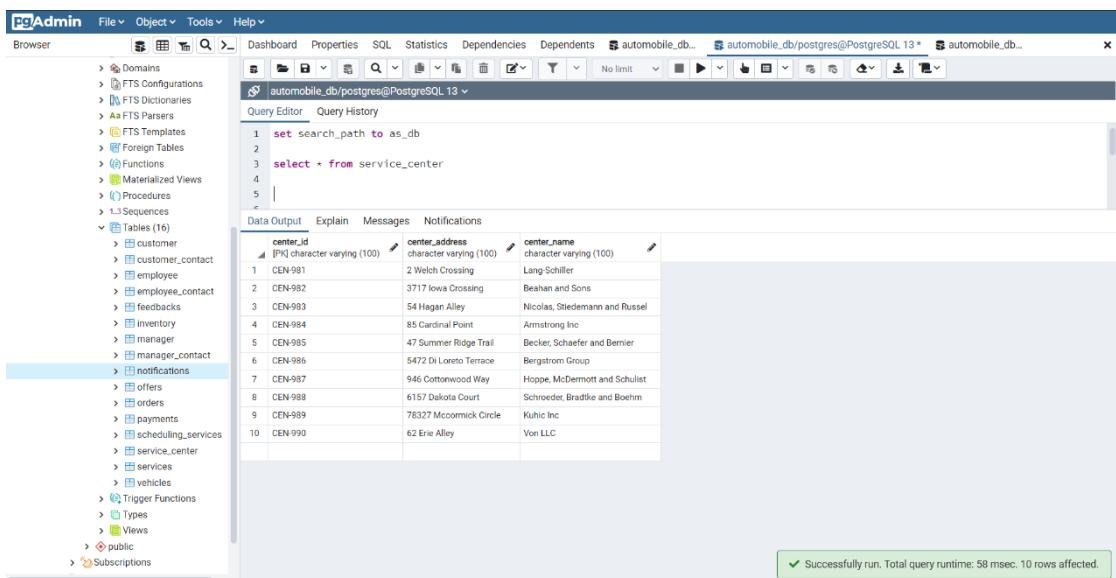


The screenshot shows the pgAdmin interface with the 'services' table selected. The table has three columns: service_id, service_charges, and service_name. The data output shows 15 rows of service details.

service_id	service_charges	service_name
SEC-7101	1200	Oil-Filter Changing
SEC-7102	1000	Wiper blades replacement
SEC-7103	800	Replace air filter
SEC-7104	2000	Scheduled maintenance
SEC-7105	1200	New tires
SEC-7106	500	Battery replacement
SEC-7107	400	Brake work
SEC-7108	1100	Antifreeze added
SEC-7109	1800	Engine tune-up
SEC-7110	750	Wheels aligned/balanced
SEC-7111	700	Coolant system services
SEC-7112	1000	washing
SEC-7113	3000	Full service
SEC-7114	50000	CNG fitting
SEC-7115	350	Headlight replacement

Successfully run. Total query runtime: 219 msec. 15 rows affected.

xv. Service_center



The screenshot shows the pgAdmin interface with the 'service_center' table selected. The table has three columns: center_id, center_address, and center_name. The data output shows 10 rows of service center details.

center_id	center_address	center_name
CEN-981	2 Welch Crossing	Lang-Schiller
CEN-982	3717 Iowa Crossing	Beahan and Sons
CEN-983	54 Hagan Alley	Nicolas, Stiedemann and Russel
CEN-984	85 Cardinal Point	Armstrong Inc
CEN-985	47 Summer Ridge Trail	Becker, Schaefer and Bernier
CEN-986	5472 Di Loreto Terrace	Bergstrom Group
CEN-987	946 Cottonwood Way	Hoppe, McDermott and Schulist
CEN-988	6157 Dakota Court	Schroeder, Bradtke and Boehm
CEN-989	78327 McCormick Circle	Kuhic Inc
CEN-990	62 Erie Alley	Von LLC

Successfully run. Total query runtime: 58 msec. 10 rows affected.

xvi. Vehicles

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables (16)' section, the 'notifications' table is selected. The main area displays a query in the Query Editor:

```

1 set search_path to as_db
2
3 select * from vehicles
4
5
6

```

The Data Output tab shows the results of the query:

vehicle_id	customer_id	vehicle_licence_plate	vehicle_type	model_id
284	VHL-5284	YL-35507	7796	3 YL-9747
285	VHL-5285	YL-35507	7858	4 CC-6575
286	VHL-5286	MU-21190	4903	4 GJ-8107
287	VHL-5287	MU-21190	8540	3 UA-5464
288	VHL-5288	MU-21190	1785	4 ZY-2844
289	VHL-5289	MQ-13798	8722	3 GB-6704
290	VHL-5290	MQ-13798	2889	4 LO-9235
291	VHL-5291	MQ-13798	6231	2 DF-4675
292	VHL-5292	ZK-60074	9710	3 OR-8207
293	VHL-5293	ZK-60074	1391	4 WP-4348
294	VHL-5294	ZK-60074	2511	3 WB-2447
295	VHL-5295	JT-18271	3567	4 Rb-1311
296	VHL-5296	JT-18271	5392	2 GR-7533
297	VHL-5297	JT-18271	7482	4 UW-2148
298	VHL-5298	KT-19731	5894	4 AH-3450
299	VHL-5299	KT-19731	6502	2 KW-8298
300	VHL-5300	KT-19731	2064	3 ET-4110

A green success message at the bottom right of the results pane says: "Successfully run. Total query runtime: 86 msec. 300 rows affected."

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%'

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables (16)' section, the 'customer' table is selected. The main area displays a query in the Query Editor:

```

1 set search_path to as_db
2
3 select * from customer where customer_name like 'M%'
4
5
6

```

The Data Output tab shows the results of the query:

customer_id	customer_name	customer_address	email
1 YA-04354	Mercie	2 Amritsice Trail	mbilling14@bluevines.com
2 LV-68530	Mof melia	8 Arthes Pass	mpgbe1d@unrich.edu
3 IO-17986	Martica	6897 Debra Parkway	motownfield20@hugedomains.com
4 JH-66096	Madlen	46587 Ridgeway Drive	moretti2b@lycos.com
5 MU-21190	Murry	2 Lien Center	mempn2n@feeburner.com

- 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

```

set search_path to as_db
select spare_part_id from inventory where spare_part_price >1800
  
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various objects like Languages, Publications, Schemas, and Tables. The main area shows a query in the Query Editor:

```

set search_path to as_db
select spare_part_id from inventory where spare_part_price >1800
  
```

The Data Output pane displays the results of the query, which are 23 tuples of spare part IDs:

spare_part_id
SPR-002
SPR-004
SPR-007
SPR-019
SPR-020
SPR-025
SPR-026
SPR-034
SPR-035
SPR-046
SPR-056
SPR-040
SPR-008
SPR-016
SPR-020
SPR-045
SPR-048
SPR-016
SPR-020
SPR-025
SPR-026
SPR-036
SPR-041

- Tuples: 55

3. Show data of uncompleted orders.

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

```

set search_path to as_db
select * from orders where order_status = 0 or order_status = 1
  
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various objects like Languages, Publications, Schemas, and Tables. The main area shows a query in the Query Editor:

```

set search_path to as_db
select * from orders where order_status = 0 or order_status = 1
  
```

The Data Output pane displays the results of the query, which are 58 tuples of order details:

order_id	customer_id	employee_id	service_id	vehicle_id
ODR-101	ML-78227	EMP-4001	SE0-7101	VHL-5
ODR-102	AM-99585	EMP-4003	SE0-7102	VHL-5
ODR-103	CU-94010	EMP-4003	SE0-7103	VHL-5
ODR-104	HO-11674	EMP-4004	SE0-7104	VHL-5
ODR-105	MT-42139	EMP-4005	SE0-7105	VHL-5
ODR-106	QO-33134	EMP-4008	SE0-7110	VHL-5
ODR-111	RF-13605	EMP-4011	SE0-7104	VHL-5
ODR-115	EK-02349	EMP-4015	SE0-7109	VHL-5
ODR-116	JZ-98223	EMP-4016	SE0-7106	VHL-5
ODR-118	JN-98651	EMP-4018	SE0-7108	VHL-5
ODR-119	QJ-51997	EMP-4019	SE0-7109	VHL-5
ODR-120	RG-22900	EMP-4020	SE0-7110	VHL-5
ODR-121	SW-98664	EMP-4021	SE0-7111	VHL-5
ODR-123	SE-16394	EMP-4023	SE0-7105	VHL-5
ODR-125	HV-29891	EMP-4025	SE0-7107	VHL-5
ODR-126	MH-47072	EMP-4026	SE0-7103	VHL-5
ODR-127	RE-45044	EMP-4027	SE0-7104	VHL-5
ODR-128	UJ-55842	EMP-4028	SE0-7105	VHL-5
ODR-129	VO-54578	EMP-4029	SE0-7106	VHL-5
ODR-133	IR-14659	EMP-4033	SE0-7104	VHL-5
ODR-135	XD-41817	EMP-4035	SE0-7106	VHL-5
ODR-136	DN-34996	EMP-4036	SE0-7107	VHL-5

- 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'

```

set search_path to as_db
select customer_id from feedbacks where visit_date
between '2018-11-01' and '2019-01-31'
  
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various tables like customer, employee, feedbacks, etc. The central area is the Query Editor containing the above SQL query. To the right is the Data Output window, which displays the results of the query as a list of 23 tuples, each consisting of a customer ID.

customer_id
CU-94010
EM-66025
BK-02349
XP-01407
SE-16394
MH-47072
VO-65478
IR-14439
ZC-11686
KC-32376
RK-68133
OV-45181
BK-79225
SR-63458
Li-48933
LE-04352
LB-63216
MN-88309
ID-78307
AZ-65559
XO-04199
ZF-44812
KT-19731

- 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'

```

set search_path to as_db
select customer_id from payments where payment_mode = 1
and payment_date between '2019-06-01' and '2019-08-01'
  
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema. The central area is the Query Editor containing the above SQL query. To the right is the Data Output window, which displays the results of the query as a list of 3 tuples, each consisting of a customer ID.

customer_id
NA-69910
MD-14733
MQ-13788

- 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

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

- Browser:** Shows the schema structure of the 'es_db' database, including tables like customer, employee, and service_center.
- Query Editor:** Contains the following SQL query:


```
1 set search_path to es_db
2
3 select * from employee
4 where ( work_end_time - work_start_time ) > '08:00:00'
5 and employee_status = 1;
```
- Data Output:** Displays the results of the query, showing 22 rows of employee data. The columns are employee_id, employee_name, joining_date, employee_email, and work_start_time.

employee_id	employee_name	joining_date	employee_email	work_start_time
EMP-4001	Davine	2021-06-13	djerosch@npr.org	09:06
EMP-4002	Delmor	2020-10-03	dcrayton@wufos.com	09:16
EMP-4004	Anastasia	2021-07-15	awimspears3@ommiture.com	09:26
EMP-4008	Marmaduke	2021-07-23	mihali7@unesco.org	09:13
EMP-4009	Harrison	2021-06-25	hgoooffield@abc.net.au	09:29
EMP-4017	Ferdie	2021-11-22	fstarbrookleg@tanmu.edu	09:22
EMP-4032	Chere	2021-04-18	cnewcombe123@reg.co.uk	09:04
EMP-4038	Merrile	2020-02-23	miettson11@vixnunet.com	09:23
EMP-4040	Aubrey	2021-08-28	alauri13@askey.com	09:21
EMP-4042	Shella-kathryn	2021-01-26	ssalinen15@milbean.gov.cn	09:21
EMP-4045	Channa	2020-09-03	coriane16@businessinside.com	09:13
EMP-4061	Brett	2021-05-25	bmacraik10@webeden.co.uk	09:07
EMP-4062	Malorie	2021-05-25	mivanakilov1@idc.gov	09:19
EMP-4063	Leeanne	2021-02-26	lmanon1a@loc.gov	09:10
EMP-4071	Vale	2020-05-31	vstoply1@yelp.com	09:08
EMP-4074	Amrie	2021-08-13	abarkow2@abc.net.au	09:09
EMP-4075	Koren	2021-07-15	kfranzzen22@linkedin.com	09:14
EMP-4080	Ariella	2021-07-18	aagrey27@sina.com.cn	09:16
EMP-4083	Mignon	2021-01-14	mcoburn12@washington.edu	09:26
EMP-4086	Burch	2021-09-23	bschmirmg2d@google.nu	09:24
EMP-4089	Aridatha	2021-12-11	atorms2g@swifly	09:18
EMP-4092	Roxy	2021-01-26	rsgrew2j@deledev.cn	09:21

- 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%')

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

- Browser:** Shows the schema structure of the 'es_db' database, including tables like customer, employee, and service_center.
- Query Editor:** Contains the following SQL query:


```
1 set search_path to es_db
2
3 select * from service_center
4 where center_id in
5 ( select center_id from employee where employee_name like 'A%' )
```
- Data Output:** Displays the results of the query, showing 7 rows of service_center data. The columns are center_id, center_address, and center_name.

center_id	center_address	center_name
CEN-981	2 Welch Crossing	Lang-Schiller
CEN-983	54 Hagan Alley	Nicolas, Stiedemann and Russel
CEN-984	85 Cardinal Point	Armstrong Inc
CEN-985	47 Summer Ridge Trail	Becker, Schaefer and Bernier
CEN-986	5472 Di Loreto Terrace	Bergstrom Group
CEN-988	6157 Dakota Court	Schoeder, Bradke and Boehm
CEN-989	78327 McCormick Circle	Kuhic Inc

- 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'

```

1
2 select * from employee where employee.employee_salary <
3 20000 and joining_date < '2021-01-01'

```

employee_id	employee_name	joining_date	employee_email	work_start_time	work_end_time	center_id	employee_status	employee_salary
38	Irwin	2020-01-18	idmimage23@ox.ac.uk	09:00:00	19:34:00	CEN-988	0	11565
39	EMP-4077	2020-09-06	tmachostek24@omn.com	09:04:00	18:12:00	CEN-988	0	13502
40	EMP-4082	2020-10-04	twomack29@scribd.com	09:17:00	20:40:00	CEN-989	1	16337
41	EMP-4088	2020-09-09	agilesase27@rmba.com	09:15:00	20:33:00	CEN-989	0	12683
42	EMP-4090	2020-03-10	odancy2h@zadret.com	09:07:00	20:01:00	CEN-999	1	12533
43	EMP-4094	2020-04-18	plander2@jimdo.com	09:18:00	18:57:00	CEN-990	0	13352
44	EMP-4097	2020-03-11	leddis20@friendfeed.com	09:04:00	18:10:00	CEN-990	1	19987
45	EMP-4099	James	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')

```

1 set search_path to as_db
2
3 select offer_id,max_discount, discount_percentage
4 from offers where service_id in
5 (select service_id from orders where order_id = 'ODR-109')
6
7

```

offer_id	max_discount	discount_percentage
1	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))

```

set search_path to as_db
1
2
3 select customer_name from customer where customer_id in
4 (select customer_id from orders where service_id in
5 (select service_id from offers where offers.discount_percentage > 5) )
6
7

```

customer_name
Boote
Stormie
Ezequiel
Arimm
Isac
Beulah
Ardith
Katy
Quintana
Shell
Katha
Hallee
Cortie
Rossy
Eliot
Scarlett
Hernia
Agace
Scarlett
Culle
Chiquita
Daryl
Adler

- 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)

```

set search_path to as_db
1
2
3 select customer_id, contact from customer_contact where customer_id in
4 (select customer_id from orders where order_status = 2)
5
6

```

customer_id	contact
KT-58461	610-763-0492
KT-58461	447-961-8937
KT-58461	612-239-4748
NA-59910	987-877-0631
NA-59910	399-355-0123
NA-59910	318-466-2156
FY-94453	742-986-6688
FY-94453	931-575-2917
FY-94453	569-185-8339
EM-68025	574-437-4615
EM-68025	994-426-7018
EM-68025	990-935-1764
FD-22169	249-758-2246
FD-22169	271-114-3163
FD-22169	640-543-9366
QW-93437	979-970-5077
QW-93437	117-369-0177
QW-93437	290-733-8280
FV-25340	288-332-5018
FV-25340	138-160-2992
FV-25340	558-826-5371
XF-01407	121-510-7591
XF-01407	715-279-2307

- 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'

```

set search_path to as_db
select * from employee where joining_date > '2020-08-01'
  
```

employee_id	employee_name	joining_date	employee_email	work_sta
EMP-4001	Devine	2021-06-13	dgerosch@npr.org	09:06
EMP-4002	Delmar	2020-10-03	dcrayton@wufuo.com	09:16
EMP-4004	Anastasia	2021-07-15	awinsepe3@ommurture.com	09:26
EMP-4005	Bren	2021-12-27	bbrucher@lilbonews.com	09:22
EMP-4007	Remington	2020-11-10	redworth6@gmail.net	09:24
EMP-4008	Marmaduke	2021-07-23	mkelle7@unesco.org	09:13
EMP-4009	Harrison	2021-09-25	hgoddell@lplib.net.au	09:29
EMP-4011	Gallard	2020-12-18	gthomerson@upenn.edu	09:24
EMP-4012	Cathryn	2021-07-27	cleburnut@t.co	09:16
EMP-4013	Reginald	2021-06-27	rmuselli@surveymonkey.com	09:25
EMP-4014	Christean	2021-03-20	cooperd@w3.org	09:14
EMP-4017	Ferde	2021-11-22	fraterbrooks@haru.edu	09:22
EMP-4021	Stephanie	2021-07-09	szivitz@eipais.com	09:17
EMP-4022	Albertine	2021-05-24	apolin@ibc.ca	09:07
EMP-4023	Priscilla	2021-10-10	pgleetom@comcast.net	09:08
EMP-4025	Bella	2021-01-19	bfullerton@vk.com	09:18
EMP-4028	Krista	2020-09-27	kboguer@berkeley.edu	09:24
EMP-4032	Chere	2021-04-18	corowomber@123-reg.co.uk	09:04
EMP-4035	Basilus	2021-11-15	bmaccossey@photobucket.com	09:26
EMP-4040	Aubrey	2021-09-28	alauren13@skyley.com	09:21
EMP-4042	Shella-kathryn	2021-07-26	ssalmen5@millean.gov.cn	09:21
EMP-4044	Chelsie	2021-10-19	cswardborough17@google.com.br	09:11

- 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_date-start_date) from offers))

```

set search_path to as_db
select offer_id, start_date, end_date, discount_percentage from offers
where (offers.end_date - offers.start_date =
      (select max(end_date-start_date) from offers))
  
```

offer_id	start_date	end_date	discount_percentage
OFFR-1008	2018-06-28	2021-10-16	8

- 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')

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

- Browser:** Shows the schema structure of the database, including Schemas (2) and Tables (16).
- Query Editor:** Contains the following SQL code:


```
1 set search_path to as_db
2
3 select employee_id,contact from employee_contact where employee_id in
4 (select employee_id from employee where joining_date > '2020-12-01')
```
- Data Output:** Displays the results of the query as a table with columns: employee_id and contact. The data consists of 23 rows of employee IDs and their corresponding contact numbers.

employee_id	contact
EMP-4001	529-490-2114
EMP-4001	593-565-3333
EMP-4004	274-576-4150
EMP-4004	333-480-2660
EMP-4005	426-570-9303
EMP-4005	427-769-0448
EMP-4008	527-260-1250
EMP-4008	847-540-2465
EMP-4009	985-425-0737
EMP-4009	764-352-3434
EMP-4011	283-947-8094
EMP-4011	999-506-1284
EMP-4012	906-817-0933
EMP-4012	406-747-6515
EMP-4013	979-900-4516
EMP-4013	969-842-3099
EMP-4014	511-449-8764
EMP-4014	210-681-5238
EMP-4017	889-119-8225
EMP-4017	800-149-0574
EMP-4021	570-162-7249
EMP-4021	995-644-8816
EMP-4022	594-208-5121

- 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'

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

- Browser:** Shows the schema structure of the database, including Schemas (2) and Tables (16).
- Query Editor:** Contains the following SQL code:


```
1 set search_path to as_db
2
3 select offer_id,start_date,end_date,discount_percentage, max_discount
4 from offers where start_date >= '2019-04-01' and end_date <= '2020-12-31'
```
- Data Output:** Displays the results of the query as a table with columns: offer_id, start_date, end_date, discount_percentage, and max_discount. The data consists of 3 rows of offer details.
- Status Bar:** Shows a message indicating the query was successfully run: "Successfully run. Total query runtime: 59 msec. 3 rows affected."

offer_id	start_date	end_date	discount_percentage	max_discount
OFR-1001	2019-04-18	2020-09-30	10	103
OFR-1009	2019-05-07	2020-07-19	2	74
OFR-1013	2019-10-04	2020-07-01	8	106

- Tuples: 3

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

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

```

1 set search_path to as_db
2
3 select vehicle_liscence_plate from vehicles
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

```

	vehicle_liscence_plate
1	9211
2	4047
3	6724
4	1972
5	8976
6	4825
7	8947
8	1988
9	7250
10	8851
11	4171
12	5823
13	6933
14	7305
15	7562
16	6693
17	1204
18	2311
19	2036
20	2819
21	7559
22	4064
23	5188

- 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

```

1 set search_path to as_db
2
3 select manager_id,spare_part_id, spare_part_stock,contact
4 from inventory natural join manager_contact
5 where spare_part_stock < 50
6 and manager_contact.manager_id = inventory.manager_id
7

```

manager_id	spare_part_id	spare_part_stock	contact
MN-1731	SFR-010	32	163-807-4440
MN-1731	SFR-010	32	701-442-6553
MN-1731	SFR-010	32	264-563-8851
MN-1731	SFR-017	49	163-807-4440
MN-1731	SFR-017	49	701-442-6553
MN-1731	SFR-017	49	264-563-8851
MN-1731	SFR-018	31	163-807-4440
MN-1731	SFR-018	31	701-442-6553
MN-1731	SFR-018	31	264-563-8851
MN-1731	SFR-021	40	163-807-4440
MN-1731	SFR-021	40	701-442-6553
MN-1731	SFR-021	40	264-563-8851
MN-1731	SFR-022	35	163-807-4440
MN-1731	SFR-022	35	701-442-6553
MN-1731	SFR-022	35	264-563-8851
MN-1731	SFR-023	37	163-807-4440
MN-1731	SFR-023	37	701-442-6553
MN-1731	SFR-023	37	264-563-8851
MN-1731	SFR-032	48	163-807-4440
MN-1731	SFR-032	48	701-442-6553
MN-1731	SFR-032	48	264-563-8851
MN-1731	SFR-036	42	163-807-4440
MN-1731	SFR-036	42	701-442-6553

- 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)

```

1 set search_path to as_db
2
3 select * from inventory where spare_part_stock in
4 (select min(spare_part_stock) from inventory)
5

```

center_id	manager_id	spare_part_id	spare_part_stock	spare_id
CEN-983	MN-1733	SPR-045	25	
CEN-985	MN-1735	SPR-040	25	
CEN-986	MN-1736	SPR-031	25	
CEN-986	MN-1736	SPR-037	25	
CEN-987	MN-1737	SPR-011	25	
CEN-987	MN-1737	SPR-015	25	
CEN-988	MN-1738	SPR-042	25	
CEN-990	MN-1740	SPR-022	25	

- 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

```

1 set search_path to as_db
2
3 create view completed_orders as
4 select order_id, customer_id, employee_id, vehicle_id
5 from orders where order_status = 2
6
7 select * from completed_orders

```

order_id	customer_id	employee_id	vehicle_id
ODR-105	KT-58461	EMP-4006	VHL-5006
ODR-107	NA-99510	EMP-4007	VHL-5007
ODR-109	FV-99453	EMP-4009	VHL-5009
ODR-110	EM-68025	EMP-4010	VHL-5010
ODR-112	FD-22169	EMP-4012	VHL-5012
ODR-113	QW-93437	EMP-4013	VHL-5013
ODR-114	FV-25340	EMP-4014	VHL-5014
ODR-117	XF-01407	EMP-4017	VHL-5017
ODR-122	SL-02402	EMP-4022	VHL-5022
ODR-124	OY-58111	EMP-4024	VHL-5024
ODR-130	H-89777	EMP-4030	VHL-5030
ODR-131	H2-09654	EMP-4031	VHL-5031
ODR-132	PK-22792	EMP-4032	VHL-5032
ODR-134	EJ-5308	EMP-4034	VHL-5034
ODR-139	RE-62069	EMP-4039	VHL-5039
ODR-140	ZC-11686	EMP-4040	VHL-5040
ODR-142	FZ-53270	EMP-4042	VHL-5042
ODR-143	OK-61234	EMP-4043	VHL-5043
ODR-144	KS-80653	EMP-4044	VHL-5044
ODR-145	KO-92376	EMP-4045	VHL-5045
ODR-149	LF-81753	EMP-4049	VHL-5049
ODR-150	LV-68530	EMP-4050	VHL-5050
ODR-153	QV-45181	EMP-4053	VHL-5053

- 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

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

- Browser:** Shows the schema structure under "sa_db", including tables like customer, employee, feedbacks, inventory, manager, notifications, offers, orders, payments, scheduling_services, service_center, services, and vehicles.
- Query Editor:** Contains the following SQL query:


```

1 set search_path to as_db
2
3 select order_id, customer_name, customer_id, vehicle_id, employee_name, employee_id
4 from completed_orders natural join customer natural join employee
5 where customer.customer_id = completed_orders.customer_id and employee.employee_id = completed_orders.employee_id
      
```
- Data Output:** Displays the results of the query, showing 44 tuples. The columns are order_id, customer_name, customer_id, vehicle_id, employee_name, and employee_id. The data includes various names and IDs such as Amrin, NA-99910, VHL-5009, Celest, etc.

- 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

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

- Browser:** Shows the schema structure under "sa_db", including tables like customer, employee, feedbacks, inventory, manager, notifications, offers, orders, payments, scheduling_services, service_center, services, and vehicles.
- Query Editor:** Contains the following SQL query:


```

1 set search_path to as_db
2
3 select customer_name, customer_id, payment_id , payment_mode
4 from customer natural join payments
5 where customer.customer_id = payments.customer_id and payments.payment_mode = 1
      
```
- Data Output:** Displays the results of the query, showing 51 tuples. The columns are customer_name, customer_id, payment_id, and payment_mode. The data includes various names and IDs such as Stormie, AM-99805, PAY-3002, 1, etc.

- 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

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Schemas (2)' section, there is a 'Tables (16)' folder which is expanded. In the 'Query Editor' tab, the following SQL code is written:

```

1 set search_path to as_db
2
3 create view manager_to_be_contacted as
4 select center_id, manager_id, spare_part_id, spare_part_stock from inventory where spare_part_stock < 50
5
6 select manager_id, contact, spare_part_id, spare_part_stock
7 from manager_to_be_contacted natural join manager_contact
8 where manager_to_be_contacted.manager_id = manager_contact.manager_id
9

```

Below the query editor, the 'Data Output' pane displays a table with 23 rows of data. The columns are: manager_id, character varying (10), contact, character varying (100), spare_part_id, character varying (100), and spare_part_stock, bigint. The data is as follows:

manager_id	character varying (10)	contact	character varying (100)	spare_part_id	character varying (100)	spare_part_stock
MN-1731		163-807-4440	SPR-010			32
MN-1731		701-442-6553	SPR-010			32
MN-1731		264-560-8851	SPR-010			32
MN-1731		163-807-4440	SPR-017			49
MN-1731		701-442-6553	SPR-017			49
MN-1731		264-560-8851	SPR-017			49
MN-1731		163-807-4440	SPR-018			31
MN-1731		701-442-6553	SPR-018			31
MN-1731		264-560-8851	SPR-018			31
MN-1731		163-807-4440	SPR-021			40
MN-1731		701-442-6553	SPR-021			40
MN-1731		264-560-8851	SPR-021			40
MN-1731		163-807-4440	SPR-022			35
MN-1731		701-442-6553	SPR-022			35
MN-1731		264-560-8851	SPR-022			35
MN-1731		163-807-4440	SPR-023			37
MN-1731		701-442-6553	SPR-023			37
MN-1731		264-560-8851	SPR-023			37
MN-1731		163-807-4440	SPR-032			48
MN-1731		701-442-6553	SPR-032			48
MN-1731		264-560-8851	SPR-032			48
MN-1731		163-807-4440	SPR-036			42
MN-1731		701-442-6553	SPR-036			42

- 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

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Schemas (2)' section, there is a 'Tables (16)' folder which is expanded. In the 'Query Editor' tab, the following SQL code is written:

```

1 set search_path to as_db
2
3 update employee
4 set employee_salary = employee_salary + 5000
5 where joining_date < '2021-01-01' and employee_salary < 20000
6

```

Below the query editor, the 'Messages' pane displays the results of the update query:

```

UPDATE 24
Query returned successfully in 125 msec.

```

- 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

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

- Browser:** Shows the schema structure of the database, including tables like customer, employee, manager, and vehicles.
- Query Editor:** Contains the following SQL code:


```
1 set search_path to as_db
2
3 select customer_id as owner_id,vehicle_id
4 from vehicles where vehicle_type = 4 group by owner_id,vehicle_id
5
```
- Data Output:** Displays the results of the query, which are 105 rows of data. The columns are owner_id and vehicle_id. The data includes various vehicle IDs such as BD-03200, FZ-53270, CE-92248, LV-68519, KT-19731, FV-25340, ZM-47914, SR-63458, IO-17958, SL-02402, JJ-84987, RF-13605, US-56642, and HV-29891.
- Status Bar:** Shows a green message: "Successfully run. Total query runtime: 120 msec. 105 rows affected."

- 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((50-spare_part_stock)*spare_part_price) as total_bill from inventory where spare_part_stock < 50 group by center_id,manager_id

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

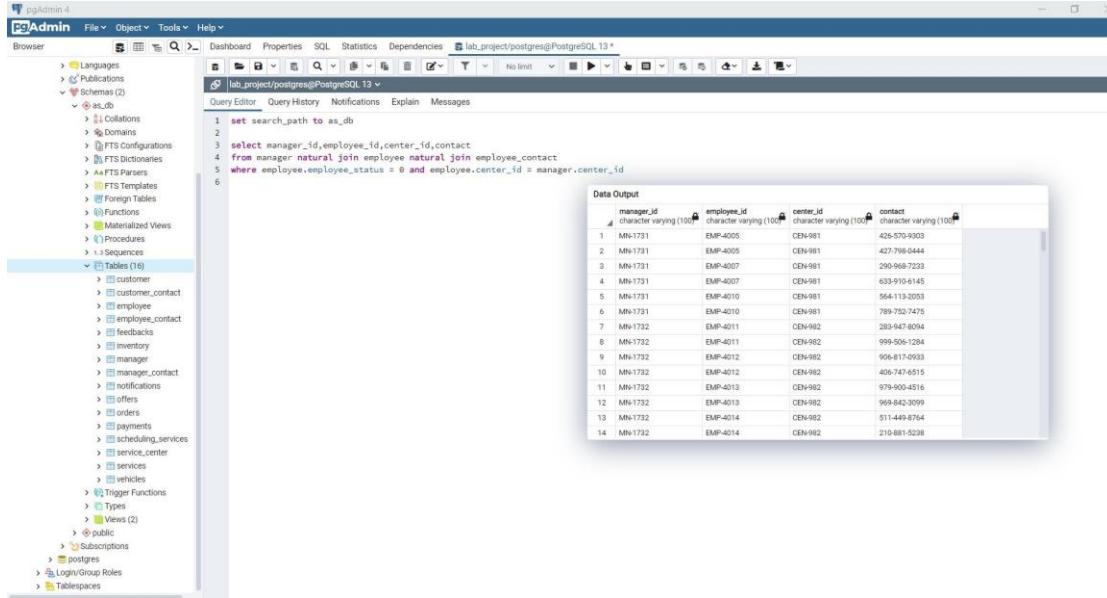
- Browser:** Shows the schema structure of the database, including tables like customer, employee, manager, and inventory.
- Query Editor:** Contains the following SQL code:


```
1 set search_path to as_db
2
3 select center_id,manager_id,sum((50-spare_part_stock)*spare_part_price) as total_bill
4 from inventory
5 where spare_part_stock < 50 group by center_id,manager_id
6
```
- Data Output:** Displays the results of the query, which are 10 rows of data. The columns are center_id, manager_id, and total_bill. The data includes various manager IDs like MH-1737, MH-1734, MH-1740, MH-1731, MH-1739, MH-1733, MH-1736, MH-1738, MH-1732, and MH-1735, with total bills ranging from 176667 to 234605.
- Status Bar:** Shows a green message: "Successfully run. Total query runtime: 103 msec. 10 rows affected."

- 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



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

- File Bar:** pgAdmin 4, File, Object, Tools, Help.
- Toolbar:** Standard icons for file operations like Open, Save, Print, etc.
- Schemas:** Current schema is lab_project/postgres@PostgreSQL 13.
- Query Editor:**

```

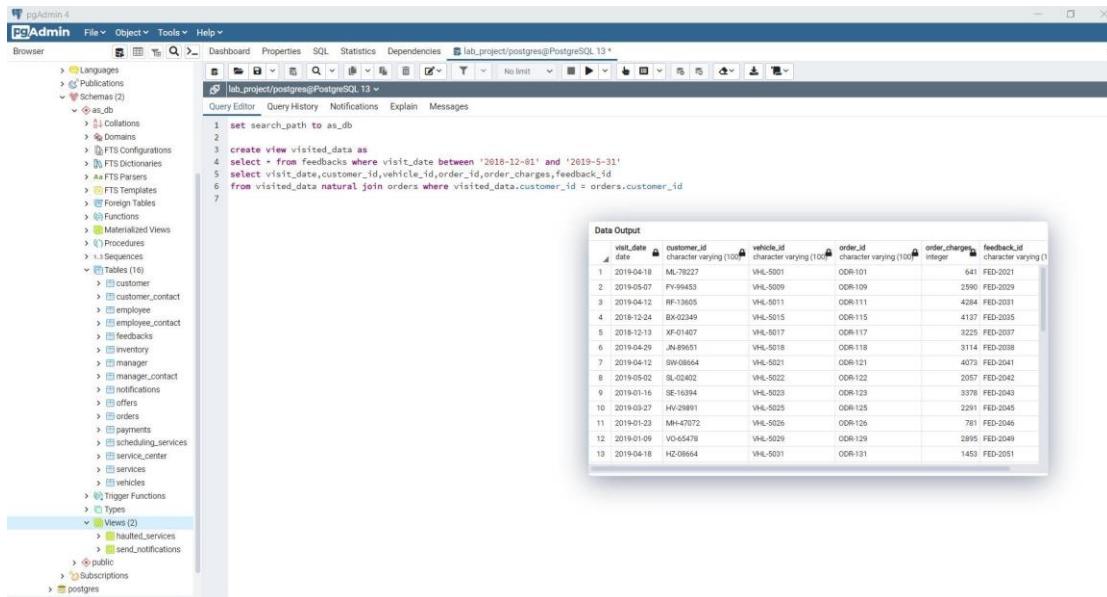
1 set search_path to as_db
2
3 select manager_id,employee_id,center_id,contact
4 from manager natural join employee natural join employee_contact
5 where employee.employee_status = 0 and employee.center_id = manager.center_id
6
    
```
- Data Output:** A table showing the results of the query. The columns are manager_id, employee_id, center_id, and contact. There are 14 rows of data.

	manager_id	employee_id	center_id	contact
1	MN1731	EMP4005	CEN981	426-570-9303
2	MN1731	EMP4005	CEN981	427-798-0444
3	MN1731	EMP4007	CEN981	297-968-7233
4	MN1731	EMP4007	CEN981	633-910-6145
5	MN1731	EMP4010	CEN981	564-113-2053
6	MN1731	EMP4010	CEN981	789-755-7475
7	MN1732	EMP4011	CEN982	283-947-8094
8	MN1732	EMP4011	CEN982	999-506-1284
9	MN1732	EMP4012	CEN982	906-817-9933
10	MN1732	EMP4012	CEN982	406-747-6515
11	MN1732	EMP4013	CEN982	979-900-4516
12	MN1732	EMP4013	CEN982	969-842-3099
13	MN1732	EMP4014	CEN982	511-449-8764
14	MN1732	EMP4014	CEN982	210-881-5238

- 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



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

- File Bar:** pgAdmin 4, File, Object, Tools, Help.
- Toolbar:** Standard icons for file operations like Open, Save, Print, etc.
- Schemas:** Current schema is lab_project/postgres@PostgreSQL 13.
- Query Editor:**

```

1 set search_path to as_db
2
3 create view visited_data as
4 select * from feedbacks where visit_date between '2018-12-01' and '2019-5-31'
5 select visit_date, customer_id, vehicle_id, order_id, order_charges, feedback_id
6 from visited_data natural join orders where visited_data.customer_id = orders.customer_id
7
    
```
- Data Output:** A table showing the results of the query. The columns are visit_date, customer_id, vehicle_id, order_id, order_charges, and feedback_id. There are 13 rows of data.

	visit_date	customer_id	vehicle_id	order_id	order_charges	feedback_id
1	2019-04-18	MU7827	VHL5001	ODR101	641	FED-2021
2	2019-05-07	FY9483	VHL5009	ODR109	2390	FED-2029
3	2019-04-12	RF13607	VHL5011	ODR111	4286	FED-2031
4	2019-12-24	BX22349	VHL5015	ODR115	4137	FED-2035
5	2018-12-13	XF01407	VHL5017	ODR117	3225	FED-2037
6	2019-04-29	JN49951	VHL5018	ODR118	3114	FED-2038
7	2019-04-12	BW08864	VHL5021	ODR121	4073	FED-2041
8	2019-05-02	SL02402	VHL5022	ODR122	2057	FED-2042
9	2019-04-16	SE16934	VHL5023	ODR123	3378	FED-2043
10	2019-04-27	HV26991	VHL5025	ODR125	2291	FED-2045
11	2019-04-23	MH47072	VHL5026	ODR126	761	FED-2046
12	2019-01-09	VO45478	VHL5029	ODR129	2895	FED-2049
13	2019-04-18	HZ09664	VHL5031	ODR131	1453	FED-2051

- 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

The screenshot shows the pgAdmin 4 interface with a database browser on the left and a query editor on the right. The query editor contains the following SQL code:

```

1 set search_path to as_db
2
3 select customer_id , vehicle_id, order_charges , order_id , drop_time
4 from orders natural join scheduling_services
5 where orders.order_id = scheduling_services.order_id and orders.order_status = 2
6

```

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

customer_id	vehicle_id	order_charges	order_id	drop_time
KT-58461	VHL-5006	506	ODR-106	16:37:00
NA-95910	VHL-5007	1277	ODR-107	16:12:00
PY-99453	VHL-5009	2590	ODR-109	16:46:00
EM-68025	VHL-5010	4926	ODR-110	15:31:00
FD-22169	VHL-5012	721	ODR-112	15:38:00
OW-93437	VHL-5013	2567	ODR-113	16:30:00
FV-25340	VHL-5014	2483	ODR-114	15:16:00
XF-01407	VHL-5017	3225	ODR-117	16:03:00
SL-02402	VHL-5022	2057	ODR-122	16:06:00
OV-98111	VHL-5024	1553	ODR-124	16:24:00
H-89777	VHL-5030	4179	ODR-130	16:52:00
HZ-08664	VHL-5031	1453	ODR-131	16:04:00
PX-22792	VHL-5032	4824	ODR-132	15:38:00
EJ-53882	VHL-5034	2045	ODR-134	15:22:00

A message at the bottom right says "Successfully run. Total query runtime: 83 ms."

- 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

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables (16)' section, there is a 'Views (2)' folder which contains two views: 'haunted_services' and 'send_notifications'. The 'Query Editor' tab is active, displaying the following SQL code:

```

1 set search_path to as_db
2
3 update orders
4 set order_charges = case
5 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)
6 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)
7 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
8 end
9
10 select * from orders
11

```

The 'Data Output' pane on the right displays the results of the query, showing 13 tuples. The columns are: order_id, customer_id, employee_id, service_id, vehicle, and charge.

order_id	customer_id	employee_id	service_id	vehicle	charge
ODR-101	ML-78227	EMP-4001	SEO-7101	VHL	
ODR-102	AM-99585	EMP-4052	SEO-7102	VHL	
ODR-103	CU-94010	EMP-4053	SEO-7103	VHL	
ODR-104	HO-11674	EMP-4054	SEO-7104	VHL	
ODR-105	MT-42139	EMP-4055	SEO-7105	VHL	
ODR-106	KT58461	EMP-4056	SEO-7106	VHL	
ODR-107	NA-95910	EMP-4007	SEO-7109	VHL	
ODR-108	QO-33134	EMP-4058	SEO-7110	VHL	
ODR-109	FV-94453	EMP-4059	SEO-7102	VHL	
ODR-110	EM-68025	EMP-4010	SEO-7103	VHL	
ODR-111	RF-13605	EMP-4011	SEO-7104	VHL	
ODR-112	FD-22169	EMP-4012	SEO-7115	VHL	
ODR-113	QW-93437	EMP-4013	SEO-7114	VHL	

- 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

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables (16)' section, there is a 'Views (2)' folder which contains two views: 'haunted_services' and 'send_notifications'. The 'Query Editor' tab is active, displaying the following SQL code:

```

1 set search_path to as_db
2
3 select center_id,customer_id
4 from orders natural join employee natural join service_center
5 where orders.employee_id = employee.employee_id and employee.center_id = service_center.center_id

```

The 'Data Output' pane on the right displays the results of the query, showing 14 tuples. The columns are: center_id and customer_id.

center_id	customer_id
CEN-981	ML-78227
CEN-981	AM-99585
CEN-981	CU-94010
CEN-981	HO-11674
CEN-981	MT-42139
CEN-981	KT58461
CEN-981	NA-95910
CEN-981	QO-33134
CEN-981	FV-94453
CEN-981	EM-68025
CEN-982	RF-13605
CEN-982	FD-22169
CEN-982	QW-93437
CEN-982	FV-25340

- Tuples: 102

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

- **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

The screenshot shows the pgAdmin 4 interface. On the left, the object browser displays the schema 'as_db' with various tables like customer, employee, feedbacks, etc. In the center, the Query Editor contains the following SQL code:

```

1 set search_path to as_db
2
3 create view haulted_services as
4 select order_id,employee_id as onleave_employee,center_id from orders natural join employee
5 where order_status = 0 and employee.employee_status = 0
6
7 select * from haulted_services
  
```

On the right, the Data Output pane shows the results of the query, which is a table with three columns: order_id, onleave_employee, and center_id. The data consists of 14 rows:

order_id	onleave_employee	center_id
ODR-105	EMP-4005	CEN-981
ODR-111	EMP-4011	CEN-982
ODR-119	EMP-4019	CEN-982
ODR-125	EMP-4025	CEN-983
ODR-126	EMP-4026	CEN-983
ODR-128	EMP-4028	CEN-983
ODR-129	EMP-4029	CEN-983
ODR-135	EMP-4035	CEN-984
ODR-141	EMP-4041	CEN-985
ODR-147	EMP-4047	CEN-985
ODR-151	EMP-4051	CEN-985
ODR-152	EMP-4052	CEN-986
ODR-154	EMP-4054	CEN-986
ODR-169	EMP-4069	CEN-987

- 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-%')

```

set search_path to as_db
(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-%')

```

feedback_id	customer_id	order_id	vehicle_id	employee_id
1 FED-2051	CE-61416	ODR-172	VH-5072	EMP-4
2 FED-2050	ZC-11686	ODR-140	VH-5040	EMP-4
3 FED-2053	GX-61234	ODR-143	VH-5043	EMP-4
4 FED-2054	EJ-53982	ODR-134	VH-5034	EMP-4
5 FED-2101	DT-10853	ODR-181	VH-5081	EMP-4
6 FED-2104	JN-66096	ODR-184	VH-5084	EMP-4
7 FED-2034	FV-25340	ODR-114	VH-5014	EMP-4
8 FED-2045	HV-29891	ODR-125	VH-5025	EMP-4
9 FED-2105	AZ-65559	ODR-186	VH-5086	EMP-4
10 FED-2108	ML-72426	ODR-188	VH-5088	EMP-4
11 FED-2095	LB-83216	ODR-175	VH-5075	EMP-4
12 FED-2058	JJ-64987	ODR-138	VH-5038	EMP-4
13 FED-2077	TH-43031	ODR-157	VH-5057	EMP-4

- 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

```

set search_path to as_db
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

```

customer_id	customer_name	vehicle_id	vehicle_liscence_plate	order_id
1 ML-78227	Boote	VHL-5001	9211	ODR-101

- 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'

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

- Browser:** Shows the schema structure of the database, including Schemas (2), Tables (16), and Views (1).
- Query Editor:** Contains the following SQL code:


```

1 set search_path to as_db
2
3 insert into orders
4   values ('ODR-201','ML-78227','EMP-4001','SEC-7109','VHL-5100',1000,1);
5 insert into orders
6   values ('ODR-202','ML-78227','EMP-4002','SEC-7103','VHL-5100',2000,1);
7 select sum(order_charges) as total_bill from orders where customer_id='ML-78227';
      
```
- Data Output:** Displays the result of the query, showing a single row with the value 4200.

- 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

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

- Browser:** Shows the schema structure of the database, including Schemas (2), Tables (16), and Views (1).
- Query Editor:** Contains the following SQL code:


```

1 set search_path to as_db
2
3 create view send_notifications as
4 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
5 from orders natural join customer natural join payments
6 where orders.order_status = 2 and orders.customer_id = customer.customer_id and customer.customer_id = payments.customer_id
7 select * from send_notifications
      
```
- Data Output:** Displays the result of the query, showing 13 tuples of data.

customer_id	customer_name	vehicle_id	payment_id	payment_date	payment_amount
KT-58461	Armin	VHL-5006	PAY-3006		
NA-9910	Enrica	VHL-5007	PAY-3007		
FY-99433	Bethah	VHL-5009	PAY-3009		
EM-68025	Damiano	VHL-5010	PAY-3010		
FD-22169	Reggili	VHL-5012	PAY-3012		
QW-93437	Bale	VHL-5013	PAY-3013		
FV-25340	Katy	VHL-5014	PAY-3014		
XF-01407	Quintana	VHL-5017	PAY-3017		
SL-02402	Hallee	VHL-5022	PAY-3022		
OV-58111	Kylla	VHL-5024	PAY-3024		
HI-89777	Elott	VHL-5030	PAY-3030		
HZ-08664	Scarlett	VHL-5031	PAY-3031		
PK-22792	Javier	VHL-5032	PAY-3032		

- 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

```

set search_path to as_db;
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

```

Data Output

customer_id	center_address
ML-72227	2 Welch Crossing
ML-72227	2 Welch Crossing
ML-72227	2 Welch Crossing
AM-99585	2 Welch Crossing
OU-94010	2 Welch Crossing
HC-11674	2 Welch Crossing
MT-42199	2 Welch Crossing
KT-58461	2 Welch Crossing
NA-59910	2 Welch Crossing
QC-03134	2 Welch Crossing
FY-99453	2 Welch Crossing
EM-68025	2 Welch Crossing
RF-13695	9717 Iowa Crossing
FD-22148	9717 Iowa Crossing

- 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'

```

set search_path to as_db;
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'

```

Data Output

time	order_id	employee_id	vehicle_id	customer_id
01:38:00	ODR-110	EMP-4010	VHL-5010	EM-68025
01:31:00	ODR-138	EMP-4038	VHL-5038	JJ-84987
01:41:00	ODR-143	EMP-4043	VHL-5043	GX-61234

- 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

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under 'Browser', there are sections for Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Functions, Materialized Views, Procedures, Sequences, Tables (16), Types, and Views (7). The 'Views' section is currently selected. In the main pane, the SQL tab contains the following code:

```

1 set search_path to as_db
2
3 create view most_recent_orders as
4 select max(payment_date),customer_id,customer_name from payments natural join customer
5 where customer.customer_id = payments.customer_id group by customer_id,customer_name
6
7 select * from most_recent_orders

```

Below the code, the 'Data Output' tab displays the results of the query:

	max_date	customer_id	customer_name
1	2018-01-16	YL-35507	Haley
2	2018-03-20	TJ-37884	Tisha
3	2019-04-29	JH-89651	Shell
4	2018-03-30	PD-56608	Isodore
5	2019-04-09	ML-78227	Boote
6	2019-04-12	DT-10853	Alexandr
7	2018-06-26	AM-99585	Stormie
8	2018-01-16	XD-61817	Guri
9	2019-06-19	NA-95910	Enrica
10	2019-10-10	DN-34996	Agace
11	2019-12-27	MQ-37799	Cheri
12	2019-11-27	YA-68117	Krista
13	2019-01-23	BK-79225	Clyde
14	2018-03-30	OV-58111	Kylla

- Tuple: 100

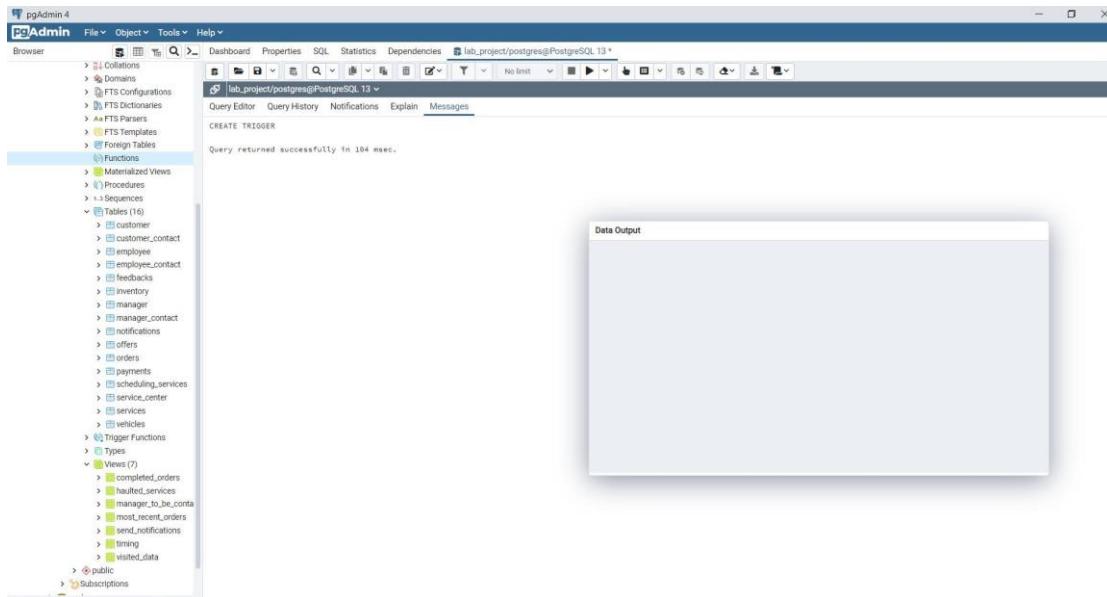
6.5.3 Trigger Function

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:**

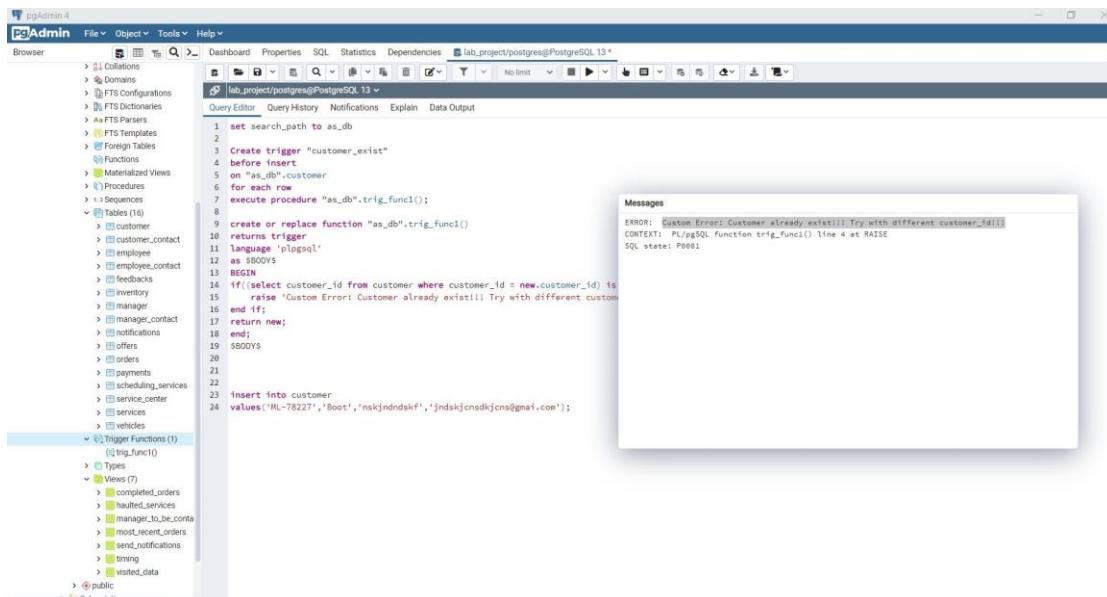
```
Create trigger "customer_exist" before
insert on "as_db".customer for each row
execute procedure "as_db".trig_func1();
```

```
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\('ML-78227','Boot','nskjndndskf','jndskjcnSDKjcn@gmail.com'\);](#)



6.5.4 Function

- 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
$BODY$
```

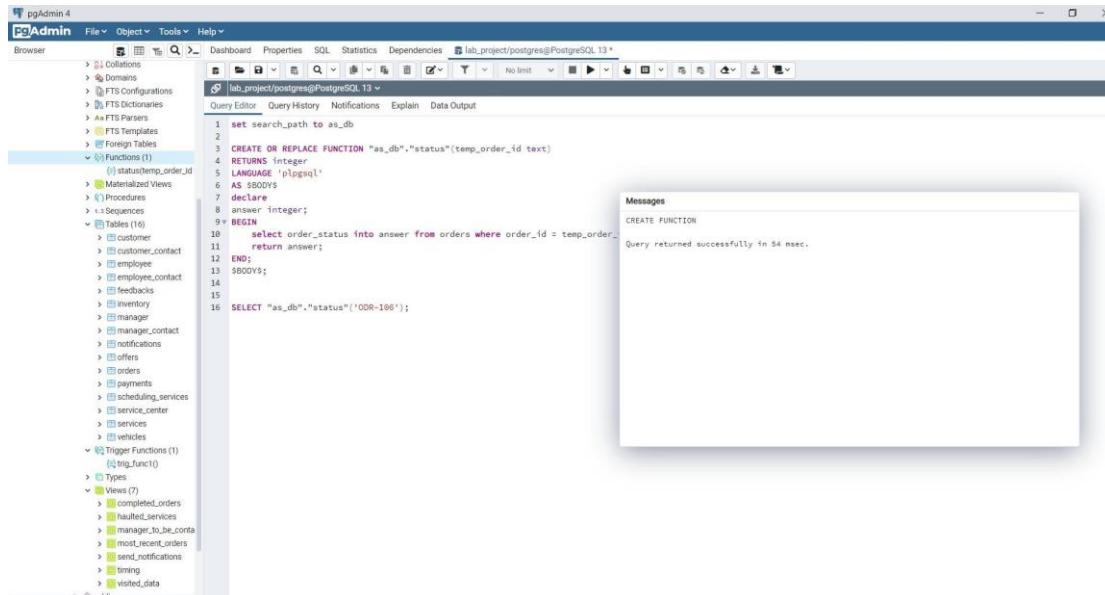
```
declare answer
integer;
```

```
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');
```

```

1 set search_path to as_db
2
3 CREATE OR REPLACE FUNCTION "as_db"."status"(temp_order_id text)
4 RETURNS integer
5 LANGUAGE 'plpgsql'
6 AS $BODY$
7 declare
8 answer integer;
9 BEGIN
10     select order_status into answer from orders where order_id = temp_order_id;
11     return answer;
12 END;
13 $BODY$;
14
15 SELECT "as_db"."status"('ODR-106');

```

Data Output
status
2

- 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.

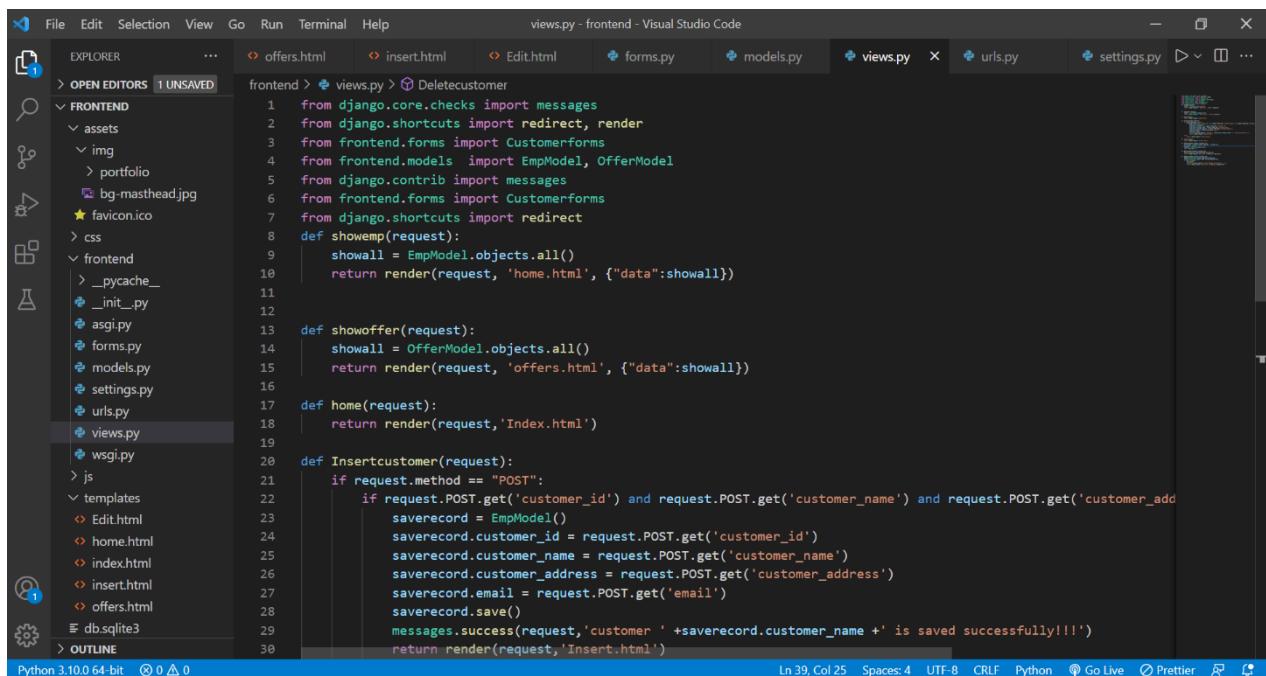
7.1 Codes File Link

<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

from django.core.checks import messages
from django.shortcuts import redirect, render
from frontend.forms import Customerforms
from frontend.models import EmpModel, OfferModel
from django.contrib import messages
from frontend.forms import Customerforms
from django.shortcuts import redirect
def showemp(request):
    showall = EmpModel.objects.all()
    return render(request, 'home.html', {"data":showall})
def showoffer(request):
    showall = OfferModel.objects.all()
    return render(request, 'offers.html', {"data":showall})
def home(request):
    return render(request,'Index.html')
def Insertcustomer(request):
    if request.method == "POST":
        if request.POST.get('customer_id') and request.POST.get('customer_name') and request.POST.get('customer_address'):
            saverecord = EmpModel()
            saverecord.customer_id = request.POST.get('customer_id')
            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')

```

Python 3.10.0 64-bit 0 0 0

Ln 39, Col 25 Spaces:4 UTF-8 CRLF Python Go Live Prettier

```

File Edit Selection View Go Run Terminal Help
views.py - frontend - Visual Studio Code

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

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!!!')
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 0 ▲ 0 In 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

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

frontend > models.py > OfferModel
from django.db import models

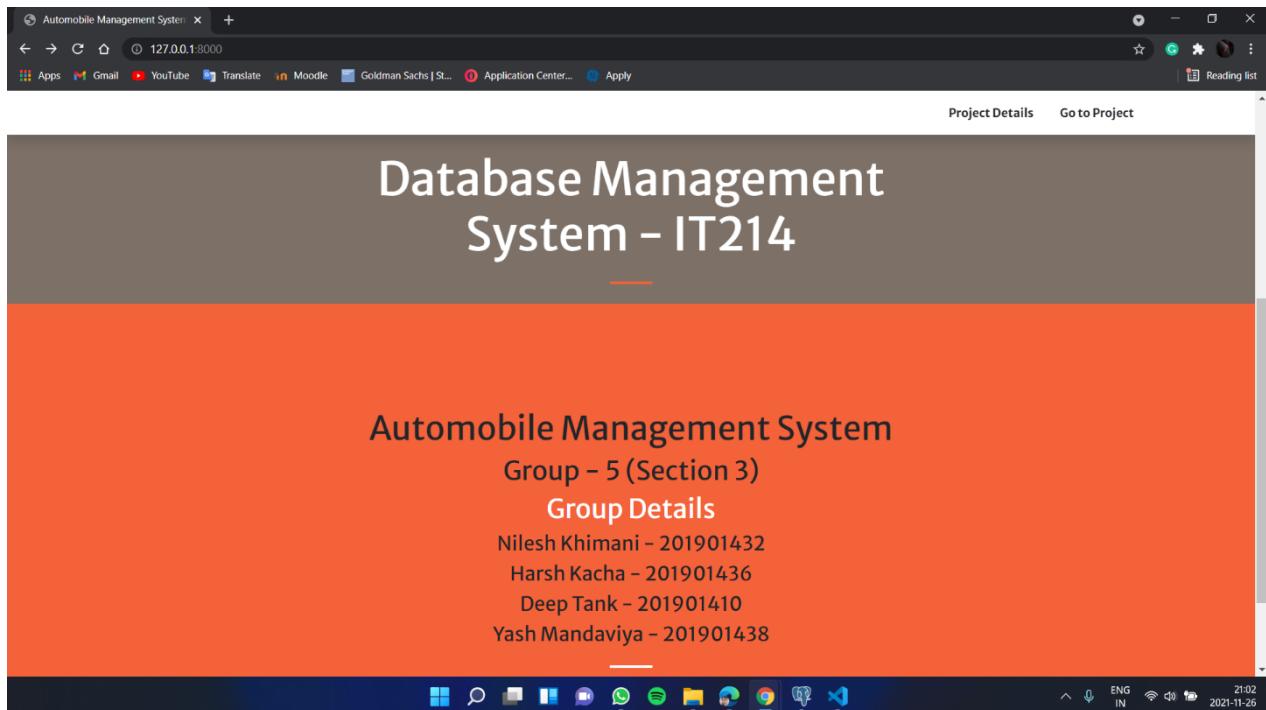
class EmpModel(models.Model):
    customer_id = models.CharField(max_length=100, primary_key=True)
    customer_name = models.CharField(max_length=100)
    customer_address = models.CharField(max_length=100)
    email = models.CharField(max_length=100)
    class Meta:
        db_table = '"as_db"."customer"'

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 0 ▲ 0 In 14, Col 50 Spaces: 4 UTF-8 CRLF Python Go Live Prettier

7.3 Snapshot of Webpage:

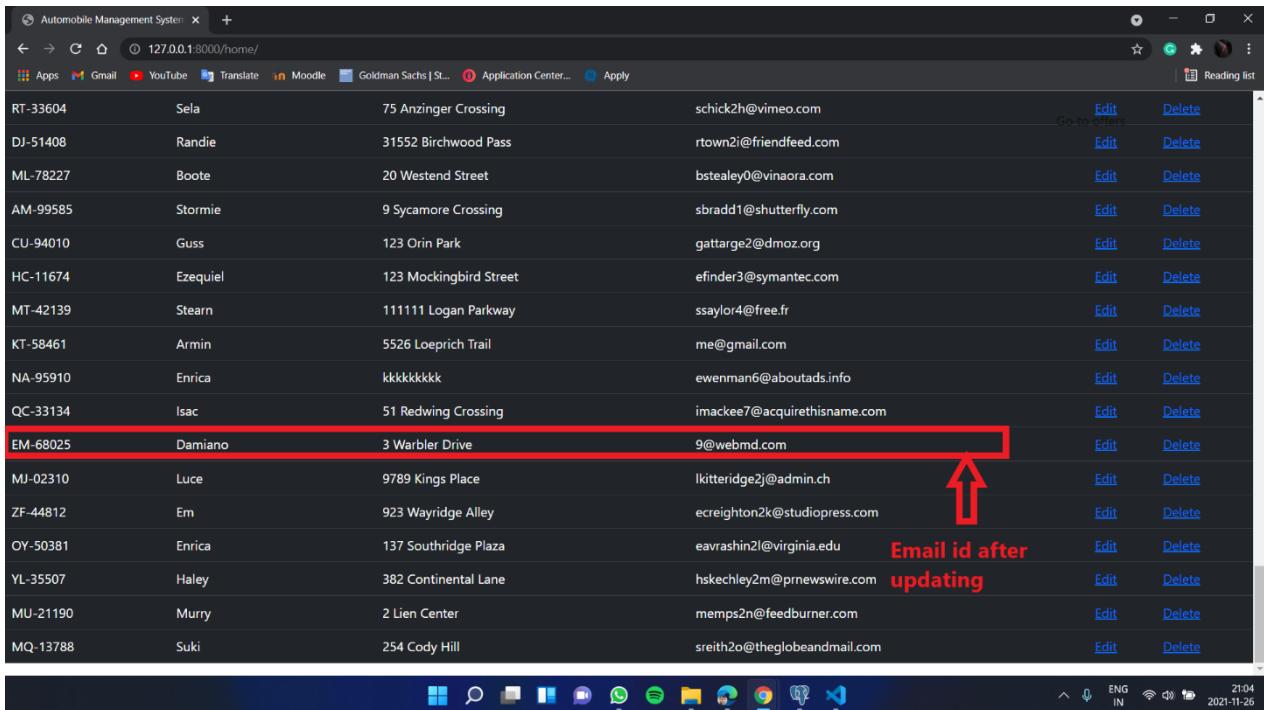


7.4 Snapshot of Queries and Outputs:

❖ Update

Email id before updating

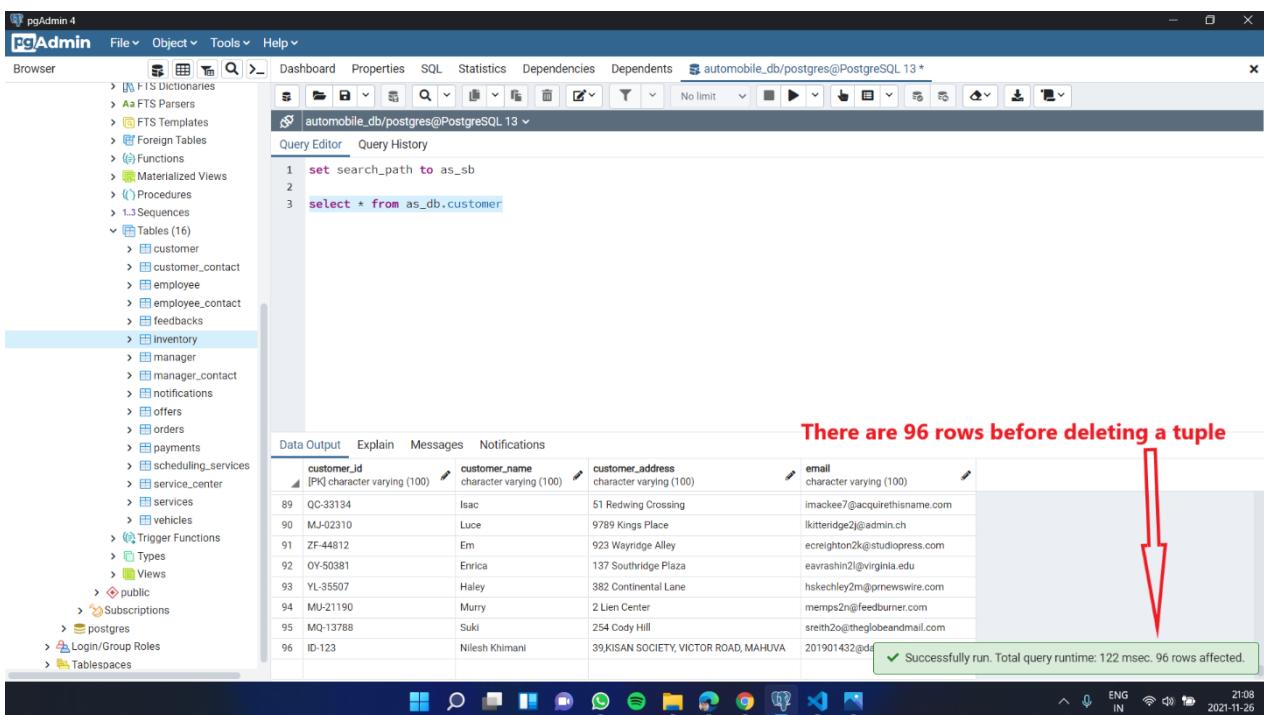
Customer ID	Customer Name	Address	Email	Edit	Delete
EM-68025	Damiano	3 Warbler Drive	dwine9@webmd.com	Edit	Delete
RF-13605	Ardith	9592 Sauthoff Crossing	acristofanoa@usgs.gov	Edit	Delete
FD-22169	Reggis	8816 Northridge Drive	rfarrarb@latimes.com	Edit	Delete
QW-93437	Bale	9184 Glacier Hill Pass	btregensoec@twitter.com	Edit	Delete
FV-25340	Katy	09 Division Place	ktrilletd@hostgator.com	Edit	Delete
BX-02349	Leona	9 Twin Pines Lane	lhanscome@google.nl	Edit	Delete
JZ-98723	Lucian	3 Washington Court	lconeybearef@unesco.org	Edit	Delete
XF-01407	Quintana	536 Spaight Terrace	qmallalg@ustream.tv	Edit	Delete
JN-89651	Shell	64057 Morrow Trail	shabberth@go.com	Edit	Delete
QJ-51997	Lavinie	90 Vidon Trail	lmacklami@about.com	Edit	Delete
RG-22090	Katha	28883 Mariners Cove Court	kgawthorpej@wikispaces.com	Edit	Delete
SW-08664	Dorena	84 Fuller Alley	dbrehenyk@latimes.com	Edit	Delete
SL-02402	Hailee	5296 School Drive	hklimentyonokl@webnode.com	Edit	Delete



Automobile Management System

Customer ID	Name	Address	Email	Action
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@shutterstock.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	11111 Logan Parkway	ssaylor4@free.fr	Edit Delete
KT-58461	Armin	5526 Loeprich Trail	me@gmail.com	Edit Delete
NA-95910	Enrica	kkkkkkkk	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	ereighton2k@studiorpress.com	Edit Delete
OY-50381	Enrica	137 Southridge Plaza	eavashin2l@virginia.edu	Edit Delete
YL-35507	Haley	382 Continental Lane	hskechley2m@prnewswire.com	Edit Delete
MU-21190	Murry	2 Lien Center	mempis2n@feedburner.com	Edit Delete
MQ-13788	Suki	254 Cody Hill	sreith2o@theglobeandmail.com	Edit Delete

❖ Delete



pgAdmin 4

File Object Tools Help

Browser

- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Procedures
- Sequences
- Tables (16)
 - customer
 - customer_contact
 - employee
 - employee_contact
 - feedbacks
 - inventory
 - manager
 - manager_contact
 - notifications
 - offers
 - orders
 - payments
 - scheduling_services
 - service_center
 - services
 - vehicles
- Trigger Functions
- Types
- Views
- public
- postgres
- Login/Group Roles
- Tablespaces

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	customer_name	customer_address	email
89	QC-33134	Isac	51 Redwing Crossing
90	MJ-02310	Luce	9789 Kings Place
91	ZF-44812	Em	923 Wayridge Alley
92	OY-50381	Enrica	137 Southridge Plaza
93	YL-35507	Haley	382 Continental Lane
94	MU-21190	Murry	2 Lien Center
95	MQ-13788	Suki	254 Cody Hill
96	ID-123	Nilesh Khimani	39 KISAN SOCIETY, VICTOR ROAD, MAHUA
			201901432@df

There are 96 rows before deleting a tuple

Successfully run. Total query runtime: 122 msec. 96 rows affected.

customer_id that we have deleted

```

1 set search_path to as_sb
2
3 select customer_id from as_db.customer where customer_id = 'EM-68025'

```

customer_id
EM-68025

Successfully run. Total query runtime: 128 msec. 0 rows affected.

There are 95 rows after deleting a tuple

customer_id	customer_name	customer_address	email
NA-95910	Enrica	kkkkkkkk	ewenman6@aboutads.info
QC-33134	Isac	51 Redwing Crossing	imackee@acquirethisname.com
MJ-02310	Luce	9789 Kings Place	lkitteridge2@admin.ch
ZF-44812	Em	923 Wayridge Alley	ecreighton2@studiodress.com
OY-50381	Enrica	137 Southridge Plaza	eavashini2@virginia.edu
YL-35507	Haley	382 Continental Lane	hskechley2m@prnewswire.com
MU-21190	Murry	2 Lien Center	mamps2n@feedburner.com
MQ-13788	Suki	254 Cody Hill	sreith2o@theglobeandmail.com

❖ Insert

Screenshot of a web browser showing a table of data. A red arrow points from the text "New inserted tuple" to the last row of the table, which is highlighted with a red border. The table has columns: ID, Name, Address, Email, Edit, and Delete.

ID	Name	Address	Email	Edit	Delete
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	kkkkkkkk	ewenman6@aboutads.info	Edit	Delete
QC-33134	Isac	51 Redwing Crossing	imackee7@acquirethisname.com	Edit	Delete
MJ-02310	Luce	9789 Kings Place	lkitteridge2j@admin.ch	Edit	Delete
ZF-44812	Em	923 Wayridge Alley	ereighton2k@studiotpress.com	Edit	Delete
OY-50381	Enrica	137 Southridge Plaza	eavashin2l@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
ID-123	Nilesh Khimani	39,KISAN SOCIETY, VICTOR ROAD, MAHUSA	201901432@daiict.ac.in	Edit	Delete

❖ Offer table

Screenshot of a web browser showing a table titled "Offers Table". The table has columns: Offer ID, Service ID, Start Date, End Date, DiscountPercentage, and Max Discount. A red arrow points from the text "Offers Table" to the title of the table.

Offer ID	Service ID	Start Date	End Date	DiscountPercentage	Max Discount
OFR-1001	SEC-7101	April 18, 2019	Sept. 30, 2020	10	103
OFR-1002	SEC-7102	June 26, 2018	Dec. 31, 2020	7	265
OFR-1003	SEC-7103	Nov. 13, 2018	May 3, 2020	3	95
OFR-1004	SEC-7104	Sept. 28, 2019	April 1, 2021	10	359
OFR-1005	SEC-7105	Jan. 16, 2018	Sept. 27, 2020	3	389
OFR-1006	SEC-7106	Oct. 25, 2019	Sept. 7, 2021	3	274
OFR-1007	SEC-7107	June 16, 2019	July 28, 2021	6	162
OFR-1008	SEC-7108	June 28, 2018	Oct. 16, 2021	8	381
OFR-1009	SEC-7109	May 7, 2019	July 19, 2020	2	74
OFR-1010	SEC-7110	Nov. 7, 2018	March 16, 2021	6	186
OFR-1011	SEC-7111	April 12, 2019	March 24, 2021	5	417
OFR-1012	SEC-7112	July 16, 2018	March 17, 2020	7	473
OFR-1013	SEC-7113	Oct. 4, 2019	July 1, 2020	8	106
OFR-1014	SEC-7114	Oct. 26, 2018	June 7, 2021	2	443
OFR-1015	SEC-7115	Dec. 24, 2018	May 15, 2020	3	473