

# **Task Management System**

**S10\_T5**

**DBMS - IT214**

**Instructor:** Prof. Minal Bhise

**Mentor TA:** Dhairy Patel

**Team Members :**

201901080 : Boricha Tarun

201901087 : Makwana Vivek

201901091 : Rohit Mahima

201901140 : Patel Khushbu



Dhirubhai Ambani Institute of  
Information and Communication  
Technology

# **Index**

## **1. Final SRS document**

Introduction  
Requirements collection  
Fact finding chart  
List requirements  
User class characteristics  
Operating Environment  
Product functions  
Privileges  
Assumption  
Business Constraints

## **2. Noun Analysis**

## **3. ER Diagram**

ER Diagram version 1  
ER Diagram version 2  
ER diagram version 3

## **4. Final ER Diagram to Relational Model**

## **5. Normalization and Schema Refinement**

## **6. SQL**

DDL Scripts  
40 SQL Queries

## **7. Project code with output screenshot**

# **Section 1**

## **Final version of SRS**

[A]

## **Introduction :**

### **1. Purpose:-**

- Provide users with an interface through which they may log into the system.
- Provide project managers with an interface through which they may initialize a new project.
- Provide project managers with an interface through which they may dynamically add or remove processes from a particular project.
- Provide project managers with a system alerting them of inconsistencies in process dependencies among the processes chosen for a project.
- Provide project managers as well as team members with an interface through which they may view all of the processes for a project in the form of a dependency graph.
- Provide project managers as well as team members with an interface through which they may update the inputs, outputs, and status of a process.
- Obtain information from and make changes to the database quickly and accurately.

### **2. Intended Audience and Reading Suggestions:-**

This section of the software requirement specification (SRS) document is written for a more general audience, this part is intended for individuals directly involved in the development of projects. This includes hardware-software developers, project consultants, finance manager and team managers and employees. In our hardware manufacturing company, we need mainly engineers who are experts in hardware components and related software like assembly language and microcontroller level coding.

In the general reading suggestion it includes goals and objectives, project scope, general system details, and some major constraints associated with the intended platform which is generally used by the manager.

People or employees who are working in designing a product, they should read the design part, which includes rules and some good samples of design. In our company it is the design of hardware chip, IC, components.

Readers interested in how to organize and handle data should consult data design who are database engineers, which covers data structures and flow patterns utilized by the system. Restrictions, Limitations, and Constraints discusses the general constraints in the project.

Testing engineers interested in the hardware and software testing process should consult the testing issue, which offers a list of test cases, expected responses, information about the testing machine and other pertinent information.

Other side of the reading suggestion is for market people, who use the company's product. For them there is documentation which includes detailed specifications about the product, for example in our company suppose the product is headphone then It includes general info(color,model name, brand name), connector type, from factor, frequency, operating system compatibility etc.

### **3. Product Scope :**

This product has a wide range of applications. So this project helps a firm to be managed very easily. By using this task management system we can keep track of the daily progress of tasks. Some of the functionalities of this task management system is described below :

login : An official can have access to this application through a login id and password.

Display task list : It will display all the tasks assigned to the employees.

Create a new project: For creating a new project

progress: If a task is going on then it will indicate how much work is done and it will keep track of the project's process without talking with the group leader or manager.

Task assigning : The higher authorities can assign a new task to the employees by using it.

Correction comments: If something is going in the wrong direction while executing the task then a correction comment will be displayed to indicate the invalidness of the execution.

Insert/Delete/Update : Manager or employee can insert/Delete/update any information /task .

### **4. Description**

One of the hardest things about running any project is dealing with the messy middle. You might have a vision of what things should look like when you're done, and a good idea of where you are now. But getting there? It's safe to say even the best teams can get lost along the way.

There is an easy tool that can guide your team through the messy middle: Task management.

Instead of huge, vague, and unclear project goals, tasks are clear, descriptive, step-by-step instructions. By breaking down every part of your project into a detailed task, you get a better picture of how you're going to bring it to life. Otherwise, it's like trying to do a puzzle without any idea of what it looks like.

Task management is both a science and an art. But with the right methods and a proper task management system in place, you can ensure your projects run smoothly from start to finish.

### **What is task management?**

Task management is a pretty simple idea. But simple doesn't always mean easy.

In its most basic form, task management is the process of managing a task through its life cycle—from planning, testing, tracking, to reporting on the outcome.

Task Management involves managing all aspects of a task, from its status and priority, to the time, human, and financial resources it needs.

When you're working with a team, proper task management is the foundation for productive days. Task management tools and techniques give you a detailed and up-to-date view of all the moving parts of a project. It helps everyone on your team stay in-sync, productive, and on schedule.

And as an added bonus, breaking projects and milestones down into specific tasks can be a huge motivator.

A powerful task management tool or software is your best friend when it comes to organizing, assigning, tracking, and reporting on tasks.

Not only will it keep all your info in one place that's visible to the entire team, but it also lets you see your task and project progress and where you might be overcommitting resources.

So, now that we understand why task management is so important to running projects, what's the best way to go about actually doing it?

The basics of task management are to capture, organize, and assign everything that needs to get done to complete your project. The more detail you can get upfront, the more people will know what needs to be done.

As with everything when it comes to project management, the only method that matters is the one that works for your team. So while this guide will give you everything you need to know to start properly managing tasks, make sure to experiment and see what your team likes or doesn't.

### How to properly prioritize tasks

You can't just look at your giant list and assume you'll know what to do next. Instead, every task needs to be properly prioritized to know what needs to happen to keep the project running smoothly. But this isn't always easy.

Some tasks will naturally be higher priority than others. But there will also be some nebulous ones that you'll be stumped by. You'll want to start by defining the scale you're going to prioritize on. Something like:

- Critical
- High

- Medium
- Low

Tasks by their very definition are tiny. But the return they bring you and your team are massive.

Take the time to break down tasks into their smallest pieces, prioritize them ruthlessly, and find a task management software that gives you visibility, flexibility, and awareness (like Planio) and you'll be on your way to turning that big, scary, audacious project into an actionable, step-by-step plan.

### **Hardware Manufacturing Task Management System :**

1). This task management system will allow us to manipulate the data very easily. Maybe in earlier times this was done on paper which was very hard to deal with. This system also includes a login system for companies and employees.

2). By this system companies which are working for each other and suppliers become more connected. And the employees who are working in various tasks connected with a virtual platform where they can share ideas and make work more perfect. Managers also have direct contact to all employees, So all employees get equal work according to their skill and expertise like software part, hardware level coding, hardware manufacturing. Managers can make sure that all employees are working properly and there is no such thing like someone is loaded with lots of work and someone is free.

3). It is very important to keep information about raw materials. How much is in stock?, From where we will get if stock is empty or in case a company is providing right now that stop providing then which company is next.

4). This system is very useful In order to update database information which comes from employees, customers, other companies(which supply raw materials, components) and for those who access the database and keep updating the Data.

5). Database has been updated with real time so that all the information like employee suggestions or other suggestions from head office, so that company can make appropriate and efficient decisions in working projects.

6). One important factor for task management is safety of workers, safety is important while working on various projects.

There must be all information in the database about the department of the company which is at high risk of accident, and precautions to be taken for working in those areas. If an accident happens then what to do to control the situation very fast. One person should monitor the activity of those areas via a cctv camera .

7). This task management system contains following :

i). Information Storage: This stores all details about employees(ID, department, expertise, phone no., etc), all manager details, required customer details, all product details and lots of things.

ii). Data security:

The major role of the software database comes when we talk of the security level of databases because it's very necessary to keep all the information private and safe from malicious activities. Because of one attack or mistake, the whole system can fail.

Data security is there so that no one can manipulate any information on other employees details or read some personal info. For this we have a login(username and password) system. An authorised person only can access particular data like manager's data can't be accessed by employees. All customers' accounts are also protected, it must be protected with high security because some of the accounts are linked with the customer's debit card, credit card etc.

iii). Customers' data: Containing all the information about customers (customer id, name, phone no. , purchased items, date of purchase).

iv). Update easily: Anyone with permission can update that particular data very easily. And when we need to update the structure of the database, It will also be easy to do by a database expert.

v). All device compatibility: In this database all the information is kept in the way such that other devices with less memory and low speed processor can easily access the data.

8). After production of a hardware, the person who is testing it has to write all the things he noticed and require changes. So it can be taken under consideration during the final testing. So there should be facility to track the testing part.

9). There is a whole section in the database for advertisement, where all our advertisement videos and posters and details about advertisers are kept. Manager can track information about all advertised mediums and how it is going. so that our product can reach the people and other job interested people apply to the company.

10). We classified customer feedback and employee feedback into two different groups so we maintain the simplicity of the database.

11). After the production is complete our task is to put the products into market and keep track of sales. Keep all details about product goes where if we are exporting in another country, profit we get from them, taxes etc.

12). Limitation of this database is that it does not support all speaking languages yet, so people who are using it need to know that particular language (ex. English, Hindi).

[B]

## 1. Background Reading

### • Description of each reading done

The IT hardware sector basically comprises activities such as need assessment and definition, development, pre-production processes, production, sales and service support.

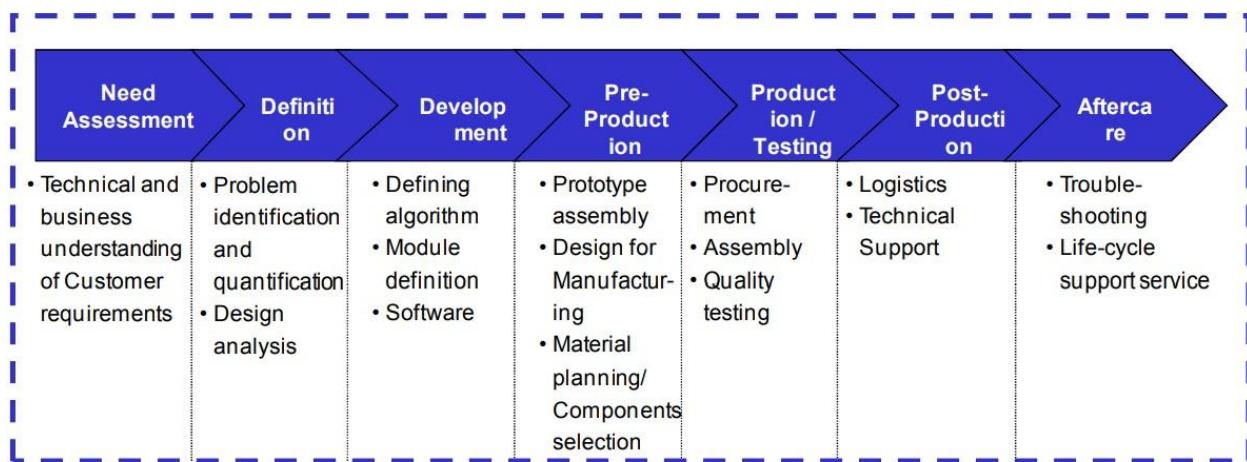


Figure : Core processes in the IT hardware sector

1. **Need Assessment:** For this, a cross functional team composed of people with background in engineering, manufacturing, materials management, and sales and marketing assess the technical and business aspects of the customer requirements. These are then taken up in further detail by the definition team.
2. **Definition:** Based on the need assessment, the engineering team works to define the customer application requirements and help finalise the exact hardware requirements/software configuration. The design concept is then chalked out considering the hardware configuration and cost estimates during the complete manufacturing process.
3. **Development:** With the aid of algorithm engineering, i.e. a combination of theoretical algorithm design with real-world data via a hardware device, verification of algorithm results and behaviour is ascertained. Definition module includes packaging of various units which have a well-defined interface with the other components, and developing the enabling software for hardware to function.
4. **Pre-Production:** The pre-production process begins with prototype assembly to identify the manufacturing process and finalise the product design. The design for manufacturing analysis is

performed to point out cost reduction and system improvements at the prototype level. Further, to achieve desired quality and price level, best available components are identified for the product design.

5. Production/Testing: Best-in-class components at competitive prices need to be procured from the existing/new vendors developed for the assembly process. Assembly services shall provide complete box-build and full system assembly for the computing hardware and finally conclude the production process with functional and system testing.
6. Post-Production: This includes arranging logistics and providing technical support for the product.
7. Aftercare: This includes post sales trouble-shooting and repair and maintenance. Managing and providing life-cycle support service for the product.

- **References**

1. [Microsoft Word - MAIT Report on Manpower for electronics industry Novemberr v0 25 \(meity.gov.in\)](https://meity.gov.in)

- **List the combined Requirements gathered from Background Reading/s.**

- 1) For manufacturing of Hardware devices, we need raw materials which are useful for production of the hardware. We can take the example of a computer system and mobile processor, for which we need ICs and circuits.
- 2) After the hardware manufacturing part is done ,we need to test it. So I can fix those issues. But even after testing some issues are still there. So we have to take the review of the customers. And work accordingly on those issues.
- 3) Hence we can give technical support, trouble-shooting and life cycle support service for the customer as an aftercare service.

## **2. Interview/s**

### **Hardware Manufacturing Company/Interview Plan**

**System:** Hardware Manufacturing System(HMS)

**Interviewee:** 1) Mahima Rohit

**Designation:** CEO at HMS

**Contact Details:**

**Email :** [mahimarohit@gmail.com](mailto:mahimarohit@gmail.com)

**Phone no:** 9510345713

**Organization Details:** 676 , Near Home Town super market ,Bengaluru

**Interviewer:** 1) Rohan Shah

**Designation:** Hardware & Networking Engineer

2) Jigar Patel

**Designation:** Hardware Engineer

**Date:** 05/10/2021 **Time:** 12:00

**Duration:** 30 minutes    **Place:** Google meet

#### **Purpose of Interview:**

Preliminary meeting to identify problems and requirements regarding security of task management system

#### **Agenda:**

Problems with security

Problem with lack of management

Initial ideas

Follow-up actions

#### **Documents to be brought to the interview:**

documents relating to current security procedures

documents relating to the suggestion

## **Interview Summary**

**System: Hardware Manufacturing System**

**Interviewee: 1)** Mahima Rohit

**Designation:** CEO at HMS

**Contact Details:**

Email : mahimarohit@gmail.com

Phone No : 9510345713

**Organization Details:** 676 , Near Home Town super market,Bengaluru

**Interviewer: 1)** Rohan Shah **Designation:** Hardware and networking engineer

**2)** Jigar Patel

**Designation:** Hardware engineer

**Date:** 5/10/2021      **Time:** 12:00

**Duration:** 30 minutes      **Place:** Google meet

### **Purpose of Interview:**

Preliminary meeting to identify problems and requirements regarding security of task management systems.

- ★ Not very effective security in the system
- ★ Secure login
- ★ Some information is accessed by only the specified person and not by all
- ★ Priority of task

### 3) Questionnaire:

The screenshot shows a Google Form titled "Employee" from a hardware manufacture company. The user is signed in as "thevivek211@gmail.com". A question asks "How many hours do you work in a day" with four options: "6 hour", "8 hour", "12 hour", and "Can't say, depending on work".

Below the form, the Windows taskbar is visible with icons for File Explorer, Google Chrome, and other applications. The system tray shows the date as 09-10-2021, the time as 12:17 AM, and the weather as 29°C Rain.

The screenshot shows a continuation of the Google Form. A question asks "Which part of working area(department) you work mostly" with six options: "Integrated circuit department", "Soldering", "Display", "Marketing, Finance", "Sales", and "Electric power managements".

Below this, another question asks "How much average time do you spend at power management department." with four options: "1 hour", "2 hour", "5 hour", and "More than 5 hour".

At the bottom of the form, there is a question "How much is machines and device which is used to manufacture hardware accurate and easy to use?" with a small icon next to it.

Below the form, the Windows taskbar is visible with icons for File Explorer, Google Chrome, and other applications. The system tray shows the date as 09-10-2021, the time as 12:17 AM, and the weather as 29°C Rain.

How much is machines and device which is used to manufacture hardware accurate and easy to use?

1      2      3      4      5

Very good                                    Very bad

Have you ever found safety issue, security issue in system?

Yes  
 No

If you found any, write here without fail.

Your answer

If not found but want more security then also write here, give suggestion.

Your answer

Any other suggestion or comment

...

S10\_T5\_SRS\_V\_1\_21-Sep-2021.xls Employee - Google Forms Employee

Employee - Google Forms

Employee

Responses 8

Questions Responses Settings

Who has responded?

Email

201901091@daiict.ac.in

201901140@daiict.ac.in

201901087@daiict.ac.in

tarunboricha2092001@gmail.com

vmakwana@gmail.com

shubhamchudasama999@gmail.com

abhishek123@gmail.com

rahul1213@gmail.com

How many hours do you work in a day

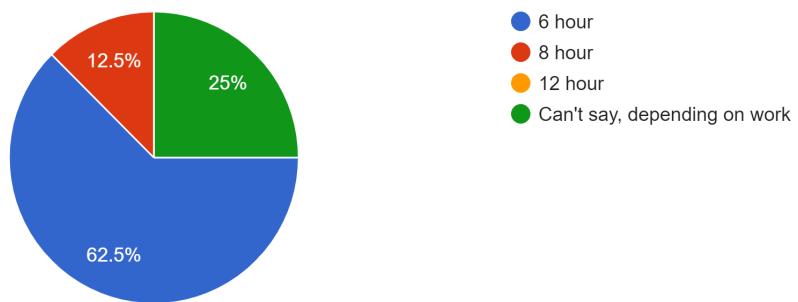
Type here to search

Windows 10 icons

Cloud 29°C Rain 12:23 AM 09-10-2021

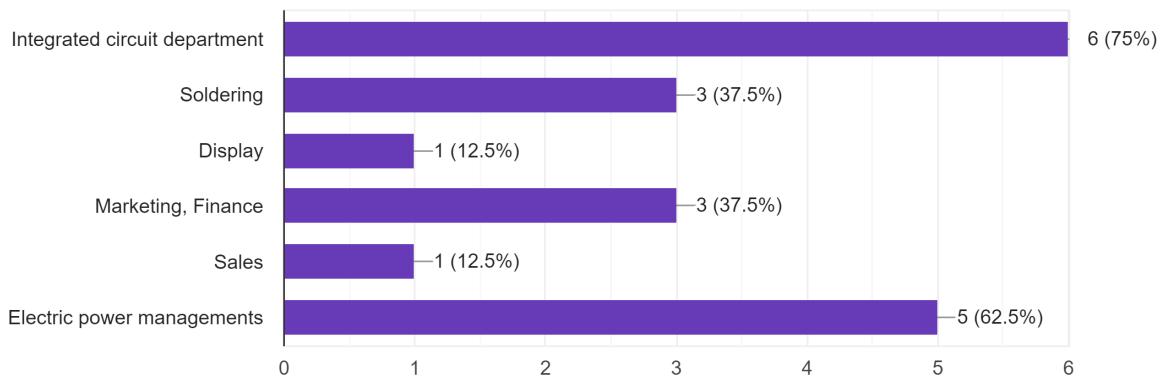
How many hours do you work in a day

8 responses



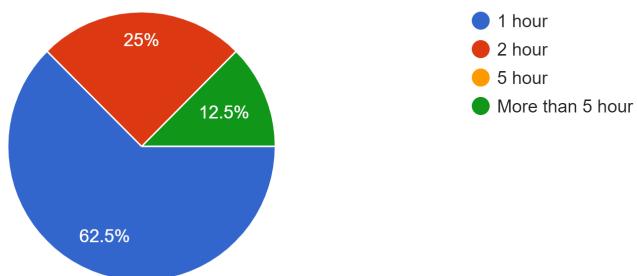
Which part of working area(department) you work mostly

8 responses



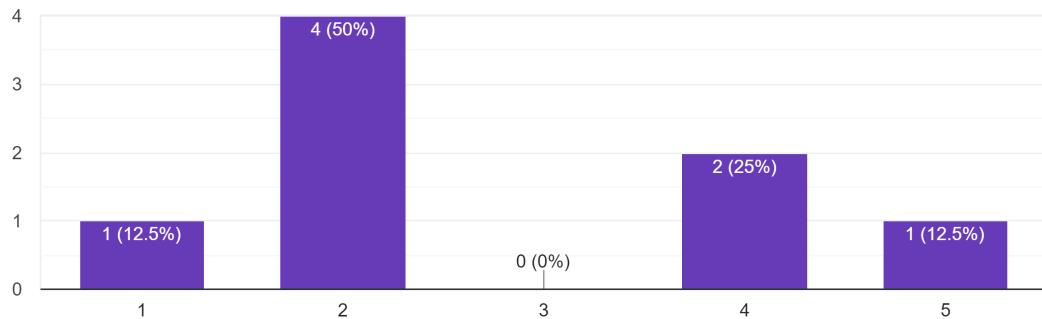
How much average time do you spend at power management department.

8 responses



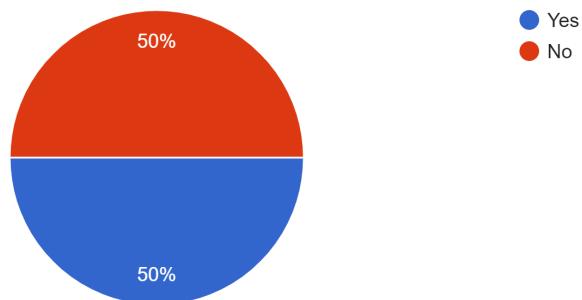
How much is machines and device which is used to manufacture hardware accurate and easy to use?

8 responses



Have you ever found safety issue, security issue in system?

8 responses



The screenshot shows a Microsoft Forms survey interface. At the top, there are navigation icons for Employee, Questions, Responses (8), Settings, and a Send button. Below the header, there are three text input boxes for different types of comments.

- If you found any, write here without fail.**  
4 responses  
security issues when machines used for long hours  
Sometimes short circuit in system  
Some old device sometimes heats a lot  
Blast in circuit
- If not found but want more security then also write here, give suggestion.**  
2 responses  
improve the security  
Remove some old device
- Any other suggestion or comment**  
2 responses  
need to improve in site  
Bring some new device where needed

S10\_T5\_SRS\_V\_1\_21-Sep-2021.j | Customer - Google Forms | Customer | +

docs.google.com/forms/d/e/1FAIpQLSf9R\_UaQuIAK1NE7\_JaP7V\_zfx5SVjQnVvkyZwWZ4lg4UA/viewform

## Customer

Fill this form as feedback after experiencing the product.

thevivek211@gmail.com (not shared) [Switch account](#)

\* Required

Your purchased item and model no:\*

Your answer

Have you ever purchased any device from our company before this one?

Yes  
 No

Are you satisfied with product and it's design? \*

1 2 3 4 5

Very much  Not at all

Type here to search

29°C Rain 12:29 AM 09-10-2021

S10\_T5\_SRS\_V\_1\_21-Sep-2021.j | Customer - Google Forms | Customer | +

docs.google.com/forms/d/e/1FAIpQLSf9R\_UaQuIAK1NE7\_JaP7V\_zfx5SVjQnVvkyZwWZ4lg4UA/viewform

How was strength and durability of the item? \*

1 2 3 4 5

Very good  Very bad

How much time did you use the purchased item? what was exact warranty time? \*

Your answer

Any suggestion, comment

Your answer

[Submit](#) [Clear form](#)

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) · [Terms of Service](#) · [Privacy Policy](#)

Type here to search

29°C Rain 12:29 AM 09-10-2021

S10\_T5\_SRS\_V\_1\_21-Sep-2021.j | Customer - Google Forms | Customer | + docs.google.com/forms/d/1qMzQY-AGrWJusmRTcm38Ue5\_-p6vMsmDxba8\_O4-uVE/edit#responses

Customer | ☆

Send M

Questions Responses 11 Settings

Your purchased item and model no:

11 responses

Smart Phone, A10  
Keyboard, K-4572  
Keyboard SM32435  
Laptop, inspiron 5584  
Nokia 2720  
Adapter , A-78  
Hard Disk, model no. - h7492x3  
Samsung A20  
Laptop hi1908wi

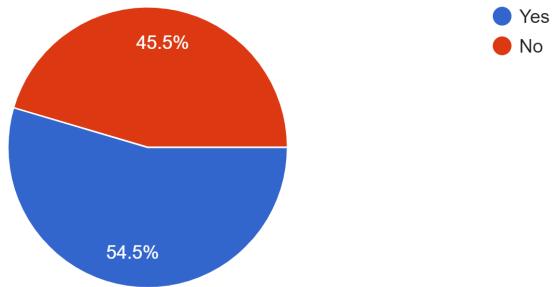
Have you ever purchased any device from our company before this one?

Type here to search

29°C Rain ENG 12:30 AM 09-10-2021

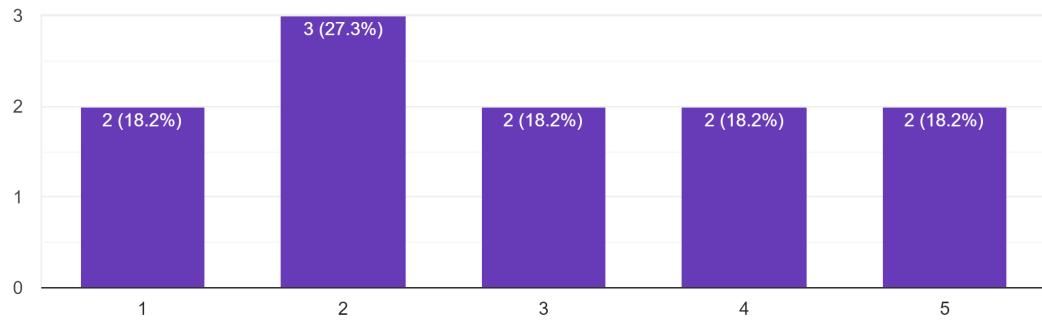
Have you ever purchased any device from our company before this one?

11 responses



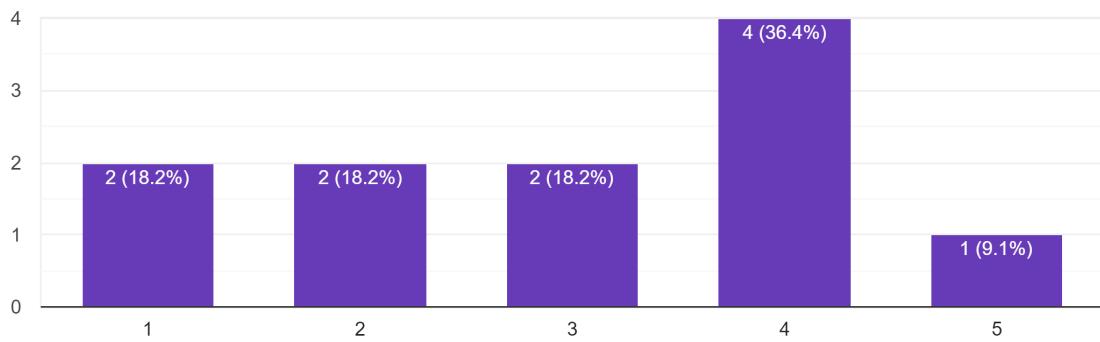
Are you satisfied with product and it's design?

11 responses



How was strength and durability of the item?

11 responses



How much time did you use the purchased item? what was exact warranty time?

11 responses

- 1.5 year, 1 year
- 10 months, 6 months
- 1 year
- 2 year, 1year
- Yes
- 7 months, 6 months
- I use often. Warranty time is 1 year.
- 1 year warranty, using from 3 ear
- 1 month and 12 month

Any suggestion, comment

7 responses

- No
- Make processor faster
- Increase warranty of the product and use strong material.
- NULL

## Requirements:

There are some complaints from employees mostly to increase security of the system. Machine heat, short circuit, old device not working properly. This requires taking it into consideration very first as soon as possible.

From the customer side they want more time in warranty, aspect faster processor. This needs to improve.

## **4. Observations: -**

### **Hardware Manufacturing Company: Observations**

**System :** Hardware Manufacturing System

**Project Reference :** CJ/WI/2019/07

**Observations by:** Tarun (Company employee)

**Date:** 10/09/2019 **Time:** 11:30

**Duration:** 30 minutes      **Place:** Head Office, Mumbai

### **Observations:**

1. Need to improve in the response and needful action time of feedback for customers.
2. Need to improve product hardware efficiency.
3. Need to upgrade some devices in the working area: computer, printer and security camera.
4. Products like mobile processors need 5nm base architecture to improve their power efficiency and improved performance.

## [C]. Fact Finding Chart

Objective	Technique	Subject(s)	Time commitment
To get background on Hardware related companies and their strategy of management	Background Reading	Hardware company sites like intel, snapdragon and mediatek and some articles	4 hour
To determine customer feedback	Questionnaire	Google form	2 days
To determine employee feedback	Questionnaire	Google Form	2 days
To solve problem of lack of security	Interview	Role play	1 hour
Summary	Observation	All above	1 hour

## [D]. List Requirements

We can collect all the required information regarding the project assigned and we can work on it on paper. After the complete task is done on paper ,we can give it to the system we are working on. This is helpful when we have some chances of changes in the information.

- **Details about security :** The security of company related information like on the current project the company is working on is a very important aspect nowadays. So it is required to provide security to the information we have and the security about the project ideas.
- **Details of Product :** We can start our task after we get all the information about the project. It is good if we can collect all the data and information that we are going to use in the task.
- **Employees Details :** To assign the task with no ambiguity ,we can simply assign the task to the employees ,if we have the necessary information about the employee like subject of expertise and the work experience. It helps a lot when we are assigning the task.
- **Customer Details :** Whenever the issue is raised by the customer for the product ,we need to gather some details of the customer. So we can easily communicate with them. It is our responsibility to provide the security to the customers details.
- **Update on the Product :** After we took an action on the product for the further development ,we need to track those changes. So in future, whenever the previous version is needed we can easily get access to that product.

## [E]. User Classes and Characteristics

Manufacturing is the basic building block of any organization and the primary reason why an organization exists. Manufacturing is the process that involves crafting of a product as per the company demand. Manufacturing is not only the work of labor but requires a number of professionals at management level who work hard to ensure the smooth running of the entire manufacturing process.

The major levels in the manufacturing management hierarchy are described as designations from the highest manufacturing management level to the lowest manufacturing management level.

1) List User Names with basic description of user role in the system/database

- **Chief Executive Officer :** The highest management head of the manufacturing company or department is always the CEO or simply saying Chief Executive Officer of the company. Entire work associated with manufacturing from planning to implementing and executing is done under the supervision of this professional who has years of vital experience of the field and is an integral part of the company.
- **Project Manager :** A project manager in the manufacturing management hierarchy is the associate professional that works in compilation with the manufacturing head on a specific project or for all the projects. Project managers can have the liability of taking care of the planning, implementation and closing of any manufacturing project in a company.
- **Senior Engineers :** Next in the manufacturing management hierarchy are the senior engineers who undertake the accountability of the management of the development, designing and manufacturing of the product being manufactured. They play a fundamental yet imperative role in the manufacturing management hierarchy.
- **Architect :** Architects are the skillful professionals who are responsible to finalize the final layout of the product to be manufactured. In some companies they are referred to as designers depending upon the work being tackled.
- **Head of Machinery Operations :** The machinery and tasks associated with the tools, equipment, machinery is under the supervision of these management professionals.
- **Head of the Sales –** The sales head supervises the promotion as well as sale of the manufactured product in order to work for the growth of the company.
  
- **Junior Engineers –** These professionals in the manufacturing management hierarchy play an important role since they perform the ground level work for the senior officials.

## [F]. Operating environment :

- 1). Computer windows, mac
- 2). Database to manage all produced products.
- 3). Wifi router for connectivity.
- 4).- Soldering machine.
  - Automatic IC Forming Machine
  - High Precision Double Spindles CNC Making Machine with Knife Magazine for Mobile Phone Tempered Glass
    - Making Machine Mobile Phone Battery Making Machine Pouch Cell Making
    - 300t High Speed Plastic Mobile Phone Case Making Machine Injection Molding Machine Machinery Price
    - Mobile Phone Touch Screen Making Machine, Automatic Mini Vacuum Lcd
    - 180t Horizontal Cold Chamber Aluminum Die Casting Machine for Making Hardware
    - High Precision Double Spindles CNC Making Machine with Tool Magazine for Hardware Fittings and Accessories

## [G]. Product Functions :

Providing information about the below objects :

- ★ Details about employees of the company
- ★ Details about project managers of the company
- ★ Projects details
- ★ Task details
- ★ Assignment of task
- ★ Keep track of the task progress
- ★ Comment on the task
- ★ Customer details

## [H]. Privileges

- ★ Details about Employees of the company  
Read - Some of the details can read all and some of the details can read only authorized person of the company  
Update : Only authorized person or that person can update
- ★ Details about project managers of the company  
Read/Update/Delete - Only authorized person can read and Update this details
- ★ Project Details

Read : All Employees of the company can read this details

Update : Only project manager can update this details

★ Task details

Read / Update : All can read and update this details

★ Assignment of task

Read : All can read

Update : Only project manager can update

★ Keep track of the task progress

Read : All can read

Update : Only the person who works on that task can update

★ Comment on task

Read : All can read

Update : Project manager or the person who works on that task can update

★ Customer Details

Read : All can read

Update : Only customer can update

## [I]. Assumptions

1. Quick implementation in new projects.
2. Companies provide proper functionality to their employees for new projects.
3. Head office quickly gives feedback on working projects.
4. Customer's problems are solved quickly.
5. Employees can update databases very easily.

## [J]. Business constraints:

Business constraints is a factor which affect negatively in the growth of the business.

1). Hardware products usually depend on many suppliers, materials, electronic components, manufacturing, etc. These things are not something you will make by yourself. Sometimes Raw materials are not available or available at high prices. So these dependencies could make hardware businesses inefficient.

- 2). Hardware requires a whole different set of skills in engineers. It is essential to have the right number of people with the appropriate skills to enable the business to achieve its business objectives, sometimes hard to hire skilled employees.
- 3). Hardware businesses suffer from high costs and low margins.
- 4) Companies need to make themselves powerful by time to time in order to stay in the market and in competition.
- 5) It is important to identify the nature of customers and their requirements through market research.
- 6) Common business constraints include budget and time restrictions, resource limitations, and resource skill limitations, size of the market, quality of direction and management, transportation costs, money management.

## **Section 2**

## **Noun Analysis**

**Table 1 :**

All Extracted Nouns &  
Verbs from Problem Description

Nouns	Verbs
employee	Assigned
Project	Creating
Task	Working
Company	Located
Product	Manufacturing
Skills	Required
Security	Ensuring
Staff	Allocate
Name	Managed
Work	Assigned
Safety	Resolved
Salary	Allotted
Date	Sentencing
System	Edit
Database	Maintained
Gender	Recorded
Attributes	Associated
Information	Included
Number	Required
Date of birth	Recorded
Managers	Managing
Hardware	Required

Software	Required
ID	Provided
Age	Recorded
Department	Handling
Expertise	Needed
Phone number	Required
Data security	Authorised
Login	Required
User name	Information
Password	Protected
Customer	Using
Customer id	Required
Item	Production
Memory	Useful
System	Regarding
Processor	Working
Raw materials	Required
Components	Making
Feedback	Useful
Safety	Required
Issues	Resolved

Employee ID	Required
Project Ideas	Information
Working Hours	Assigned
Marketing Strategy	Information
Payment Details	Information
Project Details	Required
Project type	Allocated
Assigned Employee	Assigned
Project ID	Required
Documents	Information
production	completed
Taxes	Pay

**Table 2 :**

Accepted Noun & Verbs list

Candidate entity Set	Candidate attribute set	Candidate relationship set
User	user_id user_password user_employeeID user.AuthenticationMode	Required

Project	Project_id Project_title Project_startTime Project_endTime project_status	Assigned
Task	task_id task_name task_employeeID task_projectTitle start_time end_time task_status	Allotted
Customer	customer_ID customer_name selected_item cutomer_contact	Purchased by
Employee	Employee_ID Employee_name Employee_address Date_of_birth Employee_phoneNo Employee_email	Working In

Product	product_ID product_name Manufacturing_date Warranty Total_No_product	Produced
Payment	payment_id customer_id payment_amount payment_date payment_mode	To be paid
Customer feedback	product_id customer_id customer_name selected_item feedback_description	feedback
Company	Company_id Company_name Contact Address	Data
	team_id	

Team	team_name No_of_employee project_id	Assign
Manager	Department Office_address	Manage

**Table 3:**

Rejected Noun & Verbs list from Problem Description

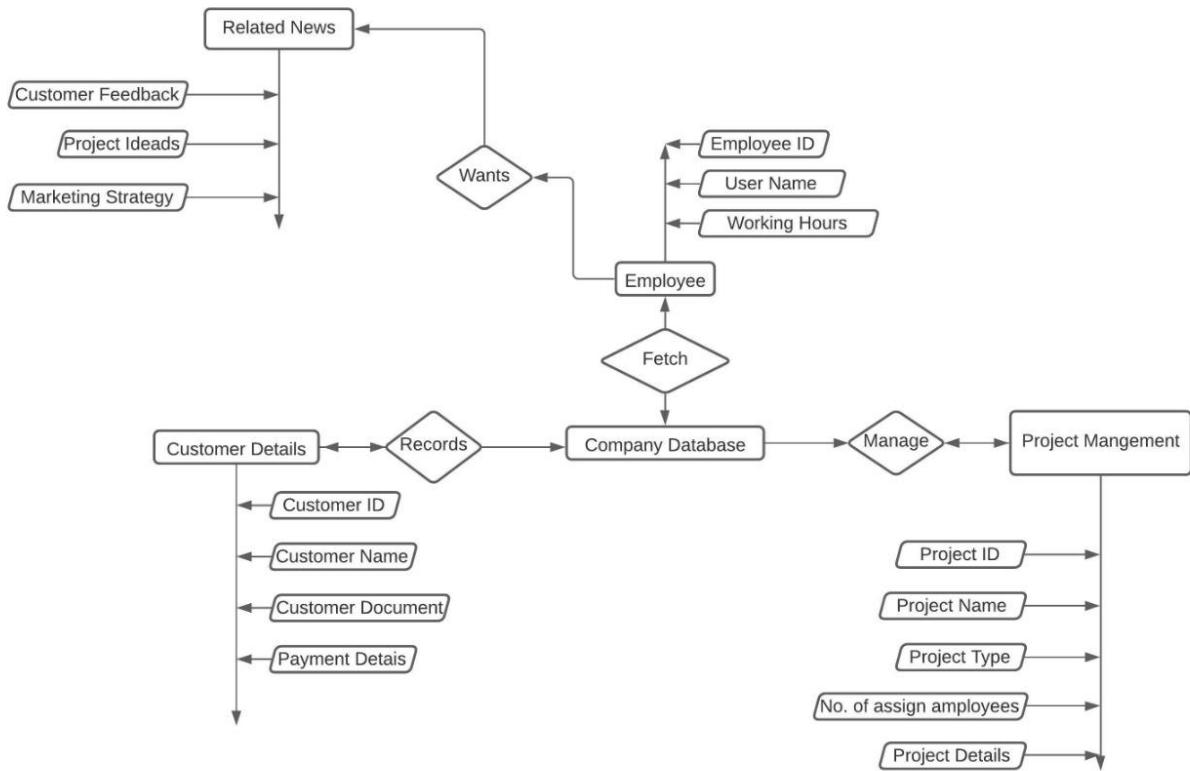
Noun	Reject Reason
Online	Irrelevant
Date	General
System	Irrelevant
Database	Irrelevant
Name	General
Information	Vague
Safety	Duplicate
Staff	General
Relation	General
Number	General
Attributes	General
System	Duplicate

# **Section 3**

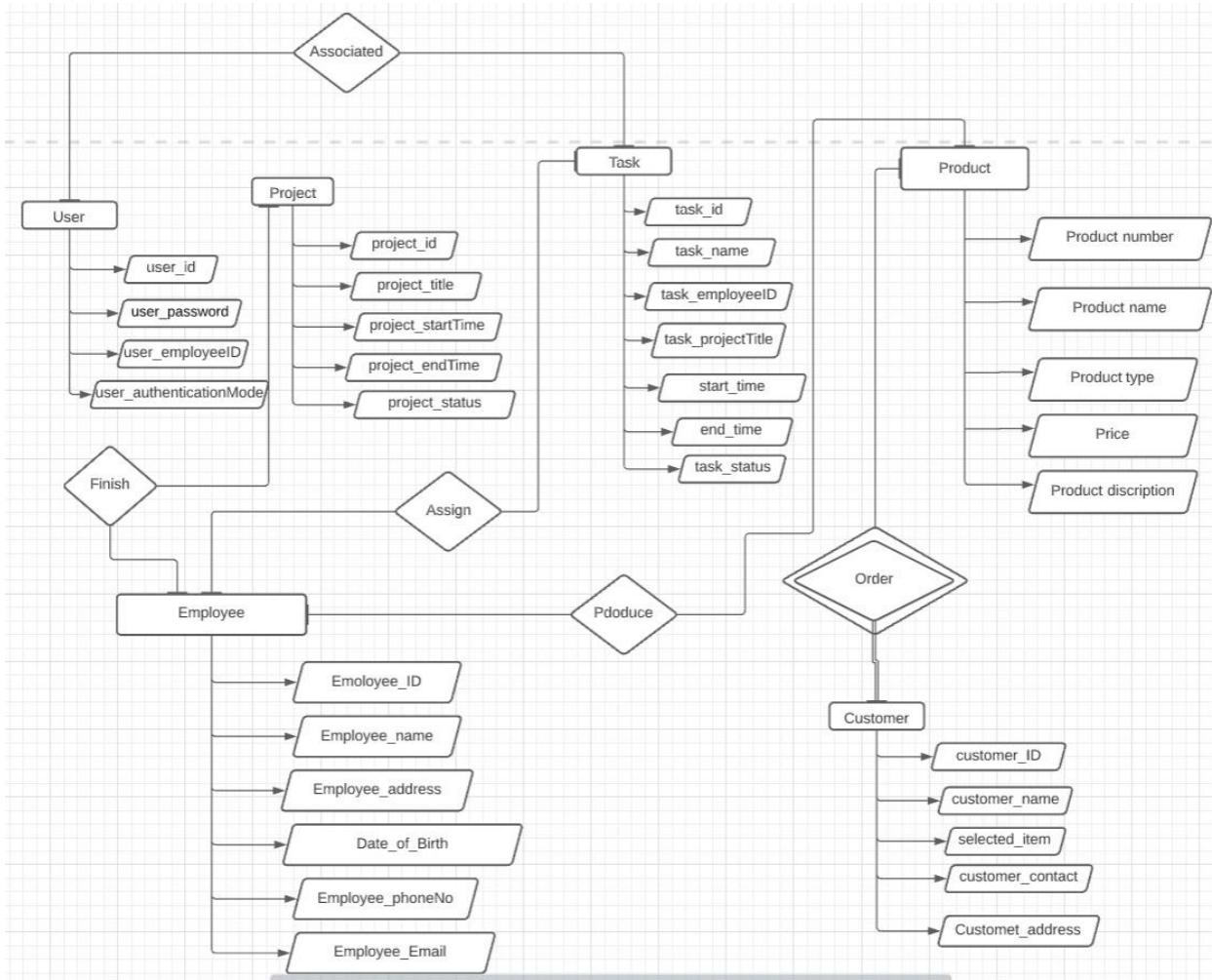
## **ER-Diagrams all versions**

## II. Develop the ER Diagram (ERD)

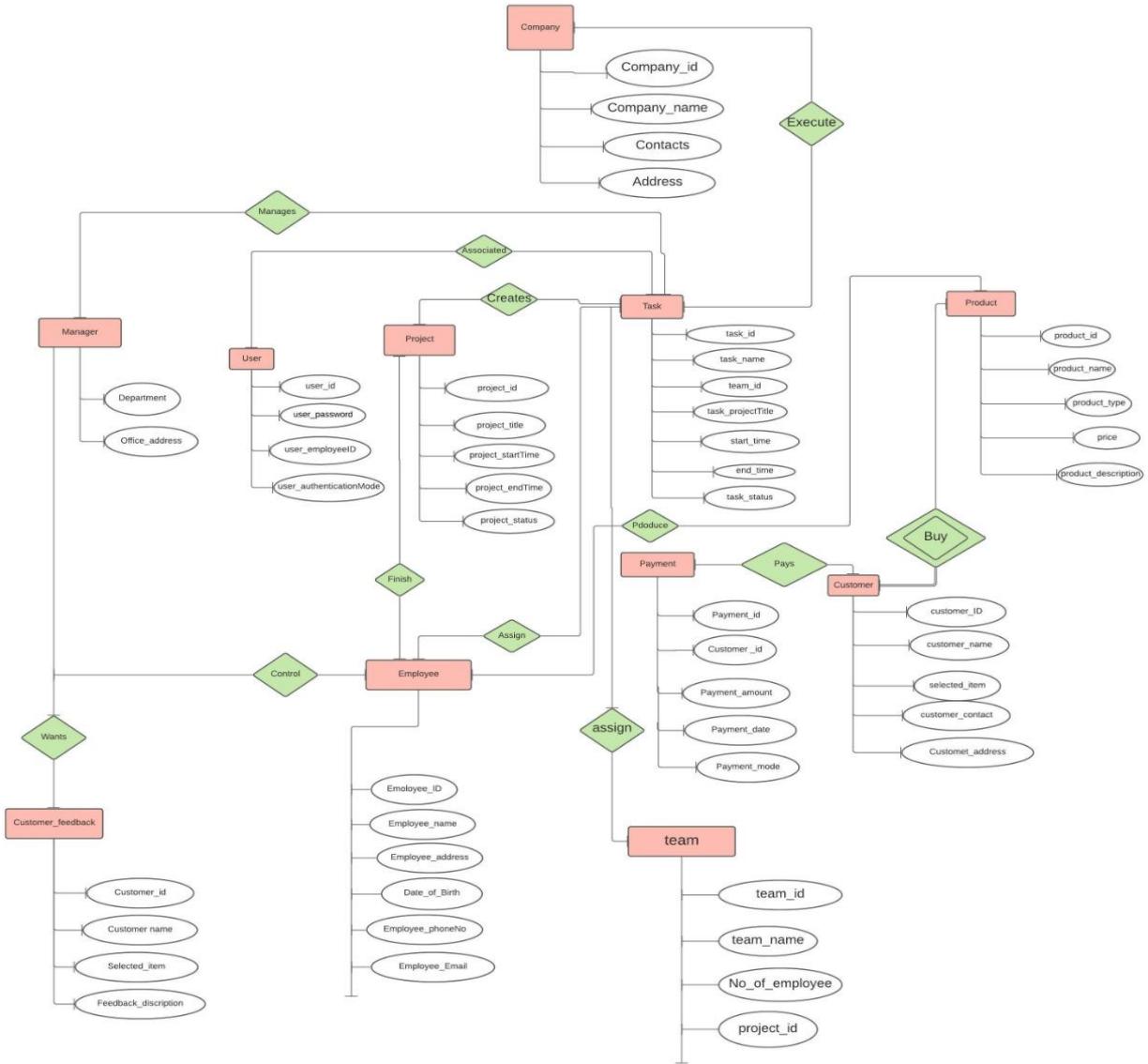
- Version 1: -



## Version 2: -



### Version 3: -



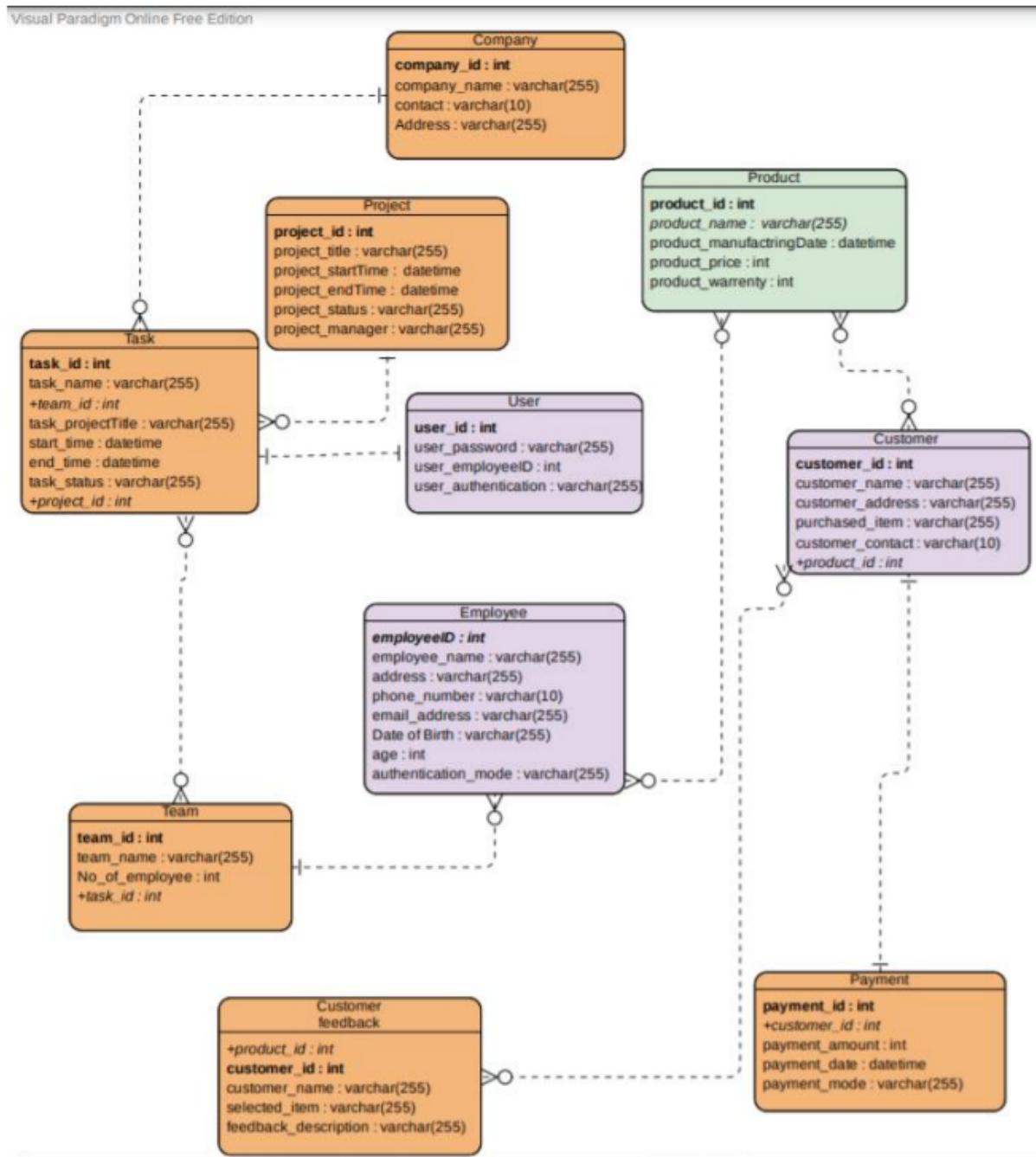
## **Section 4**

# **Conversion of Final ER-Diagram to Relational Model**

## **Mapping E-R Model to Relational Model: -**

- Company(company\_id, company\_name, contacts, address)
- User(user\_id, user\_password, user\_employeeID, user\_authenticationMode)
- Manager(id, department, office\_address)
- Employee(employee\_id, employee\_name, employee\_address, date\_of\_birth, employee\_phoneNo, employee\_email)
- Project(project\_id, project\_title, project\_startTime, project\_endTime, project\_status)
- Team(team\_id, team\_name, no\_of\_employee, project\_id)
- Task(task\_id, task\_name, task\_employeeID, task\_projectTitle, task\_status, task\_startTime, task\_endTime)
- Product(Product\_numer, Product\_name, Product\_type, Product\_price, Product\_description)
- Costumer(customer\_id, customer\_name, selected\_item, customer\_contact, customer\_address)
- Payment(payment\_id, paymet\_amount, Payment\_date, customer\_id, payment\_mode)
- Customer Feedback(customer\_id, customer\_name, selected\_item, feedback\_description)

## RELATIONAL SCHEMA: -



# **Section 5**

## **Normalization and Schema Refinement**

## **Normalization & Schema Refinement :**

### **1. Company(company\_id, company\_name, contacts, address)**

**PK dependency :** company\_id  $\rightarrow$  company\_name, contacts, address

**FD :** company\_id  $\rightarrow$  company\_name  
company\_id  $\rightarrow$  contacts  
company\_id  $\rightarrow$  address

**Partial Key Dependency :** None

**Transitive Dependency :** None

**Redundancies :** if a company has two different contacts then there will be redundancy in the table.

**Anomalies:** -

**Insert:** - we cannot insert company\_name, contacts, address without company\_id.

**Deletion:** - There won't be any anomaly in deletion.

**Update:** - There won't be any anomaly in update

**1NF:**- The table is not in 1NF because there can be multiple contact numbers. To remove this company\_id, company\_name, address will be repeated.

**2NF:** - there is no partial dependency so the table is already in 2NF.

**3NF:** - there is no transitive dependency so the table is already in 3NF.

### **2. User(user\_id, user\_name, user\_password)**

**PK dependency :** user\_id  $\rightarrow$  user\_password, user\_name

**FD :** user\_id  $\rightarrow$  user\_password  
user\_id  $\rightarrow$  user\_name

**Partial Key Dependency :** None

**Transitive Dependency :** None

**Redundancies :** None.

**Anomalies:** -

There won't be any anomaly in insert, update and deletion.

**1NF:** - There are no multi-valued attributes here, so it is already in 1NF

**2NF:** - there is no partial dependency so the table is already in 2NF.

**3NF:** - there is no transitive dependency so the table is already in 3NF.

**3. Employee(employee\_id, employee\_name, employee\_address, date\_of\_birth, employee\_contact, employee\_email)**

**PK dependency :**

{employee\_id}  $\rightarrow$  employee\_name, employee\_address, date\_of\_birth, employee\_contact, employee\_email}

**FD :**  $\text{employee\_id} \rightarrow \text{employee\_name}$   
 $\text{employee\_id} \rightarrow \text{employee\_address}$   
 $\text{employee\_id} \rightarrow \text{date\_of\_birth}$   
 $\text{employee\_id} \rightarrow \text{employee\_contact}$   
 $\text{employee\_id} \rightarrow \text{employee\_email}$

**Partial Key Dependency :** here partial dependency occurs because if an employee has two contacts then there will be the same employee\_email with different contacts.

**Transitive Dependency :** None

**Anomalies:**

**Insert:** - there is no anomaly in insert.

**Update:** - if we update email id then it will occur more than once because the same employee has two different contacts.

**Delete:** - there is no anomaly in delete.

**1NF:** - if an employee has two different addresses like current and permanent then there will be 2 values in the same attribute. So we need change primary key (employee\_id employee\_contact)

**Employee(employee\_id employee\_contact, employee\_name, employee\_cur\_address, employee\_per\_address, date\_of\_birth, employee\_email)**

**2NF:** -

There is partial dependency in the table so we need to divide this table.

Employee(employee\_id, employee\_contact)

Employeeinfo(employee\_id, employee\_name, employee\_cur\_address, employee\_per\_address, date\_of\_birth, employee\_email)

Here employee\_id in employeeinfo table is foreign key.

**3NF:** - there is no transitive dependency in the table so it is already in 3NF.

So now it is on BCNF.

#### 4. Project(project\_id, project\_title, project\_startTime, project\_workstatus, project\_endTime)

**PK dependency :** project\_id  $\rightarrow$  project\_title, project\_startTime, project\_workstatus, project\_endTime

**FD :** project\_id  $\rightarrow$  project\_title

project\_id  $\rightarrow$  project\_startTime

project\_id  $\rightarrow$  project\_workstatus

project\_id  $\rightarrow$  project\_endTime

Project\_status  $\rightarrow$  project\_endTime

**Partial Key Dependency :** None

**Transitive Dependency :** Project\_status can find project end time which is transitive dependency.

**Redundancies :** Some projects are starting or ending at the same time so information will repeat.

**Anomalies:** -

**Insert:** - we can not insert project\_endTime without inserting project\_startTime.

**Update:** - There is no update anomaly.

**Delete:** - There is no delete anomaly.

**1NF:-** There are no multi-valued attributes here, so it is already in 1NF

**2NF:** - there is no partial dependency so the table is already in 2NF.

**3NF:** - there is transitive dependency in the table so we need to divide this table.

Project(project\_id, project\_title, project\_startTime)  
Projectinfo(Project\_id, project\_workstatus, project\_endTime)

Now it is already in BCNF.

## 5. Team(team\_id, team\_name, no\_of\_employee, project\_id)

**PK dependency :** team\_id  $\rightarrow$  team\_name, no\_of\_employee, project\_id

Here project\_id is foreign key

**FD :** team\_id  $\rightarrow$  team\_name  
team\_id  $\rightarrow$  no\_of\_employee  
team\_id  $\rightarrow$  project\_id

**Partial Key Dependency :** None

**Transitive Dependency :** None

**Redundancies :** None

**Anomalies:** -

**Insert:** - there is no anomaly in insert.

**Update:** - there is no anomaly in the update.

**Delete:** - if we delete project\_id then all project details will lost

**1NF:-** There are no multi-valued attributes here, so it is already in 1NF

**2NF:** - there is no partial dependency so the table is already in 2NF.

**3NF:** - there is no transitive dependency so the table is already in 3NF.

Hence table is in BCNF

## 6. Task(task\_id, task\_name, employee\_id, task\_status, task\_startTime, task\_endTime)

**PK dependency :** task\_id  $\rightarrow$  task\_name, employee\_id, task\_status, task\_startTime, task\_endTime

**FD :** task\_id  $\rightarrow$  task\_name  
task\_id  $\rightarrow$  task\_employeeID  
task\_id  $\rightarrow$  task\_projectTitle

task\_id -> task\_status  
task\_id -> task\_startTime  
task\_id -> task\_endTime  
task\_status -> task\_endTime

**Partial Key Dependency :** There won't be any partial dependencies.

**Transitive Dependency :** There is transitive dependency that we can find on task\_endTime by current task\_status.

**Redundancies :** There can be multiple employees working on the same task. It can create redundancy.

**Anomalies:-**

**Insert:-** If the next task is supposed to be done on a particular day, this can't be inserted in tables without task\_id.

**Delete:-** There is no delete anomaly.

**Update:-** There is no update anomaly

**1NF:-** There can be multiple employees working on the same task. To Convert in 1NF task\_id, task\_name, task\_status, task\_startTime, task\_endTime will be repeated.

**2NF:-** Table already in 2NF.

**3NF:-** There is transitive dependency from task\_status to task\_endTime, so we have to create another table containing task\_id, task\_status, task\_endTime like this,  
task\_completion(task\_id, task\_status, task\_endTime)  
Task(task\_id, task\_name, employee\_id, task\_startTime)

## 7. Product(Product\_id, Product\_name, Product\_type, Product\_price, Product\_description)

**PK dependency :** Product\_id -> Product\_name, Product\_type, Product\_price, Product\_description

**FD :** Product\_id -> Product\_name  
Product\_id -> Product\_type  
Product\_id -> Product\_price  
Product\_id -> Product\_description  
Product\_name -> product\_description  
Product\_name -> product\_price

**Partial Key Dependency :** There is no partial dependency.

**Transitive Dependency :** There is transitive dependency. Because from product\_name we can get product\_price and product description.

**Redundancies :** Product name, price and description can be repeated for same product.

### **Anomalies:-**

**Insert:** We can't add product description and product type without id, even if we know product description before production.

**Delete:** There is no delete anomaly.

**Update:** To update a price or description we need to change all price and description.

**1NF:** Table is already in 1NF.

**2NF:** It is also in 2NF.

**3NF:** There is transitive dependency so we need to remove that to convert into 3NF.

Product\_name  $\rightarrow$  product\_description, product\_price

Product\_id  $\rightarrow$  Product\_name, Product\_type

### **8. Customer(customer\_id, customer\_name, selected\_item, customer\_contact, customer\_address)**

**PK dependency :** customer\_id  $\rightarrow$  customer\_name, selected\_item, customer\_contact, customer\_address  
(Customer\_name, customer\_contact)  $\rightarrow$  customer\_id, selected\_item, customer\_address.

**FD :** customer\_id  $\rightarrow$  customer\_name  
customer\_id  $\rightarrow$  selected\_item  
customer\_id  $\rightarrow$  customer\_contact  
customer\_id  $\rightarrow$  customer\_address  
(Customer\_name, customer\_contact)  $\rightarrow$   
Customer\_id  
(Customer\_name, customer\_contact)  $\rightarrow$   
selected\_item  
(Customer\_name, customer\_contact)  $\rightarrow$   
customer\_address

**Partial Key Dependency :** None

**Transitive Dependency :** There is no transitive dependency.

**Redundancies :** If customers purchase more than one item then other information will repeat.

### **Anomalies :-**

**Insert:-** can't insert product item without customer's id or name, contact.

**Delete:-** There is no deletion anomaly.

**Update:-** If a customer's contact or address will change then it will update on all purchased items.

**1NF:-** Table is not in 1NF as there can be more than one value in purchased item and contact number, to convert in 1NF we have to repeat customer\_id, customer\_name, customer\_address

**2NF:-** There is no partial dependency, so the table is in 2NF.

**3NF:** There is no transitive dependency so it is in 3NF.

## **9. Payment(payment\_id, payment\_amount, Payment\_date, customer\_id, payment\_mode)**

**PK dependency :** payment\_id  $\rightarrow$  payment\_amount, Payment\_date, customer\_id, payment\_mode

**FD :** payment\_id  $\rightarrow$  payment\_amount  
payment\_id  $\rightarrow$  payment\_date  
payment\_id  $\rightarrow$  customer\_id  
payment\_id  $\rightarrow$  payment\_mode  
(customer\_id, payment\_mode)  $\rightarrow$  payment\_id  
    (customer\_id, payment\_mode)  $\rightarrow$   
payment\_date  
    (customer\_id, payment\_mode)  $\rightarrow$   
Payment\_amount  
    customer\_id  $\rightarrow$  payment\_id  
    customer\_id  $\rightarrow$  payment\_amount  
    customer\_id  $\rightarrow$  Payment\_date

**Partial Key Dependency :** There is partial dependency as customer\_id can find payment\_id, payment\_amount and payment\_date. To remove this we can make new tables.

**Transitive Dependency :** There is no transitive dependency.

**Redundancies :** There is no redundancy.

**Anomalies:-**

**Inset:**- No insertion anomalies.

**Delete:**- No delete anomalies.

**Update:**- No updation anomalies.

**1NF:**- Table is already in 1NF as there is no multivalued attributes.

**2NF:**- It is not in 2NF, there is partial dependency so we make new table as follows,

Payment\_id  $\rightarrow$  customer\_id

Payment\_id  $\rightarrow$  payment\_amount, payment\_date, payment\_mode.

**3NF:**- Table is in 3rd normal form already as there is no transitive dependency.

## **10. Customer Feedback(customer\_id, product\_id, selected\_item, feedback\_description)**

**PK dependency :** customer\_id  $\rightarrow$  customer\_name, selected\_item, feedback\_description

**FD :** customer\_id  $\rightarrow$  customer\_name

customer\_id  $\rightarrow$  selected\_item  
customer\_id  $\rightarrow$  feedback\_description  
Item\_id  $\rightarrow$  selected\_item

**Partial Key Dependency :** Here selected item is non prime attribute and it is functionally dependent on customer\_id.

**Transitive Dependency :** There is no dependency.

**Redundancies :** Lot's of customers find the same item so selected\_item and product\_id are the same for many customers.

#### **Anomalies:-**

**Insert**:- can't insert customer\_name, selected\_item, Item\_id, feedback\_description without customer\_id.

**Delete**:- No delete anomalies.

**Update**:- No update anomalies.

**1NF**:- It is already in 1NF.

**2NF**:- For this we divide this table into two table,

Customer\_id  $\rightarrow$  product\_id

Customer\_id  $\rightarrow$  selected\_item, feedback\_description

**3NF**:- already in 3NF. Yet it is in BCNF.

# **Section 6**

# **SQL**

## Final schema

1. **company\_id** -> { company\_name, contacts, address }  
**Key:-** company\_id
2. **user\_id** -> {user\_password, user\_name }  
**Key:-** user\_id
3. **Employee\_id, employee\_contact** -> { employee\_id, employee\_contact}  
**Key:-** employee\_id, employee\_contact
4. **employee\_id** -> { employee\_name, employee\_cur\_address, employee\_per\_address, date\_of\_birth, employee\_email}  
**Key:-** employee\_id
5. **Project\_id** -> {project\_title, project\_startTime}  
Project\_id -> project\_workstatus,  
project\_endTime  
  
**Key :-** project\_id
6. **Team\_id, project\_id** -> { team\_name, no\_of\_employee, project\_id}  
**Key :-** team\_id
7. **task\_id** -> { task\_name, employee\_id, task\_startTime }  
**task\_id** -> {task\_status, task\_endTime,}  
  
**Key :-** task\_id
8. **Product\_id** -> { Product\_name, Product\_type}  
**Product\_name** -> product\_description, product\_price  
**Key :-** Product\_id
9. **customer\_id** -> { customer\_name, selected\_item, customer\_contact, customer\_address }  
**Key :-** customer\_id
10. **Payment\_id** -> payment\_mode  
**customer\_id** -> payment\_id, payment\_amount, payment\_date.  
**Key :-** payment\_id, customer\_id
11. **Customer\_id** -> product\_id  
**Customer\_id** -> selected\_item, feedback\_description  
  
**Key :-** customer\_id, product\_id

## **DDL Script: -**

### **1. Company: -**

```
CREATE TABLE IF NOT EXISTS prct."Company"
(
    company_id bigint NOT NULL,
    company_name character varying(255),
    contacts bigint,
    address character varying(255),
    PRIMARY KEY (company_id)
);
```

### **2. User: -**

```
CREATE TABLE IF NOT EXISTS prct."User"
(
    user_id bigint NOT NULL,
    user_name character varying(255),
    user_password character varying(255),
    PRIMARY KEY (user_id)
);
```

### **3. Employee: -**

```
CREATE TABLE IF NOT EXISTS prct."Employee"
(
    employee_id bigint NOT NULL,
    employee_contact bigint NOT NULL,
    PRIMARY KEY (employee_id, employee_contact),
    CONSTRAINT employee_id FOREIGN KEY (employee_id)
        REFERENCES prct."Employee info" (employee_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

### **4. Employeeinfo: -**

```
CREATE TABLE IF NOT EXISTS "S10"."Employee info"
(
    employee_id bigint NOT NULL,
    employee_name character varying(255),
    employee_cur_address character varying(255),
    employee_per_address character varying(255),
    date_of_birth date,
    employee_email character varying(255),
    PRIMARY KEY (employee_id)
);
```

**5. Projectinfo: -**

```
CREATE TABLE IF NOT EXISTS prct."Projectinfo"
(
    project_id bigint NOT NULL,
    project_workstatus date,
    "project_endTime" date,
    PRIMARY KEY (project_id)
);
```

**6. Project: -**

```
CREATE TABLE IF NOT EXISTS prct."Project"
(
    project_id bigint NOT NULL,
    project_title character varying(255),
    "project_startTime" date,
    PRIMARY KEY (project_id),
    CONSTRAINT project_id FOREIGN KEY (project_id)
        REFERENCES prct."Projectinfo" (project_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

**7. Team: -**

```
CREATE TABLE IF NOT EXISTS prct."Team"
(
    team_id bigint NOT NULL,
    team_name character varying(255),
    no_of_employee bigint,
    project_id bigint NOT NULL,
    PRIMARY KEY (team_id, project_id),
    CONSTRAINT project_id FOREIGN KEY (project_id)
        REFERENCES prct."Projectinfo" (project_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

**8. Task: -**

```
CREATE TABLE IF NOT EXISTS prct."Task"
(
    task_id bigint NOT NULL,
    task_name character varying(255),
    employee_id bigint,
    "task_startTime" date,
    PRIMARY KEY (task_id)
);
```

**9. Task Completion: -**

```
CREATE TABLE IF NOT EXISTS prct."Task_completion"
(
    task_id bigint NOT NULL,
    task_status date,
    "task_endTime" date,
    PRIMARY KEY (task_id),
    CONSTRAINT task_id FOREIGN KEY (task_id)
        REFERENCES prct."Task" (task_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

**10. Product: -**

```
CREATE TABLE IF NOT EXISTS prct."Product"
(
    product_id bigint NOT NULL,
    "Product_name" character varying(255),
    "Product_type" character varying(255),
    PRIMARY KEY (product_id)
);
```

**11. Productinfo: -**

```
CREATE TABLE IF NOT EXISTS prct."Product_info"
(
    product_id bigint NOT NULL,
    product_price bigint,
    product_desc character varying(255),
    PRIMARY KEY (product_id),
    CONSTRAINT product_id FOREIGN KEY (product_id)
        REFERENCES prct."Product" (product_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

**12. Customer: -**

```
CREATE TABLE IF NOT EXISTS prct."Customer"
(
    customer_id bigint NOT NULL,
    customer_name character varying(255),
```

```
selected_item character varying(255),
customer_contact bigint,
customer_address character varying(255),
PRIMARY KEY (customer_id)
);
```

**13. Payment: -**

```
CREATE TABLE IF NOT EXISTS prct."Payment"
(
    "Payment_id" bigint NOT NULL,
    payment_amount bigint,
    payment_date date,
    payment_mode character varying(255),
    PRIMARY KEY ("Payment_id")
);
```

**14. Paymentinfo: -**

```
CREATE TABLE IF NOT EXISTS prct."Paymentinfo"
(
    payment_id bigint NOT NULL,
    customer_id bigint NOT NULL,
    PRIMARY KEY (payment_id, customer_id),
    CONSTRAINT payment_id FOREIGN KEY (payment_id)
        REFERENCES prct."Payment" ("Payment_id") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT customer_id FOREIGN KEY (customer_id)
        REFERENCES prct."Customer" (customer_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

**15. Customer Feedback: -**

```
CREATE TABLE IF NOT EXISTS prct."Customer Feedback"
(
    customer_id bigint NOT NULL,
    selected_item character varying(255),
    feedback_description character varying(255),
    PRIMARY KEY (customer_id)
);
```

## 16. Customer productinfo: -

```
CREATE TABLE IF NOT EXISTS prct."Customer Productinfo"
(
    customer_id bigint NOT NULL,
    product_id bigint NOT NULL,
    PRIMARY KEY (customer_id, product_id),
    CONSTRAINT customer_id FOREIGN KEY (customer_id)
        REFERENCES prct."Customer" (customer_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT product_id FOREIGN KEY (product_id)
        REFERENCES prct."Product" (product_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

Customer: -

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' tab:

```
1 SELECT * FROM prct."Customer"
2 ORDER BY customer_id ASC
```

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

	customer_id	customer_name	selected_item	customer_contact	customer_address
95	95	Randall	Pannier	9280057327	Wilczyce
96	96	Tate	Hatty	9961886759	San Nicolas
97	97	Wren	Y-find	9823834413	Mezinovskiy
98	98	Doroteja	Bitwolf	9320885493	Kokkinóchoma
99	99	Gillian	Zamit	9488330910	Newmarket
100	100	Lauren	Alphazap	9495630154	Rzeczenica

Employee: -

The screenshot shows the pgAdmin 4 interface with the 'Employee' table selected in the left sidebar. The main area displays a query in the Query Editor:

```
1 SELECT * FROM prct."Employee"
2 ORDER BY employee_id ASC, employee_contact ASC
```

The Data Output tab shows the results of the query:

employee_id	employee_contact
95	9553960410
96	9600532038
97	9899591710
98	9999923600
99	9490461946
100	9549060135

Employeeinfo: -

The screenshot shows the pgAdmin 4 interface with the 'Employeeinfo' table selected in the left sidebar. The main area displays a query in the Query Editor:

```
1 SELECT * FROM prct."Employeeinfo"
2 ORDER BY employee_id ASC
```

The Data Output tab shows the results of the query:

employee_id	employee_name	employee_cur_address	employee_per_address	date_of_birth	employee_email
95	Yolande Tredgold	71128 Blaine Trail	same as cur	1986-07-08	ytreddgold2m@xinhuanet.com
96	Britt Strahan	9 Myrtle Terrace	same as cur	1984-02-09	bstrahan2n@ask.com
97	Celestina Breukelman	716 Anhalt Lane	716 Anhalt Lane	1986-06-07	cbreukelman2o@facebook.com
98	Ruby Elderton	685 Lunder Parkway	685 Lunder Parkway	1982-12-11	relderton2p@example.com
99	Niki Mustoo	1 Cordelia Circle	1 Cordelia Circle	1986-07-08	nmustoo2q@usa.gov
100	Barret Filov	21623 Moose Alley	21623 Moose Alley	1986-03-07	bfilov2r@123-reg.co.uk

## Payment: -

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query is:

```
1 SELECT * FROM prct."Payment"
2 ORDER BY "Payment_id" ASC
```

The results table has four columns: Payment\_id, payment\_amount, payment\_date, and payment\_mode. The data is as follows:

Payment_id	payment_amount	payment_date	payment_mode
95	4178	1986-07-08	Björn
96	13714	1984-02-09	Angélique
97	4879	1986-06-07	Rui
98	3091	1982-12-11	Gérald
99	14055	1986-07-08	Bérénice
100	6510	1986-03-07	Léane

## Paymentinfo: -

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query is:

```
1 SELECT * FROM prct."Paymentinfo"
2 ORDER BY payment_id ASC, customer_id ASC
```

The results table has two columns: payment\_id and customer\_id. The data is as follows:

payment_id	customer_id
95	95
96	96
97	97
98	98
99	99
100	100

## Product: -

The screenshot shows the pgAdmin 4 interface with the 'Product' table selected in the left sidebar under 'Tables (16)'. The 'Query Editor' tab is active, displaying the SQL query: `SELECT * FROM prct."Product"`. The 'Data Output' tab is selected, showing the following table data:

product_id	Product_name	Product_type
95	Cardify	Lasiorhinus latifrons
96	Konklux	Phalaropus fulicarius
97	Mat Lam Tam	Dolichitus patagonum
98	Otcom	Echimys chrysurus
99	Aerified	Damaliscus dorcas
100	Subin	Nycticorax nycticorax

## Product\_info: -

The screenshot shows the pgAdmin 4 interface with the 'Product\_info' table selected in the left sidebar under 'Tables (16)'. The 'Query Editor' tab is active, displaying the SQL query: `SELECT * FROM prct."Product_info"`. The 'Data Output' tab is selected, showing the following table data:

product_id	product_price	product_desc
95	14883	Huels, Sawayn and Thompson
96	3026	Gutmann-Pfeffer
97	14853	Schowalter, Green and Schuster
98	12974	Rohan-Effertz
99	11736	Renner LLC
100	7222	Yost and Sons

## Project: -

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query is:

```
1 SELECT * FROM prct."Project"
2 ORDER BY project_id ASC
```

The results are displayed in a Data Output table:

project_id	project_title	project_startTime
94	94 Flowdesk	2021-07-19
95	95 Home Ing	2021-05-08
96	96 Voyatouch	2021-03-05
97	97 Bidex	2021-08-14
98	98 Bidex	2021-01-31
99	99 Quo Lux	2021-07-15
100	100 Opela	2021-09-22

A green message bar at the bottom right indicates: **Successfully run. Total query runtime: 159 msec. 100 rows affected.**

## Projectinfo: -

The screenshot shows the pgAdmin 4 interface with the 'Query Editor' tab active. The query is:

```
1 SELECT * FROM prct."Projectinfo"
2 ORDER BY project_id ASC
```

The results are displayed in a Data Output table:

project_id	project_workstatus	project_endTime
95	95 2021-05-08	2021-06-26
96	96 2021-03-05	2021-06-03
97	97 2021-08-14	2021-09-22
98	98 2021-01-31	2021-03-04
99	99 2021-07-15	2021-09-13
100	100 2021-09-22	2021-05-02

A green message bar at the bottom right indicates: **Successfully run. Total query runtime: 218 msec. 100 rows affected.**

Task: -

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects: FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, and Tables (16). The 'Tables (16)' section is expanded, and the 'Task' table is selected, highlighted with a blue background. The main area is the 'Query Editor' tab, which contains the following SQL code:

```
1 SELECT * FROM prct."Task"
2 ORDER BY task_id ASC
```

Below the query editor is the 'Data Output' tab, which displays the results of the query. The table has four columns: task\_id, task\_name, task\_employeeID, and taskStartTime. The data is as follows:

task_id	task_name	task_employeeID	taskStartTime
95	Zaarn-Dox	95	2021-01-11
96	Konklab	96	2021-01-11
97	Biodelx	97	2021-05-11
98	Namfix	98	2021-03-11
99	Redhold	99	2021-04-11
100	Zoolab	100	2021-07-11

A green success message at the bottom right of the data output pane states: "Successfully run. Total query runtime: 191 msec. 100 rows affected."

Task\_completion: -

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects: FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, and Tables (16). The 'Tables (16)' section is expanded, and the 'Task\_completion' table is selected, highlighted with a blue background. The main area is the 'Query Editor' tab, which contains the following SQL code:

```
1 SELECT * FROM prct."Task_completion"
2 ORDER BY task_id ASC
```

Below the query editor is the 'Data Output' tab, which displays the results of the query. The table has three columns: task\_id, task\_status, and task\_endTime. The data is as follows:

task_id	task_status	task_endTime
95	2021-01-11	2021-10-11
96	2021-01-11	2021-04-11
97	2021-05-11	2021-07-08
98	2021-03-11	2021-10-11
99	2021-04-11	2021-09-11
100	2021-07-11	2021-02-11

A green success message at the bottom right of the data output pane states: "Successfully run. Total query runtime: 288 msec. 100 rows affected."

Team: -

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'Tables (16)', with 'Team' selected. The main area has tabs for 'Query Editor' and 'Query History'. The 'Query Editor' tab contains the following SQL code:

```
1 SELECT * FROM prct."Team"
2 ORDER BY team_id ASC, project_id ASC
```

The 'Data Output' tab displays the results of the query in a table:

team_id	team_name	no_of_employee	project_id
95	Bitwolf	35	95
96	Lotstring	47	96
97	Voyatouch	31	97
98	Zathin	41	98
99	Transcof	37	99
100	Viva	45	100

A green message bar at the bottom right indicates: 'Successfully run. Total query runtime: 287 msec. 100 rows affected.'

User: -

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'Tables (16)', with 'User' selected. The main area has tabs for 'Query Editor' and 'Query History'. The 'Query Editor' tab contains the following SQL code:

```
1 SELECT * FROM prct."User"
2 ORDER BY user_id ASC
```

The 'Data Output' tab displays the results of the query in a table:

user_id	user_name	user_password
95	PANDAR MAYUR ARVINDBHAI	TYIHFHDq
96	ROHIT MAHIMABEN SHANTILAL	tBjffG4HBl
97	RATHOD PARTH BHUPENDRBHAI	6k2GEUJ
98	PRAJAPATI JINALBEN KAMLESHBHAI	puFjnT46ly
99	GOHIL DARSHAN HARSHADBHAI	dmlFpu
100	MAPARA PRASANTA SANDIP	OkZQyQlbv1n

## QUERY: -

- 1) select payment\_amount from prct."Payment" where "Payment\_id"=3

```

1   nt_amount from prct."Payment" where "Payment_id"=3
2   ent_id" from prct."Payment" order by payment_amount
3   ner_name as "Name", customer_address as "Address" from prct."Customer" where "customer_id"><4
4   &CT employee_per_address from prct."Employeeinfo"
5   n prct."Payment" where "Payment_id" in (select "payment_id" from prct."Paymentinfo" where customer_id<11)

```

Payment_id	payment_amount	payment_date	payment_mode
1	10153	1987-06-08	Cleopatre
2	2		
3	3	1984-06-01	Maélanne
4	4	1988-01-04	Åsa
5	5	1989-04-05	Dafnë
6	6	1988-08-09	Nadège
7	7	1986-11-07	Maëlla
8	8	1987-07-03	Söng

- 2) select "Payment\_id" from prct."Payment" order by payment\_amount

```

1   select payment_amount from prct."Payment" where "Payment_id"=3
2   select "Payment_id" from prct."Payment" order by payment_amount
3   select customer_name as "Name", customer_address as "Address" from prct."Customer" where "customer_id"><4
4   select DISTINCT employee_per_address from prct."Employeeinfo"
5   select * from prct."Payment" where "Payment_id" in (select "payment_id" from prct."Paymentinfo" where cu

```

Payment_id	
1	44
2	13
3	58
4	38
5	10
6	5
7	63
8	?

Successfully run. Total query runtime: 142 msec. 100 rows affected.

- 3) select customer\_name as "Name", customer\_address as "Address" from prct."Customer" where "customer\_id"<4

<4; select DISTINCT employee\_per\_address from prct.Employeeinfo; select \* from prct.Payment where "Payment\_id" in (select "payment\_id" from prct.Paymentinfo where cu. The data output pane shows results for the third query: Name: Aube, Address: San Miguel; Name: Taber, Address: Metz; Name: Jacqueline, Address: Ribamar."/>

```

1 select payment_amount from prct.Payment where "Payment_id"=3
2 select "Payment_id" from prct.Payment order by payment_amount
3 select customer_name as "Name", customer_address as "Address" from prct.Customer where "customer_id"><4
4 select DISTINCT employee_per_address from prct.Employeeinfo
5 select * from prct.Payment where "Payment_id" in (select "payment_id" from prct.Paymentinfo where cu

```

Name	Address
Aube	San Miguel
Taber	Metz
Jacqueline	Ribamar

- 4) select DISTINCT employee\_per\_address from prct."Employeeinfo"

```

1 select payment_amount from prct.Payment where "Payment_id"=3
2 select "Payment_id" from prct.Payment order by payment_amount
3 select customer_name as "Name", customer_address as "Address" from prct.Customer where "customer_id"><4
4 select DISTINCT employee_per_address from prct.Employeeinfo
5 select * from prct.Payment where "Payment_id" in (select "payment_id" from prct.Paymentinfo where cu

```

employee_per_address
908 Gerald Road
21823 Moose Alley
157 Bellfuss Street
862 Oak Street
1 Cordelia Circle
29 Forster Lane
685 Lunder Parkway
616 Kenneth Terrace

✓ Successfully run. Total query runtime: 97 msec. 38 rows affected.

5) select "project\_startTime" from "S10"."Project" where "project\_id"=1;

The screenshot shows the PgAdmin 4 interface. The left sidebar displays a tree view of database objects under the 'S10' schema, including tables like Customer, Employee, and ProjectInfo. The 'ProjectInfo' table is currently selected. The main window contains a query editor with the following SQL code:

```
1
2 select "project_startTime" from "S10"."Project" where "project_id"=1;
```

The results pane shows a single row of data:

project_startTime
2021-08-12

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 139 msec. 1 rows affected."

6) select "project\_title" from "S10"."Project"

The screenshot shows the PgAdmin 4 interface. The left sidebar displays a tree view of database objects under the 'S10' schema, including tables like Customer, Employee, and ProjectInfo. The 'ProjectInfo' table is currently selected. The main window contains a query editor with the following SQL code:

```
1 select "project_title" from "S10"."Project"
```

The results pane shows multiple rows of data:

project_title
Candguard
Zontrax
Span
Aerified
Lotetting
Konklux
Rank
Bamity
Holdamis

7) select project\_workstatus from "S10"."Projectinfo" where "project\_id" = 3

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with the 'Projectinfo' table selected. The 'Query Editor' tab is active, containing the following SQL code:

```
1 select project_workstatus from "S10"."Projectinfo" where "project_id" = 3
```

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

project_workstatus	date
1	2021-07-03

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 99 msec. 1 rows affected."

8) select \* from prct."Payment" where "Payment\_id" in (select "payment\_id" from prct."Paymentinfo" where customer\_id<11)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with the 'Paymentinfo' table selected. The 'Query Editor' tab is active, containing the following SQL code:

```
1 select payment_amount from prct."Payment" where "Payment_id"=3
2 select "Payment_id" from prct."Payment" order by payment_amount
3 select customer_name as "Name", customer_address as "Address" from prct."Customer" where "customer_id" < 11
4 select DISTINCT employee_per_address from prct."Employeeinfo"
5 select * from prct."Payment" where "Payment_id" in (select "payment_id" from prct."Paymentinfo" where customer_id < 11)
```

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

Payment_id	payment_amount	payment_date	payment_mode
1	1	13646	1981-06-10
2	2	10153	1987-06-08
3	3	7572	1984-06-01
4	4	14282	1988-01-04
5	5	2488	1989-04-05
6	6	3013	1988-08-09
7	7	2492	1986-11-07
8	8	10649	1987-07-09

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 167 msec. 10 rows affected."

9) select product\_id from “S10”. “product” where Product\_type=‘Input’ order by product\_id desc

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects, including Tables (16), Functions, Procedures, Sequences, and so on. The 'Product' table under 'Tables (16)' is selected. The main area shows a query editor with the following SQL code:

```
1 select product_id from s10."Product"
2 where "Product_type"='Input'
3 order by product_id desc
```

Below the query editor is a data grid titled 'Data Output' showing the results of the query. The columns are 'product\_id' and 'product\_type'. The data is as follows:

product_id	product_type
19	Input
18	Input
17	Input
16	Input
15	Input
14	Input
13	Input
12	Input
11	Input
10	Input
9	Input
8	Input
7	Input
6	Input
5	Input
4	Input
3	Input
2	Input
1	Input

10) Select all details from Product\_info table where product\_price is between rupees 2000 to 5000.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects, including Tables (16), Functions, Procedures, Sequences, and so on. The 'Product\_info' table under 'Tables (16)' is selected. The main area shows a query editor with the following SQL code:

```
1 select * from s10."Product_info"
2 where "product_price" between 2000 and 5000
3
```

Below the query editor is a data grid titled 'Data Output' showing the results of the query. The columns are 'product\_id', 'product\_price', and 'product\_desc'. The data is as follows:

product_id	product_price	product_desc
6	4464	Input
11	4613	Input
12	4652	Input
17	3996	Input
19	2000	Input
20	4862	Output
21	4445	Output
22	4982	Output
25	2756	Output
20	4343	Input-Output
31	3364	Input-Output
32	2971	Input-Output
36	3628	Port

11) select task\_id where task\_status is 20 and task\_endTime is 12-7-2021.

The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar says "final/postgres@PostgreSQL 13". The query editor contains the following SQL code:

```
1 select task_id from s10."Task_completion"
2 where "task_status"=20 and "task_endTime"='12-07-2021'
```

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

task_id
21

12) Select all details from Employee info table where date of birth of employee is greater than or equal to 1-1-1988

The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar says "final/postgres@PostgreSQL 13". The query editor contains the following SQL code:

```
1 select * from s16."Employee_info"
2 where "date_of_birth" >= '01-01-1988'
```

Below the code, there are tabs for "Data Output", "Explain", "Messages", and "Notifications". The "Data Output" tab is selected, showing a table with 14 rows of employee information. The columns are: employee\_id, employee\_name, employee\_cur\_address, employee\_per\_address, date\_of\_birth, and employee\_email. A green message at the bottom right of the table area says "Successfully run. Total query runtime: 151 msec, 14 rows affected."

employee_id	employee_name	employee_cur_address	employee_per_address	date_of_birth	employee_email
1	4 PRYAL SURESHBHAI RAJ	76 Cascade Alley	same as cur	1988-01-04	membellon3@ask.com
2	5 CHAUDHAN SIDDHARTH DINESHBHAI	15 Derek Street	same as cur	1989-04-05	dcesford4@eclinedaily.com
3	6 SONANI RIDHAM RAMESHBHAI	862 Oak Street	49992 Surrey Place	1988-08-09	lbennholt5@360.cn
4	17 BHALALA JAY BHARATBHAI	91871 Reilne Plaza	same as cur	1988-12-04	nburleyg@tiny.cc
5	32 MOVALYA PARAS BHARATEBHAI	73587 Graceland Hill	same as cur	1989-06-11	flivingew@independent.co.uk
6	34 PARIGH VEDANT ATUL	46 Golf Course Street	same as cur	1988-03-05	pfaulixz@tco
7	41 PATEL HARSH MANISHKUMAR	95911 Bartlett Point	same as cur	1989-10-01	swilkenson14@github.com
8	43 THAKKAR AMIT HASMURIBHAI	49992 Surrey Place	same as cur	1989-06-11	nlehler16@pcworld.com
9	45 PANCHAL MEET HIRENKUMAR	157 Bellfus Street	49992 Surrey Place	1988-03-05	holdcock18@unilog.fr
10	53 BHALANI AYUSH VIRJIBHAI	9 Superior Trill	same as cur	1988-03-05	rsperesbrick1g@princeton.edu
11	59 MORADIYA JAIMINBHAI SHAILESHBHAI	93 Reilne Pass	685 Lunder Parkway	1988-03-05	dasberry1m@cbsnews.com
12	79 VORA DEV KASHYAP	7794 Onsgard Court	685 Lunder Parkway	1989-07-01	jbullard26@wiley.com
13	82 HIRPARA UTSAV PARESHBHAI	16684 Burrowe Center	same as cur	1990-02-07	chowfley25@greaterstar.com
14	83 MOKARIYA JAYDEPMUAR DEVRAMBHAI	554 Dawn Parkway	same as cur	1989-01-05	gkupke2@gmail.com

- 13) Select customer id, customer name and customer address from Customer table where selected item is 'Head Phone'

The screenshot shows a PostgreSQL query editor interface. The title bar says "final/postgres@PostgreSQL 13". The query editor tab is selected. The query text is:

```
1 select customer_id, customer_name, customer_address from s1e."Customer"
2 where "selected_item"='Head Phone'
```

Below the query, there are tabs for "Data Output", "Explain", "Messages", and "Notifications". The "Data Output" tab is selected, displaying a table with three columns: customer\_id, customer\_name, and customer\_address. The data is as follows:

	customer_id	customer_name	customer_address
1	12	Medison	Garibaldi
2	30	Sarah	Sabenagrende
3	51	Grace	Hinunangen
4	62	Steve	Zungeru
5	72	Alex	San Fernando

- 14) Select Product\_name except all Product where "Product\_type"='Input-Output'

The screenshot shows a PostgreSQL query editor interface. The title bar says "final/postgres@PostgreSQL 13". The query editor tab is selected. The query text is:

```
1 select "Product_name" from s1e."Product" except all select "Product_name" from s1e."Product" where "Product_type"='Input-Output'
```

Below the query, there are tabs for "Data Output", "Explain", "Messages", and "Notifications". The "Data Output" tab is selected, displaying a table with one column: Product\_name. The data is as follows:

Product_name
2756
1710
4415
4813
5253
14291
14746
5286
12906
13362
13749
14364
4652
11588
1504
1872
3628

In the bottom right corner of the data output area, there is a green message box with a checkmark that says "Successfully run. Total query runtime: 133 msec. 40 rows affected."

- 15) Select all details from the task table of the employees who are born before 23-10-1983.

final/postgres@PostgreSQL 13 ▼

Query Editor    Query History

```
1 SELECT * FROM s10."Task"
2 WHERE employee_id IN (
3   SELECT employee_id
4   FROM s10."Employee_info"
5   WHERE date_of_birth < '23-10-1983'
6 )
7
```

Data Output   Explain   Messages   Notifications

	task_id [PK] bigint	task_name character varying (255)	employee_id bigint	task_startTime date	
1	3	taskA3	28	2021-08-18	
2	4	taskB1	26	2021-06-25	
3	18	taskF3	19	2021-06-04	
4	19	taskG1	48	2021-06-21	
5	21	taskG3	20	2021-07-08	
6	26	taskI2	1	2021-02-03	
7	27	taskI3	18	2021-02-11	
8	34	taskL1	52	2021-09-02	
9	38	taskM2	23	2021-11-14	
10	40	taskN1	37	2021-01-10	
11	43	taskO1	11	2021-10-09	
12	44	taskO2	14	2021-10-19	
13	49	taskQ1	56	2021-02-27	
14	50	taskQ2	13	2021-03-04	

- 16) Select team\_name as “Team Name” and project\_title as “Project Name” from Team and Project table.

```
final/postgres@PostgreSQL:13 ~
Query Editor  Query History
1 select team_name as "Team Name", project_title as "Project Name"
2 from s10."Team" join s10."Project"
3 on s10."Team".team_id=s10."Project".project_id
4
5
```

Data Output		Explain	Messages	Notifications
	Team Name character varying (255)	Project Name character varying (255)		
1	Karma Gigantic	ProjectA		
2	The Titans	ProjectB		
3	Internet Explorers	ProjectC		
4	Tech Connect	ProjectD		
5	Blue Screens	ProjectE		
6	Temp	ProjectF		
7	Jazz Our Souls	ProjectG		
8	Solarbreeze	ProjectH		
9	Zenit	ProjectI		
10	Sub-Ex	ProjectJ		

- 17) Find out "Employee info"."employee\_id","Employee info"."employee\_name","Employee info"."employee\_contact".

```
final/postgres@PostgreSQL:13 ~
Query Editor  Query History
1 Select "Employee info"."employee_id", "Employee info"."employee_name", "Employee"."employee_contact"
2 from s10."Employee info" full join s10."Employee"
3 on "Employee"."employee_id" = "Employee info"."employee_id";
4
```

Data Output		Explain	Messages	Notifications
	employee_id bigint	employee_name character varying (255)	employee_contact bigint	
1	1	PATEL VISHVAKUMARI BHARATEBHAI	9452445437	
2	2	SUBAPANENI SRI AKHIL	9797969040	
3	3	DHAVALIKUMAR HASMUKHBHAI SOLANKI	9730463356	
4	4	PRIYAL SURESHBHAI RAJ	9650778056	
5	5	CHAIRMAN SIDDEARTH DINESHBHAI	9572496054	
6	6	SOONANI RIDHAM RAMESHBHAI	9446598752	
7	7	DIYORA MANAN MANOJBHAI	9429538863	
8	8	VAGHASIYA CHIRAG JIVRAJBHAI	9726620624	
9	9	DHOLA AXIT KISHORBHAI	9416361746	
10	10	PATEL AKSH BHARATLAL	9502208341	
11	11	SAVALIYA KEVAL DINESHKUMAR	9976867473	
12	12	SHAH HARMEEN ATULBHAI	9186191069	

✓ Successfully run. Total query runtime: 275 msec. 100 rows affected.

18) Find out product id,product name,product type,product price.

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

```
1 select "Product".product_id,"Product"."Product_name","Product"."Product_type","Product_info".product_price
2 from s10."Product_info" LEFT JOIN s10."Product"
3 ON s10."Product".product_id = s10."Product_info".product_id
4
5
```

The results table has columns: product\_id, Product\_name, Product\_type, and product\_price. The data is as follows:

	product_id	Product_name	Product_type	product_price
1	1	14746	Input	14746
2	2	14291	Input	14291
3	3	8277	Input	8277
4	4	5253	Input	5253
5	5	14364	Input	14364
6	6	4464	Input	4464
7	7	8074	Input	8074
8	8	9862	Input	9862
9	9	11375	Input	11375
10	10	14753	Input	14753
11	11	4813	Input	4813
12	12	4852	Input	4852
13	13	7506	Input	7506
14	14	1872	Input	1872
15	15	13362	Input	13362
16	16	13749	Input	13749
17	17	3596	Input	3596

19) select distinct Product\_id,product\_price from "S10"."Product" natural right outer join "S10"."Product\_info"

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

```
1 select distinct Product_id,product_price from "S10"."Product" natural right outer join "S10"."Product_info"
2
```

The results table has columns: product\_id and product\_price. The data is as follows:

	product_id	product_price
1	75	3454
2	2	14291
3	47	3009
4	94	14629
5	12	4852
6	27	10310
7	52	9760
8	65	11602
9	62	10865
10	20	4862

20) select task\_name from "S10"."Task" except all select task\_name from "S10"."Task" where employee\_id='5'

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects, including tables like Customer, Employee, and Task. The main window shows a query editor with the following SQL code:

```
1 select task_name from "S10"."Task" except all select task_name from"S10"."Task" where employee_id='5'
```

The results pane shows a table titled "Data Output" with the following data:

task_name
Opela
Opela
Opela
Temp
Bitwolf
Bamity
Bamity
Regrant
Hatty
Redhold

21) select "Team".team\_id, "Project".project\_title, "Team".no\_of\_employee from "S10"."Team" inner join "S10"."Project" on "Team".project\_id="Project".project\_id;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects, including tables like Customer, Employee, and Project. The main window shows a query editor with the following SQL code:

```
1 select "Team".team_id, "Project".project_title, "Team".no_of_employee
2 from "S10"."Team"
3 inner join "S10"."Project"
4 on "Team".project_id="Project".project_id;
```

The results pane shows a table titled "Data Output" with the following data:

team_id	project_title	no_of_employee
1	ProjectA	33
2	ProjectB	42
3	ProjectC	47
4	ProjectD	21
5	ProjectE	47
6	ProjectF	32
7	ProjectG	49
8	ProjectH	27
9	ProjectI	48
10	ProjectJ	29

A green success message at the bottom right of the results pane states: "Successfully run. Total query runtime: 63 msec. 20 rows affected."

22) select "Task"."task\_name","Task\_completion"."task\_endTime" from "S10"."Task" left join "S10"."Task\_completion" on "Task".task\_id = "Task\_completion".task\_id;

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

- Browser:** Shows the database structure under the schema "S10", specifically the "Tables (16)" section.
- Query Editor:** Contains the following SQL code:
 

```
1 select "Task"."task_name", "Task_completion"."task_endTime"
2   from "S10"."Task"
3  left join "S10"."Task_completion"
4    on "Task".task_id = "Task_completion".task_id;
5
```
- Data Output:** Displays a table with two columns: "task\_name" and "task\_endTime". The data is as follows:
 

	task_name	task_endTime
1	Holdlamis	2021-08-14
2	Alphazap	2021-08-17
3	Zoolab	2021-08-31
4	Fix San	2021-06-28
5	Toughjoyfax	2021-06-30
6	Matsoft	2021-07-20
7	Zathin	2021-07-07
8	Aerified	2021-08-01
9	Toughjoyfax	2022-08-09
10	Flexidy	2021-10-20
- Messages:** A green success message: "Successfully run. Total query runtime: 104 msec. 100 rows affected."

23) select "Product"."Product\_name","Product\_info"."product\_price" from "S10"."Product" right join "S10"."Product\_info" on "Product".product\_id = "Product\_info".product\_id;

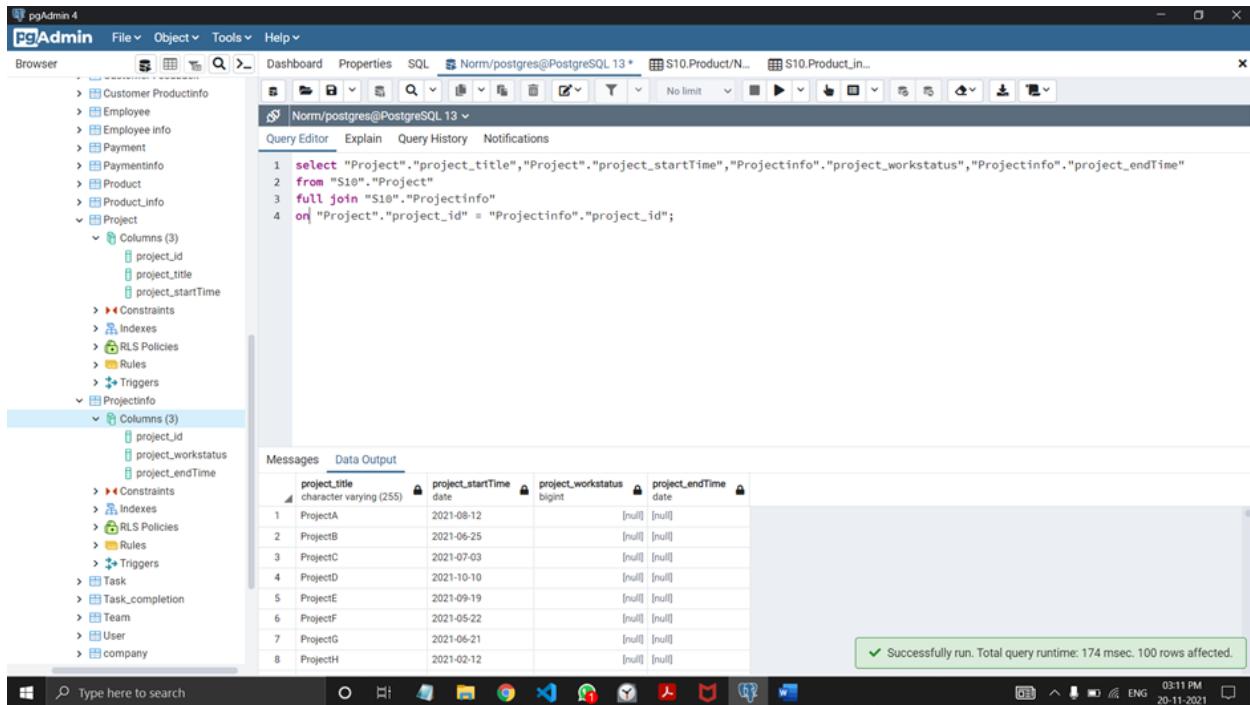
The screenshot shows the pgAdmin 4 interface with the following details:

- Browser:** Shows the database structure under the schema "S10", specifically the "Tables (16)" section.
- Query Editor:** Contains the following SQL code:
 

```
1 select "Product"."Product_name", "Product_info"."product_price"
2   from "S10"."Product"
3  right join "S10"."Product_info"
4    on "Product".product_id = "Product_info".product_id;
5
```
- Data Output:** Displays a table with two columns: "Product\_name" and "product\_price". The data is as follows:
 

	Product_name	product_price
1	Stringtough	14746
2	Sonsing	14291
3	Duobam	8277
4	Bigtax	5253
5	Holdlamis	14364
6	Horne Ing	4464
7	Sub-Ex	8074
8	Otocom	9882
9	Fintone	11375
10	Mat Lam Tam	14753
- Messages:** A green success message: "Successfully run. Total query runtime: 186 msec. 100 rows affected."

24) Select "Project"."project\_title", "Project"."project\_startTime", "Projectinfo"."project\_workstatus", "Projectinfo"."project\_endTime" from "S10"."Project" full join "S10"."Projectinfo" on "Project"."project\_id" = "Projectinfo"."project\_id";



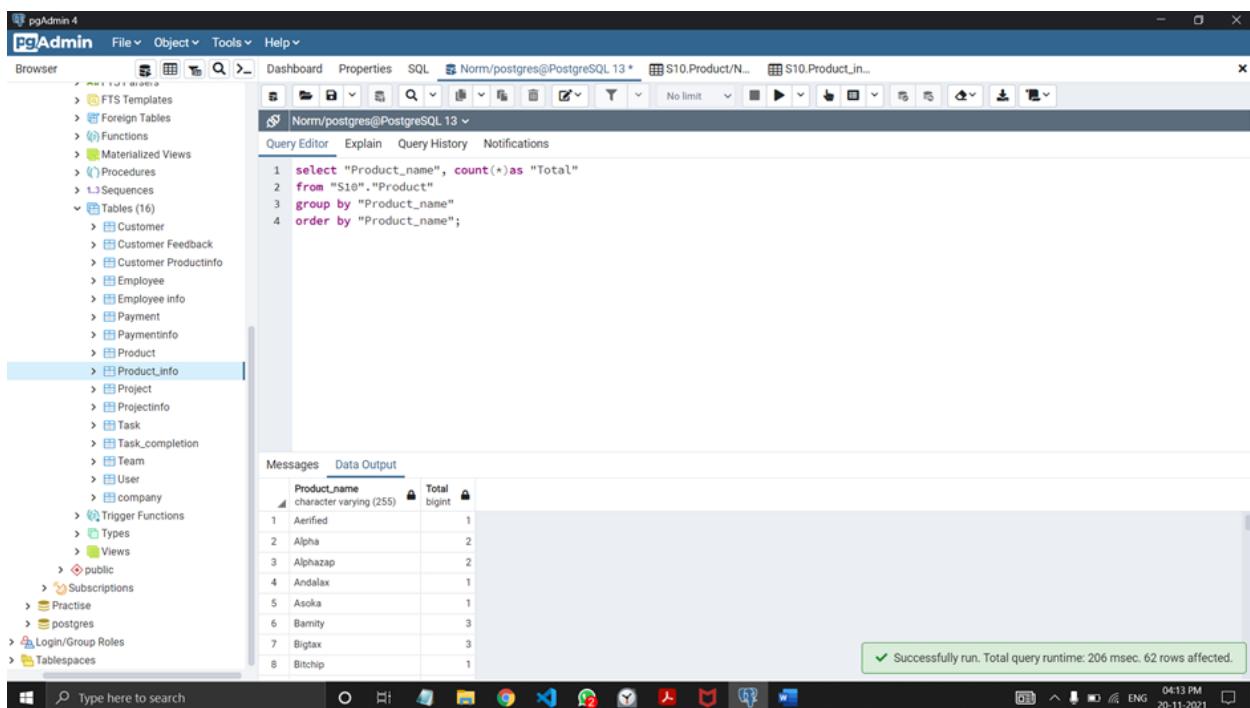
```

1 select "Project"."project_title", "Project"."project_startTime", "Projectinfo"."project_workstatus", "Projectinfo"."project_endTime"
2 from "S10"."Project"
3 full join "S10"."Projectinfo"
4 on "Project"."project_id" = "Projectinfo"."project_id";
    
```

	project_title	project_startTime	project_workstatus	project_endTime
1	ProjectA	2021-08-12	[null]	[null]
2	ProjectB	2021-06-25	[null]	[null]
3	ProjectC	2021-07-03	[null]	[null]
4	ProjectD	2021-10-10	[null]	[null]
5	ProjectE	2021-09-19	[null]	[null]
6	ProjectF	2021-05-22	[null]	[null]
7	ProjectG	2021-06-21	[null]	[null]
8	ProjectH	2021-02-12	[null]	[null]

Successfully run. Total query runtime: 174 msec. 100 rows affected.

25) select "Product\_name", count(\*) as "Total" from "S10"."Product" group by "Product\_name" order by "Product\_name";



```

1 select "Product_name", count(*) as "Total"
2 from "S10"."Product"
3 group by "Product_name"
4 order by "Product_name";
    
```

	Product_name	Total
1	Aerified	1
2	Alpha	2
3	Alphazap	2
4	Andalax	1
5	Asoka	1
6	Bamity	3
7	Bigtaz	3
8	Bitchip	1

Successfully run. Total query runtime: 206 msec. 62 rows affected.

26) Find total number of product which have price less than 5000.

```
SELECT count(*) FROM prct."Product_info" WHERE product_price < 5000
```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, listing various database objects like FTS Parsers, Functions, Materialized Views, Procedures, Sequences, Tables (16), and so on. The 'Tables (16)' section is expanded, and 'Product\_info' is selected. The right pane is the 'Query Editor' where the following SQL query is written:

```
1 SELECT count(*)
2 FROM prct."Product_info"
3 WHERE product_price < 5000
4
5
6
7
8
```

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

count
18

A green status bar at the bottom right indicates: Successfully run. Total query runtime: 109 ms.

27) Find all the team from “Team” table which have 7 employee in Team.

```
SELECT team_name FROM prct."Team" Where no_of_employee = 7
```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, listing various database objects. The 'Tables (16)' section is expanded, and 'Team' is selected. The right pane is the 'Query Editor' where the following SQL query is written:

```
1 SELECT team_name
2 FROM prct."Team"
3 WHERE no_of_employee = 7
4
5
6
7
8
```

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

team_name
Karma Gigantic
Solarbreeze
Ronstring

28) Find task from “Task” table which is assigned to employee id 1 to 10.

```
SELECT task_name FROM prct."Task" ORDER BY employee_id limit 10
```

The screenshot shows the pgAdmin 4 interface. The left sidebar lists various database objects like FTS Parsers, Functions, Sequences, Tables, and Views. The main area shows a query editor with the following SQL code:

```
1 SELECT task_name
2 FROM prct."Task"
3 ORDER BY employee_id limit 10
```

The results pane displays a table with the following data:

task_name
taskI2
taskE3
taskH1
taskC2
taskA1
taskD2
taskB2
taskK2
taskF2
taskO1

A green success message at the bottom right indicates "Successfully run. Total query runtime: 120 msec. 10 rows affected."

29) Find all the details of customer from “Customer” table who bought camera.

```
SELECT * FROM prct."Customer" WHERE selected_item = 'Camera'
```

The screenshot shows the pgAdmin 4 interface. The left sidebar lists various database objects like FTS Parsers, Functions, Sequences, Tables, and Views. The main area shows a query editor with the following SQL code:

```
1 SELECT *
2 FROM prct."Customer"
3 WHERE selected_item = 'Camera'
```

The results pane displays a table with the following data:

customer_id	customer_name	selected_item	customer_contact	customer_address
16	Grace	Camera	9579030369	Cempaka
29	Allison	Camera	9819750353	Al Mazra'ah ash Sharqiyah
44	Maria	Camera	9454315227	Umm Kaddah
53	Marge	Camera	9303917479	Breda
75	Oliver	Camera	9120977729	Gus'-Khrustal'nyy

30) Find all the employee whose cur\_address is same as per\_address.

```
SELECT * FROM prct."Employee info" WHERE employee_per_address = 'same as cur'
```

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

- Browser:** Shows the database schema with the "Employee info" table expanded, displaying columns like employee\_id, employee\_name, employee\_cur\_address, employee\_per\_address, etc.
- Query Editor:** Contains the following SQL query:
 

```
1 SELECT *
2 FROM prct."Employee info"
3 WHERE employee_per_address = 'same as cur'
```
- Data Output:** Displays the results of the query, showing 30 rows of employee data. The columns are employee\_id, employee\_name, employee\_cur\_address, employee\_per\_address, date\_of\_birth, employee\_email, and date.

employee_id	employee_name	employee_cur_address	employee_per_address	date_of_birth	employee_email
1	PATEL VISHVAKUMARI BHARATBHAI	45441 Dryden Terrace	same as cur	1981-06-10	nollfaunt0@sfgate.com
2	SURAPANENI SRI AKHIL	87 Farragut Park	same as cur	1987-06-08	ebevis1@chron.com
3	DHAVALKUMAR HASMUKHBHAI SOLANKI	80915 Lunder Way	same as cur	1984-06-01	ljoseleitch2@cargo collective.com
4	PRIYAA SURESHBHAI RAJ	76 Cascade Alley	same as cur	1988-01-04	rhambleton3@ask.com
5	CHAUHAN SIDDHARTH DINESHBHAI	15 Derek Street	same as cur	1989-04-05	dcaisford4@sciencedaily.com
6	PATEL AKSH BHARATLAL	25181 Elgar Street	same as cur	1987-09-01	mbarron9@xrea.com
7	SAVALIYA KEVAL DINESHKUMAR	5943 Oak Valley Center	same as cur	1980-02-11	ethawa@dagondesign.com
8	SHAH HARSHIBEN ATULBHAI	990 Glendale Street	same as cur	1986-04-07	gpinwillb@sohu.com
9	BHIKADIYA DHRUVIL MANJIBHAI	26 Gateway Hill	same as cur	1981-07-01	lzanrec@wufoo.com
10	BHALALA JAY BHARATBHAI	91871 Reinde Plaza	same as cur	1988-12-04	nburlayg@tiny.cc
11	DALSANIA NEVILKUMAR JITENDRABHAI	756 Farragut Court	same as cur	1982-03-11	cbangleh@issuu.com
12	SHAH HARSHAL SANDEEP	7146 Brown Point	same as cur	1983-06-05	cmaconi@yellowbook.com
13	CHAUHAN VISHVARAJSINH PRAVINSINH	76203 Ohio Lane	same as cur	1982-01-02	wtunnockj@barnesandnoble.com
14	PATEL HARSH SHAILESHBHAI	9326 Melby Avenue	same as cur	1987-02-01	dpossekk@wiley.com
15	PRAJAPATI MEET ASHOKBHAI	66138 Delladonna Plaza	same as cur	1987-10-07	franciskiewiczt@europa.eu

31) Find all the products from product table which have price greater than 10000.

```
SELECT * FROM prct."Product" WHERE product_id in ( SELECT product_id FROM prct."Product_info" WHERE product_price > 10000 )
```

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

- Browser:** Shows the database schema with the "Product" table expanded, displaying columns like product\_id, Product\_name, Product\_type, etc.
- Query Editor:** Contains the following SQL query:
 

```
1 SELECT *
2 FROM prct."Product"
3 WHERE product_id IN ( SELECT product_id
4   FROM prct."Product_info"
5   WHERE product_price > 10000 )
```
- Data Output:** Displays the results of the query, showing 39 rows of product data. The columns are product\_id, Product\_name, and Product\_type.

product_id	Product_name	Product_type
1	Keyboard	Input
2	Microphone	Input
3	Scanner	Input
4	Electronic Whiteboard	Input
5	Barcode reader	Input
6	Graphics Tablets	Input
7	Trackballs	Input
8	Magnetic Ink character reader	Input
9	Monitor	Output
10	Speaker	Output
11	Film Recorder	Output
12	Modems	Input-Output
13	Audio Cards	Input-Output
14	CPU	Hardware

- 32) Perform Right Outer Join between “Employee\_info” table and “Task” table. Find employee\_name from “employee\_info” table and task\_name from “Task” table.

```
SELECT employee_name,task_name FROM prct."Employee_info" RIGHT JOIN prct."Task" ON prct."Employee_info".employee_id = prct."Task".employee_id
```

employee_name	task_name
CHAUHAN SIDDARTH DINESHBHAI	taskA1
MAPARA PRASANTA SANDIP	taskA2
SHAH CHIRAG MUKESHBHAI	taskA3
PATEL ARYAN SHAILESHKUMAR	taskB1
DIYORA MANAN MANOJBHAI	taskB2
TULSIYANI KAMAL RAMESHBHAI	taskB3
SHAH NILAY JAYESH	taskC1
PRIYAL SURESHBHAI RAJ	taskC2
PRAJAPATI MEET ASHOKBHAI	taskC3
GOHIL DARSHAN HARSHADBHAI	taskD1
SONANI RIDHAM RAMESHBHAI	taskD2
KORI VIKAS SURESHBHAI	taskD3
NARSINGHANI DHIRAJKUMAR NAreshKumar	taskE1
PATEL BHUMINKUMAR VISHNUPRASAD	taskE2
SURAPANENI SRI AKHIL	taskE3

- 33) Perform Left Outer Join between “Team” table and “Project” table and find team\_name, no\_of\_employee from “Team” table and project\_name from “Project” table.

```
SELECT team_name,no_of_employee,project_title as Project_name FROM prct."Team" LEFT JOIN prct."Project" ON prct."Team".project_id = prct."Project".project_id
```

team_name	no_of_employee	project_name
Karma Gigantic	7	ProjectA
The Titans	5	ProjectB
Internet Explorers	4	ProjectC
Tech Connect	6	ProjectD
Blue Screens	3	ProjectE
Temp	5	ProjectF
Java Our Souls	2	ProjectG
Solarbreeze	7	ProjectH
Zamit	4	ProjectI
Sub-Ex	6	ProjectJ
Namfix	3	ProjectK
Bitwolf	10	ProjectL
Cardguard	3	ProjectM
Tempsoft	2	ProjectN
It	4	ProjectO

34) Find all the payment where payment\_amount is greater than 10000 and payment\_mode is Cash.

```
SELECT * FROM prct."Payment" WHERE payment_amount > 10000 and payment_mode = 'Cash'
```

The screenshot shows the pgAdmin 4 interface. The left sidebar has a tree view of database objects under 'project/postgres@PostgreSQL 13'. The 'Tables' node is expanded, showing 'Payment' and other tables like 'Customer', 'Employee', etc. The 'Payment' table is selected. The 'Query Editor' tab is active, containing the following SQL code:

```
1 SELECT *
2 FROM prct."Payment"
3 WHERE payment_amount > 10000 and payment_mode = 'Cash'
```

The 'Data Output' tab is selected, showing the results of the query. The columns are 'Payment\_Id', 'payment\_amount', 'payment\_date', and 'payment\_mode'. The data shows 15 rows of payment records, all with payment\_mode as 'Cash'.

Payment_Id	payment_amount	payment_date	payment_mode
1	13646	2021-10-12	Cash
2	10153	2021-10-12	Cash
3	13003	2021-10-12	Cash
4	12781	2021-10-13	Cash
5	12447	2021-10-13	Cash
6	12612	2021-10-14	Cash
7	12931	2021-10-14	Cash
8	12887	2021-10-14	Cash
9	12504	2021-10-15	Cash
10	11808	2021-10-15	Cash
11	11964	2021-10-16	Cash
12	13137	2021-10-16	Cash
13	13058	2021-10-16	Cash
14	13714	2021-10-16	Cash
15	14055	2021-10-16	Cash

35) Find all employees who has not assigned any task.

```
SELECT employee_id,employee_name FROM prct."Employee_info" WHERE employee_id not in ( SELECT employee_id FROM prct."Task" )
```

The screenshot shows the pgAdmin 4 interface. The left sidebar has a tree view of database objects under 'project/postgres@PostgreSQL 13'. The 'Tables' node is expanded, showing 'Employee\_info' and other tables like 'Customer', 'Employee', etc. The 'Employee\_info' table is selected. The 'Query Editor' tab is active, containing the following SQL code:

```
1 SELECT employee_id,employee_name
2 FROM prct."Employee_info"
3 WHERE employee_id not in (SELECT employee_id
4 FROM prct."Task" )
5
```

The 'Data Output' tab is selected, showing the results of the query. The columns are 'employee\_id' and 'employee\_name'. The data shows 15 rows of employee records, all without assigned tasks.

employee_id	employee_name
8	VAGHASIYA CHIRAG JIVRAJBHAI
15	SANGHANI OMIK KANTILAL
16	DUDHATRA HIMANSHU HARASUKHLAL
21	PATEL HARSH SHAILESHBHAI
33	KEDAR BHAVYA HASMUKHLAL
36	DHAMELIYA SANNY GOBARBHAI
41	PATEL HARSH MANISHKUMAR
44	KIKAN ISHA MAHESHBHAI
61	PATEL SMIT ASHYINBHAI
62	PATEL SHIVAM HARESHKUMAR
63	PAREKH ANISH BRUESH
64	MISTRY AYUSH CHETAN
66	GORASIYA DHRUVIL SURESHBHAI
67	BORAD PARAS CHIMANBHAI
69	KHOKHAR KAUTIKKUMAR DINESHBHAI

36) select count(\*) from prct."Payment" where "payment\_amount" <=10000

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

- Browser:** Shows the schema structure under the "Customer" table, including columns like customer\_id, customer\_name, selected\_item, customer\_contact, and customer\_address.
- Query Editor:** Contains the following SQL query:

```
1 select count(*) from prct."Payment" where "payment_amount" <=10000
```
- Data Output:** Displays the result of the query:

count
60
- Message Bar:** Shows a green message: "Successfully run. Total query runtime: 58 msec. 1 rows affected."

37) select \* from prct."Payment" where "Payment\_id" in (select "Payment\_id" from prct."Paymentinfo" where "Payment\_id" in (select "employee\_id" from prct."Employeeinfo" where "employee\_id" <=5));

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

- Browser:** Shows the schema structure under the "Customer" table, including columns like customer\_id, customer\_name, selected\_item, customer\_contact, and customer\_address.
- Query Editor:** Contains the following SQL query:

```
1 select count(*) from prct."Payment" where "payment_amount" < -10000
2 select * from prct."Payment" where "Payment_id" in (select "Payment_id" from prct."Paymentinfo" where "Payme
3 select "customer_contact" from prct."Customer" intersect select employee_contact from prct."Employee";
4 select * from prct."Task_completion" where task_id in(select "task_id" from prct."Task" where "task_id">>=15)
5 select * from prct."Employeeinfo" order by "date_of_birth";
```
- Data Output:** Displays the result of the query, showing five rows of payment information:

Payment_id	payment_amount	payment_date	payment_mode
1	13646	1981-06-10	Cash
2	10153	1987-06-08	Cash
3	7572	1984-06-01	Debit card
4	14282	1988-01-04	Check
5	2488	1989-04-05	Cash
- Message Bar:** Shows a green message: "Successfully run. Total query runtime: 133 msec. 5 rows affected."

38) select "customer\_contact" from prct."Customer" intersect select employee\_contact from prct."Employee";

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with the Employeeinfo table selected. The right pane contains a Query Editor with the following SQL code:

```
1 select count(*) from prct."Payment" where "payment_amount"><=10000
2 select * from prct."Payment" where "Payment_id" in (select "Payment_id" from prct."Paymentinfo" where "Payme
3 select "customer_contact" from prct."Customer" intersect select employee_contact from prct."Employee";
4 select * from prct."Task_completion" where task_id in(select "task_id" from prct."Task" where "task_id">>=15)
5 select * from prct."Employeeinfo" order by "date_of_birth";
```

The Data Output tab shows the result of the query, which is a single column named 'customer\_contact' with a data type of 'bigint'. A message at the bottom right indicates 'Successfully run. Total query runtime: 111 msec. 0 rows affected.'

39) select \* from prct."Task\_completion" where task\_id in(select "task\_id" from prct."Task" where "task\_id">>=15)

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with the Employeeinfo table selected. The right pane contains a Query Editor with the same SQL code as the previous screenshot:

```
1 select count(*) from prct."Payment" where "payment_amount"><=10000
2 select * from prct."Payment" where "Payment_id" in (select "Payment_id" from prct."Paymentinfo" where "Payme
3 select "customer_contact" from prct."Customer" intersect select employee_contact from prct."Employee";
4 select * from prct."Task_completion" where task_id in(select "task_id" from prct."Task" where "task_id">>=15)
5 select * from prct."Employeeinfo" order by "date_of_birth";
```

The Data Output tab shows the results of the query, which is a table with columns 'task\_id', 'task\_status', and 'task\_endTime'. The data is as follows:

task_id	task_status	task_endTime
15	30	2021-11-11
16	43	2021-05-28
17	54	2021-06-03
18	25	2021-06-21
19	96	2021-06-30
20	67	2021-07-07
21	20	2021-07-12
22	60	2021-07-21

A message at the bottom right indicates 'Successfully run. Total query runtime: 146 msec. 46 rows affected.'

40) select \* from prct."Employeeinfo" order by "date\_of\_birth";

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Employeeinfo' node, the 'Columns' section is expanded, showing six columns: employee\_id, employee\_name, employee\_cur\_address, employee\_per\_address, date\_of\_birth, and employee\_email. The 'Query Editor' tab contains the following SQL code:

```
1 select count(*) from prct."Payment" where "payment_amount"><10000
2 select * from prct."Payment" where "Payment_id" in (select "Payment_id" from prct."Paymentinfo" where "Payme
3 select "customer_contact" from prct."Customer" intersect select employee_contact from prct."Employee";
4 select * from prct."Task_completion" where task_id in(select "task_id" from prct."Task" where "task_id">>=15)
5 select * from prct."Employeeinfo" order by "date_of_birth";
```

The 'Data Output' tab displays the results of the fifth query, which lists seven employees with their details:

	employee_id	employee_name	employee_cur_address	employee_per_address	date_of_birth	employee_email
1	11	SAVALIYA KEVAL DINESHKUMAR	5943 Oak Valley Center	same as cur	1980-02-11	ethawa@dagondesk.
2	1	PATEL VISHVAKUMARI BHARATBHAI	45441 Dryden Terrace	same as cur	1981-06-10	nolifaunt0@sfgate.c
3	13	BHIKADIYA DHIRUVIL MANJIBHAI	26 Gateway Hill	same as cur	1981-07-01	lzanrec@wufoo.com
4	89	SAVAN SMIT ASHIRVAD BHAI	55 Sunnyside Parkway	716 Anhalt Lane	1981-08-09	afrenschew29@compe
5	44	KIKANI ISHA MAHESHBHAI	06502 McGuire Crossing	same as cur	1981-09-03	dianbury17@attmu
6	52	SOMAIYA DHAIRYAA MEHUL	87 Fair Oaks Pass	same as cur	1981-09-03	f1f@homestead.co
7	33	KEDAR BHAVYA HASMUKHLAL	796 Acker Street	same as cur	1981-09-03	sakittie@google.it

## **Section 7**

# **Project code with output screenshots**

**Tool: - Visual Studio**

**Language: - Python**

**Code: -**

```
import time
import pymonetdb
import psycopg2
#l pip3 install paymonetdb

#connection = pymonetdb.connect(username="monetdb", password ="monetdb",
hostname="Localhost", database="demo")
#conn=psycopg2.connect("dbname=suppliers user=postgres
password="postgres")
connection = psycopg2.connect(host="localhost", database="Final_Project",
user="postgres", password="admin")

cursor=connection.cursor()
cursor.execute('SELECT version() ')
db_version=cursor.fetchone()
print(db_version)

opt = int(input("Enter option, For Insert press '1', For SQL Query press
'2' and For exit press 3 : "))
#insert data into tables
if opt == 1:
    print("""Table no. and it's corresponding relations are as follows:
1 : Customer
2 : Task
3 : Employee
4 : Employee info
""")
    val=input("Enter Table Number : ")
    v = int(val)
    if(v == 1):
        Insert = 'Insert into
prct.customer(customer_id, customer_name, selected_item, customer_contact, cus
tomer_address) values(%s,%s,%s,%s,%s)'
        value = (

```

```

        input("Enter customer_id : "), input("Enter customer_name : "),
input("Enter selected_item: "),
        input("Enter customer_contact : "), input("Enter customer_address
: "))
        cursor.execute(Insert, value)
        connection.commit()
    elif(v == 2):
        Insert = 'Insert into prct.task(task_id, task_name,
task_employeeid, task_startTime) values(%s,%s,%s,%s)'
        value = (
            input("Enter task_id : "), input("Enter task_name : "),
input("Enter task_employeeid: "),
            input("Enter task_startTime : "))
        cursor.execute(Insert, value)
        connection.commit()
    elif(v == 3):
        Insert = 'Insert into prct.employee(employee_id, employee_contact)
values(%s,%s,%s,%s,%s)'
        value = (
            input("Enter employee_id : "), input("Enter employee_contact : "))
        cursor.execute(Insert, value)
        connection.commit()
    elif(v == 4):
        Insert = 'Insert into prct.employeeinfo(employee_id,
employee_name, employee_cur_address, employee_per_address, date_of_birth,
employee_email) values(%s,%s,%s,%s,%s,%s)'
        value = (
            input("Enter employee_id : "), input("Enter employee_name : "),
input("Enter employee_cur_address: "),
            input("Enter employee_per_address : "), input("Enter date_of_
birth : "), input("Enter employee_email : "))
        cursor.execute(Insert, value)
        connection.commit()
    #Execute query of User's choice
elif opt == 2:
    Query = input("Enter your query : ")
    cursor.execute(Query)
    print(cursor.fetchall())
elif opt == 3:
    exit(0)

```

- Insert into Customer: -

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** On the left, showing a folder structure with files like `__init__.py`, `customer.py`, `Employee.py`, `task.py`, and `Customer`.
- Run and Debug:** A sidebar on the left with options to run and debug, and links to customize launch.json and show automatic configurations.
- Terminal:** The main area shows the output of a Python script named `Hello.py`. The script prints a menu of table relations and then interacts with the user to select a table and provide specific information for that table.
- Output:** Shows the command-line interface used to run the script.
- Problems:** Shows no errors or warnings.
- Python Debug:** A status bar at the bottom indicates the Python extension is active.

```
if opt == 1:  
    print("Table no. and it's corresponding relations are as follows:  
1 : Customer  
2 : Task  
3 : Employee  
4 : Employee info  
  
Enter Table Number : 1  
Enter customer_id : 107  
Enter customer_name : vivek  
Enter selected_item: mobile  
Enter customer_contact : 9339455678  
Enter customer_address : bhavnagar  
PS C:\Users\tarun\Desktop\Python>
```

- **Output:** -

pgAdmin 4

File ▾ Object ▾ Tools ▾ Help ▾

Browser

Dashboard Properties SQL Statistics Dependencies Dependents prct.customer/Final\_Project/postgres@PostgreSQL 13

Procedures Sequences Tables (16)

Company Customer\_Feedback Customer\_Productinfo Payment Paymentinfo Product Product\_info Project Projectinfo Task\_completion Team User customer employee employeeinfo task

Trigger Functions Types Views > public Subscriptions

postgres temp

Anakin/Anakin Online

Query Editor Query History

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

Data Output Explain Messages Notifications

	customer_id	customer_name	selected_item	customer_contact	customer_address
77	77	John	CPU	9229538328	Concepción
78	78	Gillian	Webcam	9677880472	Ratiškovice
79	79	Amy	Microphone	9102960336	San Pablo
80	80	Jenny	Mouse	9945118326	Kitaibaraki
81	105	tarun	keyboard	9374930055	sihor
82	107	vivek	mobile	9339455678	bhavnagar

- Insert into Employee info: -

```

Hello.py - Python - Visual Studio Code
RUN AND DEBUG RUN ... Hello.py 2 x
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.

18 if opt == 1:
19     print("""Table no. and it's corresponding relations are as follows:
20         1 : Customer
21         2 : Task
22         3 : Employee
23         4 : Employee info
24
25         Enter Table Number : 4
26         Enter employee_id : 101
27         Enter employee_name : tarun
28         Enter employee_cur_address: sihar
29         Enter employee_per_address : sihar
30         Enter date_of_birth : 01/07/2002
31         Enter employee_email : hello@gmail.com
32
33         PS C:\Users\tarun\Desktop\Python> & C:/Python/python.exe c:/Users/tarun/Desktop/Python/Hello.py
34 ('PostgreSQL 13.4, compiled by Visual C++ build 1914, 64-bit')
35 Enter option, For Insert press '1', For SQL Query press '2' and For exit press 3 : 1
36 Table no. and it's corresponding relations are as follows:
37         1 : Customer
38         2 : Task
39         3 : Employee
40         4 : Employee info
41
42         Enter Table Number : 4
43         Enter employee_id : 101
44         Enter employee_name : tarun
45         Enter employee_cur_address: sihar
46         Enter employee_per_address : sihar
47         Enter date_of_birth : 01/07/2002
48         Enter employee_email : hello@gmail.com
49
50         PS C:\Users\tarun\Desktop\Python>

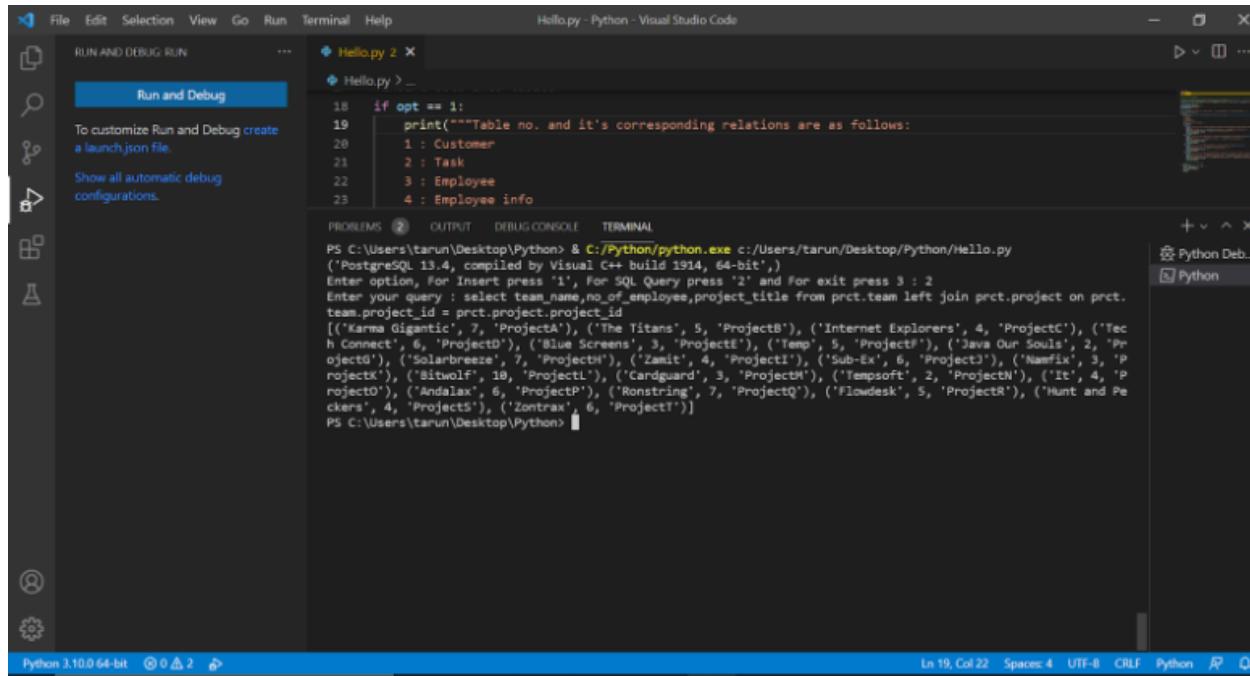
```

## Output: -

employee_id	employee_name	employee_cur_address	employee_per_address	date_of_birth	employee_email
101	tarun	sihar	sihar	2002-07-01	hello@gmail.com

## Execute Query: -

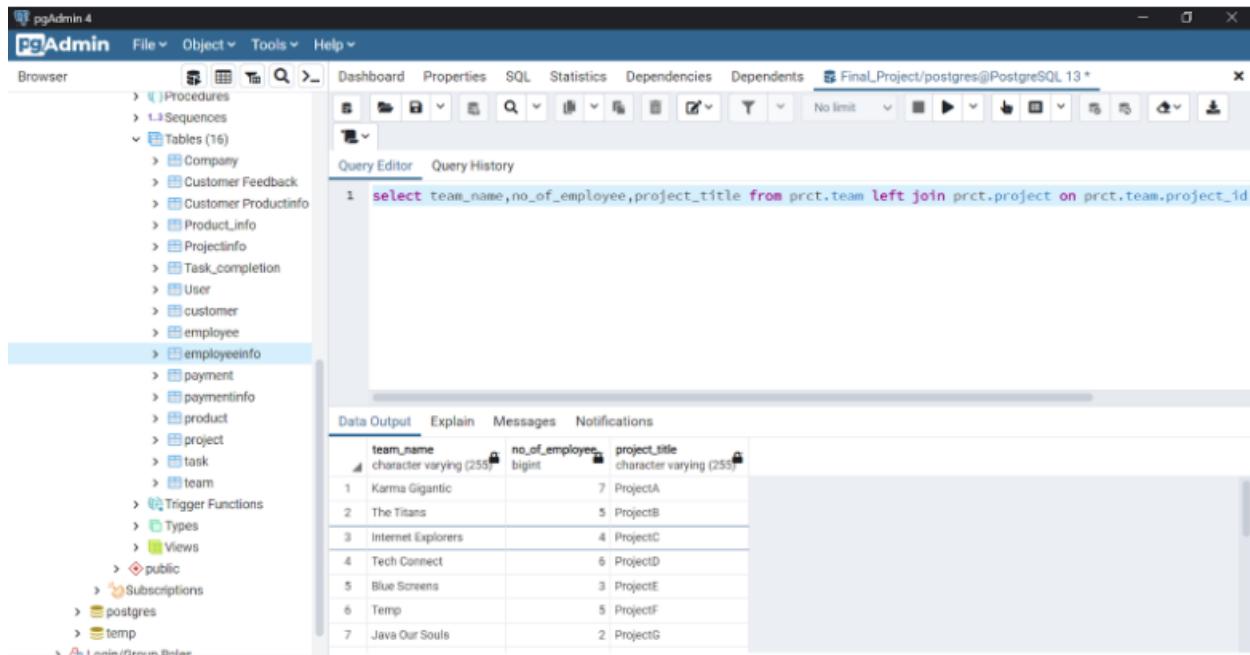
- select team\_name,no\_of\_employee,project\_title from prct.team left join prct.project on prct.team.project\_id = prct.project.project\_id



The screenshot shows the Visual Studio Code interface with a Python file named 'Hello.py' open. The code contains a script that prints a table structure and then executes a SQL query. The output window shows the results of the query:

```
PS C:\Users\tarun\Desktop\Python> & C:/Python/python.exe c:/Users/tarun/Desktop/Python/Hello.py
('PostgreSQL 13.4, compiled by Visual C++ build 1914, 64-bit')
Enter option, For Insert press '1', For SQL Query press '2' and For exit press 3 : 2
Enter your query : select team_name,no_of_employee,project_title from prct.team left join prct.project on prct.team.project_id = prct.project.project_id
[("Karma Gigantic", 7, "ProjectA"), ("The Titans", 5, "ProjectB"), ("Internet Explorers", 4, "ProjectC"), ("Tech Connect", 6, "ProjectD"), ("Blue Screens", 3, "ProjectE"), ("Temp", 5, "ProjectF"), ("Java Our Souls", 2, "ProjectG"), ("Solarbreeze", 7, "ProjectH"), ("Zmit", 4, "ProjectI"), ("Sub-Ex", 6, "ProjectJ"), ("Namfix", 3, "ProjectK"), ("Bitwolf", 10, "ProjectL"), ("Cardguard", 3, "ProjectM"), ("Tempsoft", 2, "ProjectN"), ("It", 4, "ProjectO"), ("Andalax", 6, "ProjectP"), ("Ronstring", 7, "ProjectQ"), ("Flowdesk", 5, "ProjectR"), ("Hunt and Peckers", 4, "ProjectS"), ("Zontrax", 6, "ProjectT")]
PS C:\Users\tarun\Desktop\Python>
```

- Output: -



The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane lists various database objects like Procedures, Sequences, and Tables. The 'Tables (16)' section is expanded, showing tables such as Company, Customer Feedback, Customer Productinfo, Product\_info, Projectinfo, Task\_completion, User, customer, employee, employeeinfo, payment, paymentinfo, product, project, task, team, Trigger Functions, Types, Views, public, Subscriptions, postgres, temp, and An 1 min/1 item Online. The 'Query Editor' pane contains the SQL query:

```
1 select team_name,no_of_employee,project_title from prct.team left join prct.project on prct.team.project_id
```

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

team_name	no_of_employee	project_title
Karma Gigantic	7	ProjectA
The Titans	5	ProjectB
Internet Explorers	4	ProjectC
Tech Connect	6	ProjectD
Blue Screens	3	ProjectE
Temp	5	ProjectF
Java Our Souls	2	ProjectG

- select \* from prct.employeeinfo order by date\_of\_birth

The screenshot shows the Visual Studio Code interface. On the left is the sidebar with icons for file operations like Run and Debug. The main area displays a Python script named `Hello.py`. The terminal tab at the bottom shows the command-line output of running the script, which includes a series of SQL queries and their results. The status bar at the bottom indicates the Python version (3.10.0 64-bit) and other settings.

```

RUN AND DEBUG RUN ... Hello.py ✘
File Edit Selection View Go Run Terminal Help
Hello.py - Python - Visual Studio Code
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\tarun\Desktop\Python> & C:/Python/python.exe c:/Users/tarun/Desktop/Python/Hello.py
('PostgreSQL 13.4, compiled by Visual C++ build 1914, 64-bit')
Enter option, For insert press '1' For SQL Query press '2' and For exit press 3 : 2
Enter your query : select * from prct.employeeinfo order by date_of_birth
([11, 'SAVALIYA KEVAL DINESHKUMAR', '5943 Oak Valley Center', 'same as cur', datetime.date(1980, 2, 11), 'ethawa@dagondesign.com'], [1, 'PATEL VISHVAKUMARI BHARATBHAI', '45441 Dryden Terrace', 'same as cur', datetime.date(1981, 6, 10), 'nolifaunt0@sfgate.com'], [13, 'BHIKADIA DHRUVIL MANJIBHAI', '26 Gateway Hill', 'same as cur', datetime.date(1981, 7, 1), 'lzanrec@wufoo.com'], [89, 'SAVAN SMIT ASHIRVAD BHAI', '55 Sunnyside Parkway', '716 Anhalt Lane', datetime.date(1981, 8, 9), 'afrensch2g@comsenz.com'], [44, 'KIKANI ISHA MAHESHBHAI', '86592 McGuire Crossing', 'same as cur', datetime.date(1981, 9, 3), 'dianbury17@tamu.edu'], [58, 'SANGHAVI KATHAN BHAVINKUMAR', '1 Jackson Park', '716 Anhalt Lane', datetime.date(1981, 9, 3), 'bbmiddle1@ftc.gov'], [39, 'KEDAR BHAVYA HASMIJHLAL', '796 Acker Street', 'same as cur', datetime.date(1981, 9, 3), 'sakitt@google.it'], [52, 'SOMALYA DHAIYRA MEHUL', '87 Fair Oaks Pass', 'same as cur', datetime.date(1981, 9, 3), 'flif@homestead.com'], [80, 'SAK ARIYA SHREYAS DILIPBHAI', '988 Gerald Road', datetime.date(1981, 11, 2), 'mullin27@google.n 1'], [28, 'SHAH CHIRAG MUKESHBHAI', '474 Dog Crossing Court', '474 Dog Crossing Court', datetime.date(1981, 12, 5), 'ydarbishiher@webmd.com'], [20, 'OMALHAN VISHVARAZISINH PRAVINSINH', '76283 Ohio Lane', 'same as cur', datetime.date(1982, 1, 2), 'wtunnockj@barnesandnoble.com'], [18, 'DALANSINGH NEVILKUMAR JITENDRBHAI', '756 Farragut Court', 'same as cur', datetime.date(1982, 3, 11), 'cbangale@issuu.com'], [23, 'KASUNDRA CHINTAN RAMNIKBHAI', '70 Petterle Junction', '70 Petterle Junction', datetime.date(1982, 6, 8), 'jmcgeever@accuweather.com'], [26, 'PATEL ARYAN SHAILESHBHAI', '3850 American Ash Trail', '3850 American Ash Trail', datetime.date(1982, 7, 9), 't strode@chron.com'], [74, 'DHOLIYA ABHI RAJUBHAI', '4 Pierstorff Street', '4 Pierstorff Street', datetime.date(1982, 12, 11), 'averrill2@xing.com'], [94, 'MAYATRA VRINDA CHANDRESHBHAI', '4869 Sachtn Street', 'same as cur', datetime.date(1982, 12, 11), 'fwillcox2@epa.gov'], [70, 'NALTYADHARA DEVANOI ARVINDBHAI', '80 Reindahl Junction', 'same as cur', datetime.date(1982, 12, 11), 'kgurunson1@tiny.cc'], [64, 'GORASIYA DHRUVIL SURESBHAI', '94889 Mitchell Crossing', '94889 Mitchell Crossing', datetime.date(1982, 12, 11), 'ewyndham1@over-blog.com'], [48, 'PATEL NAKUL BHAVESH', '8400 Kedzie Plaza', '45 Holy Cross Way', datetime.date(1982, 12, 11), 'kbatiselle@
In 19, Col 22 Spaces: 4 UTF-8 CRLF Python R D

```

- Output: -

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure with tables like Company, Customer Feedback, Product\_info, ProjectInfo, Task\_completion, User, customer, employee, payment, paymentinfo, product, project, task, team, Trigger Functions, Types, Views, public, postgres, temp, and main/Tables Below. The right pane shows the Query Editor with the query `select * from prct.employeeinfo order by date_of_birth`. The Data Output tab displays the results of the query in a table format.

	employee_id	employee_name	employee_cur_address	employee_per_address	date_of_birth	employee_email
1	11	SAVALIYA KEVAL DINESHKUMAR	5943 Oak Valley Center	same as cur	1980-02-11	ethawa@dagondesign.com
2	1	PATEL VISHVAKUMARI BHARATBHAI	45441 Dryden Terrace	same as cur	1981-06-10	nolifaunt0@sfgate.com
3	13	BHIKADIA DHRUVIL MANJIBHAI	26 Gateway Hill	same as cur	1981-07-01	lzanrec@wufoo.com
4	89	SAVAN SMIT ASHIRVAD BHAI	55 Sunnyside Parkway	716 Anhalt Lane	1981-08-09	afrensch2g@comsenz.com
5	44	KIKANI ISHA MAHESHBHAI	06592 McGuire Crossing	same as cur	1981-09-03	dianbury17@tamu.edu
6	58	SANGHAVI KATHAN BHAVINKUMAR	1 Jackson Park	716 Anhalt Lane	1981-10-11	bbmiddle1@ftc.gov
7	39	KEDAR BHAVYA HASMIJHLAL	70 Petterle Junction	70 Petterle Junction	1982-03-11	jmcgeever@accuweather.com

Successfully run. Total query runtime: 490 msec. 101 rows affected.