# Task Management System

**Database Management Systems**

**(IT - 214)**


Assigned by: Prof. Minal Bhise

Mentor TA: Dhairya

## Section-10 Group-3

Team Members:

1. Prasansa Mapara 201901097
2. Dharti Ikharwala 201901243
3. Harsh Mehta 201901277
4. Dhwanil Shah 201901450

# Table of Contents

**Section1: SRS** **4**

**Section2: Noun Analysis** **19**

**Section3: ER diagrams** **29**

**Section4: ER to Relational Mapping** **33**

**Section 5: Normalization and Schema Refinement** **39**

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

**Section7: Project Code for Frontend** **90**

# Section1: SRS

# A. Description of Case study:

## 1. Purpose

Task Management is one of the most crucial aspects of Project Management. Any typical work breakdown structure divides a project into manageable portions. Traditionally, tools like checklists, To-Do lists, Calendars, Kanban boards, etc. can be used for smooth organization, assignment, and prioritization of tasks. But with big amounts of data, these tools prove ineffective as work will be spread across too many tools. Moreover, in PMI's 2017 Global Project Management survey, 37% of executive leaders said "the primary cause of failure [of projects in their organization] was a lack of clearly defined objectives and milestones to measure progress" and "a lack of discipline when implementing strategy".

The purpose of this Task Management System is to overcome these issues and allow us to set goals and milestones, manage deadlines, and keep track of deliverables. Complex projects are broken down into many related milestones, task lists, tasks, and subtasks, hence eliminating the need to remember all the tasks. This makes the team more productive as the system also sends timely reminders. It integrates all the work to ensure a quality product for clients. Such a system is needed for overall improvisation of the workflow by detailed goal setting, and more efficiency.

**Real World working:** In the Real World, many software like Jira, Asana, Zoho Projects, etc. are examples of Task/Project Management Softwares. These have features like agile reporting, roadmaps, kanban boards, scrum boards, Gantt charts, dependencies, calendar, automation, etc. all in one place.

**Issues in the existing workflow with no Task Management System in use:**
Traditional tools like kanban boards and calendars are used in place of central software. But it faces issues like clustering of tasks as the tasks are spread over multiple websites. Dividing, and tracking the work, staying in sync with the teams,following strict deadlines, reporting the work done, as well as keeping up with the emerging technologies gets difficult in absence of Task Management Software.

## 2. Intended Audience and Reading Suggestions

Any company involving a large number of employees is an intended audience for such software. It also depends on the scale at which the organization operates. For eg. premium, enterprise versions of this software generally attract Multinational Companies like Google, Atlassian, Deloitte, Amazon, Snap INC. It even attracts government agencies and large-scale foundations like NASA, Bill and Melinda Gates Foundation. Various domains of industries like media and entertainment, beauty, fintech, tech, and even universities like Harvard, Northwestern, etc. use such software. While the free, or standard versions generally attract schools, local shops, individuals, etc.

## 3. Product scope

This product has issue trackers, customizable workflows to increase transparent project management, team member accountability, and improved customer satisfaction. It also has features like estimation and work logging, email notifications, project backlogs. Progress reporting, as we plan, implement, and ship any products. It also integrates security throughout the lifecycle of any product. There is a feature called timeline to visualize the relationship between tasks and keep track of work. There are calendars for easy access to schedules that are visible to everybody. Monitoring, and task assignment features for managers. Features for easy collaboration, shared access to tasks are also present. Developers can access the system at any point in time and add/modify features depending on the requirements of the company for efficient usage of the system.

### 4. Description

Our software replaces the traditional task management tools with a large-scale central system and ensures getting more work done, with much more ease. It helps teams plan, assign, track, report, and manage their work. It has 24/7 customer support. Its primary goal is to collaborate and connect the workforce, no matter the scale. The workflow can be customized as per clients' requirements. It will help us keep track of the team tasks. Assess their workloads, use timers, go through their team reports, and communicate with multiple teams at once, as well as get instant updates.
This Task management software helps the companies track the status of their ongoing projects along with the information of every employee working on a particular project. Further our solution of the task management system also provides privileges to different classes of users using the software. For eg., the administration in the company would be able to see the data of all the projects along with the details of the employees working in it. Secondly, a project/team lead would only be able to see/update the status and information of the projects and employees working in his team and not the information about all the ongoing projects. Further, the clients of the company would only be able to see the finalized product that is available for buying and not the ongoing projects or the information of any of the employees in the company.

## B. Fact-finding Techniques

### 1. Background Reading

- **Books**

  Database system concepts by Henry F. Korth, Abraham Silberschatz, and S. Sudarshan

- **Readings**

  We explored a few tasks management software that are currently leading in the industry and learned about its features, and how those features come in handy while working on large-scale projects.

  ➔ Jira software by Atlassian We learned about the various features in a large-scale project management software and got an insight on the working of such software.

  ➔ Asana's software for Task Management We got an insight on what a project management tool is, why do organizations/individuals need it, and how this tool helps in collaborating tasks across various departments like sales, engineering, marketing, human resources, etc.

  ➔ Task Management Software | Free Task Management System It is a guide on what goes behind managing team tasks, how a task management system like ZOHO makes it less tedious. We also got to learn about how to pick the right task management software for our teams, and finally the features and benefits this software provides.

  ➔ How to develop a Task Management System This is a case study on how DDI Development created a task management software. We got to know the difference between project management, and task management. The project structure of the task management platform, its development process, the team needed, and a brief description of the structure, benefits, features, and working of their software.

## 2. Interviews

**Interview 1:**

**HOPE**: (Role Play) Interview Plan
**System**: Task Management System Database Company

**Interviewee:** 1) Bhavi Panchal(Role Play) **Designation:** Admin
**Interviewer:** 1) Dharti Ikharwala(Role Play) **Designation:** Developer

**Date:** 7/10/2021 **Time:** 16:30
**Duration:** 45 minutes **Place:** Google Meet

**Agenda:**
Problems with security
Overall Productivity
Increase in Company sales
Follow-up actions

Documents to be brought to the interview:
Schema of the existing database
Any documents relating to current project management methods.

**HOPE: Interview Summary**
System: Task Management System Database Company

**Interviewee**: 1) Bhavi Panchal(Role Play) **Designation**: Admin
**Interviewer**: 1) Dharti Ikharwala((Role Play) **Designation**: Developer

**Date**: 7/10/2021 **Time**: 16:30
**Duration**: 45 minutes **Place**: Google Meet

**Purpose of Interview**:

Preliminary meeting to discuss the security concerns, identify user and employee requirements and get an insight on how this software affected the companies' sales and overall productivity of employees.

1. What do you think about the future scope of the Task Management System?
2. How is your current company working with security problems?
3. What is the overall productivity you observed using the Task Management Software?
4. How were you able to meet customer requirements? How much were you satisfied using Task Management Software?

**Interview 2**:

**HOPE**: (Role Play) Interview Plan
**System**: Task Management System Database

**Interviewee**: 1) Rakesh Shah(Role Play) **Designation**: Project Manager at HOPE
**Interviewer**: 1) Harsh Mehta(Role Play) **Designation**: Developer

**Date**: 8/10/2021 **Time**: 15:00
**Duration**: 45 minutes **Place**: Google Meet

**Agenda**:

Problems with existing system/software
What needs to change
Project structure discussion

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

The overview of the annual project plan
Papers related to the working structure of the company

**HOPE: Interview Summary**
**System**: Task Management System Database

**Interviewee**: 1) Rakesh Shah(Role Play) **Designation**: Project Manager at HOPE
**Interviewer**: 1) Harsh Mehta(Role Play) **Designation**: Developer

**Date**: 8/10/2021 **Time**: 15:00
**Duration**: 45 minutes **Place**: Google Meet

**Purpose of Interview**:

To discuss the following points.

1. How many projects do you assign to the teams monthly?
2. Approximately, at a time, how many employees work under you?
3. How much information do you acquire for each employee?
4. What is the approximate budget of all the projects done in a year?
5. Is the communication between you and the team leaders working under you proper and smooth?

**Interview 3:**

**HOPE**: (Role Play) Interview Plan
**System**: Task Management System Database

**Interviewee**: 1) Aman Patel(Role Play) **Designation**: Team Leader at HOPE
**Interviewer**: 1) Dhwanil Shah(Role Play) **Designation**: Developer

**Date**: 7/10/2021 **Time**: 18:30
**Duration**: 45 minutes **Place**: Google Meet

**Agenda**:

Problems with existing systems/software.
Project structure discussion and completing the project with the budget assigned.

Maintaining a team and assigning tasks to different members such that none of the members is overloaded with work.

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

The working structure of the team.
Documents related to the team members.

**HOPE: Interview Summary**
System: Task Management System Database

**Interviewee**: 1) Aman Patel(Role Play) **Designation**: Team Leader at HOPE
**Interviewer**: 1) Dhwanil Shah(Role Play) **Designation**: Developer

**Date**: 7/10/2021 **Time**: 18:30
**Duration**: 45 minutes **Place**: Google Meet

**Purpose of Interview**:

Discussion on the following questions:

1. How do you divide the tasks between the different members of your team?
2. How do you prioritize the tasks of the project?
3. What is the average size of the team that you lead?
4. How many projects get completed before the deadline?
5. Is the communication between you and the team members working under you smooth?

**Interview 4**:

**HOPE: (Role Play) Interview Plan**

**System:** Task Management System Database

**Interviewee:** 1) Anuj Kapadia(Role Play) **Designation:** Employee at HOPE

**Interviewer:** 1) Prasansa Mapara(Role Play) **Designation:** Developer (Prabandh)

**Date:** 7/10/2021 **Time:** 14:30
**Duration:** 45 minutes **Place:** Google Meet

**Agenda:**

Problems with features and any other concerns
Current features
Initial ideas
Follow-up actions

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

Rough plan of building's IT infrastructure.

Any documents relating to current project management methods.

**HOPE: Interview Summary**
**System:** Task Management System Database company

**Interviewee:** 1) Anuj Kapadia(Role Play) Designation: Employee at HOPE
**Interviewer:** 1) Prasansa Mapara(Role Play) Designation: Developer

**Date**: 7/10/2021 **Time**: 14:30
**Duration**: 45 minutes **Place**:Google Meet

**Purpose of Interview**:

Preliminary meeting to identify problems and requirements regarding features at the HOPE'S site.

1. What are the different projects that your company is currently working on?
2. Does your company currently use any task management software? If Yes, which one? If No, what difficulties do you currently face (if any) in organizing the project/task management and managing the employees for different projects of your company?
3. What are the different features that you want a task management system to have?
4. What are the different levels of security that you wish the database to have?
5. What size of the database are you expecting considering the growth of your company and with a considerable number of projects currently in the pipeline?

## 3. Questionnaire

We conducted a survey through google forms. We got around 60 responses. Through the survey, we got a better understanding of people's perspectives on the task management system which might help us improve our model.

Our questionnaire consisted of the following questions:

# Task Management System

201901097@daiict.ac.in (not shared) Switch account

* Required

Name *

Your answer

Company/Institution *

Your answer

Domain of your Company *

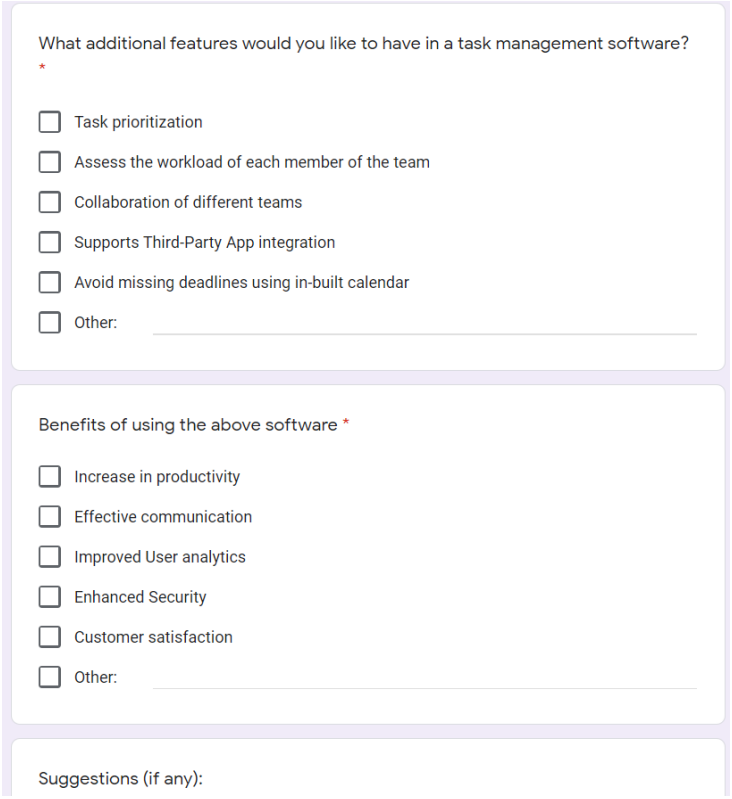
- E-commerce
- Fintech
- Educational
- IT
- IT
- Media and Entertainment
- Beauty
- Other:

Have you ever used task/project management software? *

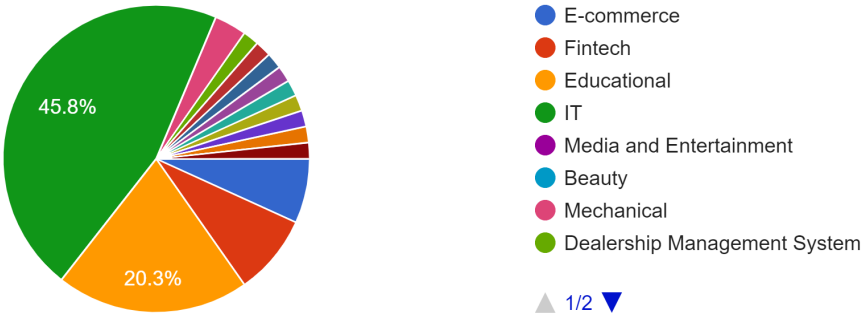
- Jira software by Atlassian
- Zoho Projects software
- Asana task management software
- Trello
- Other:

How much were you satisfied by the software? *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Not much | ○ | ○ | ○ | ○ | ○ | Best |

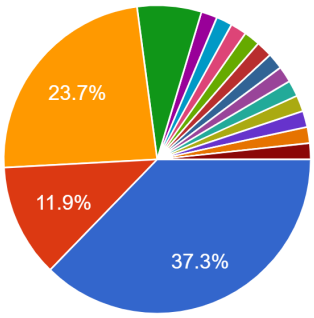



**The domain of your company:**

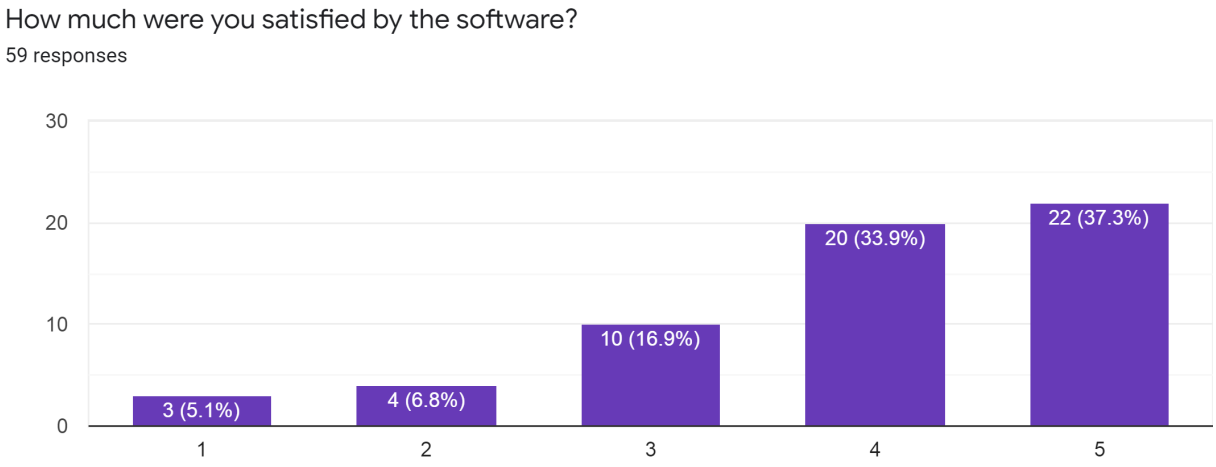


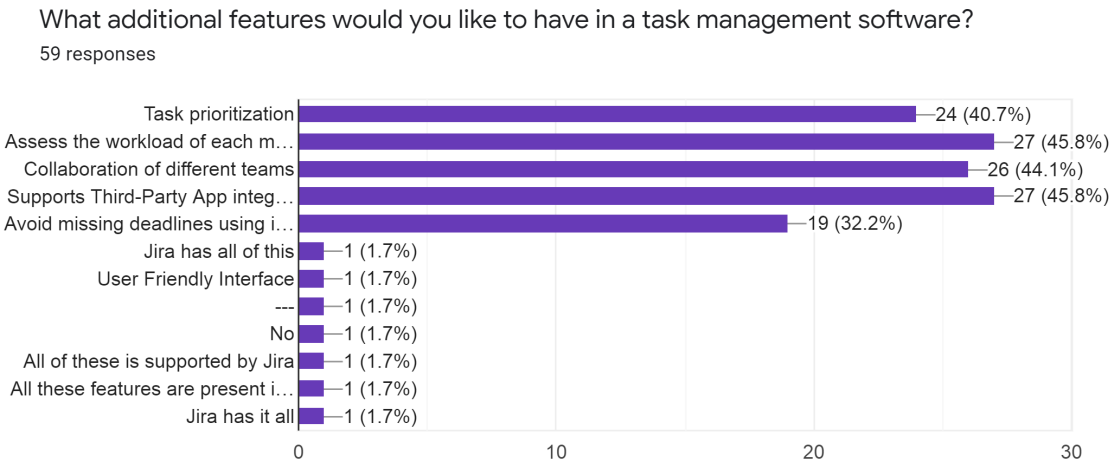

45.8% of our respondents work at Software companies, 20.3% are students from DAIICT, 8.5% work at Fintech while remaining respondents work at other domains listed in the figure above.

**Have you ever used task/project management software?**

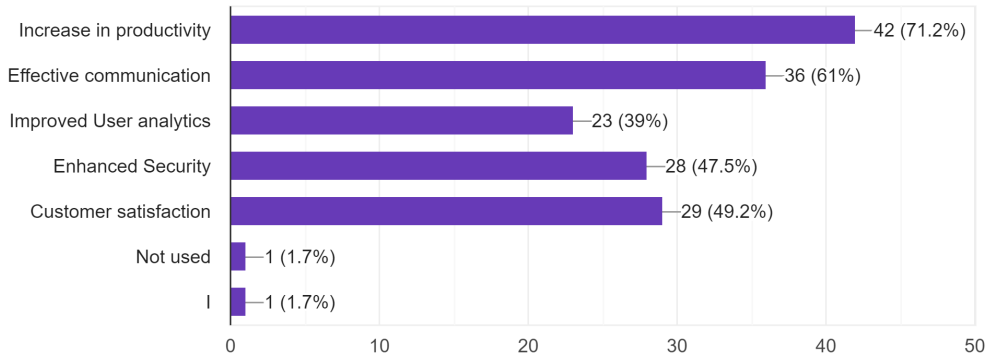



37.3% of our respondents used Jira Software.
23.7% of our respondents used Asana Task Management Software.
11.9% of our respondents used ZOHO Projects Software.
1.7% of our respondents have never used Task Management Software.
While 25.4% of our respondents used other Task Management Softwares like Trello, Microsoft Project, etc.






**Benefits of using the above software:**




## 4. Observations

From the above questionnaire and interviews, the following key points were observed:

1. To improve system functionalities so that the process would be faster, quicker, and secure. This in turn would enhance customer satisfaction and engagement.
2. Data Security: The majority of the companies want their data to be safe and hence, their data must be protected using different access policies.
3. Easy updating: To efficiently update the database from time to time.
4. There must be a feedback system for improving customer experience.
5. To optimize the data storage by removing redundant information in order to reduce the load in the software.
6. From the survey, the most important features to be included in the software are workload assessment of each member and support for 3rd party-app integration.
7. Finally, among the people who have used the task management software, the majority of them are satisfied but expect some improvements on minor issues.

## C. Fact-Finding Chart

| Objective | Technique | Subject | Time Commitment |
|---|---|---|---|
| To get basic knowledge of the currently used task management software | Background reading | Company reports, articles, journals | 4-5 hours |
| To know the requirements/needs of the users of task management software | Google Form | General users of the software | Data collected over a period of 2 days |
| To know the process of the project management | Interview | Project Manager | 1 hour |
| To know the administration process | Interview | Admin | 1 hour |
| To know the testing and debugging process for a project | Observation | Testing Manager | 0.5 day |
| To know the core working of the company | Interview | 1 Team Lead<br>1 Team Member | 1 hour |

## D. Requirements

- **Functional Requirements:**

- Users have to sign up to create their account via email or their phone number as per the customer's choice.
- If the user forgets the password, they can change the password using the 'Forgot password' option after verifying their account.
- If the user takes any premium plans, the mail regarding the due date of payment will be sent via email.
- The queries of the customers will be answered through emails.
- The customer will get all the updates about the new plans and policies via mail.

- **Performance Requirements:**

- The website should be responsive
- Make website speed a priority
- The website should work well on both desktop and mobile view
- Make Information people look for easier to find

## E. User Classes and Characteristics

1) **Admin**: The administrator is involved in the management who supports and supervises the whole system and ensures that the company runs smoothly. He/she also looks after the security of the database and controls its access.

2) **Project Manager:** The project manager is the authority that is responsible for planning, direction, and closing of a single or multiple projects and makes sure that the projects are running on time and within the budget provided for that particular task.

3) **Tester**: A test manager controls the testing activities and tasks for the projects that have been completed but not yet been approved. If there are any bugs identified, then he/she would notify the project manager to look into it.

4) **Team leader:** Team leader instructs, monitors, and motivates a group of members to work for the project assigned, He/she communicates with the seniors along with communicating the goals and targets to the team members. The team leader reports to the project manager and is responsible for maintaining the quality of the project.

5) **Team member/employee:** Team member works at the core level and execute the tasks assigned by the team leader. He/she works with fellow team members to achieve the daily/weekly/monthly targets.

## F. Operating Environment

This product will run on an online platform, The main communication mechanisms are going to be SQL script and the content management system talking with the database.

### 1. Hardware, Software, or Connectivity Requirements

**Hardware:** We need at least an i3 processor for managing databases. In the future, if we start expanding we might need multiple server sets in different areas of different cities to divide load forming a data distribution system.

**Software:** We need PostgreSQL for running SQL commands and structuring our database. We will require a domain name to run the website online. The supported operating system is windows.

**Connectivity:** Stable internet connection to connect our databases with the central server.

### 2. External Interface Requirements

**Hardware Interfaces:** In the future, if we start expanding we might need our server to store information in the form of cache or we might need multiple server sets in different areas of different cities to divide the load.

**Software Interfaces:** The server will fetch system descriptions from the task management database that we created. To store and manage the data, we will require a system like PostgreSQL. We will require a domain name to run the web application online. The user will have to use any browser to access the web application.

**Communications Interfaces:** All communication between the user and the task management system will be end to end encrypted. Each user paying for the software will get a mail regarding the same with the details. There will be a feedback section where the users can share their issues, moreover, a customer service number is also available.

**Third-Party APIs:** The system can be integrated with the third-party apps that are owned by Google, Microsoft. Apart from these, many other third-party apps are also supported.

## G. Product Functions

1) **Profile Creation:** Any user using the software for the first time will have to create his/her profile. This profile will then be authorized by an existing admin after which he/she can use the software as per the role assigned.

2) **Profile Management**: This module includes the efficient management of the profiles of the employees, i.e. projects assigned to them and his/her team lead. It also manages the privileges assigned to different users and the feature of easy updating in case of data added/deleted.

3) **Project Assessment:** The project assessment module includes tracking the status of all the ongoing projects along with the details of the project, i.e. its domain, requirements, budget, etc. It also gives the details of the project manager, team leads, and team members working for the team leads.

4) **Testing and Defect tracking**: Testing and defect tracking help software teams identify, record, and analyze bugs in their software. It's important to detect and record bugs for every member and even more important to assign them to the right team member and at the right time. Resolving these issues at the right time helps the teams to prioritize their targets properly.

5) **Notifications**: This module notifies or pings the users regarding the tasks assigned to them and informs them about the scheduled meetings and discussions with the project manager, team lead, or any member working coherently.

## H. Privileges

1) **Admin**: The administrator working for the company is responsible for solving customers' problems and is involved in management. They have access to the entire database and hence can insert, update as well as delete into the database. He/she can assign as well as revoke the privileges of different users of the database.

2) **Project Manager**: The project manager can access the details related to the projects assigned to him and has no privileges to extract or edit the information related to other projects/tasks not assigned to him. He can appoint the employees to specific teams for different projects and decide the budget required for the projects under his/her domain.

3) **Tester**: The tester can access the details of all the projects that have been completed but not yet approved and inform the project manager about the bugs detected(if any). However, he/she cannot update the details of the members working on that project.

4) **Team leader**: The team leader can track the status of the project assigned to him and get the details of the team members working under his supervision. He/she cannot change the team members without the approval of the project manager. But he can divide the team into sub-groups and assign them specific tasks related to the project.

5) **Team member**: The employees cannot insert, delete or update data from the database. They can view the details of their team members and communicate with them as well as the team leader regarding the project assigned.

6) **Client:** Clients cannot update/insert/delete from the database. They have the privilege to view the current projects going on in the company. So they can only view the project data.

## I. Assumptions

1) It is assumed that the user will have stable internet connectivity for using the website.
2) The user has an account registered with the company for any further interactions.
3) The user is familiar with the working environment of the website.
4) A server of the system should work for all days and at any time.
5) It is assumed that the user's system satisfies operating requirements.

## J. Business Constraints

- The project will go ahead only after the approval of the admin and project manager after a discussion on the main goals and budget related to that particular project.
- The system/software must have at least one administrator so that other company employees and team members can register after his/her authorization.
- One team leader can lead only 1 team and a team member can work for only 1 team so that the productivity of each employee increases and can give his best for the project assigned to him.
- The profile details of any employee of the company should be correct and specific.

# Section2: Noun Analysis

# Problem Description:

Task Management is one of the most crucial aspects of Project Management. Any typical work breakdown structure divides a project into manageable portions. Traditionally, tools like checklists, To-Do lists, Calendars, Kanban boards, etc. can be used for smooth organization, assignment, and prioritization of tasks. But with big amounts of data, these tools prove ineffective as work will be spread across too many tools. Running a startup or any kind of business requires extensive practice, skills, knowledge, and expertise. Moreover, balancing the time, tasks and people is difficult as well. According to a recent study, over 10,500 projects found that only 2.5% of companies complete their projects successfully. That's why using **task management software** would be a way out. Thanks to the task management software. Over 77% of high-performing activities have been completed and met with success. Not only does it help you keep all processes smooth, but you can also better plan the day-to-day workload.

The main purpose of any Task Management System is to overcome these issues and allow us to set goals and milestones, manage deadlines, and keep track of deliverables within the budget provided. Complex projects are broken down into many related milestones, task lists, tasks, and subtasks, hence eliminating the need to remember all the tasks. This makes the team more productive as the system also sends timely reminders. It integrates all the work to ensure a quality product for clients. Such a system is needed for overall improvisation of the workflow by detailed goal setting, and more efficiency.

**Real World working:** In the Real World, many software like Jira, Asana, Zoho Projects, etc. are examples of Task/Project Management Softwares. These have features like agile reporting, roadmaps, kanban boards, scrum boards, Gantt charts, dependencies, calendar, automation, etc. all in one place.

**Issues in the existing workflow with no Task Management System in use:**
Traditional tools like kanban boards and calendars are used in place of central software. But it faces issues like clustering of tasks as the tasks are spread over multiple websites. Task prioritization becomes a challenge for companies not using any task management tool. Dividing, and tracking the work, staying in sync with the teams, following strict deadlines, reporting the work done, as well as keeping up with the emerging technologies gets difficult in absence of Task Management Software. Our software replaces the traditional task management tools with a large-scale central system and ensures getting more work done, with much more ease. It helps teams plan, assign, track, report, and manage their work. It has 24/7 customer support. Its primary goal is to collaborate and connect the workforce, no matter the scale. The workflow can be customized as per clients' requirements. It will help us keep track of the team tasks. Assess their workloads, use timers, go through their team reports, and communicate with multiple teams at once, as well as get instant updates.

This Task management software helps the companies track the status of their ongoing projects along with the information of every employee working on a particular project. Further our solution of the task management system also provides privileges to different classes of users using the software. For eg., the administration in the company would be able to see the data of all the projects along with the details of the employees working in it. Secondly, a project/team lead would only be able to see/update the status and information of the projects and employees working in his team and not the information about all the ongoing projects. Further, the clients of the company would only be able to see the finalized product that is available for buying and not the ongoing projects or the information of any of the employees in the company.

This product has issue trackers, customizable workflows to increase transparent project management, team member accountability, and improved customer satisfaction. It also has features like estimation and work logging, email notifications, project backlogs. Progress reporting, as we plan, implement, and ship any products. It also integrates security throughout the lifecycle of any product. There is a feature called timeline to visualize the relationship between tasks and keep track of work. There are calendars for easy access to schedules that are visible to everybody. Monitoring, and task assignment features for managers. Features for easy collaboration, shared access to tasks are also present. Developers can access the system at any point in time and add/modify features depending on the requirements of the company for efficient usage of the system.

Task management software provides a systematic approach for all the work being done by an individual or a company. It prevents distraction from the ongoing workflow and maintains the focus on the end goal.

Any company involving a large number of employees can use such software. It also depends on the scale at which the organization operates. For eg. premium, enterprise versions of this software generally attract Multinational Companies like Google, Atlassian, Deloitte, Amazon, Snap INC. It even attracts government agencies and large-scale foundations like NASA, Bill, and Melinda Gates Foundation. Various domains of industries like media and entertainment, beauty, fintech, tech, and even universities like Harvard, Northwestern, etc. use such software. While the free, or standard versions generally attract schools, local shops, individuals, etc.

**Review of existing Task Management Software:**

**Jira Software** is being used as a planning and tracking module in our organization, and we have successfully integrated various workflows within the Jira Software to monitor issues and streamline the development environment to cater to any failures. We have created different sets of workflows and software implementation for different teams as per their working requirements and the feature of need base modification makes Jira Software a great solution for multi teams and multi-functional organizations like us. Software availability for different operating platforms is also very helpful; it enables us to work remotely without any issues. We have integrated dozens of in-house data sets and applications like Zephyr and Salesforce with Jira Software to customize our working with the module in more functionalized ways. We are using Jira Software in multiple sections of our organization like technology, Marketing and IT, etc**.**

**Trello** is a very helpful tool for planning what you're working on and the status of that work. My team uses it within our own team to manage workloads and as a manager, I can go in and see what my direct reports are currently working on and the status of that work. I also use it with the IT support team that supports the website I manage. It is very useful for prioritizing work and lets me see the current status of tasks I have assigned to IT. I like all the functionality within the Trello cards, especially the ability to set deadlines, tag users in responses, and link directly to other Trello cards or outside URLs.

## Table - 1 (All nouns and verbs)

| Nouns (Entities) | Verbs (Relationships) |
|---|---|
| Task | divide |
| Management | use |
| aspect | organize |
| project | assign |
| structure | prioritize |
| portion | prove |
| to-do lists | said |
| checklists | define |
| Kanban boards | measure |
| amount | implementing |
| data | overcome |
| tools | allow |
| work | set |

| | |
|---|---|
| spread | manage |
| survey | keep |
| leaders | broke |
| cause | eliminate |
| failure | remember |
| organization | makes |
| lack | sends |
| objectives | integrates |
| milestone | ensure |
| progress | needed |
| discipline | setting |
| strategy | reporting |
| purpose | faces |
| issues | clustering |
| goals | spread |
| milestones | dividing |
| deadlines | tracking |
| deliverables | done |
| task lists | keeping |
| subtasks | staying |
| system | following |
| reminders | shared |
| clients | depending |
| improvisation | track |
| workflow | email |
| efficiency | improved |
| software | logging |
| Jira | reporting |
| asana | implement |
| Zoho projects | integrates |
| examples | visualize |
| features | access |
| agile | add |
| category | modify |
| scrum boards | buy |
| Gantt chart | provide |
| dependencies | assign |

| | |
|---|---|
| calendar | plan |
| automation | manage |
| websites | collaborate |
| technologies | customize |
| absence | connect |
| software | assess |
| workflow | track |
| client | involving |
| requirements | intended |
| workforce | depends |
| Team report | operates |
| users | attracts |
| administration | has |
| team leader | |
| project | |
| product | |
| employee | |
| company | |
| audience | |
| scale | |
| premium | |
| enterprise | |
| agencies | |
| government | |
| domains | |
| industries | |
| data | |
| solution | |
| information | |
| privileges | |
| individuals | |
| versions | |
| Management | |
| aspect | |
| checklist | |
| To-do list | |
| calendars | |

| | |
|---|---|
| product | |
| work | |
| Progress | |
| product | |
| product | |
| relationship | |
| Developers | |
| features | |
| features | |
| tasks | |
| features | |
| work | |
| notifications | |
| Kanban boards | |
| amounts | |
| data | |
| tools | |
| survey | |
| leaders | |
| cause | |

## Table-2 (Accepted Nouns - ER Model)

| **Candidate Entity Set** | **Candidate Attribute** | **Candidate Relationship** |
|---|---|---|
| Project | project_id<br>project_name<br>project_desc<br>client_id<br>manager_id<br>start_date<br>due_date | Project is **divided into** tasks<br>Project **has** a project manager |
| Task | task_id<br>project_id<br>start_date<br>due_date<br>task_name<br>Task_category | Task **has** task updates |

| | | |
|---|---|---|
| Admin | Admin_id<br>manager_id<br>project_id<br>first_name<br>last_name<br>email | Admin **assigns** a project manager<br>Admin **validates** project |
| Project Manager | manager_id<br>project_id<br>first_name<br>last_name<br>email | The project manager **is** an employee<br>Project manager **accesses** project |
| Team Leader | team_lead_id<br>first_name<br>last_name<br>email<br>task_id<br>manager_id | Team leader **reviews** team member's work<br>Team lead **checks** tasks' status<br>Team lead **completes** tasks<br>Team lead **reports** project manager |
| Team Member | Id<br>task_id<br>first_name<br>last_name<br>email<br>team_lead_id | Team member **accesses** task |
| Tester | id<br>task_id<br>first_name<br>last_name | The tester can **test** tasks<br>Tester **updates** tasks' status |
| Task Update | task_id<br>task_name<br>last_updated<br>update_info | |
| Employee | employee_id<br>designation<br>first_name<br>last_name | Admin, Project Manager, Team Leader, Team members, and tester are employees. |
| Client | client_id<br>first_name<br>last_name<br>comp_name | Client **requests** projects. |

## Table-3 (Rejected Nouns - ER Model)

| **Noun** | **Reject Reason** |
|---|---|
| Task | Duplicate |
| Management | irrelevant |
| aspect | vague |

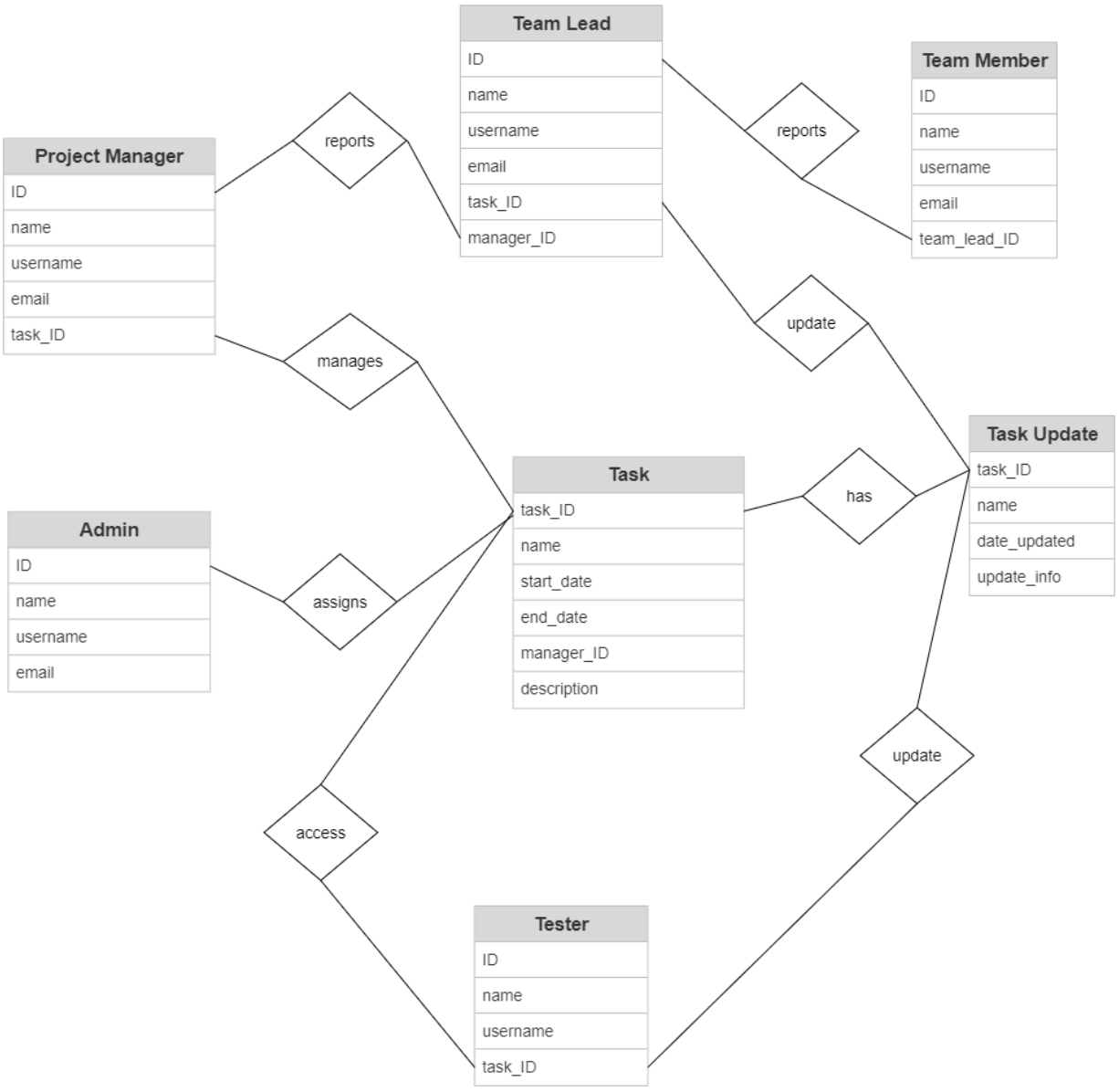
| | |
|---|---|
| project | duplicate |
| structure | irrelevant |
| portion | vague |
| to-do lists | attribute |
| checklists | vague |
| kanban boards | irrelevant |
| amount | attribute |
| data | attribute |
| tools | general |
| work | vague |
| spread | irrelevant |
| survey | vague |
| leaders | irrelevant |
| cause | vague |
| failure | vague |
| organization | general |
| lack | vague |
| objectives | vague |
| milestone | general |
| progress | association |
| discipline | vague |
| strategy | general |
| purpose | vague |
| issues | vague |
| goals | attribute |
| milestones | general |
| deadlines | attribute |
| deliverables | irrelevant |
| task lists | general |
| subtasks | duplicate |
| system | general |
| reminders | attribute |
| clients | duplicate |
| improvisation | vague |
| workflow | vague |
| efficiency | attribute |
| software | duplicate |

| | |
|---|---|
| jira | duplicate |
| asana | duplicate |
| zoho projects | duplicate |
| examples | general |
| features | attribute |
| agile | vague |
| roadmap | association |
| scrum boards | vague |
| Gantt chart | irrelevant |
| dependencies | vague |
| calendar | attribute |
| automation | irrelevant |
| websites | vague |
| technologies | general |
| absence | irrelevant |
| software | duplicate |
| workflow | vague |
| client | attribute |
| requirements | general |
| workforce | general |
| Team report | attribute |
| users | duplicate |
| administration | duplicate |
| project | duplicate |
| product | vague |
| company | general |
| audience | irrelevant |
| scale | attribute |
| premium | attribute |
| enterprise | irrelevant |
| agencies | general |
| government | vague |
| domains | attribute |
| industries | general |
| data | vague |
| solution | vague |
| information | duplicate |

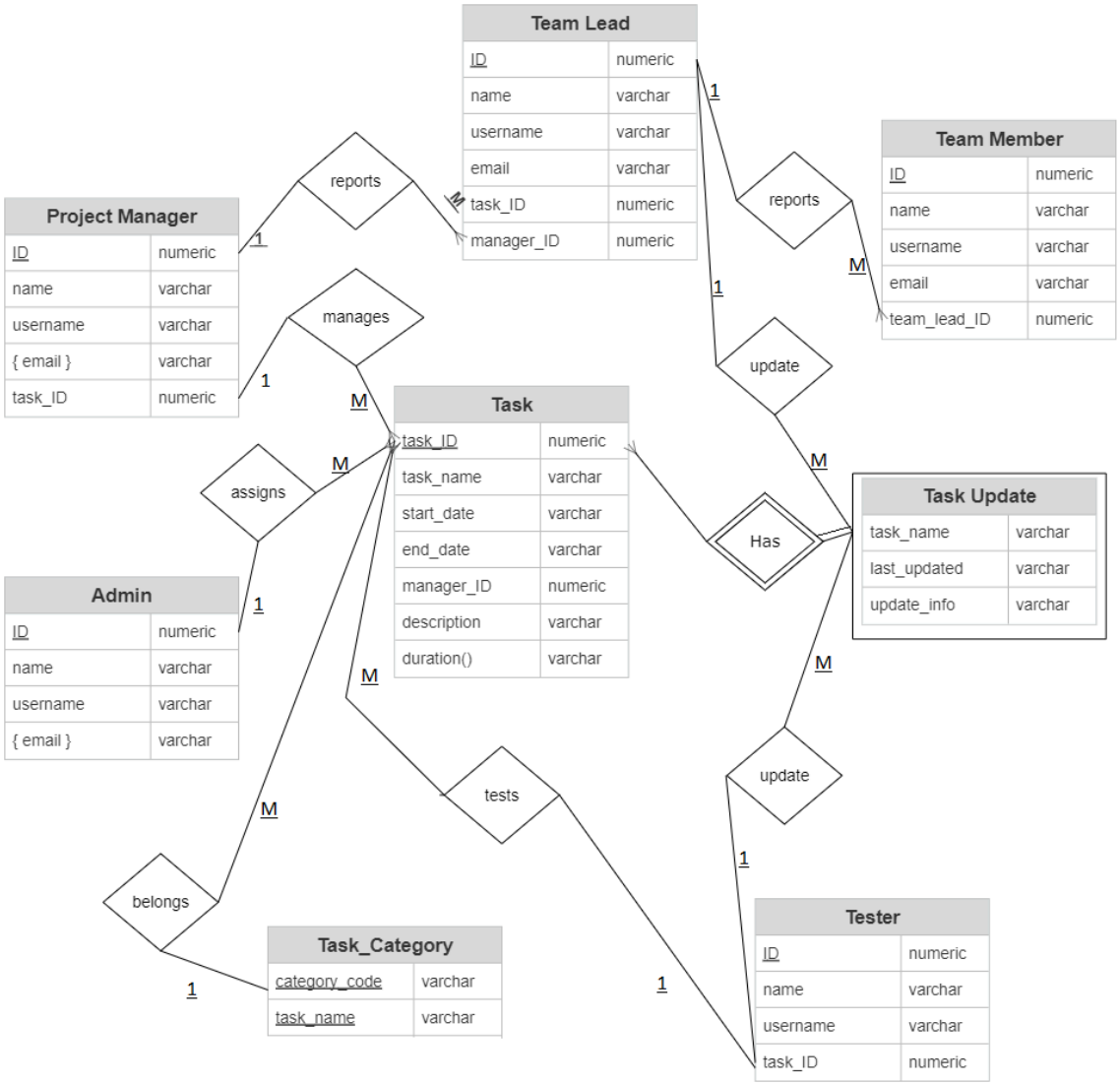
| | |
|---|---|
| privileges | duplicate |
| individuals | general |
| versions | irrelevant |
| Management | irrelevant |
| aspect | vague |
| checklist | attribute |
| To-do list | attribute |
| calendars | attribute |
| product | duplicate |
| work | vague |
| Progress | association |
| relationship | vague |
| Developers | irrelevant |
| features | attribute |
| tasks | duplicate |
| work | general |
| notifications | vague |
| Kanban boards | attribute |
| amounts | vague |
| tools | vague |
| survey | vague |
| leaders | duplicate |
| cause | vague |

# Section3: ER diagrams

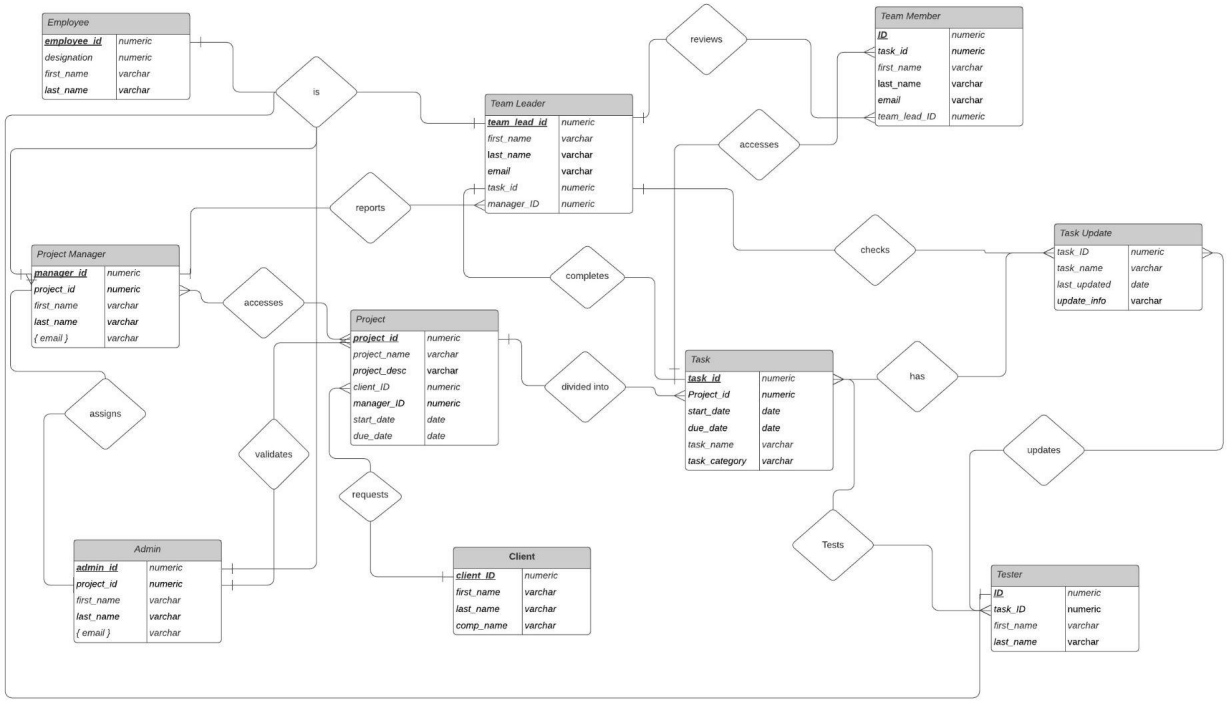
# ER Model (Version 1)



# ER Model (Version 2)



# Final ER Model



# Section4: ER to Relational Mapping

## Mapping ER model to Relational Model:

**Project(<u>project_id</u>**, project_name, project_desc, client_ID, manager_ID, start_date, due_date)

**Task(<u>task_id</u>**, project_id, start_date, due_date, task_name, task_category)

**Client(<u>client_id</u>**, first_name, last_name, comp_name)

**Employee(<u>employee_id</u>**, designation, first_name, last_name)

**Admin(<u>admin_id</u>**, project_id, first_name, last_name)

**Project_Manager(<u>manager_id,</u>** project_id, first_name, last_name**)**

**Team_Leader(<u>team_lead_id</u>**, first_name, last_name, task_id, manager_id**)**

**Team_Member(<u>id</u>**, task_id, first_name, last_name, team_lead_id)

**Tester(<u>id,</u>** task_id, first_name, last_name**)**

**Task_Update(<u>task_id,</u>** task_name, last_updated, update_info**)**

**Email(<u>id</u>, <u>email_id</u>)**

## DDL Scripts:

**<u>Project:</u>**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Project(
        project_id          character varying(20),
        project_desc        character varying(255),
        project_name        character varying(30),
        client_ID           character varying(20),
        manager_ID          character varying(20),
        start_date          date,
        due_date            date,

        PRIMARY KEY (project_id),

        FOREIGN KEY (manager_ID)
        REFERENCES "S10_T3".Project_manager(manager_ID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
```

```
        FOREIGN KEY (client_ID)
        REFERENCES “S10_T3”.Client(client_ID)
        ON UPDATE CASCADE
        ON DELETE CASCADE

);
```

**Task:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Task(
        task_id                 character varying(20),
        project_id               character varying(20)
        start_date               date,
        due_date                 date,
        task_name                character varying(30),
        task_category            character varying(25),

        PRIMARY KEY (task_id),
        FOREIGN KEY (project_id)
        REFERENCES "S10_T3".Project (project_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

**Client:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Client(
        client_id                character varying(20),
        first_name               character varying(30),
        last_name                character varying(30),
        comp_name           character varying(100),

        PRIMARY KEY (client_id)
);
```

**Employee:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Employee(
        employee_id         character varying(20),
        designation          character varying 30),
        first_name           character varying (30),
        last_name            character varying (30),

      PRIMARY KEY (employee_id)
);
```

**Admin:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Admin(
        admin_id                character varying(20),
        project_id              character varying (20),
        first_name              character varying (30),
        last_name               character varying (30),

        CONSTRAINT  unique_proj UNIQUE(project_id)

        PRIMARY KEY (admin_id),

        FOREIGN KEY (project_id)
        REFERENCES "S10_T3".Project (project_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE


);
```

**Project Manager:**

```
CREATE TABLE IF NOT EXISTS “S10_T3”.Project_manager(
        manager_id              character varying(20),
        project_id              character varying(20),
        first_name              character varying(30),
        last_name               character varying(30),

        PRIMARY KEY (manager_id)
      FOREIGN KEY (project_id)
        REFERENCES "S10_T3".Project (project_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

**Team_Leader:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Team_Leader(
        team_lead_id            character varying(20),
        task_id                 character varying(20),
        first_name              character varying (30),
        last_name               character varying (30),
        manager_id              character varying(20),

        PRIMARY KEY (team_lead_id),
        FOREIGN KEY (manager_id) references “S10_T3”.project_manager (manager_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
```

```
        FOREIGN KEY (task_id) references “S10_T3”.Task (task_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE

);
```

**Team_Member:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Team_Member(
        id                                  character varying(20),
        task_id                             character varying(20),
        first_name                          character varying (30),
        last_name                           character varying (30),
        team_lead_id                        character varying(20),

        PRIMARY KEY (id),
        FOREIGN KEY (team_lead_id) references “S10_T3”.Team_Leader (team_lead_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE

        FOREIGN KEY (task_id) references “S10_T3”.Task (task_id)
      ON DELETE CASCADE
      ON UPDATE CASCADE

);
```

**Tester:**

```
CREATE TABLE IF NOT EXISTS “S10_T3”.Tester(
        id                          character varying(20),
        task_id                     character varying(20),
        first_name                  character varying(30),
        last_name                   character varying(30),

        PRIMARY KEY (id),
        FOREIGN KEY (task_id)
        REFERENCES “S10_T3”.Task(task_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE

);
```

**Task Update:**

```
CREATE TABLE IF NOT EXISTS “S10_T3”.Task_update(
        task_id                     character varying(20),
```

```
        task_name               character varying(30),
        last_updated            date,
        update_info             text,

        PRIMARY KEY (task_id)
);
```

**Email:** (multivalued attribute)

```
CREATE TABLE IF NOT EXISTS "S10_T3".Email(
        id                      character varying(20),
        email                   character varying(100),

        PRIMARY KEY (id),
        PRIMARY KEY (email)
);
```

# Section 5: Normalization and Schema Refinement

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

**Project(<u>project_id</u>**, project_name, project_desc, client_ID, manager_ID, start_date, due_date)

**Task(<u>task_id</u>**, project_id, start_date, due_date, task_name, task_category)

**Client(<u>client_id</u>**, first_name, last_name, comp_name)

**Employee(<u>employee_id</u>**, designation, first_name, last_name)

**Admin(<u>admin_id</u>**, project_id, first_name, last_name)

**Project_Manager(<u>manager_id,</u>** project_id, first_name, last_name**)**

**Team_Leader(<u>team_lead_id</u>**, first_name, last_name, task_id, manager_id**)**

**Team_Member(<u>id</u>**, task_id, first_name, last_name, team_lead_id)

**Tester(<u>id,</u>** task_id, first_name, last_name**)**

**Task_Update(<u>task_id,</u>** task_name, last_updated, update_info**)**

**Email(<u>id</u>, <u>email_id</u>)**

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

| Relation | Functional Dependencies |
|---|---|
| Project | Project_id -> project_desc,<br>Project_id -> project_name,<br>Project_id -> client_ID,<br>Project_id -> manager_ID,<br>Project_id -> start_date,<br>Project_id -> due_date |
| Task | Task_id -> project_id,<br>Task_id -> start_date,<br>Task_id -> due_date,<br>Task_id -> task_name,<br>Task_id -> task_category |

| | |
|---|---|
| Client | Client_id -> first_name<br>Client_id -> last_name<br>Client_id -> comp_name |
| Employee | Employee_id -> first_name<br>Employee_id -> last_name<br>Employee_id -> designation |
| Admin | Admin_id -> project_id<br>Admin_id -> first_name<br>Admin_id -> last_name<br>Admin_id -> email |
| Project Manager | Manager_id -> project_id<br>Manager_id -> first_name<br>Manager_id -> last_name<br>Manager_id -> email |
| Team Leader | team_lead_id -> first_name<br>team_lead_id -> last_name<br>team_lead_id -> task_id<br>team_lead_id -> manager_id<br>team_lead_id -> email |
| Team Member | id -> task_id,<br>id -> first_name,<br>id -> last_name,<br>id -> Team_lead_id,<br>id -> email |
| Tester | Id -> task_id,<br>Id -> first_name,<br>Id -> last_name |
| Task Update | task_id -> task_name<br>task_id -> last_updated<br>task_id -> update_info |

## Normalization up to 1NF:

The relations:
Project, Task, Client, Employee, and Task Update are already in 1NF because all the key attributes are defined. All the non-key attributes depend on the primary key. No attributes are either composite or multi-valued.

For Admin, the attributes project_id, and email are multi-valued.
For Project Manager, the attribute project_id, and email are multi-valued.
For Team leaders, email is a multivalued attribute.
For Team members, email is a multivalued attribute.
For Tester, the attribute task_id is multi-valued.

## Documentation:

For email we created a different table that holds the emails of all the employees of the company having a composite primary key of employee_id and email, hence making it 1NF.

To make the Admin table 1NF, we divided it into two tables admin_details and admin_project. In the admin_project table, both admin_id and project_id constitute the primary key, whereas the admin_details table satisfies all constraints for 1NF.

As project_id in Project Manager was multi-valued in the same way as Admin, we created the table manager_details and project_manager similar to admin_details and admin_project. In the project_manager table, manager_id, and project_id, both are part of the primary key.

To make the Tester table 1NF we divided it into two tables tester_details (similar to admin_details) and tester (similar to admin_project table).

To make the Task table 1NF we divided it into two tables: task, and project_task . In project_task, both project_id, and task_id are a part of the primary key. We did this because we had multiple tasks for one project.

For Team leaders and team members, email was a multivalued attribute, and we made a separate table for that.

## Normalization up to 2NF:

All our relations are in 1NF and every non-PK attribute is fully dependent on the PK, so all our relations are already in 2 NF.

## Normalization up to 3NF:

All our relations are in 1NF and 2 NF and no non-PK attribute is transitively dependent on the PK, so all relations are already in 3 NF.

## Normalization up to BCNF:

All our relations have only 1CK which is our PK, so they are already in BCNF.

## Anomalies:

PROJECT: Manager_id and Client_id should be inserted in Project_manager and client tables respectively, before inserting them in the Project table.
When either manager_id or client_id is deleted from their schemas, the related tuple is deleted from the project table as well.

TEAM LEADER: Task_id and manager_id should be inserted in Task and manager_details tables, before inserting a row in the team_leader table.
If task_id or manager_id is deleted from task and manager_details table, the respective tuple from the team_leader schema is also deleted.

TEAM MEMBER: Task_id and team_lead_id must be inserted in Task and Team_Leader relations, before inserting in the team_member table.
When task_id and team_lead_id are deleted, the tuple from the team_member table is also deleted.

# Final Relations and Schemas

**Project_details(project_id**, project_desc, project_name)

**Project(project_id**, client_ID, manager_ID, start_date, due_date)

**Task(task_id,** start_date, due_date, task_name, task_category)

**Project_Task(project_id, task_id)**

**Client(client_id,** first_name, last_name, comp_name)

**Employee(employee_id**, designation, first_name, last_name)

**Admin_details(admin_id,** first_name, last_name)

**Admin_project(admin_id, project_id)**

**Manager_details(manager_id,** first_name, last_name)

**Project_Manager(manager_id, project_id)**

**Team_Leader(team_lead_id**, first_name, last_name, task_id, manager_id)

**Team_Member(id**, task_id, first_name, last_name, team_lead_id)

**Tester_details(id,** first_name, last_name)

**Tester(id, task_id)**

**Task_Update(task_id**, task_name, last_updated, update_info)

**Email(employee_id, email_id)**

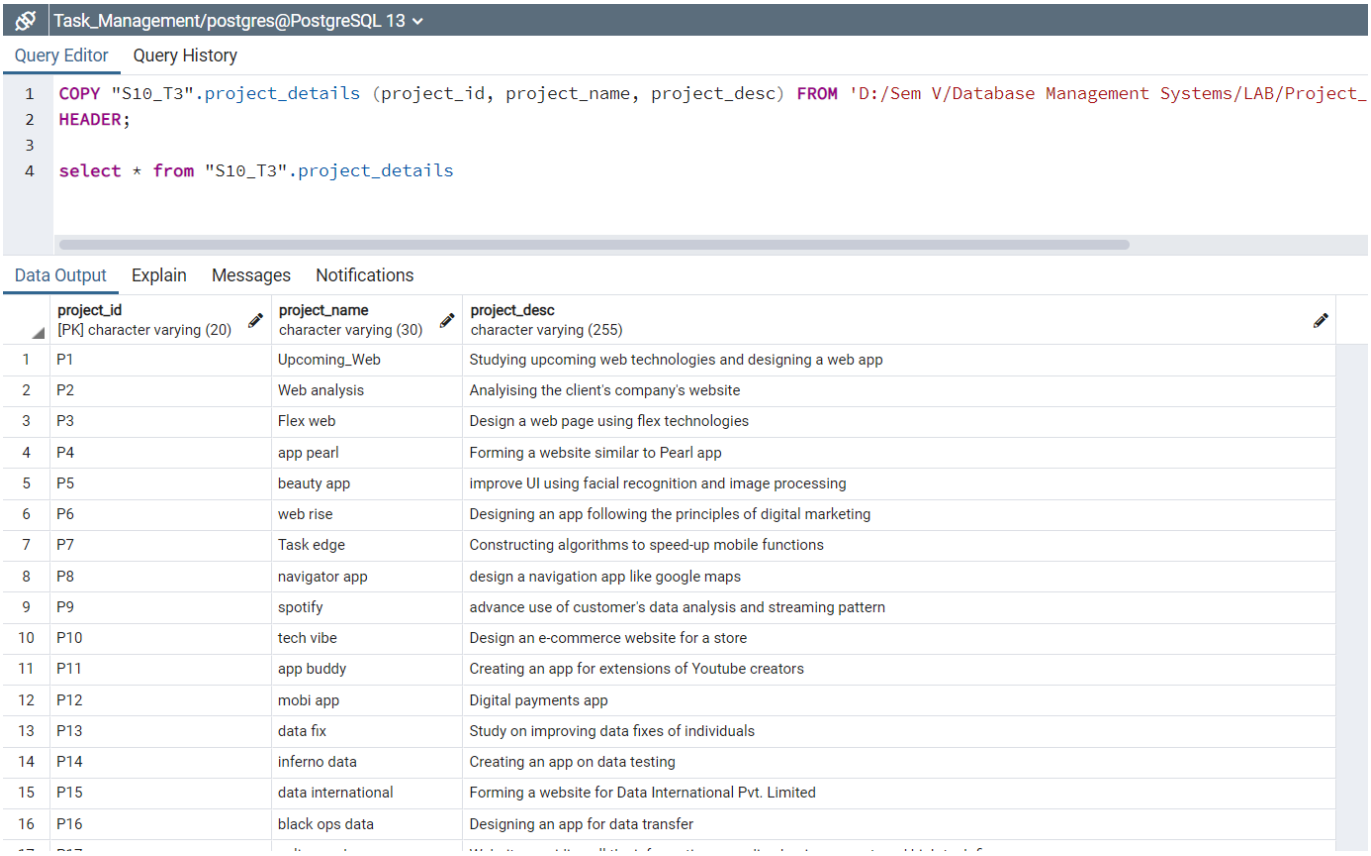
# Section6: SQL: Final DDL Scripts, Insert statements, 40 SQL Queries with Snapshots of the output of each query.

# DDL Scripts and Snapshots of Tables

**Project Details:**

**CREATE TABLE IF NOT EXISTS "S10_T3".Project_details(**
project_id character varying(20),
project_name character varying(30),
project_desc character varying(255),

PRIMARY KEY (project_id),
);




| | project_id [PK] character varying (20) | project_name character varying (30) | project_desc character varying (255) |
|---|---|---|---|
| 1 | P1 | Upcoming_Web | Studying upcoming web technologies and designing a web app |
| 2 | P2 | Web analysis | Analysing the client's company's website |
| 3 | P3 | Flex web | Design a web page using flex technologies |
| 4 | P4 | app pearl | Forming a website similar to Pearl app |
| 5 | P5 | beauty app | improve UI using facial recognition and image processing |
| 6 | P6 | web rise | Designing an app following the principles of digital marketing |
| 7 | P7 | Task edge | Constructing algorithms to speed-up mobile functions |
| 8 | P8 | navigator app | design a navigation app like google maps |
| 9 | P9 | spotify | advance use of customer's data analysis and streaming pattern |
| 10 | P10 | tech vibe | Design an e-commerce website for a store |
| 11 | P11 | app buddy | Creating an app for extensions of Youtube creators |
| 12 | P12 | mobi app | Digital payments app |
| 13 | P13 | data fix | Study on improving data fixes of individuals |
| 14 | P14 | inferno data | Creating an app on data testing |
| 15 | P15 | data international | Forming a website for Data International Pvt. Limited |
| 16 | P16 | black ops data | Designing an app for data transfer |

**Project:**

**CREATE TABLE IF NOT EXISTS "S10_T3".Project(**

```
        project_id          character varying(20),
        client_ID           character varying(20),
        manager_ID          character varying(20),
        start_date          date,
        due_date            date,

        PRIMARY KEY (project_id),

        FOREIGN KEY (manager_ID)
        REFERENCES "S10_T3".Manager_details(manager_ID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,

        FOREIGN KEY (client_ID)
        REFERENCES "S10_T3".Client(client_ID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
```

);

Query Editor | Query History

```
1  COPY "S10_T3".project(project_id,client_id,manager_id,start_date,due_date) FROM 'E:/DBMS LAB DAIICT/project.csv' DEL
2  HEADER;
3
4  select * from project
5
```

Explain | Messages | Notifications | Data Output

| | project_id [PK] character varying (20) | client_id character varying (20) | manager_id character varying (20) | start_date date | due_date date |
|---|---|---|---|---|---|
| 1 | P1 | CL1 | PM1 | 2021-05-07 | 2021-05-13 |
| 2 | P2 | CL2 | PM2 | 2021-07-02 | 2021-08-20 |
| 3 | P3 | CL3 | PM3 | 2020-01-06 | 2020-01-04 |
| 4 | P4 | CL4 | PM4 | 2020-02-04 | 2020-02-26 |
| 5 | P5 | CL5 | PM5 | 2019-11-13 | 2019-12-24 |
| 6 | P6 | CL6 | PM6 | 2020-05-26 | 2020-06-03 |
| 7 | P7 | CL7 | PM7 | 2020-11-15 | 2020-12-31 |
| 8 | P8 | CL8 | PM8 | 2020-09-30 | 2020-10-16 |
| 9 | P9 | CL9 | PM9 | 2020-03-20 | 2020-04-21 |
| 10 | P10 | CL10 | PM10 | 2021-04-16 | 2021-05-03 |
| 11 | P11 | CL11 | PM11 | 2020-05-25 | 2020-06-08 |
| 12 | P12 | CL12 | PM12 | 2021-02-26 | 2021-04-04 |
| 13 | P13 | CL13 | PM13 | 2021-01-23 | 2021-02-09 |
| 14 | P14 | CL14 | PM14 | 2020-01-03 | 2020-01-25 |

**Task:**

**CREATE TABLE IF NOT EXISTS "S10_T3".Task(**

```
    task_id             character varying (20),
    start_date                  date,
    due_date                    date,
    task_name                   character varying(30),
    task_category       character varying(30),

    PRIMARY KEY (task_id)
);
```

Query Editor | Query History

```
COPY "S10_T3".task (task_id,start_date,due_date,task_name,task_category) FROM 'E:/DBMS LAB DAIICT/task.csv' DELIMITER
HEADER;

select * from task;
```

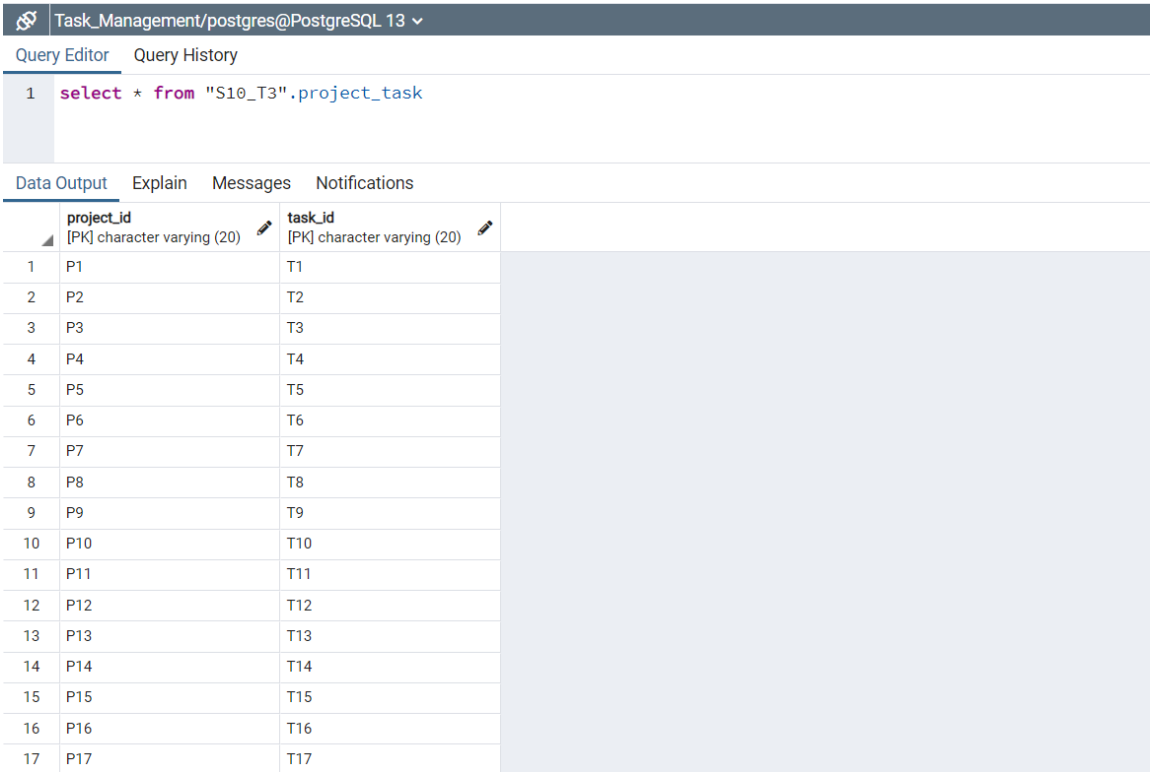
Explain | Messages | Notifications | Data Output

| | task_id [PK] character varying (20) | start_date date | due_date date | task_name character varying (30) | task_category character varying (30) |
|---|---|---|---|---|---|
| 1 | T1 | 2020-09-04 | 2020-10-23 | Otcom | testing |
| 2 | T2 | 2020-02-22 | 2020-03-05 | Stronghold | testing |
| 3 | T3 | 2020-11-03 | 2020-11-17 | Tresom | data visualization |
| 4 | T4 | 2021-08-05 | 2021-08-03 | Vagram | invoice |
| 5 | T5 | 2020-01-04 | 2020-02-28 | Vagram | testing |
| 6 | T6 | 2021-02-09 | 2021-03-04 | Tresom | deployment |
| 7 | T7 | 2020-05-08 | 2020-06-10 | Opela | statistics |
| 8 | T8 | 2019-11-20 | 2019-12-30 | Bitwolf | statistics |
| 9 | T9 | 2021-01-14 | 2021-01-17 | Hatity | website hosting |
| 10 | T10 | 2021-03-29 | 2021-04-30 | Overhold | deployment |
| 11 | T11 | 2021-02-20 | 2021-03-24 | Job | statistics |
| 12 | T12 | 2020-09-19 | 2020-10-11 | Flowdesk | data visualization |
| 13 | T13 | 2019-11-14 | 2019-11-06 | Aerified | back end |
| 14 | T14 | 2020-09-21 | 2020-10-14 | Gembucket | data visualization |

**Project_Task:**

**CREATE TABLE IF NOT EXISTS "S10_T3".Project_task(**

```
    project_id              character varying (20),
    task_id                 character varying (20),

    PRIMARY KEY (project_id, task_id)
);
```




| | project_id [PK] character varying (20) | task_id [PK] character varying (20) |
|---|---|---|
| 1 | P1 | T1 |
| 2 | P2 | T2 |
| 3 | P3 | T3 |
| 4 | P4 | T4 |
| 5 | P5 | T5 |
| 6 | P6 | T6 |
| 7 | P7 | T7 |
| 8 | P8 | T8 |
| 9 | P9 | T9 |
| 10 | P10 | T10 |
| 11 | P11 | T11 |
| 12 | P12 | T12 |
| 13 | P13 | T13 |
| 14 | P14 | T14 |
| 15 | P15 | T15 |
| 16 | P16 | T16 |
| 17 | P17 | T17 |

**Client:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Client(
        client_id               character varying(20),
        first_name              character varying(30),
        last_name               character varying(30),
        comp_name       character varying(100),

        PRIMARY KEY (client_id)
);
```

Query Editor | Query History

```
1  COPY "S10_T3".client (client_id,first_name,last_name,comp_name) FROM 'E:/DBMS LAB DAIICT/client.csv' DELIMITER ',' CSV
2  HEADER;
3
4  select * from client;
```

Explain | Messages | Notifications | Data Output

| | client_id [PK] character varying (20) | first_name character varying (30) | last_name character varying (30) | comp_name character varying (100) |
|---|---|---|---|---|
| 1 | CL1 | Judd | Weighell | Goldner and Sons |
| 2 | CL2 | Lou | Blomefield | Hackett, Labadie and Marks |
| 3 | CL3 | Baudoin | Zorener | Schimmel, Feest and Heaney |
| 4 | CL4 | Binky | Bryan | Runte, Weber and Cremin |
| 5 | CL5 | Vin | MacTague | Kris-Batz |
| 6 | CL6 | Baryram | Lumsden | Bradtke LLC |
| 7 | CL7 | Sande | Krollman | Balistreri-Dibbert |
| 8 | CL8 | Lynea | Morbey | Hand, Langosh and Schimmel |
| 9 | CL9 | Lexis | Shelley | Johnston, Willms and Marvin |
| 10 | CL10 | Roz | Bedbrough | Bernier Group |
| 11 | CL11 | Marietta | Dinesen | Shields-Bartell |
| 12 | CL12 | Sherwin | Nuzzetti | Hahn-Gleichner |

**Employee:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Employee(
        employee_id     character varying(20),
        designation             character varying (30),
        first_name              character varying (30),
        last_name               character varying (30),

    PRIMARY KEY (employee_id)
);
```

Query Editor | Query History

```
1  COPY "S10_T3".employee (employee_id,designation,first_name,last_name) FROM 'E:/DBMS LAB DAIICT/employee.csv' DELIMIT
2  HEADER;
3
4  select * from employee
5
6
```

Explain | Messages | Notifications | Data Output

| | employee_id [PK] character varying (20) | designation character varying (30) | first_name character varying (30) | last_name character varying (30) |
|---|---|---|---|---|
| 1 | AD1 | Admin | Kienan | Gosse |
| 2 | AD2 | Admin | Marice | Matthaus |
| 3 | AD3 | Admin | Harwell | Westfield |
| 4 | AD4 | Admin | Shandee | Frangione |
| 5 | AD5 | Admin | Rutherford | Cheevers |
| 6 | AD6 | Admin | Margaretha | Ebenezer |
| 7 | AD7 | Admin | Fabio | Trevor |
| 8 | AD8 | Admin | Doug | Gillard |
| 9 | AD9 | Admin | Dianne | Maasz |
| 10 | AD10 | Admin | Lester | Giral |
| 11 | AD11 | Admin | Krystle | Pinching |
| 12 | AD12 | Admin | Dee dee | Barnard |
| 13 | AD13 | Admin | Catrina | Noel |
| 14 | AD14 | Admin | Johnathan | Charer |

**Admin_details:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Admin_details(
        admin_id            character varying (20),
        first_name          character varying (30),
        last_name           character varying (30),

    PRIMARY KEY (admin_id)
);
```

Query Editor  Query History

```
1  COPY "S10_T3".admin_details (admin_id,first_name,last_name) FROM 'E:/DBMS LAB DAIICT/admin_details.csv' DELIMITER ',
2  HEADER;
3
4  select * from admin_details
5
6
```

Explain  Messages  Notifications  Data Output

| | admin_id [PK] character varying (20) | first_name character varying (30) | last_name character varying (30) |
|---|---|---|---|
| 1 | AD1 | Kienan | Gosse |
| 2 | AD2 | Marice | Matthaus |
| 3 | AD3 | Harwell | Westfield |
| 4 | AD4 | Shandee | Frangione |
| 5 | AD5 | Rutherford | Cheevers |
| 6 | AD6 | Margaretha | Ebenezer |
| 7 | AD7 | Fabio | Trevor |
| 8 | AD8 | Doug | Gillard |
| 9 | AD9 | Dianne | Maasz |
| 10 | AD10 | Lester | Giral |
| 11 | AD11 | Krystle | Pinching |
| 12 | AD12 | Dee dee | Barnard |
| 13 | AD13 | Catrina | Noel |
| 14 | AD14 | Johnathan | Charer |

**Admin_Project:**

**CREATE TABLE IF NOT EXISTS "S10_T3".Admin_project(**

admin_id character varying (20),
project_id character varying (20),

PRIMARY KEY (admin_id, project_id)
**);**

Query Editor | Query History

```
1  COPY "S10_T3".admin_project (admin_id,project_id) FROM 'E:/DBMS LAB DAIICT/admin_project.csv' DELIMITER ',' CSV
2  HEADER;
3
4  select * from admin_project;
5
6
```

Explain | Messages | Notifications | Data Output

| | admin_id [PK] character varying (20) | project_id [PK] character varying (20) |
|---|---|---|
| 1 | AD1 | P1 |
| 2 | AD2 | P2 |
| 3 | AD3 | P3 |
| 4 | AD4 | P4 |
| 5 | AD5 | P5 |
| 6 | AD6 | P6 |
| 7 | AD7 | P7 |
| 8 | AD8 | P8 |
| 9 | AD9 | P9 |
| 10 | AD10 | P10 |
| 11 | AD11 | P11 |
| 12 | AD12 | P12 |
| 13 | AD13 | P13 |
| 14 | AD14 | P14 |

**Manager_details:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Manager_details(
        manager_id          character varying (20),
        first_name          character varying (30),
        last_name           character varying (30),

    PRIMARY KEY (manager_id),
);
```

Query Editor | Query History

```
1  COPY "S10_T3".manager_details (manager_id,first_name,last_name) FROM 'E:/DBMS LAB DAIICT/manager_details.csv' DELIMI
2  HEADER;
3
4  select * from manager_details
5
6
```

Explain | Messages | Notifications | Data Output

| | manager_id [PK] character varying (20) | first_name character varying (30) | last_name character varying (30) |
|---|---|---|---|
| 1 | PM1 | Janifer | Osipenko |
| 2 | PM2 | Ronnie | Saywell |
| 3 | PM3 | Jean | Kovelmann |
| 4 | PM4 | Alberik | Jowitt |
| 5 | PM5 | Jillana | Jeavons |
| 6 | PM6 | Hunt | Raynes |
| 7 | PM7 | Flora | Tersay |
| 8 | PM8 | Berti | McGrady |
| 9 | PM9 | Rickie | Gawkroge |
| 10 | PM10 | Merell | Carl |
| 11 | PM11 | Briggs | Kulic |
| 12 | PM12 | Sax | Ivushkin |
| 13 | PM13 | Con | Helgass |
| 14 | PM14 | Elfrieda | Spens |

**Project_Manager:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Project_manager(
        manager_id              character varying (20),
        project_id              character varying (20),
        PRIMARY KEY (manager_id, project_id)
);
```

Query Editor | Query History

```
1  COPY "S10_T3".project_manager (manager_id,project_id) FROM 'E:/DBMS LAB DAIICT/project_manager.csv' DELIMITER ',' CSV
2  HEADER;
3
4  select * from project_manager;
```

Explain | Messages | Notifications | Data Output

| | manager_id [PK] character varying (20) | project_id [PK] character varying (20) |
|---|---|---|
| 1 | PM1 | P1 |
| 2 | PM2 | P2 |
| 3 | PM3 | P3 |
| 4 | PM4 | P4 |
| 5 | PM5 | P5 |
| 6 | PM6 | P6 |
| 7 | PM7 | P7 |
| 8 | PM8 | P8 |
| 9 | PM9 | P9 |
| 10 | PM10 | P10 |
| 11 | PM11 | P11 |
| 12 | PM12 | P12 |

**Team_Leader:**

**CREATE TABLE IF NOT EXISTS "S10_T3".Team_Leader(**

team_lead_id character varying (20),
first_name character varying (30),
last_name character varying (30),
task_id character varying (20),
manager_id character varying (20),

PRIMARY KEY (team_lead_id),

FOREIGN KEY (manager_id) references “S10_T3”.manager_details (manager_id)
ON DELETE CASCADE
ON UPDATE CASCADE,

FOREIGN KEY (task_id) references “S10_T3”.Task (task_id)
ON DELETE CASCADE
ON UPDATE CASCADE

**);**

Query Editor  Query History

```
1  COPY "S10_T3".team_leader(team_lead_id,first_name,last_name,task_id,manager_id) FROM 'E:/DBMS LAB DAIICT/team_leader
2  HEADER;
3
4  select * from team_leader
5
```

Explain  Messages  Notifications  Data Output

| | team_lead_id [PK] character varying (20) | task_id character varying (20) | first_name character varying (30) | last_name character varying (30) | manager_id character varying (20) |
|---|---|---|---|---|---|
| 1 | TL1 | T1 | Waverley | Laxton | PM1 |
| 2 | TL2 | T2 | Carl | Mioni | PM2 |
| 3 | TL3 | T3 | Chilton | Bellanger | PM3 |
| 4 | TL4 | T4 | Nicolai | Wain | PM4 |
| 5 | TL5 | T5 | Lars | Pol | PM5 |
| 6 | TL6 | T6 | Tabbie | Danielovitch | PM6 |
| 7 | TL7 | T7 | Roselle | Bisatt | PM7 |
| 8 | TL8 | T8 | Wendi | Selman | PM8 |
| 9 | TL9 | T9 | Gabrielle | Maudlen | PM9 |
| 10 | TL10 | T10 | Rodolphe | Moverley | PM10 |
| 11 | TL11 | T11 | Merwyn | Chittim | PM11 |
| 12 | TL12 | T12 | Timmy | Thor | PM12 |
| 13 | TL13 | T13 | Mateo | Oakenfull | PM13 |
| 14 | TL14 | T14 | Amil | Manjin | PM14 |

**Team_Member:**

```
CREATE TABLE IF NOT EXISTS "S10_T3".Team_Member(
        id                      character varying (20),
        task_id                 character varying (20),
        first_name              character varying (30),
        last_name               character varying (30),
        team_lead_id            character varying (20),

        PRIMARY KEY (id),

        FOREIGN KEY (team_lead_id) references “S10_T3”.Team_Leader (team_lead_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,

        FOREIGN KEY (task_id) references “S10_T3”.Task (task_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Query Editor  Query History

```
1  COPY "S10_T3".team_member(id,task_id,first_name,last_name,team_lead_id) FROM 'E:/DBMS LAB DAIICT/team_member.csv' DE
2  HEADER;
3
4  select * from team_member
5
6
```

Explain  Messages  Notifications  Data Output

| | id [PK] character varying (20) | task_id character varying (20) | first_name character varying (30) | last_name character varying (30) | team_lead_id character varying (20) |
|---|---|---|---|---|---|
| 1 | TM1 | T1 | Fran | Tooze | TL1 |
| 2 | TM2 | T2 | Milly | Duggon | TL2 |
| 3 | TM3 | T3 | Mehetabel | Culter | TL3 |
| 4 | TM4 | T4 | Neel | Erskine Sandys | TL4 |
| 5 | TM5 | T5 | Tasia | Ovens | TL5 |
| 6 | TM6 | T6 | Pepe | Wheeldon | TL6 |
| 7 | TM7 | T7 | Sebastian | Ceeley | TL7 |
| 8 | TM8 | T8 | Corri | Mordecai | TL8 |
| 9 | TM9 | T9 | Ardelle | Dwire | TL9 |
| 10 | TM10 | T10 | Daniel | Finders | TL10 |
| 11 | TM11 | T11 | Bartholomeus | Annandale | TL11 |
| 12 | TM12 | T12 | Essy | Glavin | TL12 |
| 13 | TM13 | T13 | Corette | Child | |
| 14 | TM14 | T14 | Ranna | Cossum | |

Successfully run. Total query runtime: 303 msec. 300 rows affected.

**Tester_details:**

**CREATE TABLE IF NOT EXISTS "S10_T3".Tester_details(**

id character varying (20),
first_name character varying(30),
last_name character varying(30),

PRIMARY KEY (id)
);

Query Editor | Query History

```
1  COPY "S10_T3".tester_details (id,first_name,last_name) FROM 'E:/DBMS LAB DAIICT/tester_details.csv' DELIMITER ',' CS
2  HEADER;
3
4  select * from tester_details
5
6
```

Explain | Messages | Notifications | Data Output

| | id [PK] character varying (20) | first_name character varying (30) | last_name character varying (30) |
|---|---|---|---|
| 1 | TS1 | Bonnibelle | Starford |
| 2 | TS2 | Gideon | Dradey |
| 3 | TS3 | Nalani | Yorke |
| 4 | TS4 | Dimitry | Lowman |
| 5 | TS5 | Matthus | Josipovic |
| 6 | TS6 | Blayne | Ewols |
| 7 | TS7 | Hilton | Rodrigues |
| 8 | TS8 | Harald | Hambers |
| 9 | TS9 | Edgardo | Playford |
| 10 | TS10 | Edik | Whitfield |
| 11 | TS11 | Skell | Farnell |
| 12 | TS12 | Benedicto | Lipmann |
| 13 | TS13 | Aleta | O'Logan |
| 14 | TS14 | Ansley | Matfin |

✔ Successfully run. Total query runtime: 107 msec. 60 rows affected.

**Tester:**

**CREATE TABLE IF NOT EXISTS "S10_T3".Tester(**

```
        id                      character varying (20),
        task_id         character varying(20),

        PRIMARY KEY (id, task_id)
);
```

Query Editor | Query History

```
1  COPY "S10_T3".tester(id,task_id) FROM 'E:/DBMS LAB DAIICT/tester.csv' DELIMITER ',' CSV
2  HEADER;
3
4  select * from tester
5
6
```

Explain | Messages | Notifications | Data Output

| | id [PK] character varying (20) | task_id [PK] character varying (20) |
|---|---|---|
| 1 | TS1 | T1 |
| 2 | TS2 | T2 |
| 3 | TS3 | T3 |
| 4 | TS4 | T4 |
| 5 | TS5 | T5 |
| 6 | TS6 | T6 |
| 7 | TS7 | T7 |
| 8 | TS8 | T8 |
| 9 | TS9 | T9 |
| 10 | TS10 | T10 |
| 11 | TS11 | T11 |
| 12 | TS12 | T12 |
| 13 | TS13 | T13 |
| 14 | TS14 | T14 |

**Task Update:**

**CREATE TABLE IF NOT EXISTS "S10_T3".Task_update(**

```
        task_id         character varying (20),
        task_name               character varying(30),
        last_updated    date,
        update_info             text,

        PRIMARY KEY (task_id)
);
```

Query Editor | Query History

```
1  COPY "S10_T3".task_update(task_id,task_name,last_updated,update_info) FROM 'E:/DBMS LAB DAIICT/task_update.csv' DELI
2  HEADER;
3
4  select * from task_update
5
6
```

Explain | Messages | Notifications | Data Output

| | task_id [PK] character varying (20) | task_name character varying (30) | last_updated date | update_info text |
|---|---|---|---|---|
| 1 | T1 | Otcom | 2020-09-05 | Passed |
| 2 | T2 | Stronghold | 2020-02-23 | Failed |
| 3 | T3 | Tresom | 2020-11-07 | Passed |
| 4 | T4 | Vagram | 2021-08-13 | Passed |
| 5 | T5 | Vagram | 2020-01-07 | Failed |
| 6 | T6 | Tresom | 2021-02-17 | Failed |
| 7 | T7 | Opela | 2020-05-11 | Failed |
| 8 | T8 | Bitwolf | 2019-11-27 | Failed |
| 9 | T9 | Hatity | 2021-01-18 | Failed |
| 10 | T10 | Overhold | 2021-04-05 | Failed |
| 11 | T11 | Job | 2021-02-22 | Passed |
| 12 | T12 | Flowdesk | 2020-09-22 | Failed |
| 13 | T13 | Aerified | 2019-11-22 | Failed |
| 14 | T14 | Gembucket | 2020-09-24 | Passed |

**Email:** (multivalued attribute)

```
CREATE TABLE IF NOT EXISTS "S10_T3".Email(
        employee_id             character varying (20),
        email                   character varying(100),

        PRIMARY KEY (employee_id, email)
);
```

Query Editor | Query History

```
1  COPY "S10_T3".email(employee_id,email) FROM 'E:/DBMS LAB DAIICT/email.csv' DELIMITER ',' CSV
2  HEADER;
3
4  select * from email
5
6
```

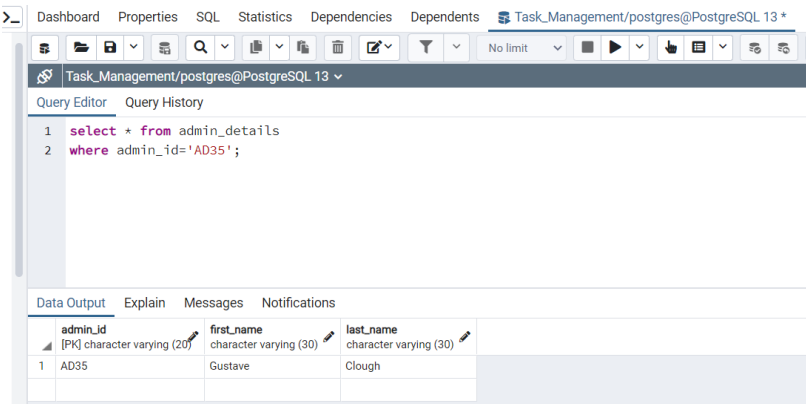
Explain | Messages | Notifications | Data Output

| | employee_id [PK] character varying (20) | email [PK] character varying (100) |
|---|---|---|
| 1 | AD1 | sdecleen0@pcworld.com |
| 2 | AD10 | rderl9@quantcast.com |
| 3 | AD10 | ssalerg2b@phpbb.com |
| 4 | AD11 | jbollera@vinaora.com |
| 5 | AD12 | ualywenb@google.es |
| 6 | AD13 | gthaxterc@barnesandnoble.com |
| 7 | AD14 | imanbyd@ucoz.com |
| 8 | AD15 | wvaldese@google.com.br |
| 9 | AD16 | aasletf@gnu.org |
| 10 | AD17 | dmcgradyg@booking.com |
| 11 | AD18 | eliddellh@cafepress.com |
| 12 | AD19 | dmcdougaldi@mayoclinic.com |
| 13 | AD2 | breast1@macromedia.com |
| 14 | AD20 | creediej@ibm.com |

# SQL Queries

1. **Select all the details of admin having ID ’AD35’:**

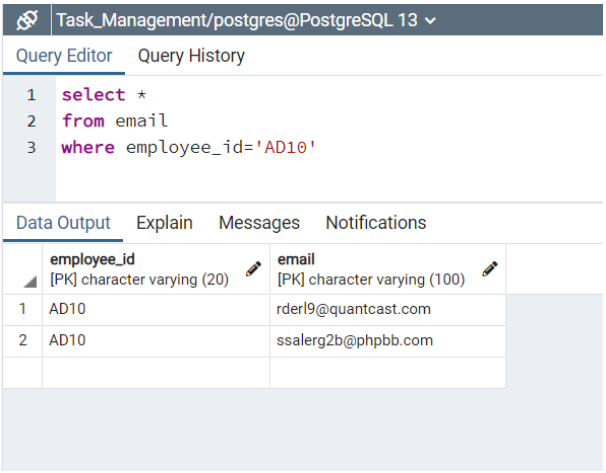
   **SQL:**
   select * from admin_details
   where admin_id='AD35';




   Tuples: 1

2. **Output emails of admin having ID ‘AD10’:**

   **SQL:**
   select *
   from email
   where employee_id='AD10'

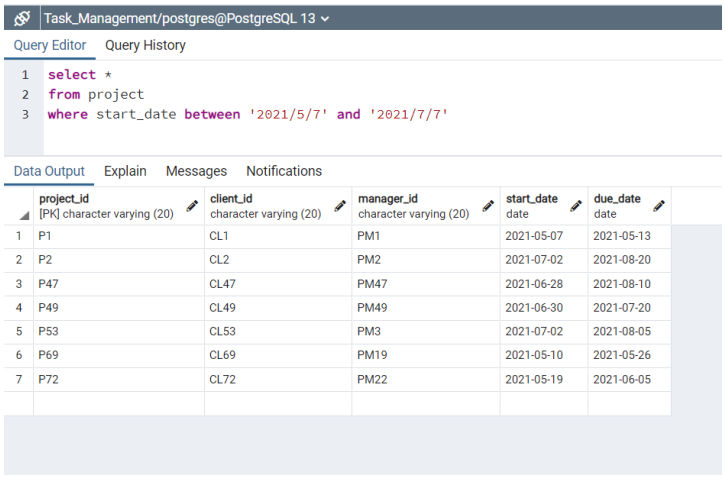



   Tuples: 2

3. **Select all the details from the project table where the start date for the project lies between 7th May 2021 and 7th July 2021.**

**SQL:**

```
select *
from project
where start_date between '2021/5/7' and '2021/7/7'
```



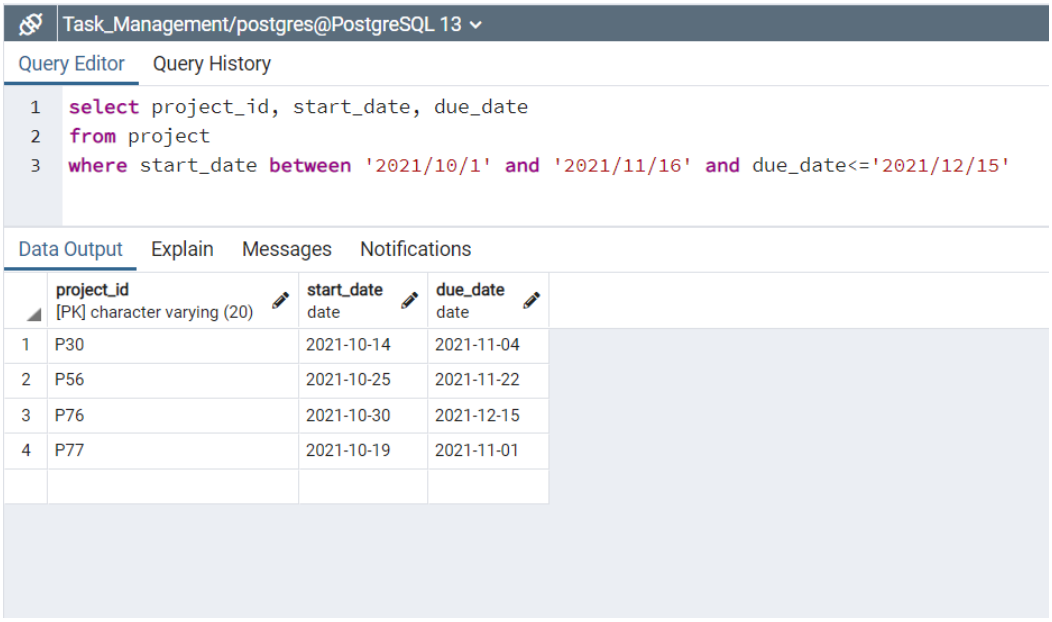

| | project_id [PK] character varying (20) | client_id character varying (20) | manager_id character varying (20) | start_date date | due_date date |
|---|---|---|---|---|---|
| 1 | P1 | CL1 | PM1 | 2021-05-07 | 2021-05-13 |
| 2 | P2 | CL2 | PM2 | 2021-07-02 | 2021-08-20 |
| 3 | P47 | CL47 | PM47 | 2021-06-28 | 2021-08-10 |
| 4 | P49 | CL49 | PM49 | 2021-06-30 | 2021-07-20 |
| 5 | P53 | CL53 | PM3 | 2021-07-02 | 2021-08-05 |
| 6 | P69 | CL69 | PM19 | 2021-05-10 | 2021-05-26 |
| 7 | P72 | CL72 | PM22 | 2021-05-19 | 2021-06-05 |

Tuples: 7

4. **Select project id, start date, due date from the project table where the start date for the project lies between 1st October 2021 and 16th November 2021, and the due_date is before 15th December 2021.**

**SQL:**

```
select project_id, start_date, due_date
from project
where start_date between '2021/10/1' and '2021/11/16' and due_date<='2021/12/15'
```



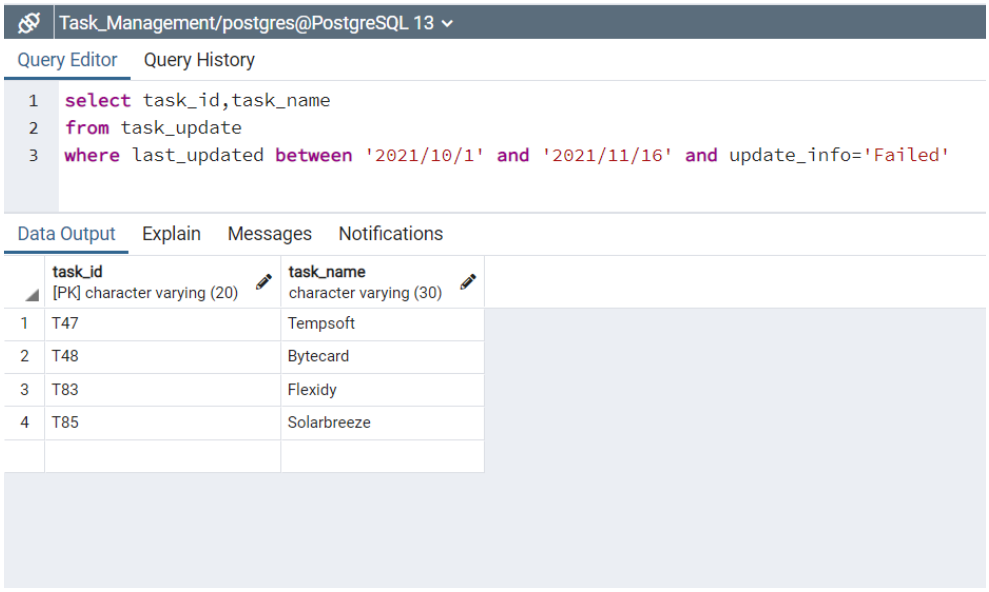

| | project_id [PK] character varying (20) | start_date date | due_date date |
|---|---|---|---|
| 1 | P30 | 2021-10-14 | 2021-11-04 |
| 2 | P56 | 2021-10-25 | 2021-11-22 |
| 3 | P76 | 2021-10-30 | 2021-12-15 |
| 4 | P77 | 2021-10-19 | 2021-11-01 |

Tuples: 4

5. **Select name and id of the task which was last updated between 1st Oct 2021 and 16 Nov 2021 whose update failed.**

**SQL:**

```
select task_id,task_name
from task_update
where last_updated between '2021/10/1' and '2021/11/16' and update_info='Failed'
```

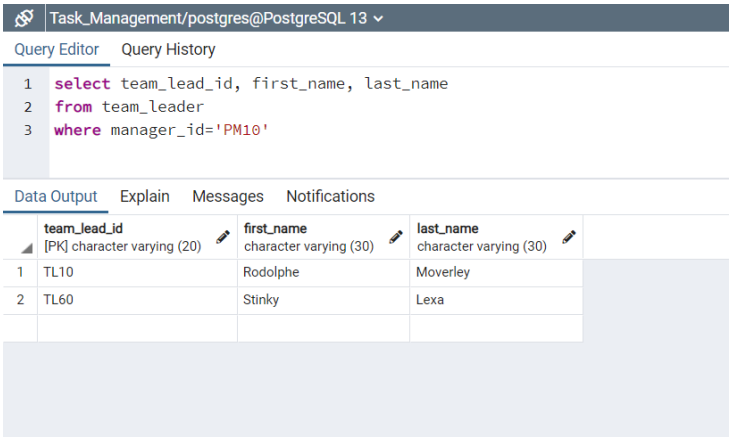



Tuples: 4

6. **Select details of team leaders working under the manager having ID 'PM10'.**

**SQL:**

```
select team_lead_id, first_name, last_name
from team_leader
where manager_id='PM10'
```

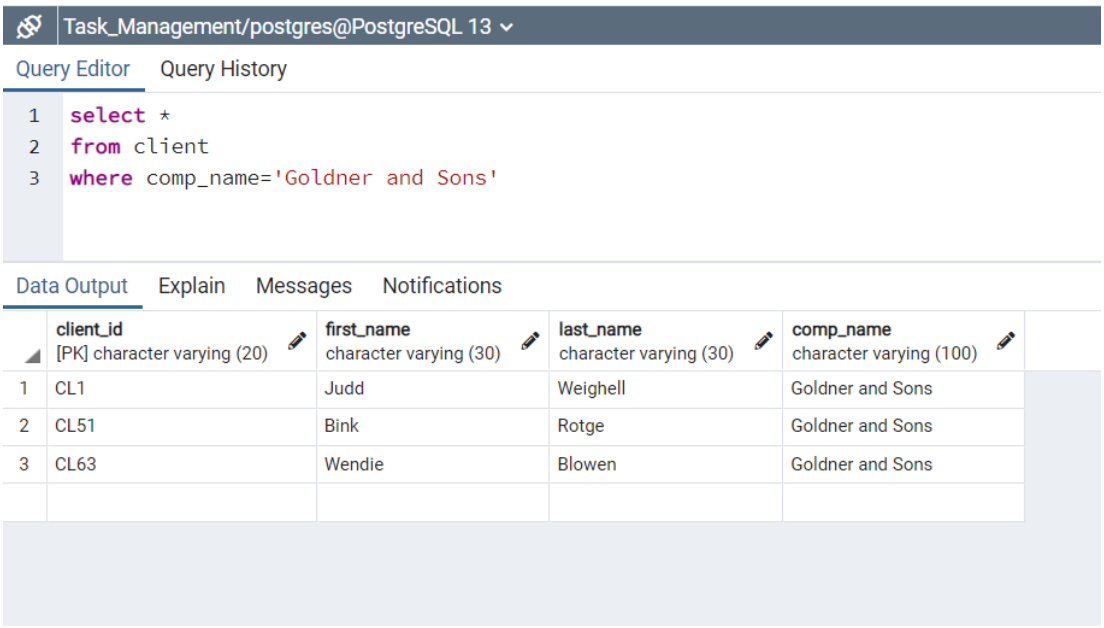



Tuples: 2

7. **Select all the details of clients from the company ‘Goldner and Sons’.**

**SQL:**

```
select *
from client
where comp_name='Goldner and Sons'
```



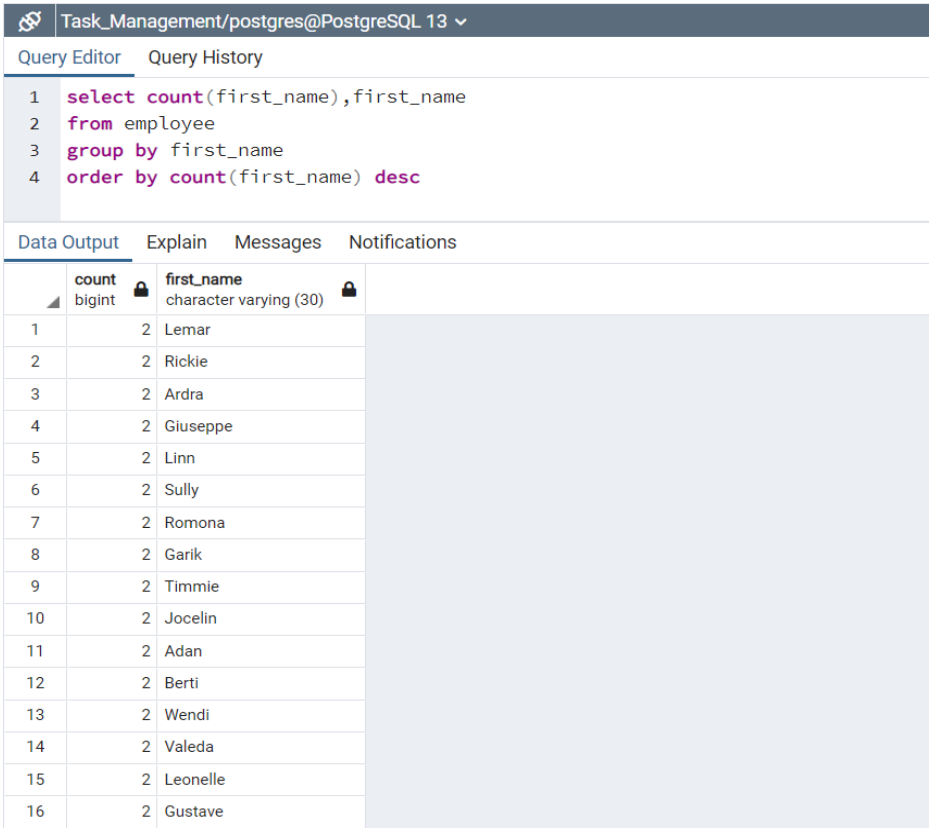

| | client_id [PK] character varying (20) | first_name character varying (30) | last_name character varying (30) | comp_name character varying (100) |
|---|---|---|---|---|
| 1 | CL1 | Judd | Weighell | Goldner and Sons |
| 2 | CL51 | Bink | Rotge | Goldner and Sons |
| 3 | CL63 | Wendie | Blowen | Goldner and Sons |

Tuples: 3

8. **Find the number of employees with the same first name.**

**SQL:**

```
select count(first_name),first_name
from employee
group by first_name
order by count(first_name) desc
```

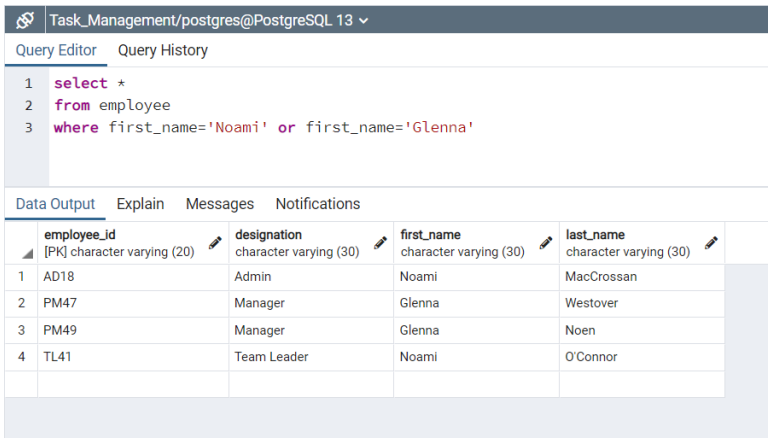



Tuples: 540

9. **Select all the details of the employees whose first name is either 'Noami' or 'Glenna'.**

**SQL:**

```
select *
from employee
where first_name='Noami' or first_name='Glenna'
```




Tuples: 4

10. **Select task id and task name of tasks coming under the Data Visualisation domain.**

**SQL:**

```
select task_id, task_name, task_category
from task
where task_category='data visualization'
```

Task_Management/postgres@PostgreSQL 13

Query Editor | Query History

```
1  select task_id, task_name, task_category
2  from task
3  where task_category='data visualization'
```

Data Output | Explain | Messages | Notifications

| | task_id [PK] character varying (20) | task_name character varying (30) | task_category character varying (25) |
|---|---|---|---|
| 1 | T3 | Tresom | data visualization |
| 2 | T12 | Flowdesk | data visualization |
| 3 | T14 | Gembucket | data visualization |
| 4 | T34 | Opela | data visualization |
| 5 | T35 | Voltsillam | data visualization |
| 6 | T61 | Otcom | data visualization |
| 7 | T62 | Duobam | data visualization |
| 8 | T70 | Konklab | data visualization |
| 9 | T73 | Prodder | data visualization |
| 10 | T75 | Namfix | data visualization |

Tuples: 10

11. **Select all the details from project_details where the project name contains ‘design’.**

**SQL:**

```
select *
from project_details
where project_name like '%design%'
```

Task_Management/postgres@PostgreSQL 13

Query Editor | Query History

```
1  select *
2  from project_details
3  where project_name like '%design%'
```

Data Output | Explain | Messages | Notifications

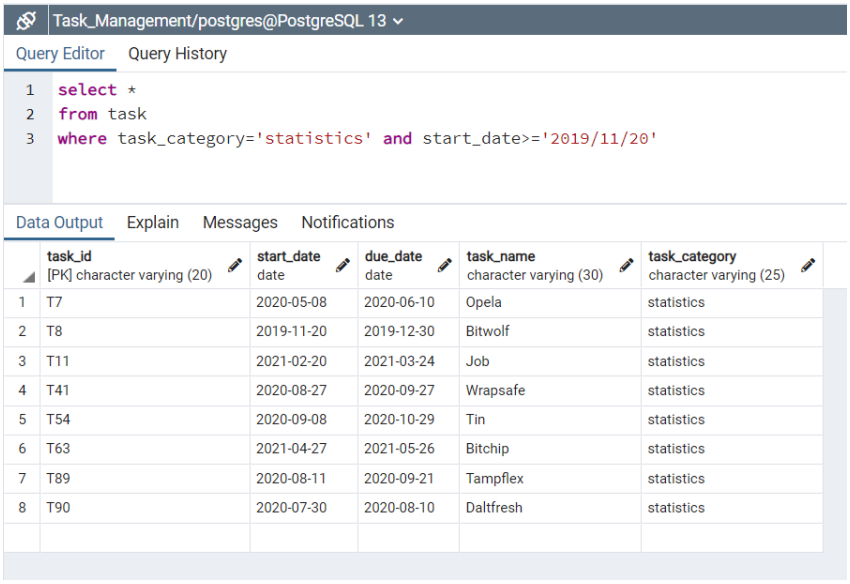
| | project_id [PK] character varying (20) | project_name character varying (30) | project_desc character varying (255) |
|---|---|---|---|
| 1 | P38 | design verse | App to suggest a good design of your home |
| 2 | P39 | cohort design | Software to design observation and nonexperimental study |
| 3 | P40 | deliver design | App for project delivery system |
| 4 | P41 | designx | Webpage for designing wall papers of hotels |
| 5 | P42 | zen design | Create a software to suggest naturilistic designs |
| 6 | P43 | vision design | Website to provide new graphic designs |
| 7 | P44 | imagine design | App to suggest innovative logos based on the name of company |
| 8 | P45 | design lux | Webpage for Engineering design |
| 9 | P48 | nurture design | Design plan for KG schools |
| 10 | P49 | luminous design | Software for interior designing |
| 11 | P50 | module design | App to suggest modules of various products |
| 12 | P51 | gonzo design | Webpage on social innovations and design of cultual projects |
| 13 | P52 | hack design | Creating an app for design thinking |

Tuples: 13

12. **Select all the details from the task where the task category is ‘statistics’ and the start date is after 20th Nov 2019.**

**SQL:**

```
select *
from task
where task_category='statistics' and start_date>='2019/11/20'
```



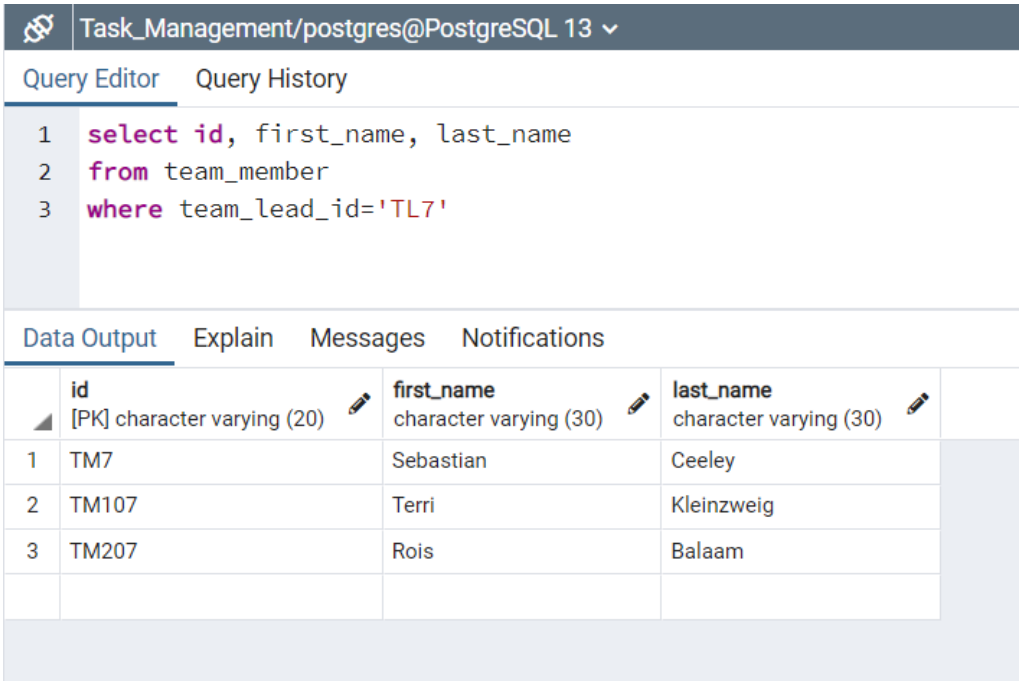

| | task_id [PK] character varying (20) | start_date date | due_date date | task_name character varying (30) | task_category character varying (25) |
|---|---|---|---|---|---|
| 1 | T7 | 2020-05-08 | 2020-06-10 | Opela | statistics |
| 2 | T8 | 2019-11-20 | 2019-12-30 | Bitwolf | statistics |
| 3 | T11 | 2021-02-20 | 2021-03-24 | Job | statistics |
| 4 | T41 | 2020-08-27 | 2020-09-27 | Wrapsafe | statistics |
| 5 | T54 | 2020-09-08 | 2020-10-29 | Tin | statistics |
| 6 | T63 | 2021-04-27 | 2021-05-26 | Bitchip | statistics |
| 7 | T89 | 2020-08-11 | 2020-09-21 | Tampflex | statistics |
| 8 | T90 | 2020-07-30 | 2020-08-10 | Daltfresh | statistics |

Tuples: 8

13. **Select id, and name of team members whose team leader id is TL7**

**SQL:**

```
select id, first_name, last_name
from team_member
where team_lead_id='TL7'
```




Tuples: 3

14. **Select details of all testers from the 'employee' table.**

**SQL:**

```
select employee_id, first_name, last_name
from employee
where designation='Tester'
```

Task_Management/postgres@PostgreSQL 13

Query Editor | Query History

```
1  select employee_id, first_name, last_name
2  from employee
3  where designation='Tester'
```

Data Output | Explain | Messages | Notifications

| | employee_id [PK] character varying (20) | first_name character varying (30) | last_name character varying (30) |
|---|---|---|---|
| 1 | TS1 | Bonnibelle | Starford |
| 2 | TS2 | Gideon | Dradey |
| 3 | TS3 | Nalani | Yorke |
| 4 | TS4 | Dimitry | Lowman |
| 5 | TS5 | Matthus | Josipovic |
| 6 | TS6 | Blayne | Ewols |
| 7 | TS7 | Hilton | Rodrigues |
| 8 | TS8 | Harald | Hambers |
| 9 | TS9 | Edgardo | Playford |
| 10 | TS10 | Edik | Whitfield |
| 11 | TS11 | Skell | Farnell |

Tuples: 60

15. **Select all the details from the task where the task name begins with ‘C’.**

**SQL:**

```
select *
from task
where task_name like 'C%'
```

Task_Management/postgres@PostgreSQL 13

Query Editor | Query History

```
1  select *
2  from task
3  where task_name like 'C%'
```

Data Output | Explain | Messages | Notifications

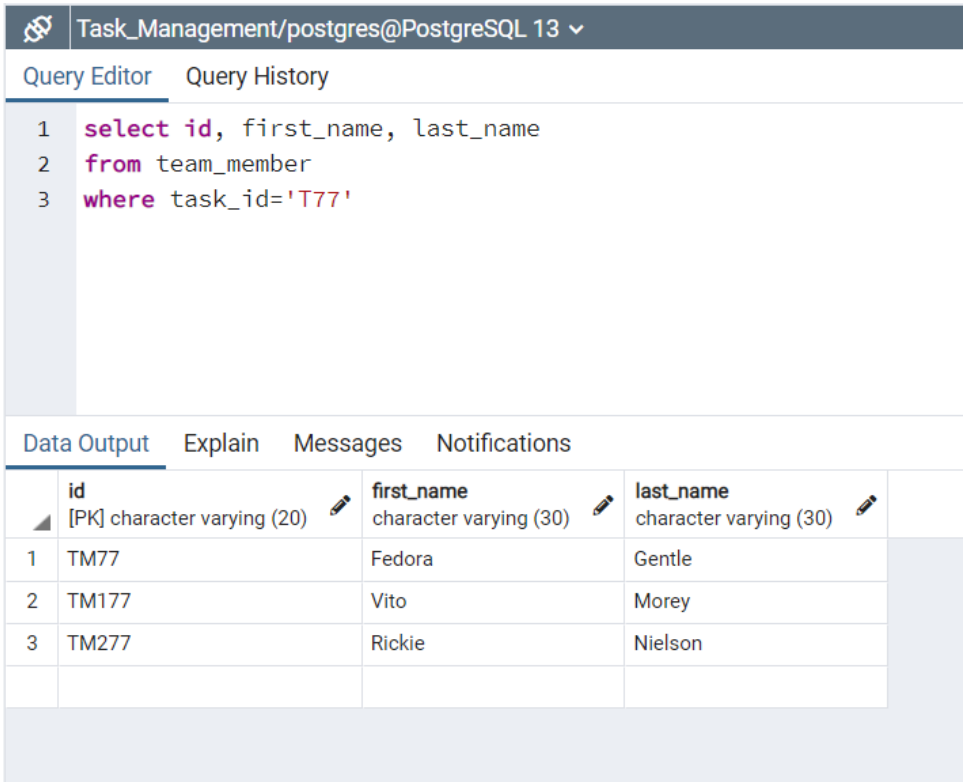
| | task_id [PK] character varying (20) | start_date date | due_date date | task_name character varying (30) | task_category character varying (25) |
|---|---|---|---|---|---|
| 1 | T39 | 2021-06-23 | 2021-07-01 | Cardify | front end |
| 2 | T69 | 2021-07-20 | 2021-08-29 | Cardguard | data loading |
| 3 | T82 | 2020-12-28 | 2021-01-06 | Cookley | data loading |

Tuples: 3

16. **Select id, and name of the team members who have the 'T77' task assigned.**

**SQL:**

```
select id, first_name, last_name
from team_member
where task_id='T77'
```

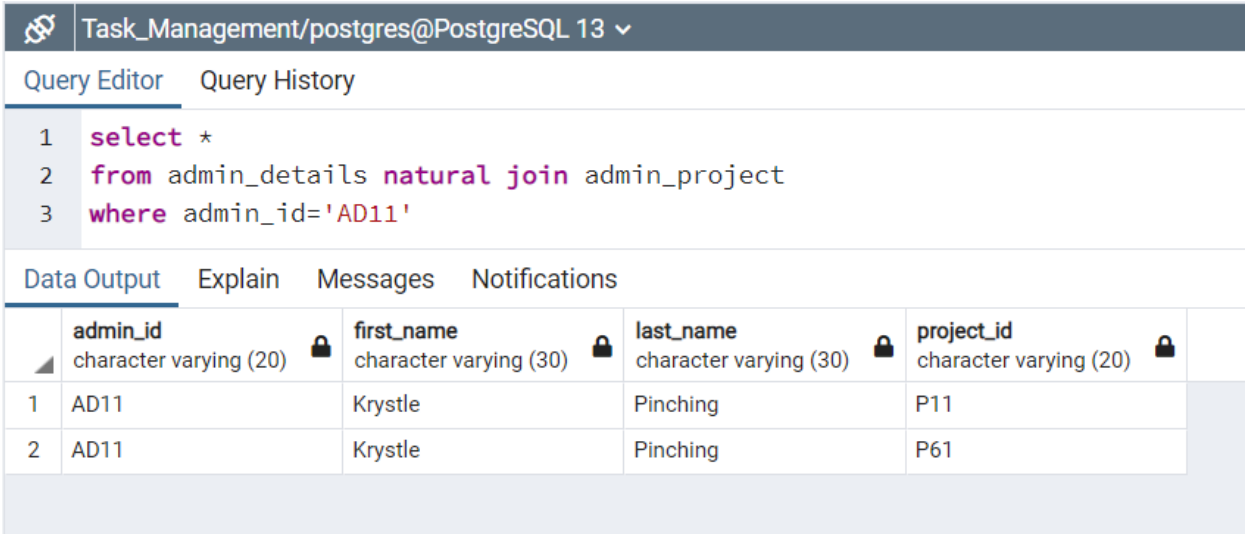



Tuples: 3

17. **Find the projects assigned to an admin whose id is 'AD11'**

**SQL:**

```
select *
from admin_details natural join admin_project
where admin_id='AD11'
```

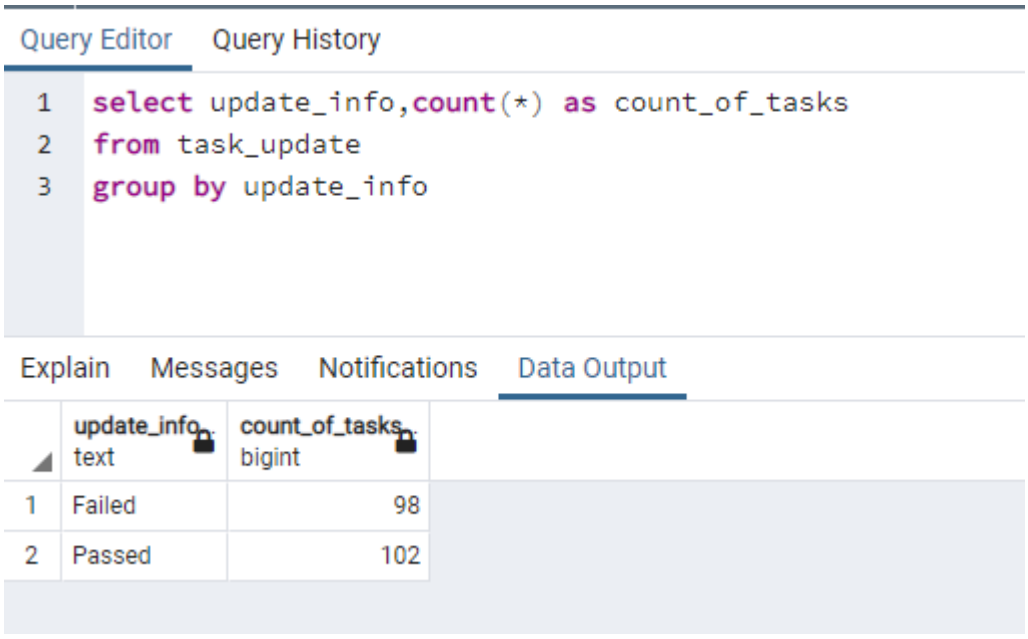



Tuples: 2

18. **Count the number of tasks that either failed or passed**

**SQL:**

```
select update_info,count(*) as count_of_tasks
from task_update
group by update_info
```

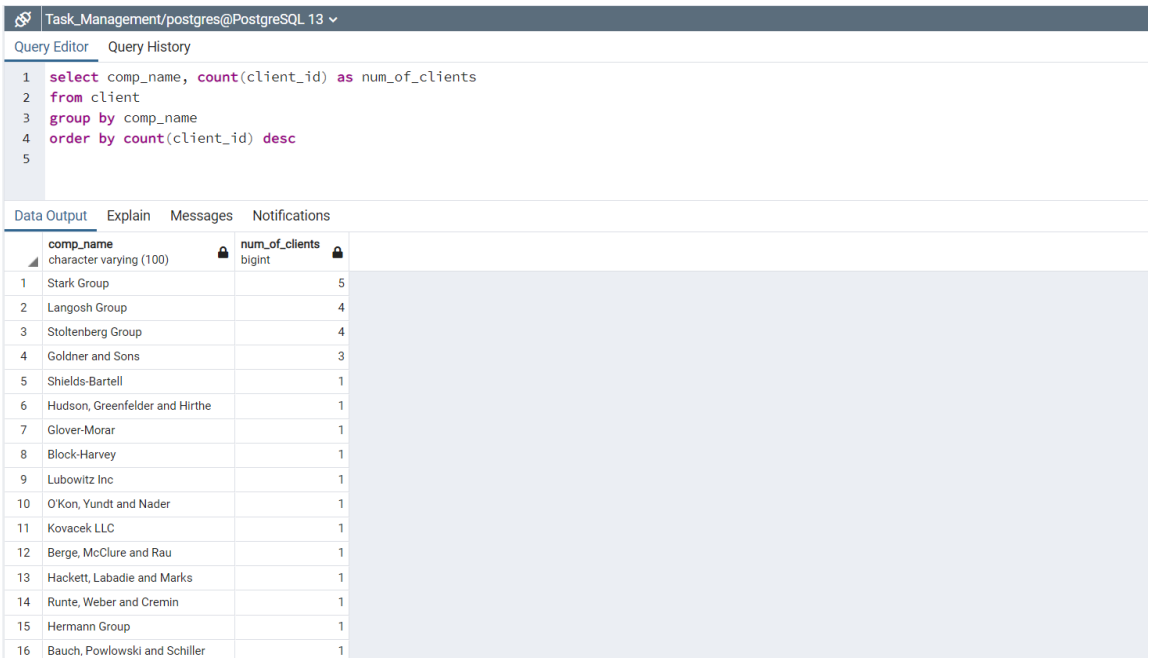



Tuples = 2

19. **Return the number of clients of each company**

**SQL:**

```
select comp_name, count(client_id) as num_of_clients
from client
group by comp_name
order by count(client_id) desc
```




| | comp_name character varying (100) | num_of_clients bigint |
|---|---|---|
| 1 | Stark Group | 5 |
| 2 | Langosh Group | 4 |
| 3 | Stoltenberg Group | 4 |
| 4 | Goldner and Sons | 3 |
| 5 | Shields-Bartell | 1 |
| 6 | Hudson, Greenfelder and Hirthe | 1 |
| 7 | Glover-Morar | 1 |
| 8 | Block-Harvey | 1 |
| 9 | Lubowitz Inc | 1 |
| 10 | O'Kon, Yundt and Nader | 1 |
| 11 | Kovacek LLC | 1 |
| 12 | Berge, McClure and Rau | 1 |
| 13 | Hackett, Labadie and Marks | 1 |
| 14 | Runte, Weber and Cremin | 1 |
| 15 | Hermann Group | 1 |
| 16 | Bauch, Powlowski and Schiller | 1 |

Tuples: 88

20. **Find admin details along with the number of projects assigned to each admin.**

**SQL:**

```
select admin_id, first_name, last_name, count(project_id)
from admin_details natural join admin_project
group by admin_id
order by count(project_id) desc
```

Task_Management/postgres@PostgreSQL 13

Query Editor | Query History

```
1  select admin_id, first_name, last_name, count(project_id)
2  from admin_details natural join admin_project
3  group by admin_id
4  order by count(project_id) desc
```

Data Output | Explain | Messages | Notifications

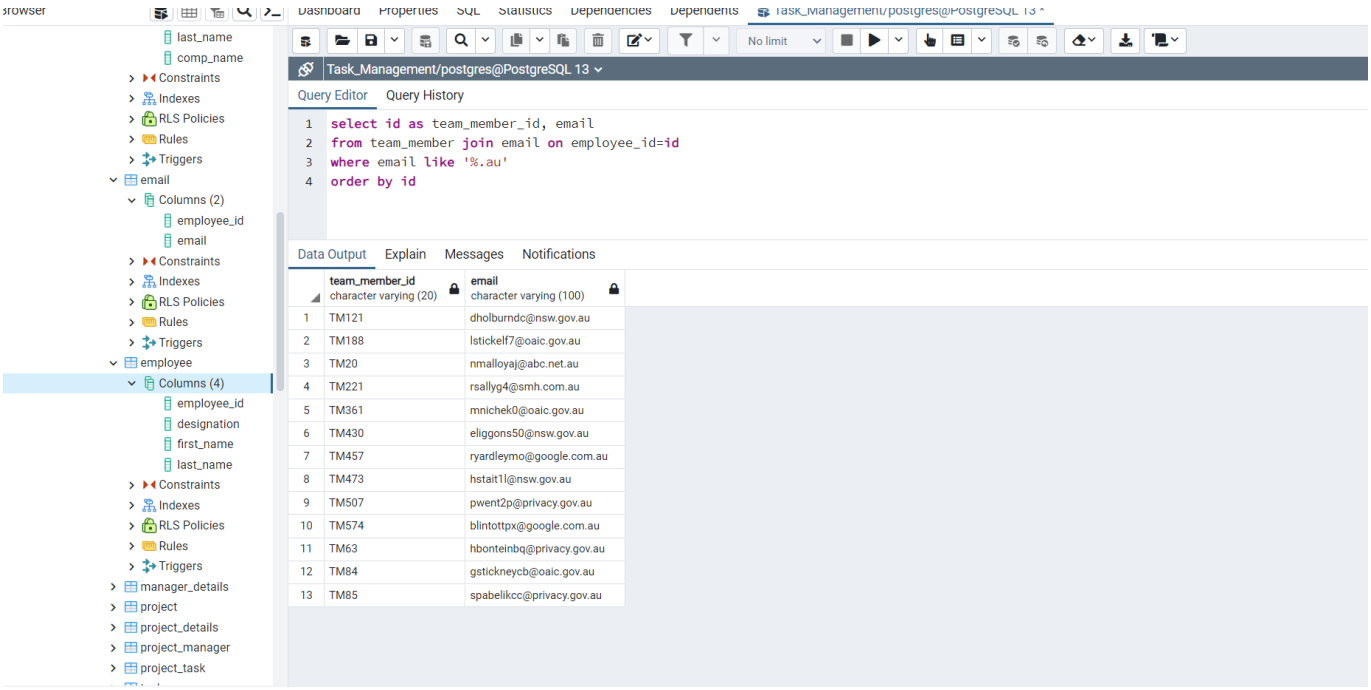
| | admin_id [PK] character varying (20) | first_name character varying (30) | last_name character varying (30) | count bigint |
|---|---|---|---|---|
| 1 | AD27 | Blanch | Hame | 2 |
| 2 | AD22 | Clair | Harback | 2 |
| 3 | AD20 | Ahmed | Crickett | 2 |
| 4 | AD29 | Kingsly | Pagitt | 2 |
| 5 | AD3 | Harwell | Westfield | 2 |
| 6 | AD12 | Dee dee | Barnard | 2 |
| 7 | AD2 | Marice | Matthaus | 2 |
| 8 | AD10 | Lester | Giral | 2 |
| 9 | AD25 | Patrizia | Yandle | 2 |

Tuples: 50

**21. Select the emails and id's of all team members who have an australian email id:**

**SQL:**

```
select id as team_member_id, email
from team_member join email on employee_id=id
where email like '%.au'
order by id
```

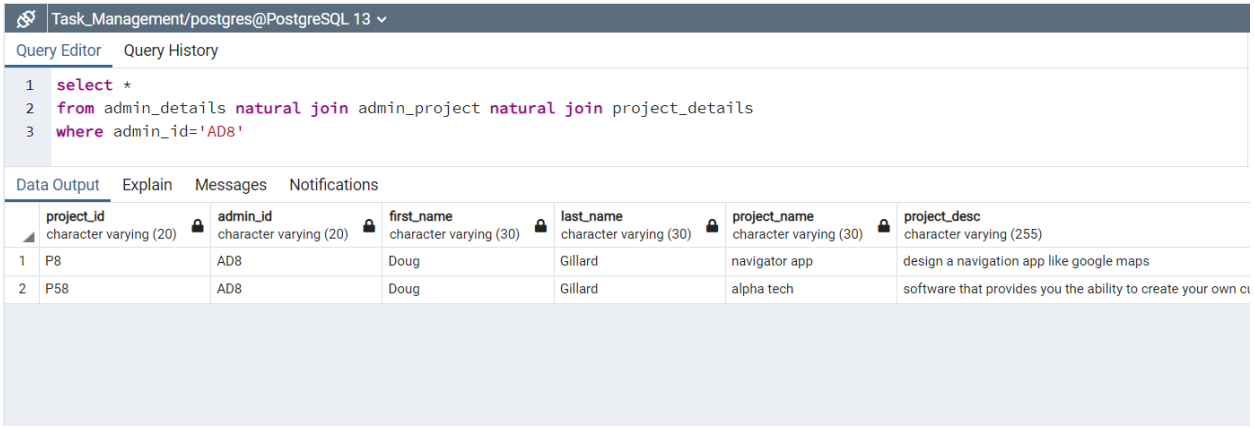


Tuples = 13

22. **Select all details of the project assigned to an admin whose id is “AD8”**

**SQL:**

```
select *
from admin_details natural join admin_project natural join project_details
where admin_id='AD8'
```



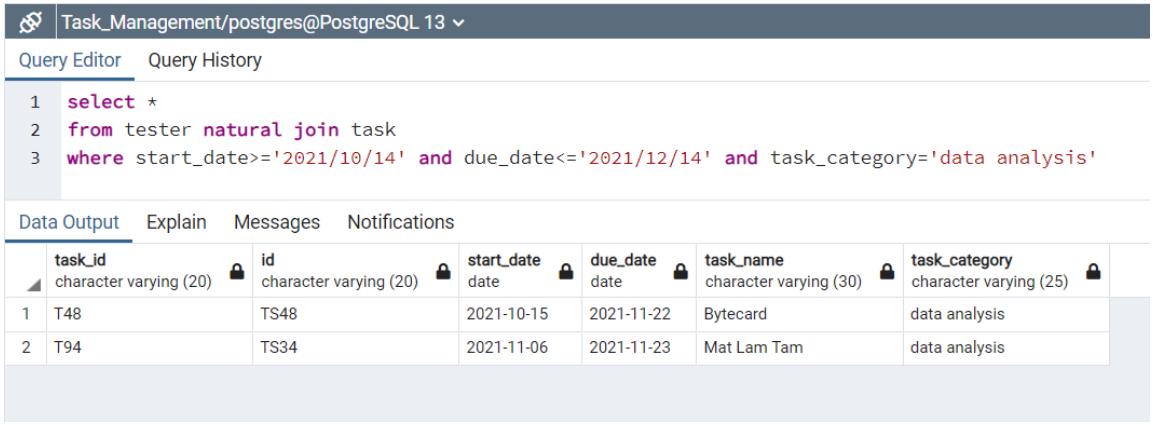

| | project_id character varying (20) | admin_id character varying (20) | first_name character varying (30) | last_name character varying (30) | project_name character varying (30) | project_desc character varying (255) |
|---|---|---|---|---|---|---|
| 1 | P8 | AD8 | Doug | Gillard | navigator app | design a navigation app like google maps |
| 2 | P58 | AD8 | Doug | Gillard | alpha tech | software that provides you the ability to create your own cu |

Tuples: 2

23. **Find all the task details whose start_date is after 14th Oct 2021 and is due before 14th Dec 2021 and whose task_category is ‘data analysis’.**

**SQL:**

```
select *
from tester natural join task
where start_date>='2021/10/14' and due_date<='2021/12/14' and task_category='data analysis'
```



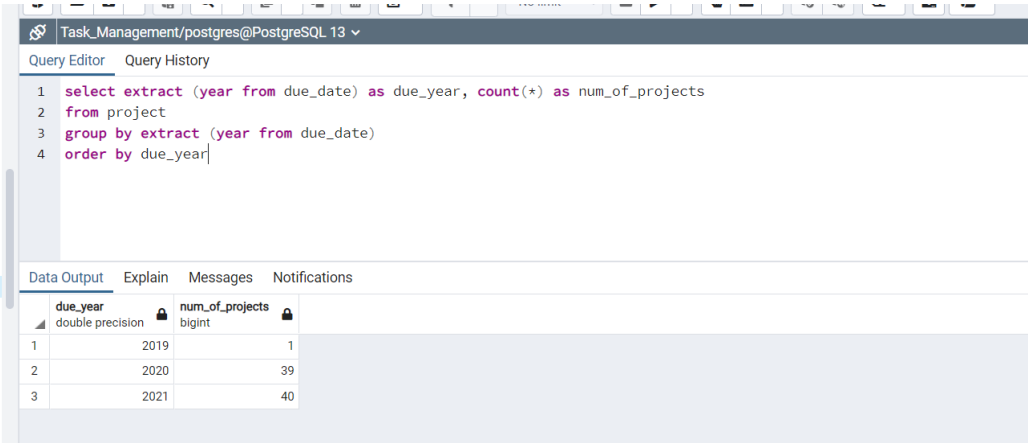

| | task_id character varying (20) | id character varying (20) | start_date date | due_date date | task_name character varying (30) | task_category character varying (25) |
|---|---|---|---|---|---|---|
| 1 | T48 | TS48 | 2021-10-15 | 2021-11-22 | Bytecard | data analysis |
| 2 | T94 | TS34 | 2021-11-06 | 2021-11-23 | Mat Lam Tam | data analysis |

Tuples: 2

**24. Select the total number of projects completed in each year.**

**SQL:**

```
select extract (year from due_date) as due_year, count(*) as num_of_projects
from project
group by extract (year from due_date)
order by due_year
```

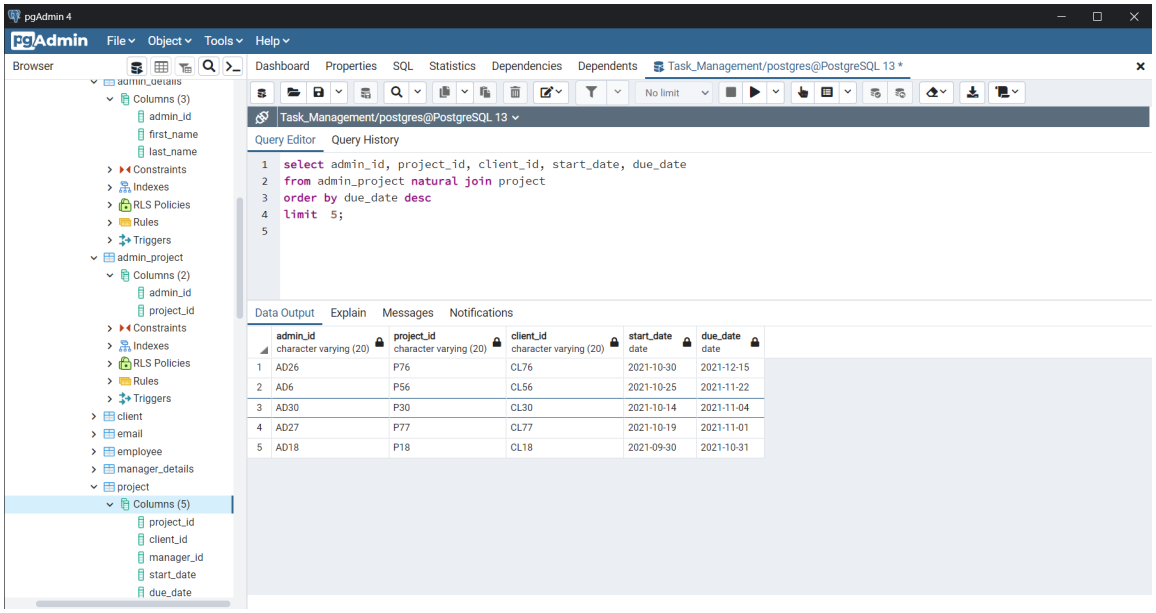



Tuples: 3

**25. Select the admin and clients is of the 5 latest projects:**

**SQL:**

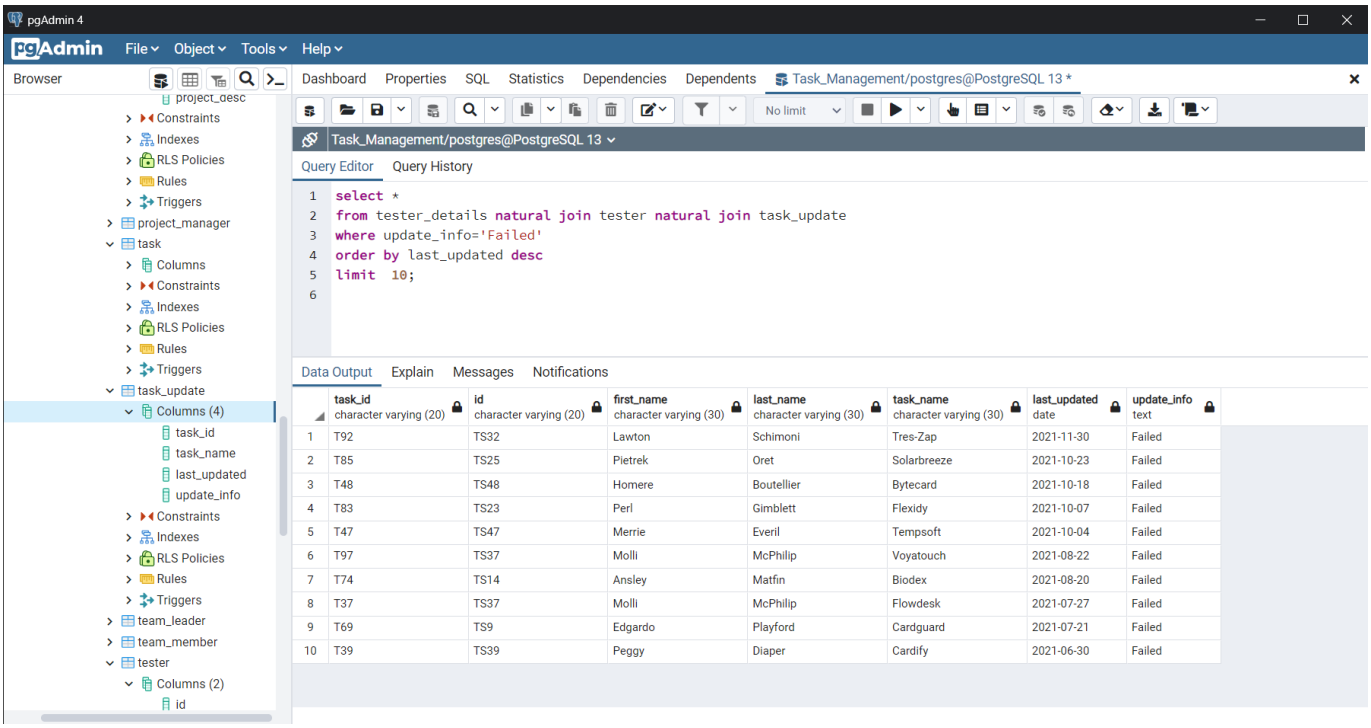
```
select admin_id, project_id, client_id, start_date, due_date
from admin_project natural join project
order by due_date desc
limit  5;
```



**26. Select the details of the tester and task of the 10 recently failed tasks.**

**SQL:**

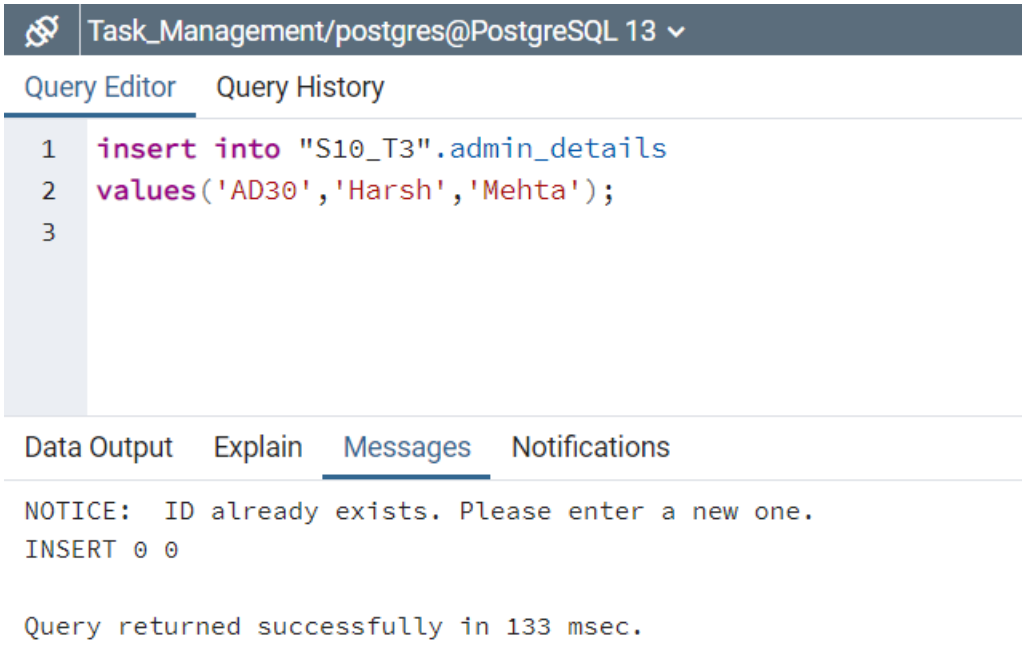
```
select *
from tester_details natural join tester natural join task_update
where update_info='Failed'
order by last_updated desc
limit  10;
```




| | task_id character varying (20) | id character varying (20) | first_name character varying (30) | last_name character varying (30) | task_name character varying (30) | last_updated date | update_info text |
|---|---|---|---|---|---|---|---|
| 1 | T92 | TS32 | Lawton | Schimoni | Tres-Zap | 2021-11-30 | Failed |
| 2 | T85 | TS25 | Pietrek | Oret | Solarbreeze | 2021-10-23 | Failed |
| 3 | T48 | TS48 | Homere | Boutellier | Bytecard | 2021-10-18 | Failed |
| 4 | T83 | TS23 | Perl | Gimblett | Flexidy | 2021-10-07 | Failed |
| 5 | T47 | TS47 | Merrie | Everil | Tempsoft | 2021-10-04 | Failed |
| 6 | T97 | TS37 | Molli | McPhilip | Voyatouch | 2021-08-22 | Failed |
| 7 | T74 | TS14 | Ansley | Matfin | Biodex | 2021-08-20 | Failed |
| 8 | T37 | TS37 | Molli | McPhilip | Flowdesk | 2021-07-27 | Failed |
| 9 | T69 | TS9 | Edgardo | Playford | Cardguard | 2021-07-21 | Failed |
| 10 | T39 | TS39 | Peggy | Diaper | Cardify | 2021-06-30 | Failed |

Tuples: 10

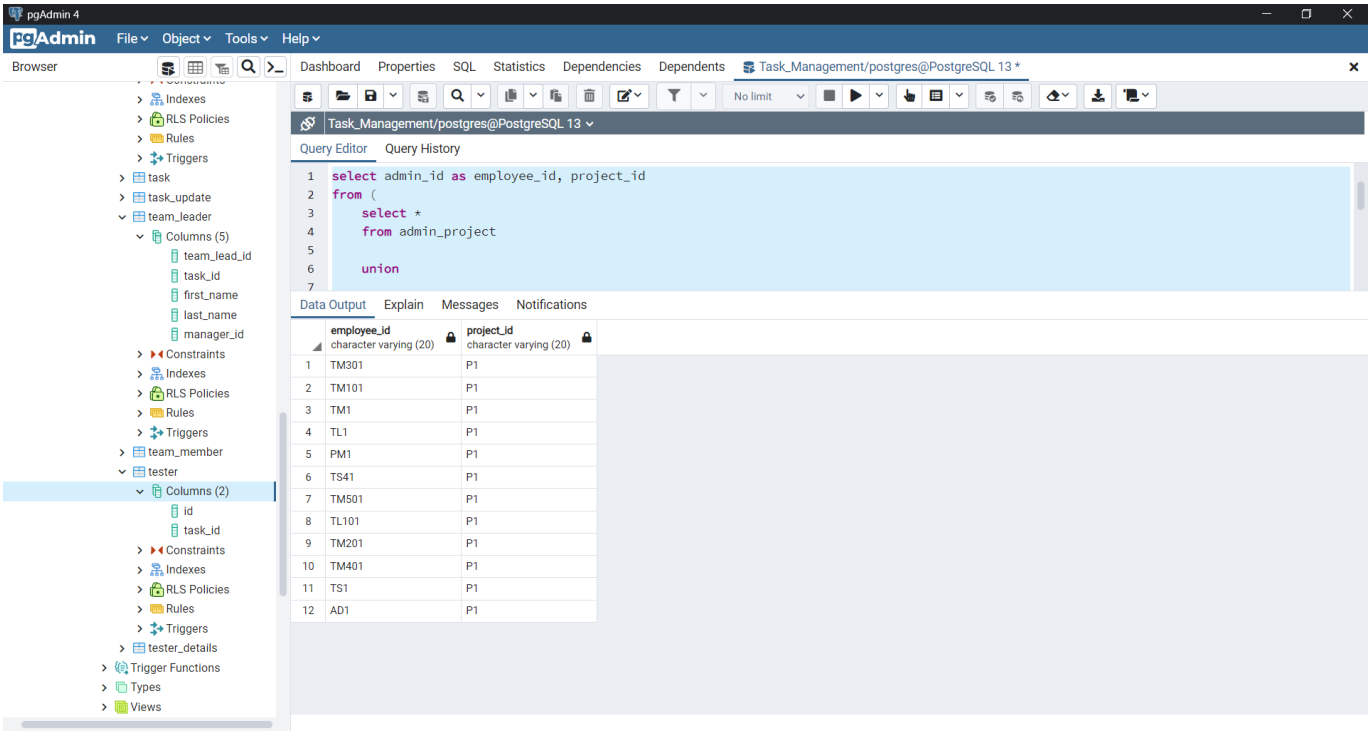
**27. Insert into the admin_details table, the values should be admin_id=AD30, the first name is Harsh, the last name is Mehta: (does not insert if it doesn't satisfy the PK constraints)**




28. **Select all the ids of all employees that are working on project P1 :**

**SQL:**

```
select admin_id as employee_id, project_id
from (
        select *
        from admin_project

        union

        select *
        from project_manager

        union

        select team_lead_id, project_id
        from team_leader natural join project_task

        union

        select id, project_id
        from team_member natural join project_task

        union

        select id, project_id
        from tester natural join project_task
) as T

where project_id='P1'
```

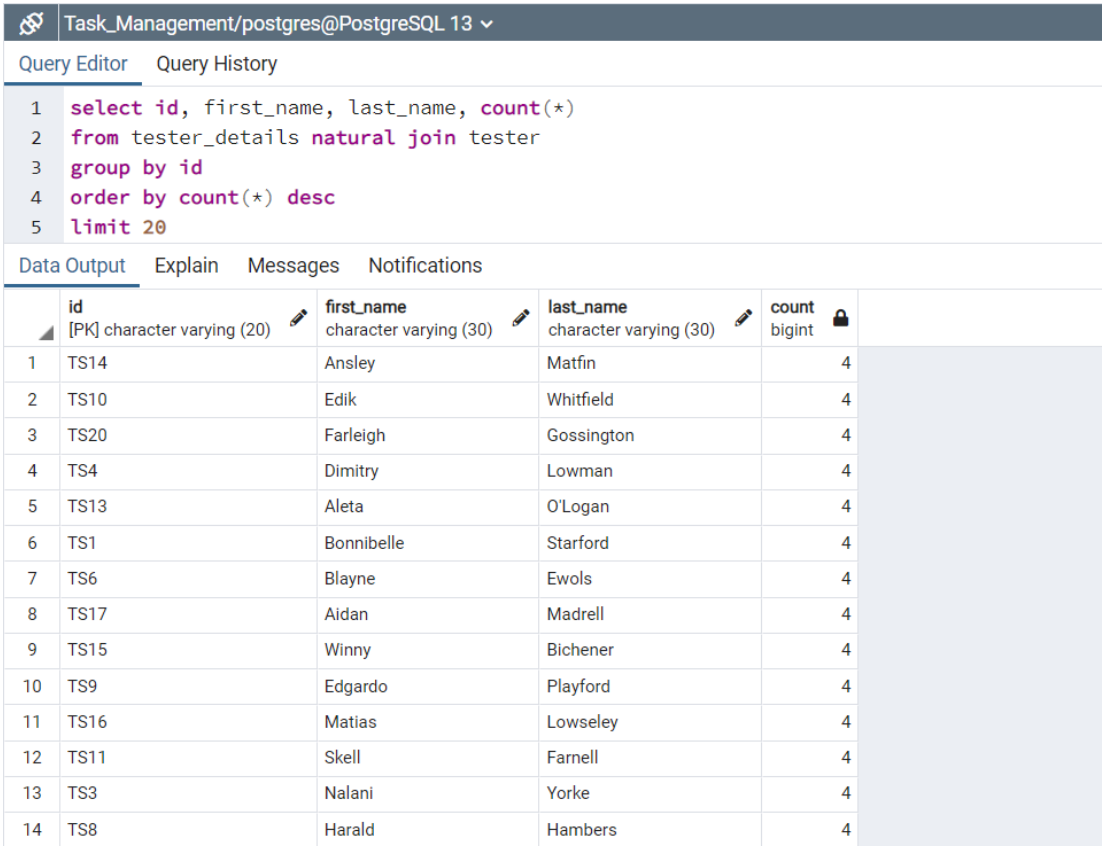



Tuples: 12

29. **Find the first 20 names and ids of all the testers who have tested the most number of tasks.**

**SQL:**

```
select id, first_name, last_name, count(*)
from tester_details natural join tester
group by id
order by count(*) desc
limit 20
```



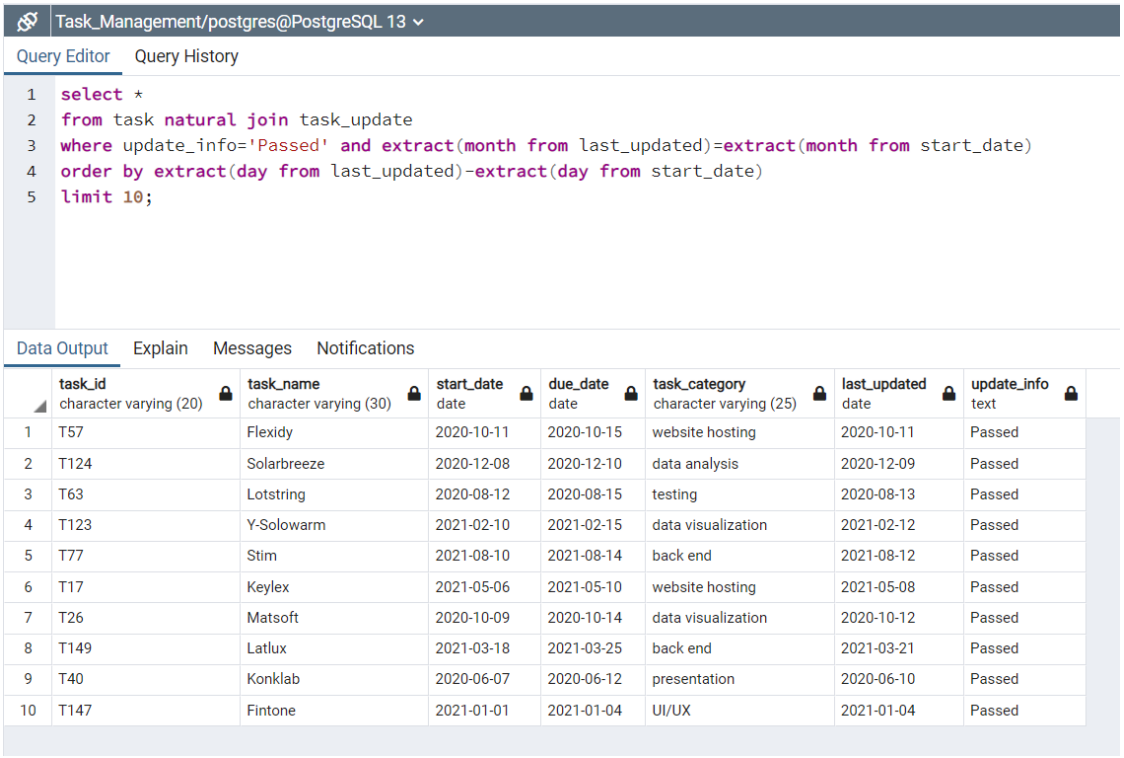

| | id [PK] character varying (20) | first_name character varying (30) | last_name character varying (30) | count bigint |
|---|---|---|---|---|
| 1 | TS14 | Ansley | Matfin | 4 |
| 2 | TS10 | Edik | Whitfield | 4 |
| 3 | TS20 | Farleigh | Gossington | 4 |
| 4 | TS4 | Dimitry | Lowman | 4 |
| 5 | TS13 | Aleta | O'Logan | 4 |
| 6 | TS1 | Bonnibelle | Starford | 4 |
| 7 | TS6 | Blayne | Ewols | 4 |
| 8 | TS17 | Aidan | Madrell | 4 |
| 9 | TS15 | Winny | Bichener | 4 |
| 10 | TS9 | Edgardo | Playford | 4 |
| 11 | TS16 | Matias | Lowseley | 4 |
| 12 | TS11 | Skell | Farnell | 4 |
| 13 | TS3 | Nalani | Yorke | 4 |
| 14 | TS8 | Harald | Hambers | 4 |

Tuples: 20

30. **Find the 10 tasks that were finished fastest.**

**SQL:**

```
select *
from task natural join task_update
where update_info='Passed' and extract(month from last_updated)=extract(month from start_date)
order by extract(day from last_updated)-extract(day from start_date)
limit 10;
```



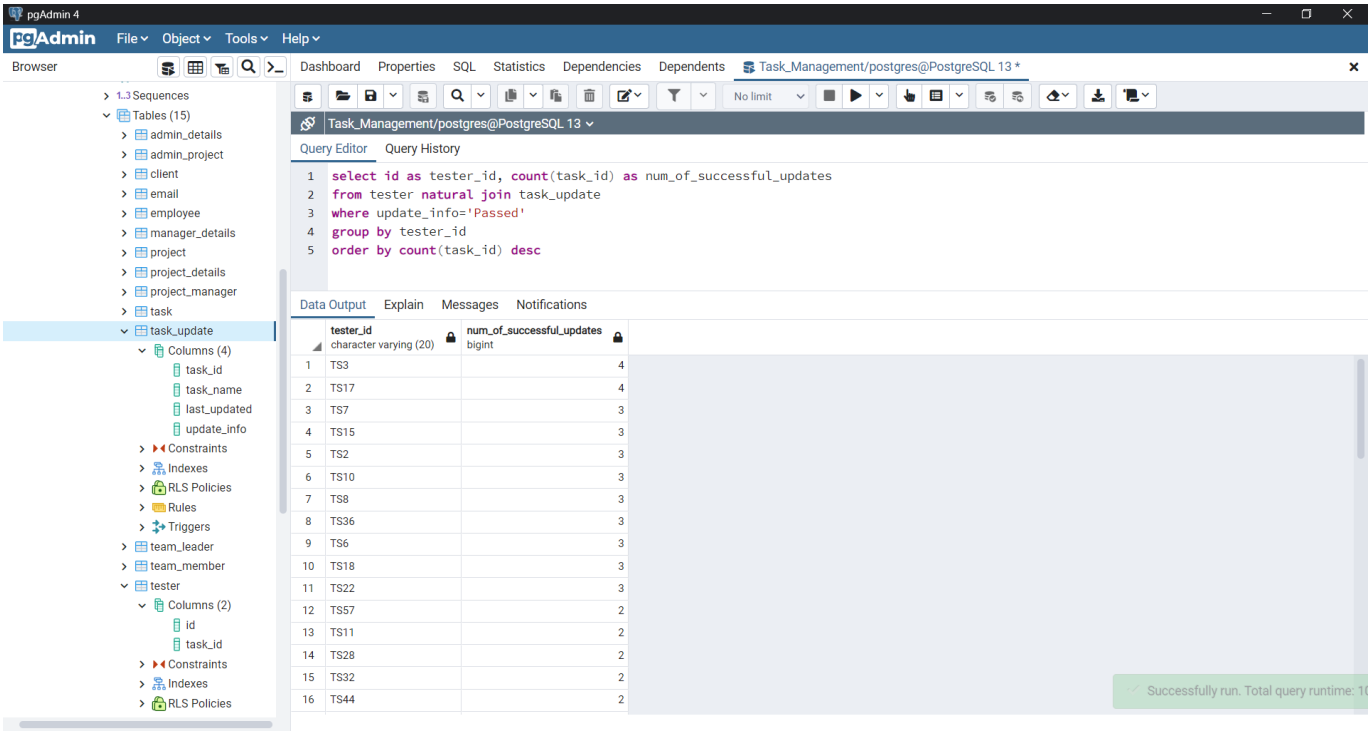

| | task_id character varying (20) | task_name character varying (30) | start_date date | due_date date | task_category character varying (25) | last_updated date | update_info text |
|---|---|---|---|---|---|---|---|
| 1 | T57 | Flexidy | 2020-10-11 | 2020-10-15 | website hosting | 2020-10-11 | Passed |
| 2 | T124 | Solarbreeze | 2020-12-08 | 2020-12-10 | data analysis | 2020-12-09 | Passed |
| 3 | T63 | Lotstring | 2020-08-12 | 2020-08-15 | testing | 2020-08-13 | Passed |
| 4 | T123 | Y-Solowarm | 2021-02-10 | 2021-02-15 | data visualization | 2021-02-12 | Passed |
| 5 | T77 | Stim | 2021-08-10 | 2021-08-14 | back end | 2021-08-12 | Passed |
| 6 | T17 | Keylex | 2021-05-06 | 2021-05-10 | website hosting | 2021-05-08 | Passed |
| 7 | T26 | Matsoft | 2020-10-09 | 2020-10-14 | data visualization | 2020-10-12 | Passed |
| 8 | T149 | Latlux | 2021-03-18 | 2021-03-25 | back end | 2021-03-21 | Passed |
| 9 | T40 | Konklab | 2020-06-07 | 2020-06-12 | presentation | 2020-06-10 | Passed |
| 10 | T147 | Fintone | 2021-01-01 | 2021-01-04 | UI/UX | 2021-01-04 | Passed |

Tuples: 10

31. **Select tester whose maximum task updates were passed:**

**SQL:**

```
select id as tester_id, count(task_id) as num_of_successful_updates
from tester natural join task_update
where update_info='Passed'
group by tester_id
order by count(task_id) desc
```




Tuples: 54

**32. Which team leaders and team members are working under project manager 'PM14'?**

**SQL:**

```
select *
from employee
where employee_id in (select id
                      from team_member
                      where team_lead_id in (select team_lead_id
                            from team_leader
                            where manager_id='PM14'
                      )) or employee_id in (select team_lead_id
                                            from team_leader
                                            where manager_id='PM14')
```

Task_Management/postgres@PostgreSQL 13

Query Editor | Query History

```
select *
from employee
where employee_id in (select id
                        from team_member
                       where team_lead_id in (select team_lead_id
                                                from team_leader
                                                where manager_id='PM14'
                        )) or employee_id in (select team_lead_id
                                                from team_leader
                                                where manager_id='PM14')
```

Data Output | Explain | Messages | Notifications

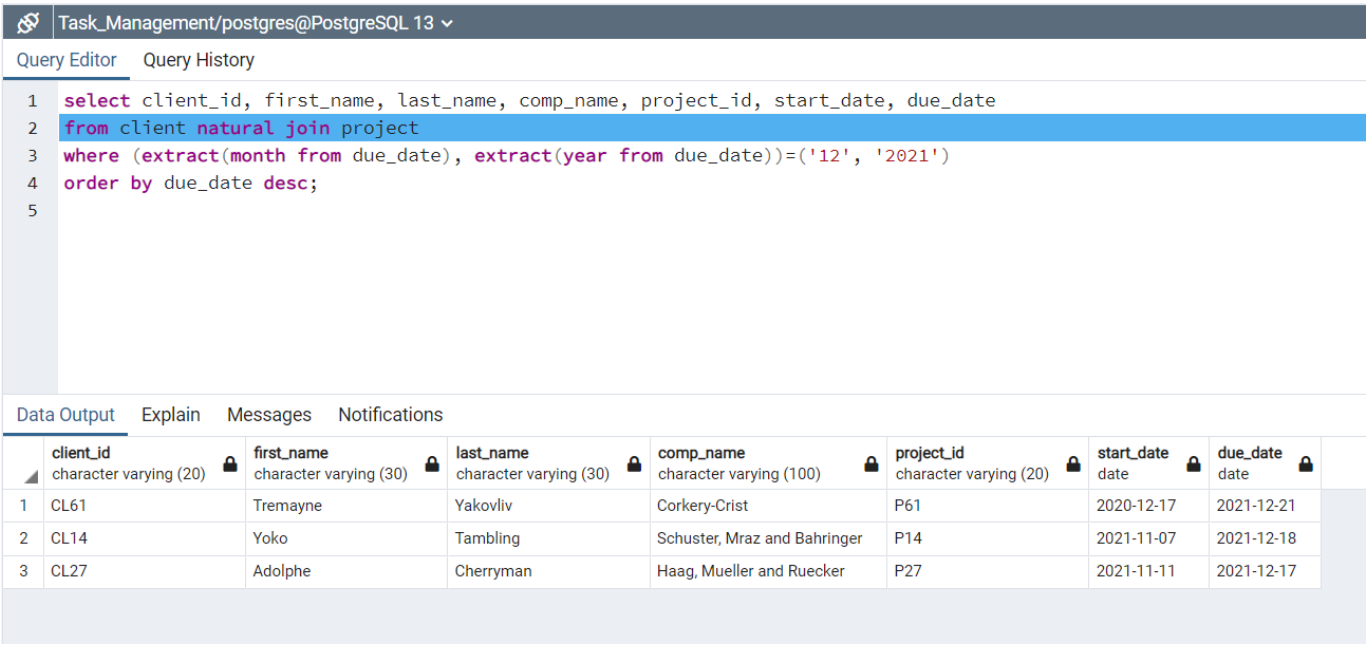
| | employee_id [PK] character varying (20) | designation character varying (30) | first_name character varying (30) | last_name character varying (30) |
|---|---|---|---|---|
| 1 | TL14 | Team Leader | Ingeberg | Mcimmie |
| 2 | TL64 | Team Leader | Louisette | Terbeek |
| 3 | TL114 | Team Leader | Junia | Baptista |
| 4 | TL164 | Team Leader | Orazio | Beazley |
| 5 | TM14 | Team Member | Ber | Boanas |
| 6 | TM64 | Team Member | Luise | MacKereth |
| 7 | TM114 | Team Member | Cristen | Dahmke |
| 8 | TM164 | Team Member | Sigrid | Ody |
| 9 | TM214 | Team Member | Ilise | Pirrey |
| 10 | TM264 | Team Member | Dion | Thake |
| 11 | TM314 | Team Member | Rupert | Keele |
| 12 | TM364 | Team Member | Amalea | Whellams |
| 13 | TM414 | Team Member | Demeter | Barrat |
| 14 | TM464 | Team Member | Hasheem | Blaxley |

Tuples: 16

33. **Give all details of the project that the clients have asked for till now, that are due in December 2021, along with client details and project timeline:**

    **SQL:**

    ```
    select client_id, first_name, last_name, comp_name, project_id, start_date, due_date
    from client natural join project
    where (extract(month from due_date), extract(year from due_date))=('12', '2021')
    order by due_date desc;
    ```



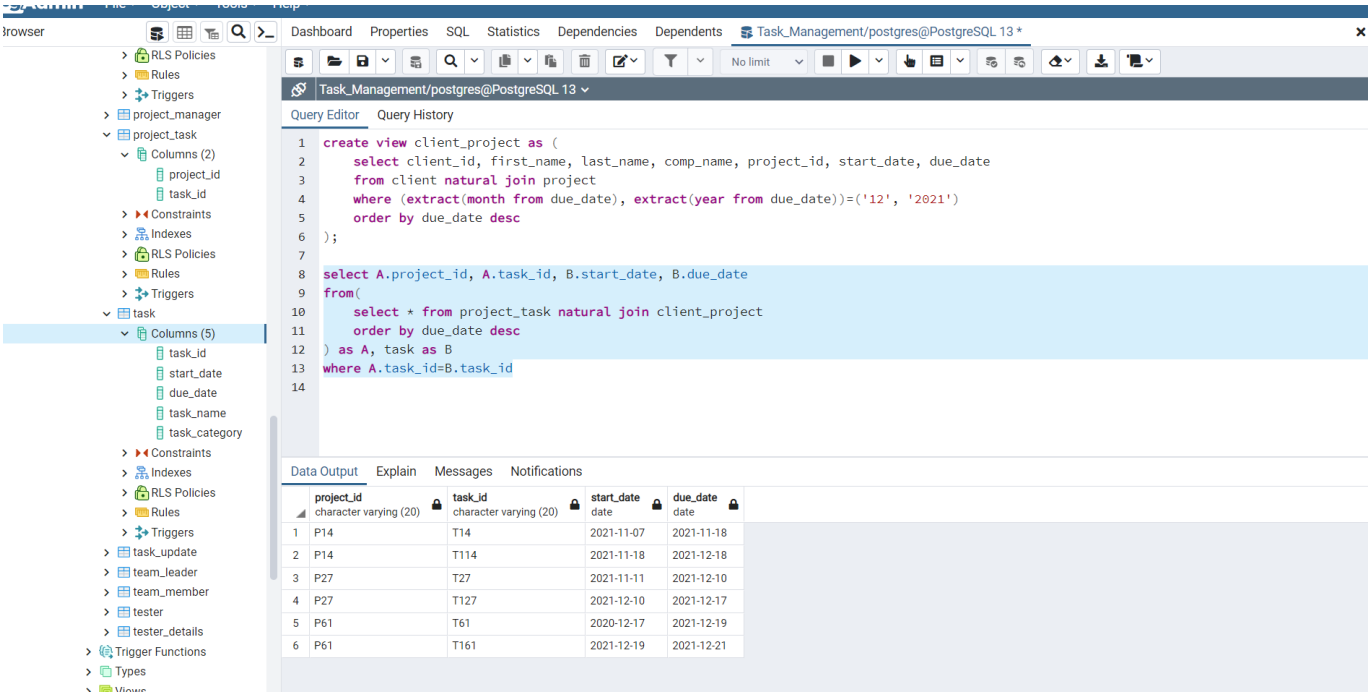

| | client_id character varying (20) | first_name character varying (30) | last_name character varying (30) | comp_name character varying (100) | project_id character varying (20) | start_date date | due_date date |
|---|---|---|---|---|---|---|---|
| 1 | CL61 | Tremayne | Yakovliv | Corkery-Crist | P61 | 2020-12-17 | 2021-12-21 |
| 2 | CL14 | Yoko | Tambling | Schuster, Mraz and Bahringer | P14 | 2021-11-07 | 2021-12-18 |
| 3 | CL27 | Adolphe | Cherryman | Haag, Mueller and Ruecker | P27 | 2021-11-11 | 2021-12-17 |

Tuples: 3

**34. Using the above table as a view, select project ids, and the timelines of the tasks they were segregated in:**

**SQL:**

```
create view client_project as (
        select client_id, first_name, last_name, comp_name, project_id, start_date, due_date
   from client natural join project
   where (extract(month from due_date), extract(year from due_date))=('12', '2021')
   order by due_date desc
);

select A.project_id, A.task_id, B.start_date, B.due_date
from(
        select * from project_task natural join client_project
        order by due_date desc
) as A, task as B
where A.task_id=B.task_id
```

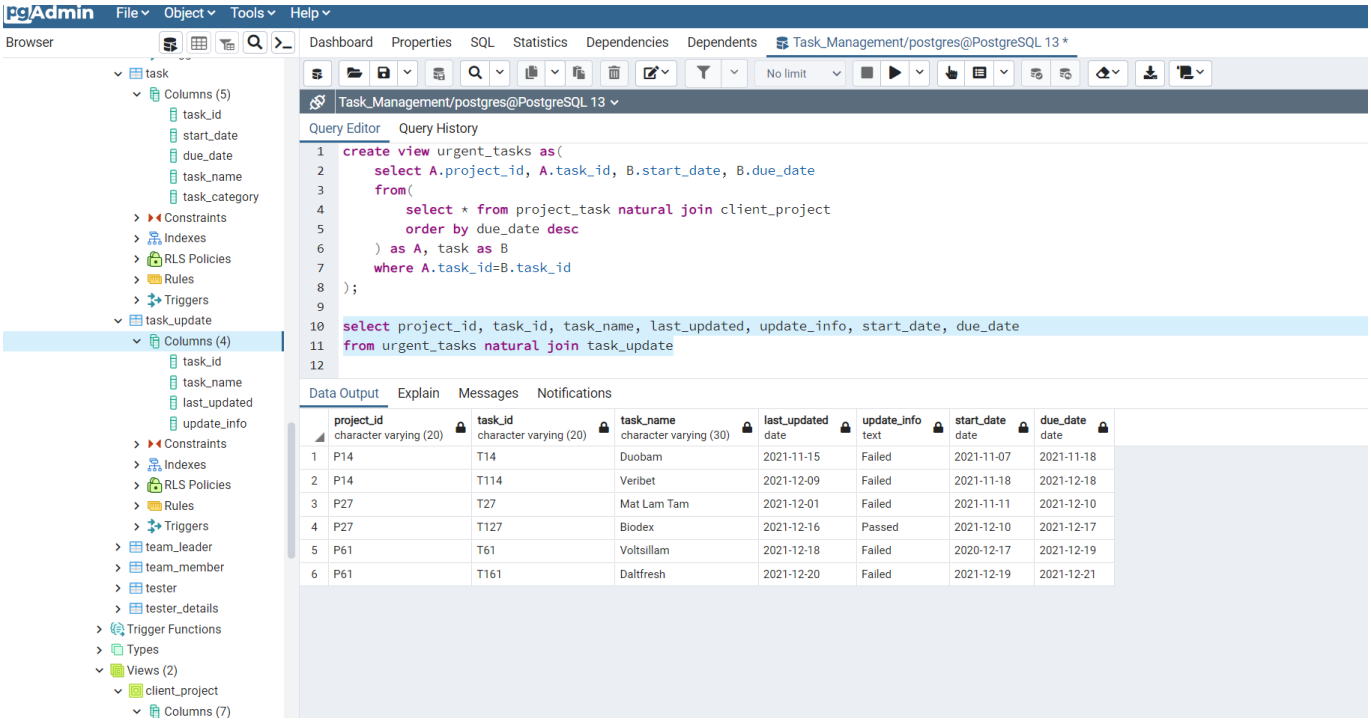



Tuples: 6

**35. Using the above table as a view, i.e. urgent tasks, return the updates that were performed on it:**

**SQL:**

```
create view urgent_tasks as(
        select A.project_id, A.task_id, B.start_date, B.due_date
   from(
        select * from project_task natural join client_project
        order by due_date desc
   ) as A, task as B
   where A.task_id=B.task_id
);

select project_id, task_id, task_name, last_updated, update_info, start_date, due_date
from urgent_tasks natural join task_update
```




Tuples: 6

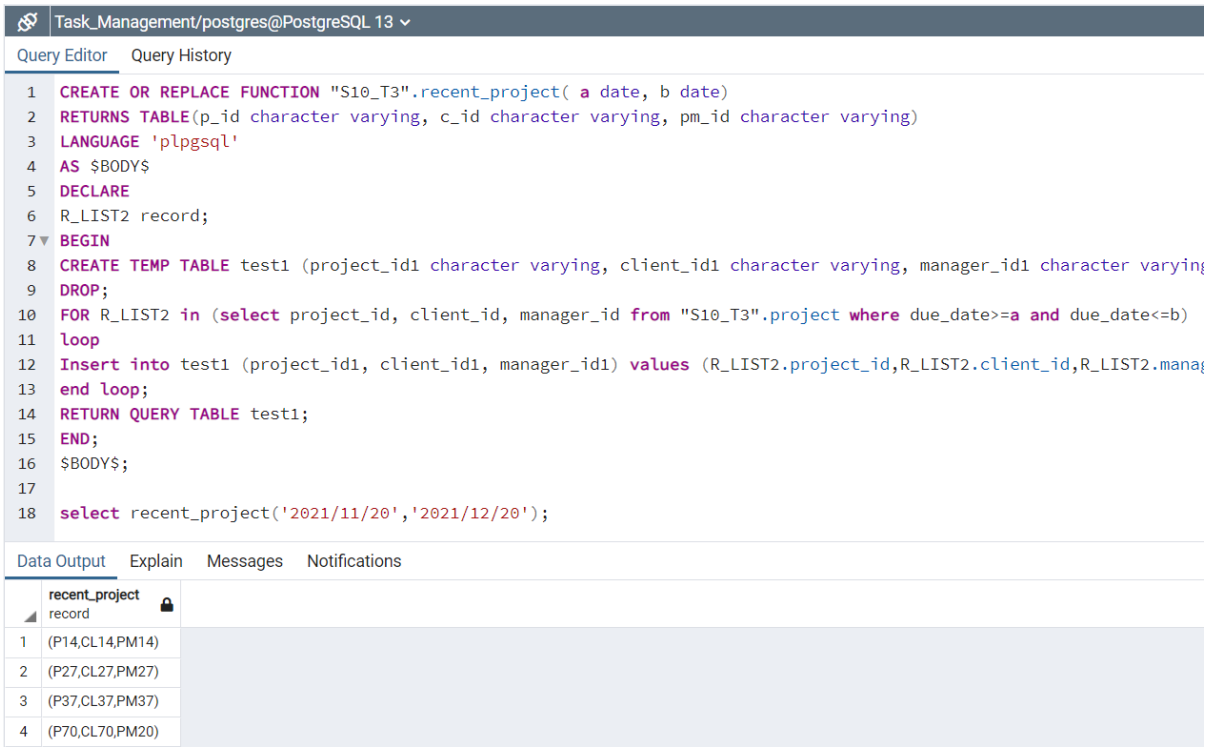
36. **Create a function that returns the list of projects along with their client and manager id's completed within the time interval specified in the input arguments.**

**Function Code:**

```
CREATE OR REPLACE FUNCTION "S10_T3".recent_project( a date, b date)
RETURNS TABLE(p_id character varying, c_id character varying, pm_id character varying)
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
R_LIST2 record;
BEGIN
CREATE TEMP TABLE test1 (project_id1 character varying, client_id1 character varying, manager_id1 character varying) ON COMMIT
DROP;
FOR R_LIST2 in (select project_id, client_id, manager_id from "S10_T3".project where due_date>=a and due_date<=b)
loop
Insert into test1 (project_id1, client_id1, manager_id1) values
(R_LIST2.project_id,R_LIST2.client_id,R_LIST2.manager_id);
end loop;
RETURN QUERY TABLE test1;
END;
$BODY$;
```

**Function call (SQL query)**:

```
select recent_project('2021/11/20','2021/12/20');
```




Tuples: 4

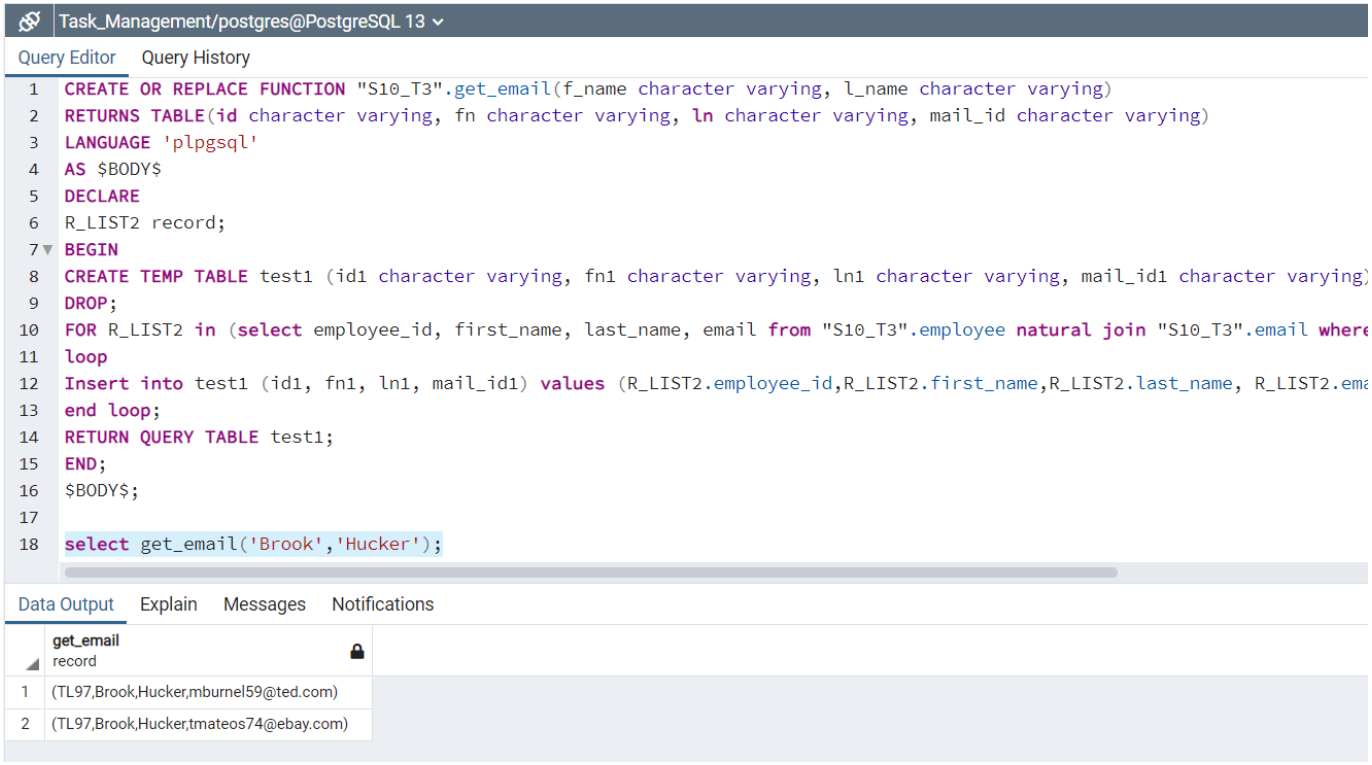
**37. This function returns the email address given the first name and the last name of the person**

**Function code:**

```
CREATE OR REPLACE FUNCTION "S10_T3".get_email(f_name character varying, l_name character varying)
RETURNS TABLE(id character varying, fn character varying, ln character varying, mail_id character varying)
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
R_LIST2 record;
BEGIN
CREATE TEMP TABLE test1 (id1 character varying, fn1 character varying, ln1 character varying, mail_id1 character varying) ON COMMIT
DROP;
FOR R_LIST2 in (select employee_id, first_name, last_name, email from "S10_T3".employee natural join "S10_T3".email where first_name=f_name and last_name=l_name)
loop
Insert into test1 (id1, fn1, ln1, mail_id1) values (R_LIST2.employee_id,R_LIST2.first_name,R_LIST2.last_name, R_LIST2.email);
end loop;
RETURN QUERY TABLE test1;
END;
$BODY$;
```

**Function call** (SQL query):

```
select get_email('Brook','Hucker');
```

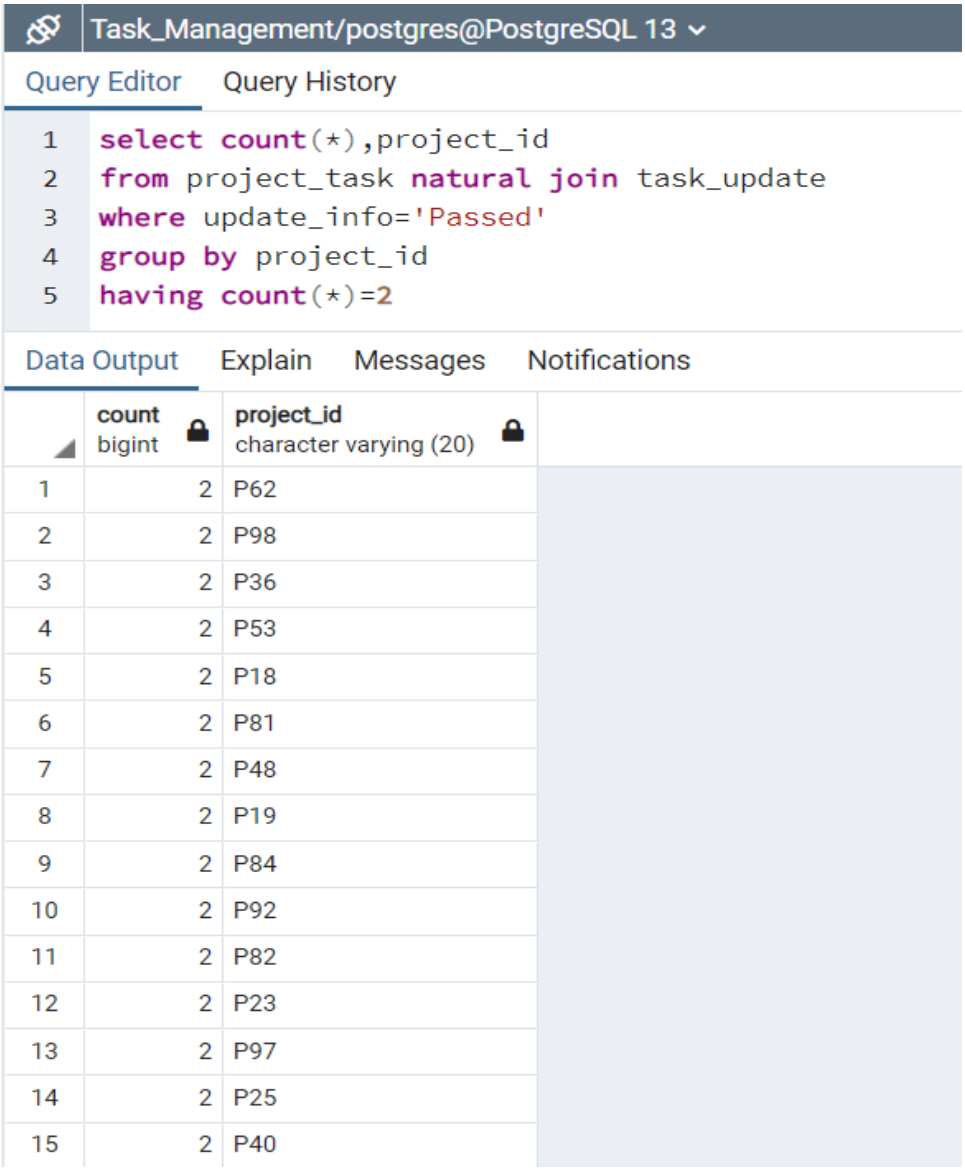



Tuples: 2

38. **Select all projects where both its subtasks have passed.**

**SQL Query:**

```
select count(*),project_id
from project_task natural join task_update where update_info='Passed'
group by project_id
having count(*)=2
```




| | count bigint | project_id character varying (20) |
|---|---|---|
| 1 | 2 | P62 |
| 2 | 2 | P98 |
| 3 | 2 | P36 |
| 4 | 2 | P53 |
| 5 | 2 | P18 |
| 6 | 2 | P81 |
| 7 | 2 | P48 |
| 8 | 2 | P19 |
| 9 | 2 | P84 |
| 10 | 2 | P92 |
| 11 | 2 | P82 |
| 12 | 2 | P23 |
| 13 | 2 | P97 |
| 14 | 2 | P25 |
| 15 | 2 | P40 |

Tuples: 27

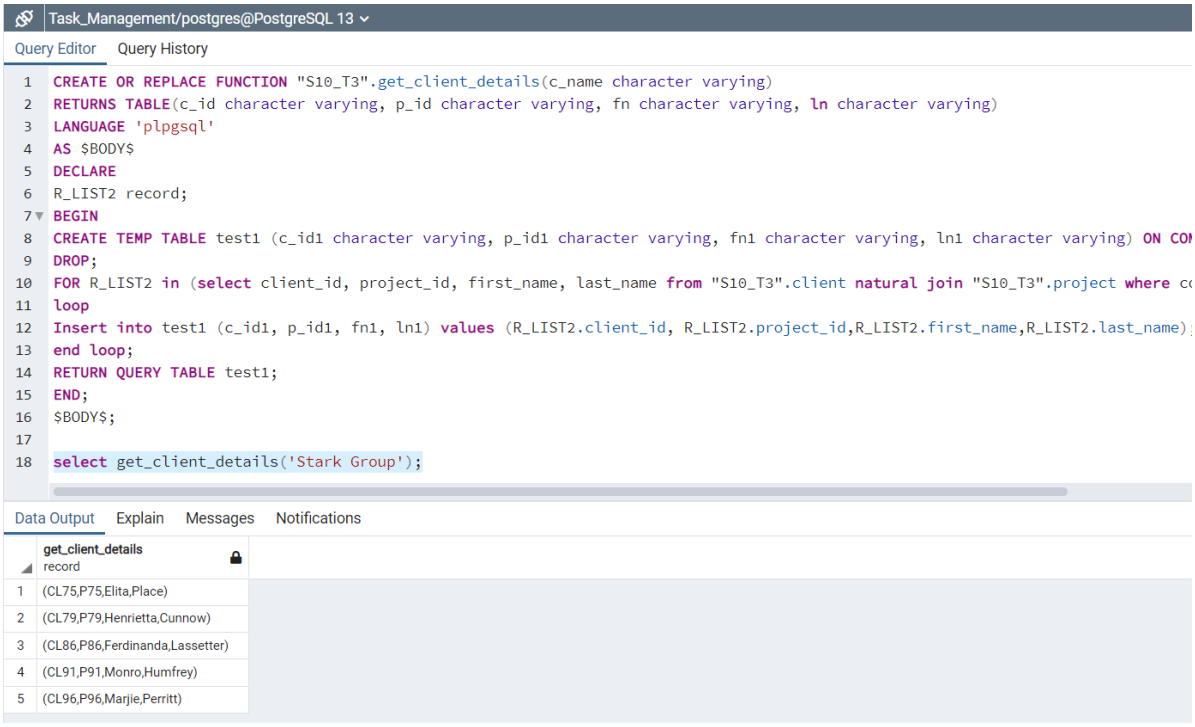
**39. Create a function that returns the details of all clients of a particular company as passed as the input parameter in the function.**

**Function code:**

```
CREATE OR REPLACE FUNCTION "S10_T3".get_client_details(c_name character varying)
RETURNS TABLE(c_id character varying, p_id character varying, fn character varying, ln character varying)
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
R_LIST2 record;
BEGIN
CREATE TEMP TABLE test1 (c_id1 character varying, p_id1 character varying, fn1 character varying, ln1 character varying) ON COMMIT
DROP;
```

FOR R_LIST2 in (select client_id, project_id, first_name, last_name from "S10_T3".client natural join "S10_T3".project where comp_name=c_name)
loop
Insert into test1 (c_id1, p_id1, fn1, ln1) values (R_LIST2.client_id, R_LIST2.project_id,R_LIST2.first_name,R_LIST2.last_name);
end loop;
RETURN QUERY TABLE test1;
END;
$BODY$;

**Function call (SQL Query)**:
select get_client_details('Stark Group');

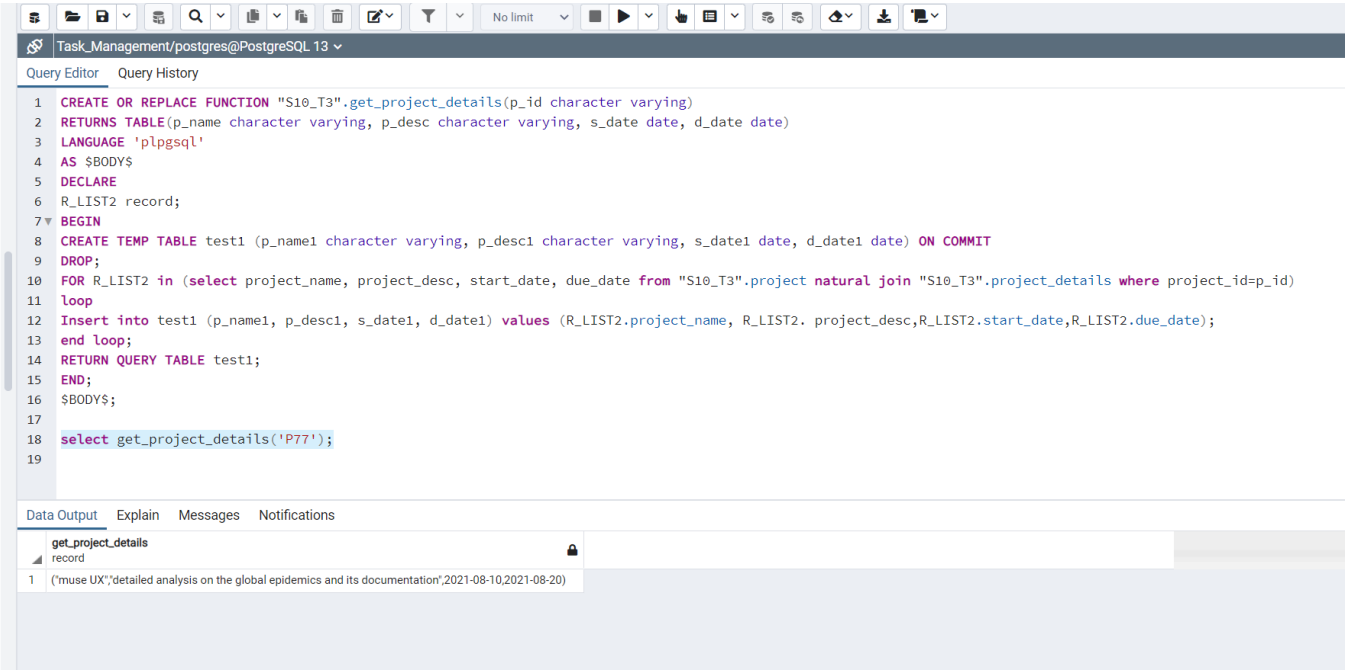



**40. Create a function that returns the details of the project id passed as the argument to the function.**

**Function Code:**
CREATE OR REPLACE FUNCTION "S10_T3".get_project_details(p_id character varying)
RETURNS TABLE(p_name character varying, p_desc character varying, s_date date, d_date date)
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
R_LIST2 record;
BEGIN
CREATE TEMP TABLE test1 (p_name1 character varying, p_desc1 character varying, s_date1 date, d_date1 date)
ON COMMIT
DROP;
FOR R_LIST2 in (select project_name, project_desc, start_date, due_date from "S10_T3".project natural join "S10_T3".project_details where project_id=p_id)

loop
Insert into test1 (p_name1, p_desc1, s_date1, d_date1) values (R_LIST2.project_name, R_LIST2.
project_desc,R_LIST2.start_date,R_LIST2.due_date);
end loop;
RETURN QUERY TABLE test1;
END;
$BODY$;

**Function_Query:**
select get_project_details('P77');



# Section7: Project Code for Frontend

**Importing Required Libraries:**

```
from ipywidgets import interact,interactive,fixed
import ipywidgets as widgets
from IPython.display import display
import pandas as pd
import PySimpleGUI as sg
import numpy as np
import psycopg2
```

**Connection Establishment with postgres database:**

```
def db_connection():
    connection=psycopg2.connect(host='localhost',database="task_management", user="postgres", password="admin")
    cursor=connection.cursor()
    return connection,cursor
```

**Main function for queries in input/output operations:**

```
def func(choice):
    if choice=='select your choice':
        pass
    elif choice=='recent projects':
        project_id=[]
        client_id=[]
        project_manager=[]
        connection,cursor=db_connection()
        sd=simple_gui("Enter the start date (yyyy-mm-dd)")
        dd=simple_gui("Enter the due date (yyyy-mm-dd)")
        cursor.execute(f'SELECT "S10_T3".recent_project(\'{str(sd)}\',\'{str(dd)}\')')
        rows=cursor.fetchall()

        for i in range(len(rows)):
            t=tuple(list(rows[i])[0][1:-1].split(','))
            project_id.append(t[0])
            client_id.append(t[1])
            project_manager.append(t[2])

        df=pd.DataFrame(np.transpose([project_id,client_id,project_manager]),list(range(1,len(rows)+1)),['Project ID','Client
ID','Project Manager'])
        display(df)

    elif choice=='client details of a particular company':
        client_id=[]
        project_id=[]
        first_name=[]
        last_name=[]
        connection,cursor=db_connection()
        comp_name=simple_gui("Enter the name of the company: ")
        cursor.execute(f'SELECT "S10_T3".get_client_details(\'{str(comp_name)}\')')
```

```
        rows=cursor.fetchall()

        for i in range(len(rows)):
            t=tuple(list(rows[i])[0][1:-1].split(','))
            client_id.append(t[0])
            project_id.append(t[1])
            first_name.append(t[2])
            last_name.append(t[3])

        df=pd.DataFrame(np.transpose([client_id,project_id,first_name,last_name]),list(range(1,len(rows)+1)),['Client ID','Project ID','First Name','Last Name'])
        display(df)

    elif choice=='task details for a particular category':
        task_id=[]
        start_date=[]
        due_date=[]
        task_name=[]
        connection,cursor=db_connection()
        task_category=complex_gui("Select the task category from the menu ")
        cursor.execute(f'SELECT  task_id, start_date, due_date, task_name from "S10_T3".task WHERE task_category=\'{str(task_category)}\'')
        rows=cursor.fetchall()

        for t in rows:
            task_id.append(t[0])
            start_date.append(t[1])
            due_date.append(t[2])
            task_name.append(t[3])

        df=pd.DataFrame(np.transpose([task_id,start_date,due_date,task_name]),list(range(1,len(rows)+1)),['Task ID','Start Date','Due Date','Task Name'])
        display(df)

    elif choice=='insert into client table':
        connection,cursor=db_connection()
        values=[]
        values.append(simple_gui("Enter the Client ID (unique): "))
        values.append(simple_gui("Enter First Name: "))
        values.append(simple_gui("Enter Last Name: "))
        values.append(simple_gui("Enter Company Name: "))

        cursor.execute(f'insert into "S10_T3".client values(\'{str(values[0])}\', \'{str(values[1])}\', \'{str(values[2])}\', \'{str(values[3])}\')')
        print("Value Inserted")
        connection.commit()
```

**Function for implementing simple gui:**

```
def simple_gui(text):
```

```
    layout = [[sg.Text(text), sg.InputText()],
              [sg.Button('OK'), sg.Button('Cancel')]]
    # Create the Window
    window = sg.Window('Enter data', layout)
    # Create the event loop
    while True:
        event, values = window.read()
        if event in (None, 'OK'):
            # User closed the Window or hit the Cancel button
            break
    window.close()
    return values[0]
```

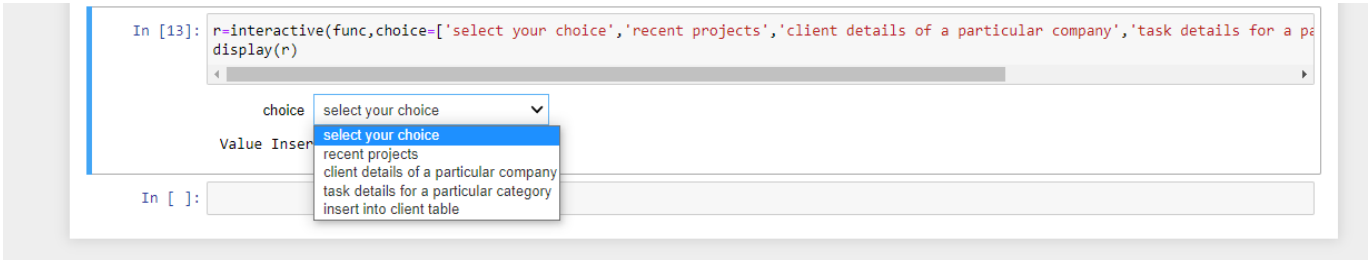
**Function for implementing complex gui:**

```
def complex_gui(text):
    layout = [[sg.Text(text), sg.Combo(["data visualization","testing","website hosting","statistics","data
analysis","invoice","UI/UX","documentation","presentation","front end","back end"])],
              [sg.Button('OK'), sg.Button('Cancel')]]
    # Create the Window
    window = sg.Window('Hello PySimpleGUI', layout)
    # Create the event loop
    while True:
        event, values = window.read()
        if event in (None, 'OK'):
            # User closed the Window or hit the Cancel button
            break
    window.close()
    return values[0]
```

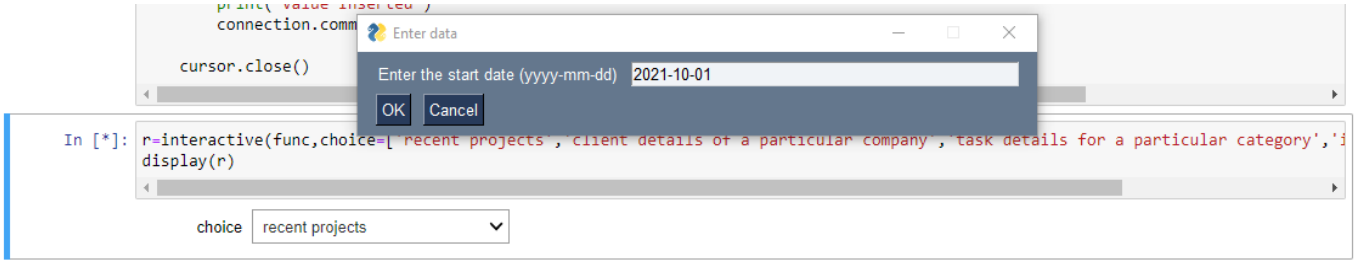
**Code for calling the interactive function for the GUI's**

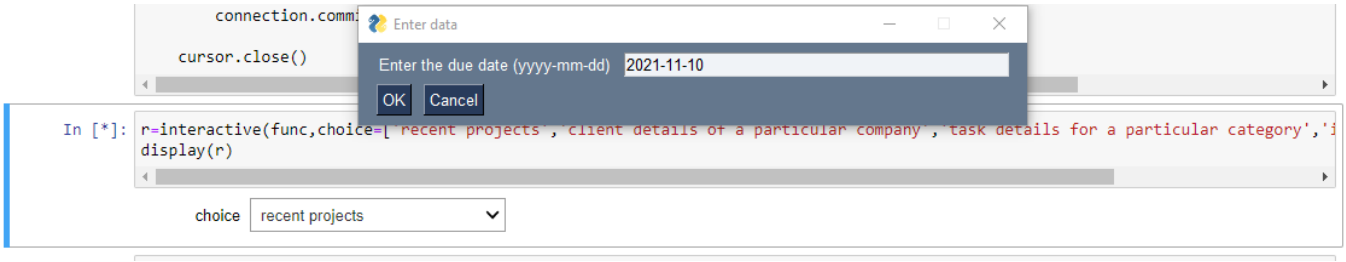
```
r=interactive(func,choice=['select your choice','recent projects','client details of a particular company','task details for a
particular category','insert into client table'])
display(r)
```

**Drop-down Menu for the 4 functions:**




**Function implementation of 'recent projects':**

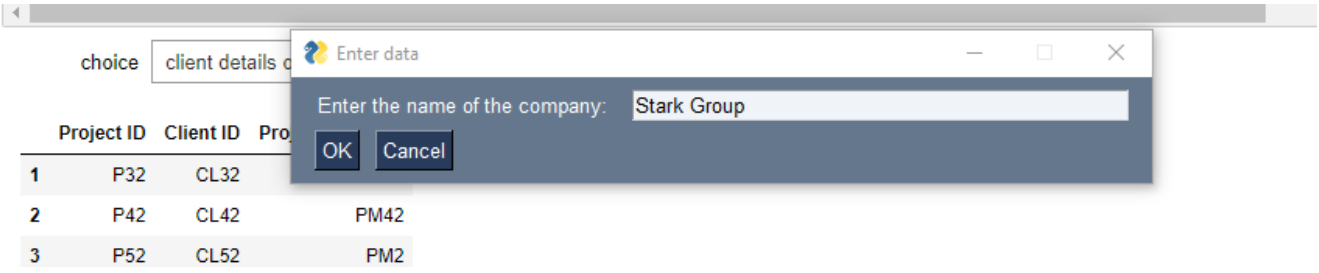




| | Project ID | Client ID | Project Manager |
|---|---|---|---|
| 1 | P32 | CL32 | PM32 |
| 2 | P42 | CL42 | PM42 |
| 3 | P52 | CL52 | PM2 |
| 4 | P54 | CL54 | PM4 |
| 5 | P90 | CL90 | PM40 |

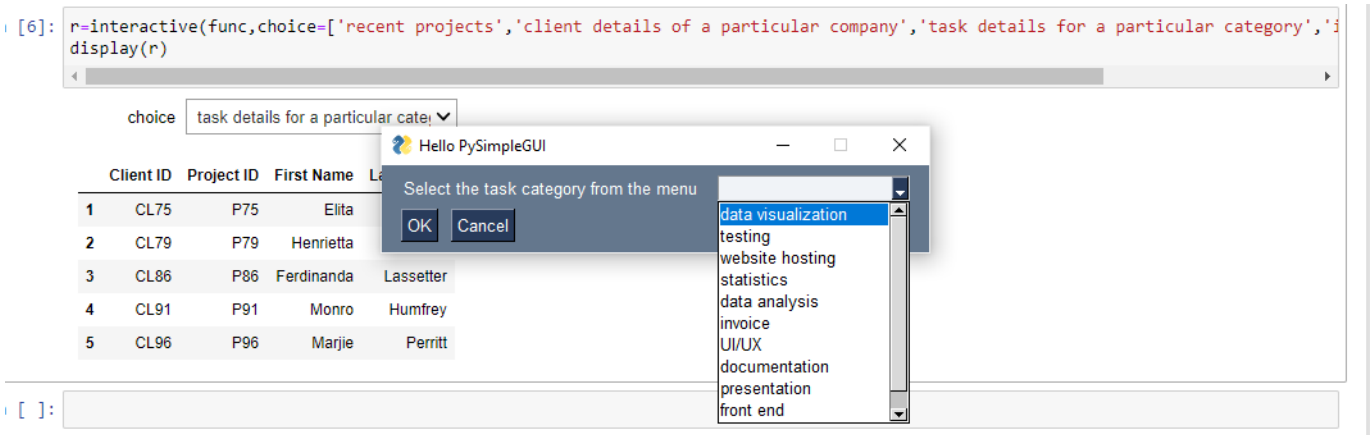
**Function implementation of 'client details of a particular company'**




choice: client details of a particular com

| | Client ID | Project ID | First Name | Last Name |
|---|---|---|---|---|
| 1 | CL75 | P75 | Elita | Place |
| 2 | CL79 | P79 | Henrietta | Cunnow |
| 3 | CL86 | P86 | Ferdinanda | Lassetter |
| 4 | CL91 | P91 | Monro | Humfrey |
| 5 | CL96 | P96 | Marjie | Perritt |

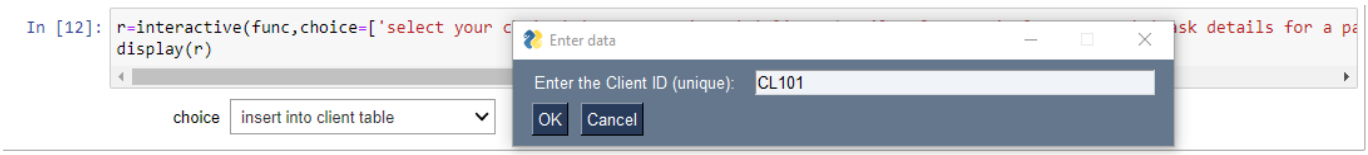
**Function implementation of task details for a particular task category:**

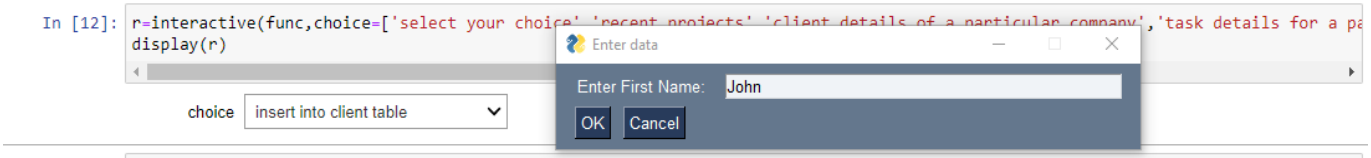


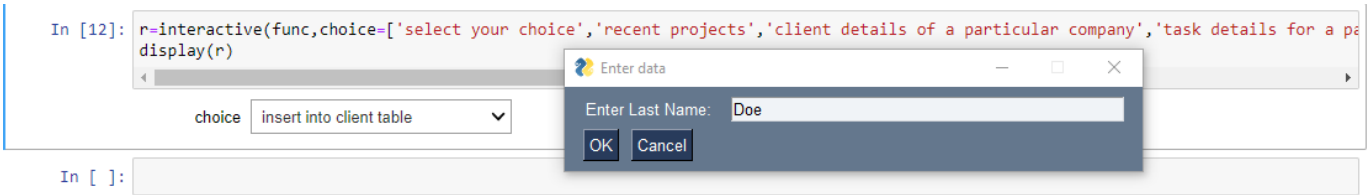
choice task details for a particular cate

| | Task ID | Start Date | Due Date | Task Name |
|---|---|---|---|---|
| 1 | T15 | 2021-09-30 | 2021-10-19 | Home Ing |
| 2 | T121 | 2019-12-10 | 2019-12-27 | Cookley |
| 3 | T122 | 2021-05-24 | 2021-06-24 | Span |
| 4 | T128 | 2020-02-01 | 2020-02-08 | Vagram |
| 5 | T129 | 2021-05-01 | 2021-05-11 | Temp |
| 6 | T36 | 2020-06-24 | 2020-06-28 | Tempsoft |
| 7 | T44 | 2020-11-25 | 2020-12-10 | Job |
| 8 | T47 | 2020-12-20 | 2021-01-01 | Bytecard |
| 9 | T5 | 2021-01-11 | 2021-02-01 | Transcof |
| 10 | T54 | 2021-09-18 | 2021-10-02 | Lotlux |
| 11 | T55 | 2019-11-23 | 2019-12-03 | Biodex |
| 12 | T158 | 2020-06-24 | 2020-07-19 | Ronstring |
| 13 | T160 | 2020-05-23 | 2020-06-02 | Hatity |
| 14 | T79 | 2021-04-05 | 2021-04-14 | Latlux |
| 15 | T185 | 2020-04-05 | 2020-05-05 | Aerified |
| 16 | T189 | 2021-03-27 | 2021-04-01 | Bytecard |
| 17 | T192 | 2021-03-04 | 2021-04-04 | Bitwolf |

**Implementation of inserting a value in the client table:**

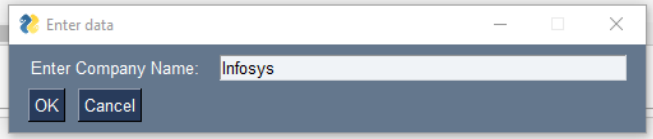


```
In [12]: r=interactive(func,choice=['select your choice','recent projects','client details of a particular company','task details for a p
         display(r)
```

choice: insert into client table




```
In [ ]:
```

```
In [13]: r=interactive(func,choice=['select your choice','recent projects','client details of a particular company','task details for a pa
         display(r)
```

choice: insert into client table

```
Value Inserted
```

Explain | Messages | Notifications | Data Output

| | client_id [PK] character varying (20) | first_name character varying (30) | last_name character varying (30) | comp_name character varying (100) |
|---|---|---|---|---|
| 95 | CL95 | Maurene | Pleasance | Bednar, Luettgen and Ankunding |
| 96 | CL96 | Marjie | Perritt | Stark Group |
| 97 | CL97 | Marja | Gilbey | Lakin and Sons |
| 98 | CL98 | Irwin | Breawood | Franecki and Sons |
| 99 | CL99 | Corny | Pumphreys | Bauch, Powlowski and Schiller |
| 100 | CL100 | Andrea | Peile | Stoltenberg Group |
| 101 | CL101 | John | Doe | Infosys |
| | | | | |