



IT214 - DBMS
Final Project Report Submission

Case Study Database: Laborlist and Wages Database

Prepared by
Team No. 17 | Group 6 | Section 10

Group Members
Dhruv Patel - 202001271
Jaykumar Navadiya - 202001465

Under the guidance of
Prof. Chhaya
Teaching Assistant - Vinay Maharaj

Date
November 23, 2022

Index

1. [Introduction](#)
 - 1.1. [Purpose](#)
 - 1.2. [Intended Audience and Reading Suggestions](#)
 - 1.3. [Product Scope](#)
 - 1.4. [Description](#)
 - 1.4.1. [Product Perspective](#)
 - 1.4.2. [Product Functions](#)
 - 1.4.3. [User Classes and Characteristics](#)
 - 1.4.4. [Operating Environment](#)
2. [Document the Requirements Collection/ Fact-Finding Phase](#)
 - 2.1. [Readings and Description](#)
 - 2.2. [Interviews](#)
 - 2.3. [Questionnaires](#)
 - 2.4. [Observations](#)
3. [Fact Finding Chart](#)
4. [List requirements](#)
5. [User categories and privileges](#)
6. [Assumptions](#)
7. [Business Constraints](#)
8. [References used in making this SRS](#)
9. [Final Problem Description](#)
 - 9.1. [Project Purpose](#)
 - 9.2. [Product Perspective](#)
 - 9.3. [Product Functions and General overview of the system](#)
 - 9.4. [User Classes, Privileges and Characteristics](#)
 - 9.5. [Operating Environment](#)
10. [Noun \(& Verb\) Analysis](#)

- 10.1. Extracted Nouns & Verbs from Problem Description using Noun Analysis Method
 - 10.2. Criteria for Truncating Initial Noun List
 - 10.3. Table 2: Accepted Nouns List
 - 10.4. Table 3: Rejected Nouns list with the reason of rejection
- 11. Developing E-R Diagram**
- 11.1. Version 1 of the E-R Diagram based on the nouns listed in Table 2.
 - 11.2. Version 2 of the E-R Diagram
 - 11.3. Final Version of the E-R Diagram
- 12. ER Diagram Final Version**
- 13. E-R Model to Relational Model Mapping**
- 14. DDL Scripts**
- 15. Normalization & Schema Refinement**
- 15.1 Relations and Schemas
 - 15.2 Types of Dependencies
 - 15.3 Schema Investigation
 - 15.3.1 List of Redundancies
 - 15.3.2 List of update, delete and insert anomalies
 - 15.4 1NF Normalization
 - 15.5 2NF Normalization
 - 15.6 3NF/BCNF Normalization
 - 15.7 Final Relations
- 16. Re-write DDL Scripts**
- 16.1 DDL Scripts for New Design (3NF/BCNF Form)
 - 16.2 DDL Snapshots
 - 16.3 Data Snapshots
- 17. SQL Queries**
~20 SQL queries and their outputs for the Case Study.
- 18. Website with CRUD functionalities and Other Features**
- 18.1 Functionalities



IT214 - DBMS
Final SRS Submission

Case Study Database: Laborlist and Wages Database

Prepared by
Team No. 17 | Group 6 | Section 10

Group Members
Dhruv Patel - 202001271
Jaykumar Navadiya - 202001465

Under the guidance of
Prof. Chhaya
Teaching Assistant - Vinay Maharaj

Date
September 30, 2022

1. Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of the database of LaborList and Wages. The database will maintain a record of all work done by laborers on a daily basis to store and distribute wages. It will explain the purpose and features of the database such as the schema and relations of the database, how the Labor Supervisor will benefit from using such a database instead of using a file management system, the constraints under which the database must operate (i.e. the constraints on some attributes). The database will be a part and backend of the web application (PERN stack) consisting of features like managing, hiring and scheduling of laborers and providing a dashboard to the firms with the help of the database at the backend.

1.2. Intended Audience and Reading Suggestions

This document is intended for users of the database, the labor supervisor, the company that hired the laborers and labor management teams, the government officials from the ministry of labor and employment, people who need labor services from time to time and want to hire laborers, firms looking for a large workforce.

The document is also intended for the laborers who are looking for jobs and want to make a great profile for themselves (Such laborers can be plumbers, electricians, maids, therapists, home painters or butlers), freelancers who gets paid on the work and the quality of work they do (designers, web developers, photographers) and also developers working on the database that might improve the database, add some functionalities later on to the database or fix some existing bugs in the database.

1.3. Product Scope

The LaborList and Wage database will provide an easy management system for all the businesses who want to manage a workforce (laborers), laborers who want to get hired and who wants to hire someone to get their work done. The Web Application will also have hiring features for normal users who will hire laborers depending on their requirements. The application will support many cities and districts of India and many categories of laborers.

The firms will be provided with an interactive dashboard for managing laborers. The application will provide an easy way for laborers to get hired based on their profiles. Their profiles will have their work experience in the field, ratings and comments of their previous work. The goal is to make an end to end system/application for businesses, laborers and normal users to have an idea of what's going on in their company while making it easy to pay employees/laborers and supervise them, creating job opportunities and making it easier for the users to hire and get their work done faster than any other conventional way.

Labor supervisors' can also be managed by firms with the help of the app and supervisors can see all aspects of the labor's work including start time of the work, end time, the type of work they are doing and availability of the labor. Supervisors will also be able to pay the laborers without calculating their wages because of the automatic wage distribution feature of the database.

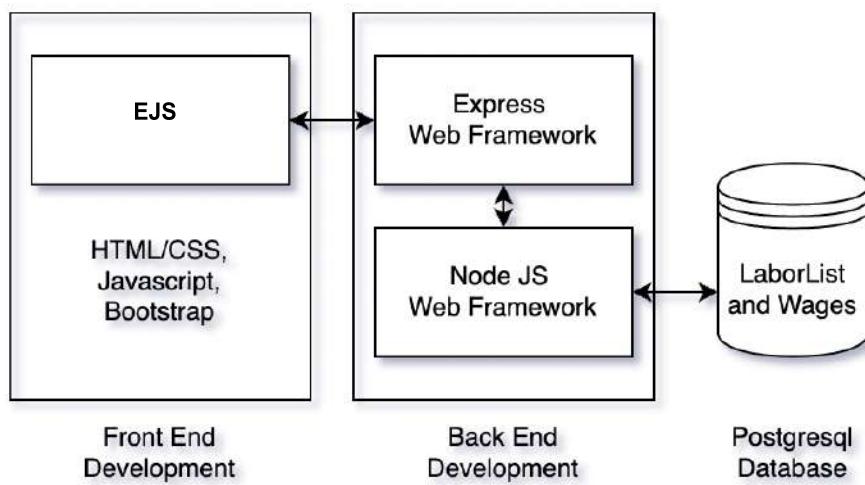
Also, special features will be implemented for the ministry of labor and employment of India such as a monitoring system that indicates any red flags towards anyone using the application and disobeying the rules and regulations.

1.4. Description

1.4.1. Product Perspective:

This database is developed for managing the workforce, getting hiring opportunities for job finders and hiring a workforce for recruiters. The database can be used and integrated with a web application, a mobile or a desktop application with the help of APIs. We will be integrating the database in our web application which uses the PEEN (PGSQL, ExpressJS,EJS, NodeJS) stack.

Below is the diagram explaining the PEEN Stack.



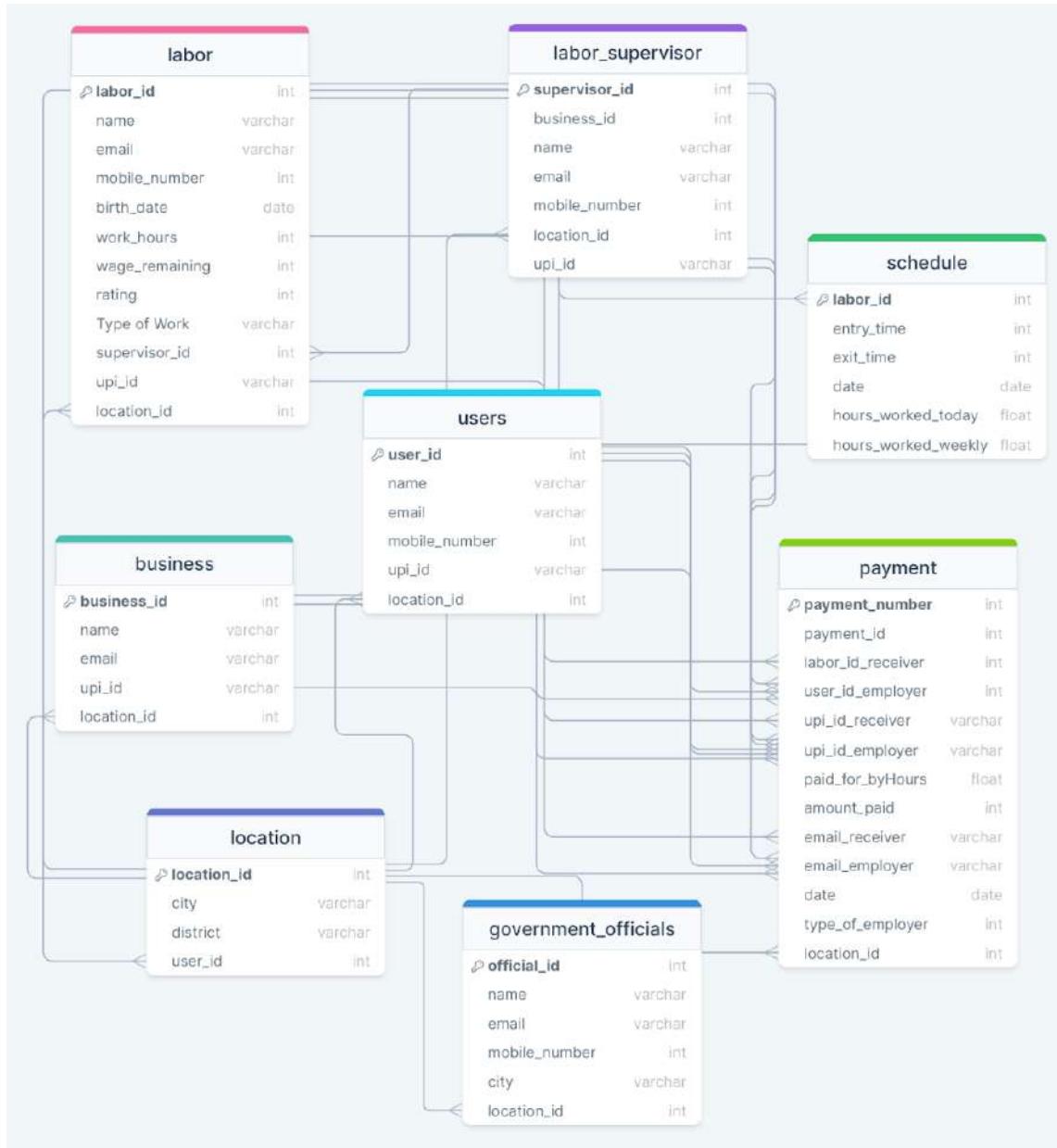
1.4.2. Product Functions:

There are various functions and features the database has. Some of the Application functions are listed below.

1. Add and delete laborers - adds or deletes the labor from the database and also saves personal details of the labors.
2. Add or delete a laborers under a supervisor - adds or deletes the labor from the supervision of a particular supervisor. (Available only for business head)

3. Show working laborers - shows the currently working laborers in the dashboard. (Available only for Supervisor head and Business head)
4. Admin - allows the developer to see the database with dummy entries and debugging features. Also, shows all the data of the database as required.
5. History of laborers - shows the history of the laborers working under a particular supervisor or a firm with the time stamps. (Available only for Supervisor head and Business head)
6. Supervisor List - Shows all the supervisors working in a business. (Available only for Supervisor head and Business head)
7. Schedule Labors (start time, end time) - gives the optimal list of employees for working in the provided time slot based on their productivity and time slot preferences.
8. Pay Laborers (Date) - distribute the wage to the laborers who worked on the given date according to their working hours.
9. Pay Laborers (Week) - only pay the laborers who worked the whole week and opted for a weekly wage option.
10. Leaves (start time, end time) - shows the list of laborers who took a leave in a given interval.
11. Hire (Type of work) - hires a specific type of worker according to an individual's needs. (for individuals)
12. Book (Type of work) - book a laborer with specific skills. (for individuals)
13. Remarks and Ratings - Give ratings to the laborers for fair hiring and wages. (For individuals and firms)
14. Automatic Payments - Automatic payments for laborers once the work is done. The money will automatically be deducted from the employer's account.
15. Bank Information - Shows the bank details of any class of users for payment purposes.

The initial version of the ER model is given below. That shows the schema of the database.



- Everything is joined by Location ID, it is foreign key in the tables in which it is present.
- Labor ID, Business ID, Supervisor ID, User ID are all primary keys in their table and foreign keys in other tables.
- The arrow ending with 3 edges () is foreign key (many to one relation) in that table and the attribute that starts with the arrow is the primary key in that table. Primary key is also indicated with the key symbol ().

1.4.3. User Classes and Characteristics:

3 types of user classes will be using the database: Hiring Managers and Firms (Labor Supervisor), Individual recruiters who will hire on a temporary basis, Laborers who want to get hired by individuals or by firms. Every type of class is equally likely to use the database. While the frequency of use might change. A firm will hire once in a while but a normal user might hire more frequently for smaller tasks like cleaning, plumbing.

1.4.4. Operating Environment:

The Database is made using postgresql and will be compatible with any operating system like Windows, MACOS X, Linux, Android, iOS and iPad OS because it can be integrated with any of the applications by using Node.JS or any other backend services or development environments such as Android Studio and XCode.

2. Document the Requirements Collection/ Fact-Finding Phase

2.1. Background Readings

There are many current workforce management systems like workforce.com, easymetrics.com and many sites for hiring laborers for individual employers such as urbancompany.com. Also, The Gazette of India has published the Code on Wages by Authority under the Ministry of Law and Justice as an act to amend and consolidate the laws relating to wages and bonus and matters connected therewith or incidental thereto.

- 1. Workforce.com:** It schedules laborers according to their convenience and the employer's needs. It provides insights on labor's productivity and shows their work on a dashboard provided to the supervisor. It also

integrates the payroll within the system. So the employees are paid automatically without the company having to manually pay them. Though normal users cannot hire laborers from the website, we have a feature of labor profile, that allows the laborer data to be used by the system and recommend the laborers to normal users.

2. [Urbancity.com](#): It has very nice features such as hiring any type of laborers individually. But it has no dashboard feature for labor supervisors or firms that have to manage a large number of laborers. It also has no weekly payment or automatic payroll integrations.
3. [Easymetrics.com](#): This website is a fully featured labor management system that has good machine learning algorithms but has no support for hiring capabilities. Nor does it have scheduling capabilities. Our database can be used for hiring, scheduling and for automatic payroll on hourly or weekly basis.

References used in this part of the SRS:

1. Urban Company Website: www.urbancity.com
2. Workforce Website: <https://workforce.com/>
3. Easymetrics Website: easymetrics.com

List of requirements from background readings:

Need a database that can automatically pay the laborers and supervisors, have the ability to allocate the laborers to particular jobs faster than conventional methods and also provide job opportunities to the laborers while providing hiring ease for individual employers.

2.2. Interviews

Interview 1 (with supervisor of canteen workers - Role Play):

Interview plan:

System: Laborlist and Wages Database

Project Reference: LLWD/2022/10

Interviewee: Rahil, Designation: Supervisor - Canteen Workers

Interviewer: Jay Navadiya, Designation: Developer - LaborList and Wages

Date: September 27, 2022 Time: 5:00 PM

Duration: 25 Minutes Place: Canteen, DA-IICT

Purpose of the interview: To find problems and requirements of the Labor Supervisor side of the database LaborList and Wages.

Agenda/Questions prepared to ask in the interview:

1. How do you keep track of the working hours of the workers?
2. How do you distribute wages between the workers?
3. If manually, would you want an automated system that takes care of the payment on time to time without any hurdles?
4. How do you keep track of the productivity of the workers?
5. How do you schedule the workers for the job that you give them?
6. If a system is developed with features such as automatic scheduling, would you use and benefit from it?

Interview summary:

Results of the Interview:

- Supervisors cannot always keep track of the work hours of the laborers.

- Wage distribution has to be done manually and it's a burden if the work hours are not tracked properly.
- Productivity of the laborers is impossible to track manually.
- Scheduling the laborers to their preferences is hard without using a scheduling algorithm.
- The supervisors at the canteen do not have any existing labor management systems either. They would love to use such a system.

Interview 2 (with one of the labor of the canteen - Role Play):

Interview plan:

System: Laborlist and Wages Database

Project Reference: LLWD/2022/10

Interviewee: Priyansh, Designation: Canteen worker (chef)

Interviewer: Dhruv Patel, Designation: Developer - LaborList and Wages

Date: September 28, 2022 Time: 3:00 PM

Duration: 40 Minutes Place: Canteen, DA-IICT

Purpose of the interview: To find problems and requirements of the Labor Supervisor side of the database LaborList and Wages.

Agenda/Questions prepared to ask in the interview:

1. Do you get paid by cash or by online banking?
2. And if you get paid by online banking, is it done automatically, regularly?
3. Do you get the fair wage for the hours of work that you do?
4. Is the work hours managed? If yes, is it managed/updated manually by the supervisor or in some structured better way?

5. Do you think that an automated payment system will help you in contrast to the conventional payment system that is implemented right now?
6. Can you shift your job and get another job that has more wages easily with your current job? How hard is it?
7. Will you work for individual employers after regular working hours for getting more wage and helping individuals?
8. Can you keep track of your working hours and productivity of work that you do?

Interview summary:

Results of the Interview:

- Employees/ Laborers get paid in cash and it makes it hard for them to remember if they were paid fully or not for the work they have done.
- It would help the laborers in many ways if the payroll system gets automated as sometimes the supervisor doesn't give the wage on time.
- Having a labor profile and rating system would really help the laborers to get new jobs (part time and full time) and also to get higher wages based on their performance.

Interview 3 (with one of the government official - Role Play):

Interview plan:

System: Laborlist and Wages Database

Project Reference: LLWD/2022/10

Interviewee: Mr. Patel, Designation: Secretary - ministry of labor and employment

Interviewer: Dhruv Patel, Designation: Developer - LaborList and Wages

Date: September 27, 2022 Time: 9:00 PM

Duration: 40 Minutes Place: Canteen, DA-IICT

Purpose of the interview: To find problems and requirements of the government official side of the database LaborList and Wages.

Agenda/Questions prepared to ask in the interview:

1. Do you need a system to manage all the employers/laborers in a particular region?
2. Does it happen that the data represented in the papers are found to be wrong later on?
3. Is live tracking of the employers/laborers helpful to the ministry of labor and employment?

Interview summary:

Results of the Interview:

- There is a need for supervision of the age of the laborers as there are many child laborers currently working in many companies.
- Real time tracking of labor activities could help in regulating the labor laws and also it helps the government officials to not go from the paperwork submitted by the companies as it's faked countless times before. Having a real time tracking feature could prevent all that.

Interview 4 (with a normal citizen - actual):

Interview plan:

System: Laborlist and Wages Database

Project Reference: LLWD/2022/10

Interviewee: Manisha Navadiya, Designation: A teacher

Email of Interviewee: manishanavadiya11@gmail.com

Interviewer: Jay Navadiya, Designation: Developer - LaborList and Wages

Date: September 29, 2022 Time: 10:00 PM

Duration: 1 Hour Place: Canteen, DA-IICT

Purpose of the interview: To find problems and requirements of the normal users who can make use of the database of Laborlist and wages.

Agenda/Questions prepared to ask in the interview:

1. In the past, had you required any kind of labor services?
2. If yes, how hard was it to hire someone?
3. Can you get someone to do your work in under an hour?
4. Would you use an application that does the hiring process for you and find you the best laborer to do the job?
5. Has it happened that the worker you hired did not satisfactorily complete the assigned work to them?

Interview summary:

Results of the Interview:

- Normal users have to hire someone by using local connections who might not be available at times.

- They sometimes get the job done very nicely but at other times have no idea of what to do in some tasks.
- Experts of a particular field cannot always be hired so there's a requirement of such a hiring feature which lets one hire experts in just one click.

Combined observations/Final List of Requirements listed from the Interviews based on the answers given by the interviewees:

- Need an automatic payment system for the laborers instead of paying in cash or paying manually.
- Need a scheduling algorithm that does the scheduling part for the labor supervisors.
- For Labors, a dashboard is also needed for them to keep track of the work they are doing.
- For normal users, a recommendation system and easy to hire feature needs to be implemented that hires the laborers/workers according to their needs.
- For government officials, a dashboard is required for monitoring for regulation purposes in a specific area.
- Government officials also need the database as the data of laborers are many times misrepresented in the offline and more conventional format. Live tracking would be better for regulating the guidelines.

2.3. Questionnaire/s

We made a google form and it was shared between laborers, friends (normal users mostly) and labor supervisors (Canteen supervisors of DA-IICT). The form screenshot is given below.

Question 1:

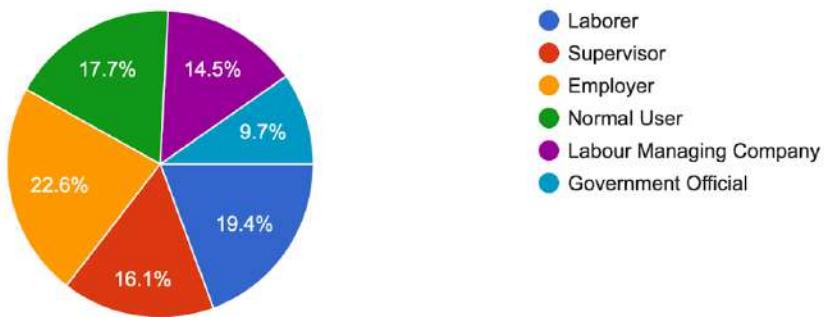
What is your role? *

- Laborer
- Supervisor
- Employer
- Normal User
- Labour Managing Company
- Government Official

Response:

What is your role?

62 responses



Intent of the question:

To get to know the major user groups of the system.

Observation from the response:

Since all the targeted users require this system, the groups are distributed almost equally with the laborers and their employers forming the majority.

Question 2:

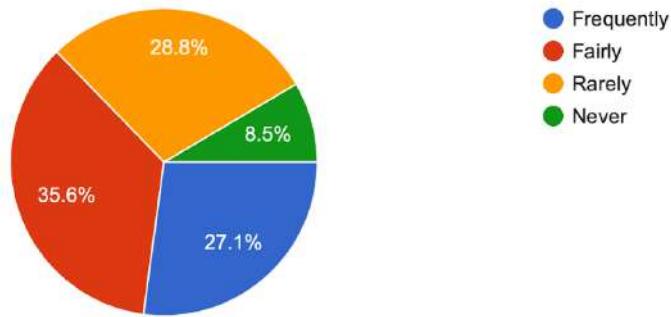
How often do you need help from laborers?
(Fill only if you are not a laborer)

- Frequently
- Fairly
- Rarely
- Never

Response:

How often do you need help from laborers?

59 responses



Intent of the question:

To gain knowledge of how often people need help with labor.

Observation from the response:

Most people require help from labor in their everyday life. There is a very small percentage of people who do not require their help.

Question 3:

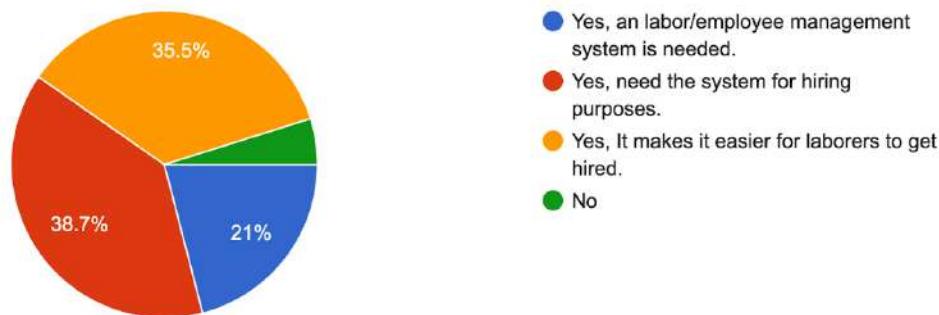
Do you feel that you need a labor/employee management system to supervise them or to simply need to hire them on the go without any delays? *

- Yes, an labor/employee management system is needed.
- Yes, need the system for hiring purposes.
- Yes, It makes it easier for laborers to get hired.
- No

Response:

Do you feel that you need a labor/employee management system to supervise them or to simply need to hire them on the go without any delays?

62 responses



Intent of the question:

The main intent of this question was to understand whether this system should be utilized for hiring purposes or not.

Observation from the response:

There is a unanimous opinion among both employers and laborers that such a system is required for hiring of laborers.

Question 4:

What is your opinion on wages of laborers?

Choose

Very High

High

Adequate

Low

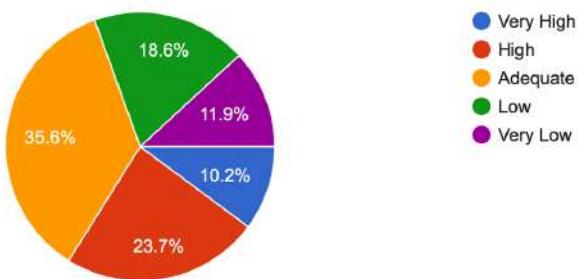
Very Low

I system for laborers? *

Response:

What is your opinion on wages of laborers?

59 responses



Intent of the question:

To get to know what is the general sentiment about the current wages of laborers among the users in order to develop the payment relation accordingly.

Observation from the response:

Relatively small percentage of people have extreme views on this question while the majority has its opinion somewhere in the middle.

Question 5:

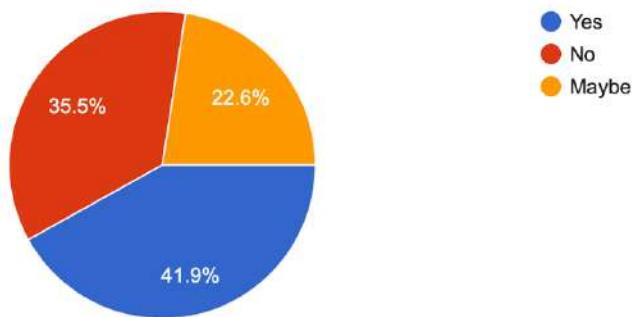
Are you in favor of tipping system for laborers? *

- Yes
- No
- Maybe

Response:

Are you in favor of tipping system for laborers?

62 responses



Intent of the question:

To get to know whether people are in favor of a tipping system for laborers and if it is required to be implemented within the system.

Observation from the response:

Not all users are in favor of the tipping system while many are divided on this topic. So, it is better to leave the tipping decision to the employer if they are impressed with the laborer's work. There is no significance of storing the data of tips received since we are already storing the ratings of hired laborers.

Question 6:

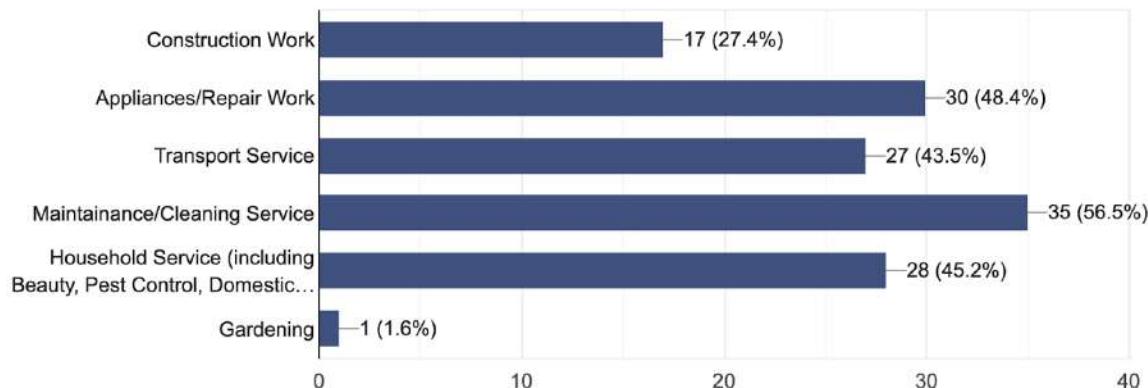
What type of labor work do you require/provide?

- Construction Work
- Appliances/Repair Work
- Transport Service
- Maintenance/Cleaning Service
- Household Service (including Beauty, Pest Control, Domestic Work, etc.)
- Other: _____

Response:

What type of labor work do you require/provide?

62 responses



Intent of the question:

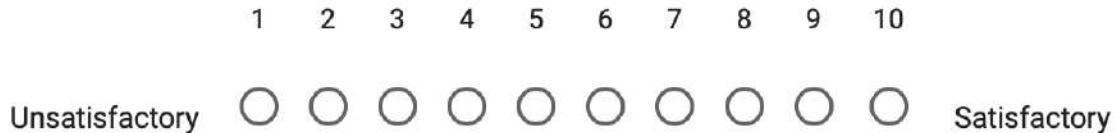
To gain knowledge of the type of work for which laborers are hired so one can define the categories for labor accordingly.

Observation from the response:

All types of labor work are required by the users almost equally. The diversity of work provided by the laborers in these general categories is enough while keeping them under the same category.

Question 7:

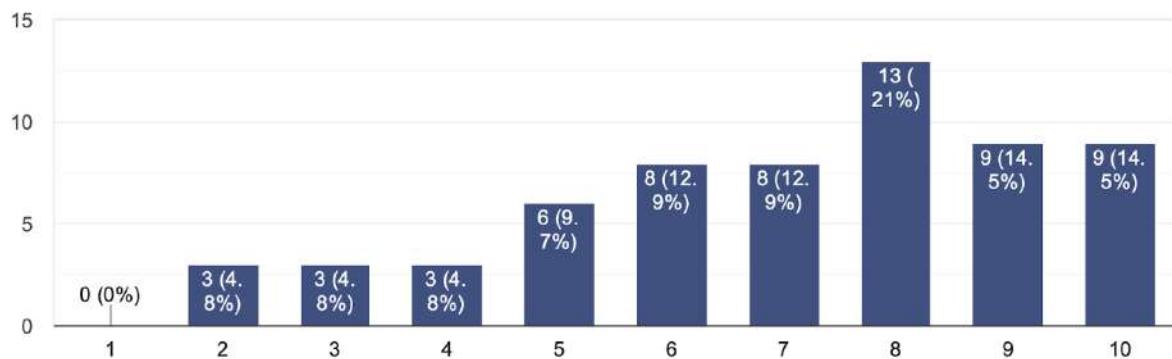
On a scale of 1 to 10, rate your experience with any previous labor management systems?



Response:

On a scale of 1 to 10, rate your experience with any previous labor management systems?

62 responses



Intent of the question:

To get to know if people were satisfied with their experience with any labor management systems they used previously.

Observation from the response:

People felt that the systems were on par with their needs but still required many improvements based on the responses.

Question 8:

Any Comments on current Labor Management Systems?

Your answer

Response:

Any Comments on current Labor Management Systems?

7 responses

Require improvement

Okay

Chaotic

Very bad

Good

Need more interaction with members

No

Intent of the question:

To get to know the user group's specific opinion on the current system of labor management.

Observation from the response:

The respondents feel the current system does not have all the required features and is quite challenging to manage.

Combined observations/Final list of requirements based on the answers of the questionnaires:

- As we can see in the responses, the users are very different from each other.
- The users are willing to use software like the one we are developing.
- No tipping system needs to be implemented for the laborers as shown in the response.
- The current labor management systems are not that responsive nor do they consist of all the features needed by the intended audience.

2.4. Observations

System: LaborList and Wages

Project Reference: LLWD/2022/10

Observation Subjects: 2 Laborers, 1 Labor Hiring Company head

Observations by: Dhruv Patel, Jay Navadiya

Date: 1/10/2022 Time: 15:00

Duration: 2 Hours Place: DAIICT Cafeteria

Observations:

- There is no proper system to collect information about availability of laborers. It needs to be filled manually by the supervisor in a register.
- There is no formal way to inform laborers about their duties. They need to be informed in-person individually.
- Even if a government official needs to be informed about the status of laborers, it is not tabulated anywhere. They need to rely on word-of-mouth.
- There is no proper method of payment to laborers. It is done on a daily basis which is not assured by the employer.
- There is no system of ratings/feedback which is used to let laborers and their employers know about their work quality.

3. Fact Finding Chart

Objective	Technique	Subject(s)	Time Commitment
To get background on current Labor Management companies and types of labor	Background Reading (Websites/Blogs)	Company Websites/Reports/blogs	1 day
To gain understanding of the current system from the subjects (laborers)	Interviews	1 Laborer	40 minutes
To understand how current Labor management works, how the payroll is managed and To establish what labor work records are kept	Interviews	1 Labor Supervisor/ Labor hiring company head	25 Minutes
To gain understanding of the Labor wages system and its business implications	Observation	2 Laborers, 1 Labor Hiring Company head	2 Hours
To find out new additional requirements to make the system streamlined and To find out how normal users hire laborers when required or if at all they require hiring laborers	Interview	1 Laborer hiring Citizen	1 Hours
To find out the needs of the government officials of the ministry of labor and employment	Interview	1 Government Official	40 Minutes

department			
To understand the user's perspective	Questionnaire	User group Respondents	3 Hours

4. List Requirements

- An automatic payment system for the laborers.
- A scheduling algorithm that does the scheduling part for the labor supervisors.
- A dashboard to keep track of the work laborers are doing.
- A recommendation system and easy to hire feature needs to be implemented to fulfill requirements.
- An algorithm needs to be developed which tracks the productivity of the laborers which lowers the load of supervisors.
- A dashboard is required for monitoring for regulation purposes in a specific area for government officials.
- Live tracking of databases for government officials to monitor the data and issue guidelines accordingly.
- Proper hardware and software for the system users to get the intended usage.

5. User Categories and Privileges

I. Laborer

Access Privileges: Relations Labor, Schedule

The Laborer will have access to all the information required by them under the two relations. They can gain knowledge of their work, payments, rating and schedules through these tables.

II. Supervisor

Access Privileges: Relations Labor, Payment, Supervisor, Location, Schedule

The Supervisor will have access to all the information required by them under these relations. They can gain knowledge of the work, payments, rating and locations of the laborers under them using these relations. Also, they would be able to schedule the laborer's work for the laborers when they are available.

III. Employer

Access Privileges: Relations Labor, Payment, Business, Supervisor, Location, Schedule

An Employer will have access to all the information required by them under these relations. They can gain knowledge of the supervisors under them and also the work, payments, rating and locations of the laborers under these supervisors using these relations. Also, they would be able to maintain the record of payments using these relations.

IV. Normal User (Citizen)

Access Privileges: Relations Labor, Normal People, Location, Schedule

A Normal Labor Hiring Person will have access to all the information required by them under these relations. They can use these to hire laborers, verify their payments, locate their laborers and also schedule the work of the laborers. Hence, they can update this information directly with the supervisor/employer.

V. Labor Managing Company

Access Privileges: Relations Labor, Payment, Business, Supervisor, Location, Schedule

A Labor Managing Company will have access to every bit of information about laborers and supervisors under the company. They can gain knowledge of the supervisors, their ids and also all the information on laborers working under these supervisors using these relations. Also, they would be able to maintain the record of payments using these relations while monitoring various businesses.

VI. Government Official

Access Privileges: Relations Labor, Business, Supervisor, Location, Government Officials

A Government Official will have access to all the businesses, their supervisors and the laborers working under them. They can use this knowledge along with their identification and location to generate a report on the laborers, their work and wage conditions. Also, they would be able to supervise all of these while being in contact with other government officials.

6. Assumptions

The following are the main assumptions made for this system:

- The user of the system has the required hardware which also fulfills the software specifications to use the system.
- The available slot of the workers is timely updated so that the supervisor and the employer can hire them accordingly.
- All the ratings and contacts provided are genuine and not used to manipulate the data in the relations.

7. Business Constraints

- All the Laborers have a registered bank account which is linked to UPI. This ID should be used to receive the payment via UPI so that the payments can be monitored.
- All the Businesses / Labor Managing Companies have a current account registered for their business which is used to carry out their business practices.
- There will be certain charges for the subscription to use the system since it needs to be hosted on a server.

8. References used for making of the SRS

1. Urban Company Website: www.urbancompany.com
2. Naukri Website: <https://www.naukri.com/>
3. Workforce Website: <https://workforce.com/>
4. Easymetrics Website: easymetrics.com
5. [IEEE Template for System Requirement Specification Documents](#)
6. Ministry of Labor and Employment, India: <https://labour.gov.in/>
7. [Airline Database SRS](#)
8. The Code on Wages, The Gazette of India:
<https://egazette.nic.in/WriteReadData/2019/210356.pdf>



IT214 - DBMS
(E-R Diagram Documentation)

Case Study Database: Laborlist and Wages Database

Prepared by
Team No. 17 | Group 6 | Section 10

Group Members
Dhruv Patel - 202001271
Jaykumar Navadiya - 202001465

Under the guidance of
Prof. Chhaya
Teaching Assistant - Vinay Maharaaj

Date
October 14, 2022

9. Final Problem Description

9.1. Purpose of the Project:

The database, Laborlist and Wages will maintain a record of all work done by laborers on a daily basis to store and distribute wages. It will explain the purpose and features of the database such as the schema and relations of

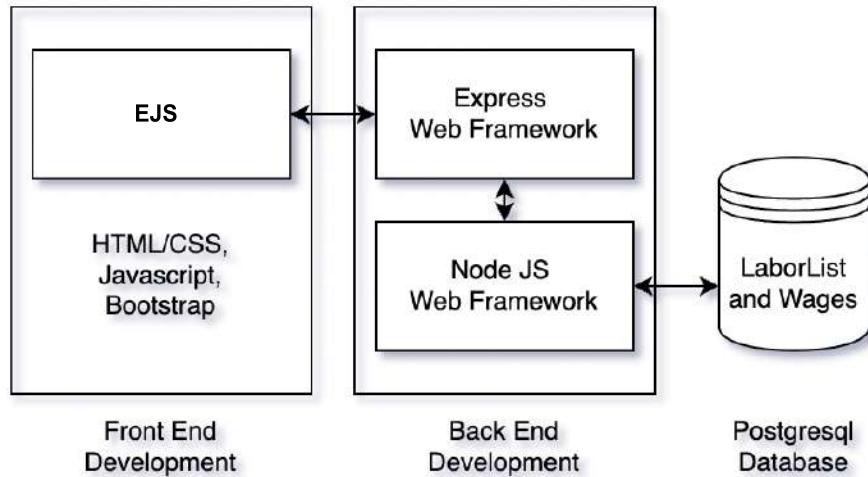
the database, how the Labor Supervisor will benefit from using such a database instead of using a file management system, the constraints under which the database must operate (i.e. the constraints on some attributes). The database will be a part and backend of the web application (PEEN stack) consisting of features like managing, hiring and scheduling of laborers and providing a dashboard to the firms with the help of the database at the backend.

The project is required for a more efficient, regulated and secure system of managing laborers and their wages. The current system which utilizes manual records and files is chaotic and not secure since data can be manipulated easily by anyone. This project makes the process completely digital and accessible to all the target user groups in order to maintain accountability.

9.2. Product Perspective:

This database is developed for managing the workforce, getting hiring opportunities for job finders and hiring a workforce for recruiters. The database can be used and integrated with a web application, a mobile or a desktop application with the help of APIs. We will be integrating the database in our web application which uses the PEEN (PGSQL, ExpressJS, EJS, NodeJS) stack.

Below is the diagram explaining the PEEN Stack.



9.3. Product Functions and General overview of the system:

Overview of the Laborlist and wages database:

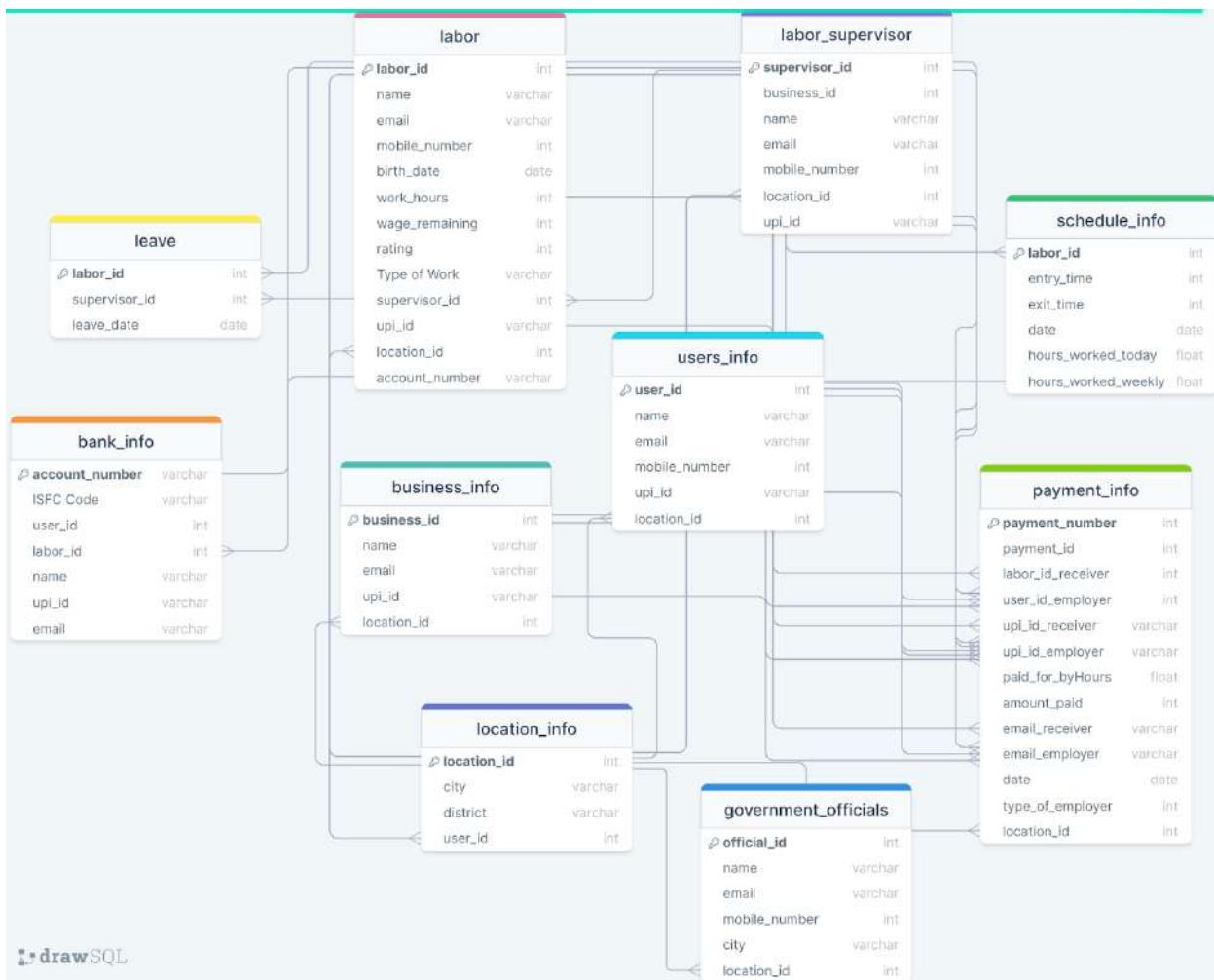
The database will consist of all the information of labor, labor supervisor, businesses who have both laborers and labor supervisors, government officials who can supervise the whole town or district with their designation in the ministry of labor and employment of India, normal users who are potential employers of the laborers who can make a profile of their own with the help of our database and software solution. Personal details may involve mobile number, email address, age and birthdate. The database will also provide solutions for scheduling of laborers and maintaining the work hours by using start and end time of working of the laborers. Also, the payroll system will be automated by using bank_details of everyone that consists of every banking related details like account number, UPI ID and IFSC Code. Every payment that happens through the platform will be stored via the payment table of the database with proper timestamps.

There are various functions and features the database has. Some of the Application functions are listed below.

1. Add and delete laborers - adds or deletes the labor from the database and also saves personal details of the labors.
2. Add or delete a laborers under a supervisor - adds or deletes the labor from the supervision of a particular supervisor. (Available only for business head)
3. Show working laborers - shows the currently working laborers in the dashboard. (Available only for Supervisor head and Business head)
4. Admin - allows the developer to see the database with dummy entries and debugging features. Also, shows all the data of the database as required.
5. History of laborers - shows the history of the laborers working under a particular supervisor or a firm with the time stamps.
(Available only for Supervisor head and Business head)
6. Supervisor List - Shows all the supervisors working in a business.
(Available only for Supervisor head and Business head)
7. Schedule Labors (start time, end time) - gives the optimal list of employees for working in the provided time slot based on their productivity and time slot preferences.
8. Pay Laborers (Date) - distribute the wage to the laborers who worked on the given date according to their working hours.
9. Pay Laborers (Week) - only pay the laborers who worked the whole week and opted for a weekly wage option.
10. Leaves (start time, end time) - shows the list of laborers who took a leave in a given interval.
11. Hire (Type of work) - hires a specific type of worker according to an individual's needs. (for individuals)
12. Book (Type of work) - book a laborer with specific skills. (for individuals)
13. Remarks and Ratings - Give ratings to the laborers for fair hiring and wages. (For individuals and firms)
14. Automatic Payments - Automatic payments for laborers once the work is done. The money will automatically be deducted from the employer's account.

15. Bank Information - Shows the bank details of any class of users for payment purposes.

The initial version of the tables with its attributes is given below. That shows the initial schema of the database.



- Everything is joined by Location ID, it is foreign key in the tables in which it is present.
- Labor ID, Business ID, Supervisor ID, User ID are all primary keys in their table and foreign keys in other tables.
- The arrow ending with 3 edges () is foreign key (many to one relation) in that table and the attribute that starts with the arrow is

the primary key in that table. Primary key is also indicated with the key symbol ().

9.4. User Classes and Characteristics:

3 types of user classes will be using the database: Hiring Managers and Firms (Labor Supervisor), Individual recruiters who will hire on a temporary basis, Laborers who want to get hired by individuals or by firms. Every type of class is equally likely to use the database. While the frequency of use might change. A firm will hire once in a while but a normal user might hire more frequently for smaller tasks like cleaning, plumbing.

User Categories and Privileges:

I. Laborer

Access Privileges: Relations Labor, Schedule

The Laborer will have access to all the information required by them under the two relations. They can gain knowledge of their work, payments, rating and schedules through these tables.

II. Supervisor

Access Privileges: Relations Labor, Payment, Supervisor, Location, Schedule

The Supervisor will have access to all the information required by them under these relations. They can gain knowledge of the work, payments, rating and locations of the laborers under them using these relations. Also, they would be able to schedule the laborer's work for the laborers when they are available.

III. Employer

Access Privileges: Relations Labor, Payment, Business, Supervisor, Location, Schedule

An Employer will have access to all the information required by them under these relations. They can gain knowledge of the supervisors under them and also the work, payments, rating and locations of the laborers under these supervisors using these relations. Also, they would be able to maintain the record of payments using these relations.

IV. Normal User (Citizen)

Access Privileges: Relations Labor, Normal People, Location, Schedule

A Normal Labor Hiring Person will have access to all the information required by them under these relations. They can use these to hire laborers, verify their payments, locate their laborers and also schedule the work of the laborers. Hence, they can update this information directly with the supervisor/employer.

V. Labor Managing Company

Access Privileges: Relations Labor, Payment, Business, Supervisor, Location, Schedule

A Labor Managing Company will have access to every bit of information about laborers and supervisors under the company. They can gain knowledge of the supervisors, their ids and also all the information on laborers working under these supervisors using these relations. Also, they would be able to maintain the record of payments using these relations while monitoring various businesses.

VI. Government Official

Access Privileges: Relations Labor, Business, Supervisor, Location, Government Officials

A Government Official will have access to all the businesses, their supervisors and the laborers working under them in certain locations. They can use this knowledge along with their identification and location to generate a report on the laborers, their work and wage conditions. Also, they would be able to supervise all of these while being in contact with other government officials.

9.5. Operating Environment:

The Database is made using postgresql and will be compatible with any operating system like Windows, MACOS X, Linux, Android, iOS and iPad OS because it can be integrated with any of the applications by using Node.JS or any other backend services or development environments such as Android Studio and XCode.

10. Noun & Verb Analysis

10.1. Extracted Nouns and Verbs from Problem Description using Noun Analysis Method

Serial no.	Nouns	Verbs
1	database	maintain
2	Laborlist	store
3	Wages	distribute

Serial no.	Nouns	Verbs
4	record	explain
5	work	benefit
6	laborers	operate
7	wages	hiring
8	schema	managing
9	relations	scheduling
10	Labor supervisor	provide
11	File	integrating
12	System	help
13	constraints	add
14	attributes	delete
15	web-application	saving
16	features	supervising
17	dashboard	showing
18	firms	allow
19	database	working
20	backend	schedule
21	product	give
22	features	pay
23	workforce	leave
24	labor	opting

Serial no.	Nouns	Verbs
25	opportunities	booking
26	desktop-application	hire
27	APIs	update
28	PEEN	consists of
29	PGSQL	access
30	ExpressJS	locate
31	EJS	verify
32	NodeJS	identify
33	stack	modify
34	diagram	insert
35	functions	sorting
36	application	completing
37	details	register
38	supervisor	hold
39	supervision	manipulate
40	business	
41	head	
42	Admin	
43	developer	
44	entries	
45	data	

Serial no.	Nouns	Verbs
46	history	
47	timestamps	
48	schedule	
49	time	
50	start-time	
51	end-time	
52	list	
53	employees	
54	time-slot	
55	preferences	
56	date	
57	hours	
58	week	
59	option	
60	Leaves	
61	Leave_date	
62	interval	
63	hire	
64	Type of work	
65	skills	
66	individuals	

Serial no.	Nouns	Verbs
67	remarks	
68	ratings	
69	Automatic payments	
70	Bank Information	
71	bank details	
72	users	
73	payment	
74	tables	
75	labor_id	
76	name	
77	email	
78	mobile_number	
79	birth_date	
80	work_hours	
81	wage_remaining	
82	rating	
83	type_of_work	
84	supervisor_id	
85	upi_id	
86	location_id	
87	labor_supervisor	

Serial no.	Nouns	Verbs
88	supervisor_id	
89	business_id	
90	schedule_laborers	
91	entry_time	
92	exit_time	
93	date	
94	hours_worked_today	
95	hours_worked_weekly	
96	payment_info	
97	payment_numer	
98	payment_id	
99	labor_id_receiver	
100	user_id_employer	
101	upi_id_receiver	
102	upi_id_employer	
103	paid_by_hours	
104	amount_paid	
105	email_receiver	
106	email_employer	
107	date	
108	type_of_employer	

Serial no.	Nouns	Verbs
109	governmentOfficials	
110	official_id	
111	location	
112	city	
113	district	
114	location_id	
115	users_info	
116	user_id	
117	business_info	
118	business_id	
119	Individual recruiters	
120	Hiring managers	
121	firms	
122	foreign key	
123	primary key	
124	symbol	
125	MACOS X	
126	Windows	
127	Android	
128	iOS	
129	Android Studio	

Serial no.	Nouns	Verbs
130	XCode	
131	Normal User	
132	account_number	
133	ISFC Code	
134	upi_id	

10.3. Accepted Nouns List

Candidate Entity Set	Candidate Attribute Set	Candidate Relationship Set
labor	Labor_ID, Work Hours, Wage_remaining, Rating, Supervisor_ID, account_number, Type of Work, Location ID, bank_number	has
labor_supervisor	Supervisor_ID, Business_ID, account_number, Email, Mobile No., Location_ID	Identifies as, hires, pays, supervises
schedule_info	Labor ID, Entry Time, Exit Time, time_slot, date, hours_worked_weekly, hours_worked_today	consists of, registers
users_info	user_id, account_id, location_id	Identify by, hires, pays, employes
payment_info	Payment_number, payment_id, labor_id_receiver, user_id_employer,	Pay using

	account_number_employer, account_number_receiver, paid_for_byHours, amount_paid, email_receiver, email_employer, date, type_of_employer, location_id	
bank_info	Account_number, ISFC_code, upi_id	consists of
location_info	Location_id, city, district, user_id	Has, consists of
business_info	Business_id, account_number, location_id	Pays, consists of, employees
government_officials	Official_id, location_id	consists of, regulates and supervises
leaves	Labor_id, supervisor_id, leave_date	Registers, modifies
personal_information	User_id, name, email, mobile_number, type_of_user	
hourly_labors	Labor_id, hourly_wage, hours_worked	
weekly_labors	Labor_id, weekly_wage, hours_worked	

10.4. Rejected Nouns list with the reason of rejection

Noun	Reject Reason
database	General

Noun	Reject Reason
Laborlist	General
Wages	General
record	General
work	Irrelevant
laborers	Duplicate
wages	Duplicate
schema	General
relations	General
File	General
System	General
constraints	General
attributes	General
web-application	General
features	General
dashboard	General
firms	Duplicate
database	General
backend	General
product	General
features	General
workforce	Vague

Noun	Reject Reason
labor	Vague
opportunities	Irrelevant
desktop-application	General
APIs	General
PEEN	General
PGSQL	General
ExpressJS	General
EJS	General
NodeJS	General
stack	Irrelevant
diagram	Irrelevant
functions	General
application	General
details	Vague
supervisor	Duplicate
supervision	Association
head	Association
Admin	Association
developer	Irrelevant
entries	General
data	General

Noun	Reject Reason
history	General
timestamps	Vague
time	Vague
start-time	Attribute
end-time	Attribute
list	General
employees	Association
time-slot	Duplicate
preferences	Vague
date	Irrelevant
hours	Irrelevant
week	Irrelevant
option	General
interval	Irrelevant
hire	General
Type of work	Duplicate
skills	Irrelevant
individuals	Vague
remarks	Duplicate
Automatic payments	Irrelevant
Bank Information	Duplicate

Noun	Reject Reason
tables	General
labor_id	Attribute
name	Attribute
email	Attribute
mobile_number	Attribute
birth_date	Attribute
work_hours	Attribute
wage_remaining	Attribute
rating	Attribute
type_of_work	Attribute
supervisor_id	Attribute
upi_id	Attribute
location_id	Attribute
labor_supervisor	Attribute
supervisor_id	Attribute
business_id	Attribute
schedule_labors	Attribute
entry_time	Attribute
exit_time	Attribute
date	Attribute
hours_worked_today	Attribute

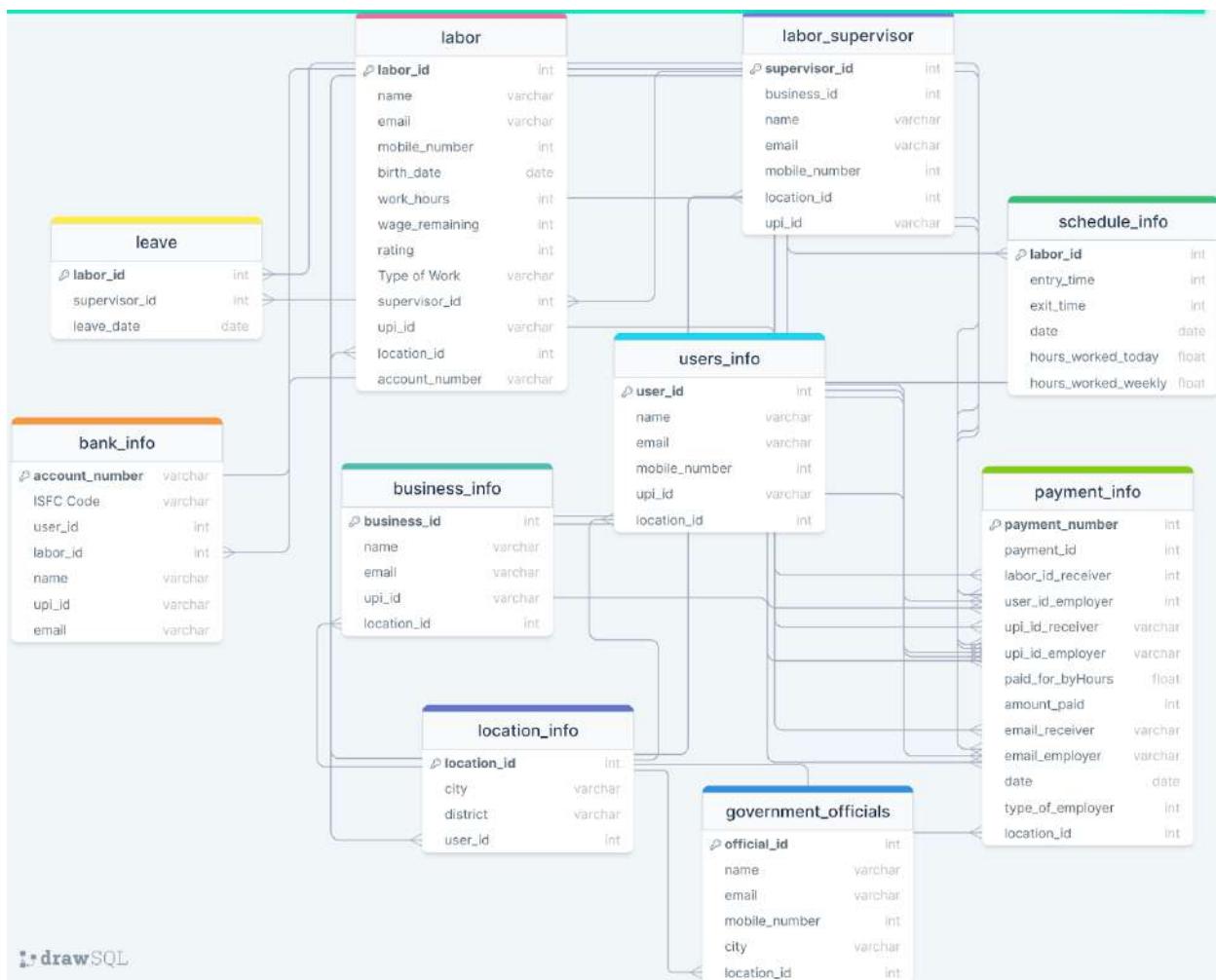
Noun	Reject Reason
hours_worked_weekly	Attribute
payment_info	Attribute
payment_numer	Attribute
payment_id	Attribute
labor_id_receiver	Attribute
user_id_employer	Attribute
UPI_id_receiver	Attribute
UPI_id_employer	Attribute
paid_by_hours	Attribute
amount_paid	Attribute
email_receiver	Attribute
email_employer	Attribute
date	Attribute
type_of_employer	Attribute
governmentOfficials	Attribute
official_id	Attribute
location	Attribute
city	Attribute
district	Attribute
location_id	Attribute
users_info	Attribute

Noun	Reject Reason
user_id	Attribute
business_info	Attribute
business_id	Attribute
Individual recruiters	Association
Hiring managers	Association
firms	Duplicate
foreign key	General
primary key	General
symbol	Irrelevant
MACOS X	General
Windows	General
Android	General
iOS	General
Android Studio	General
XCode	General
Normal User	Duplicate
Leave date	Attribute
Account number	Attribute
IFSC Code	Attribute
UPI ID	Duplicate

11. ER Diagrams

11.1. Version 1:

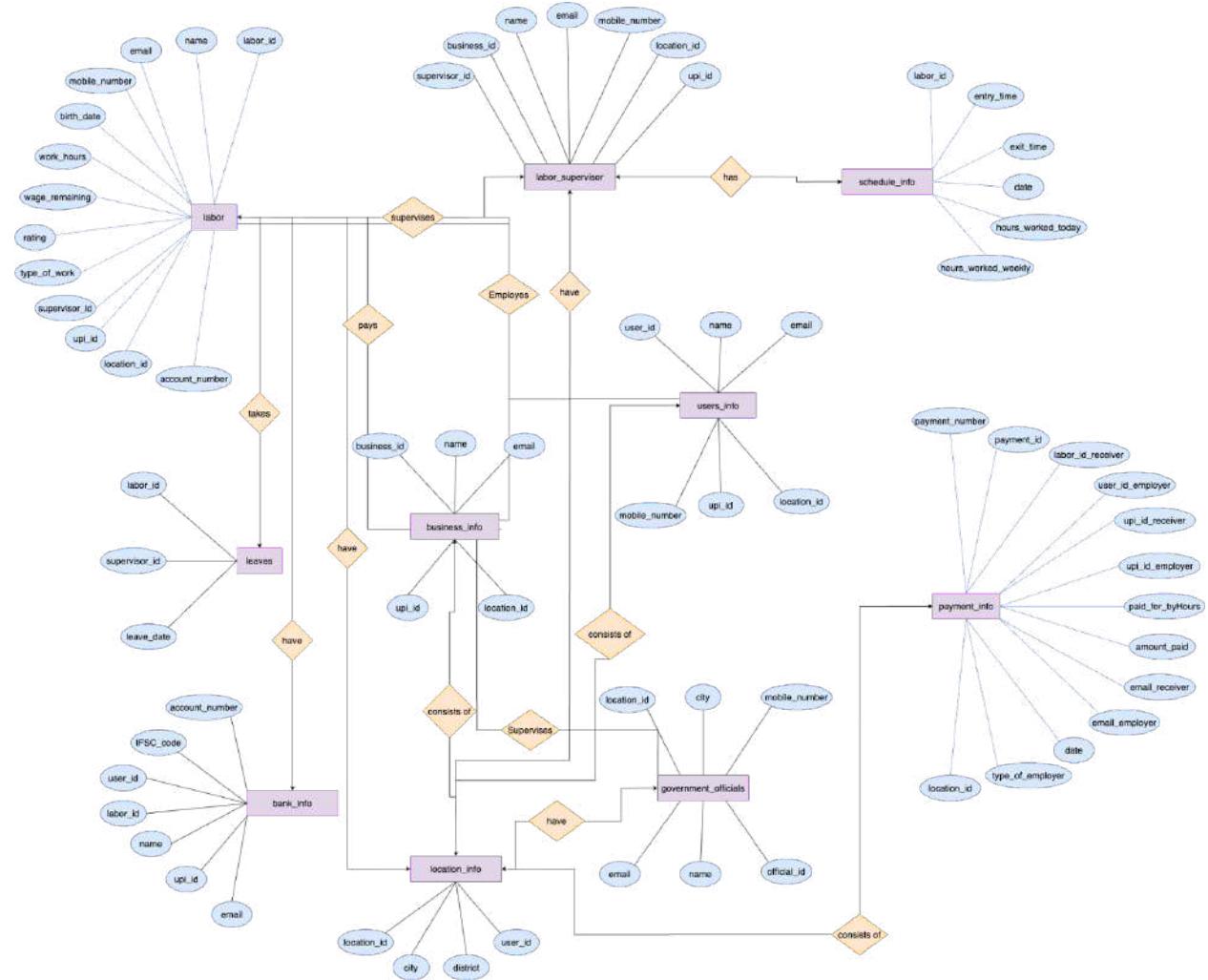
- This version contains entities and attributes in the form of a basic diagram only.
- This diagram is used just to visualize the database system for future versions of the E-R diagram. The E-R diagram will be built upon this illustration using the draw.io tool.



11.2. Version 2:

- This version consists of Entities, Attributes and relations between entities.

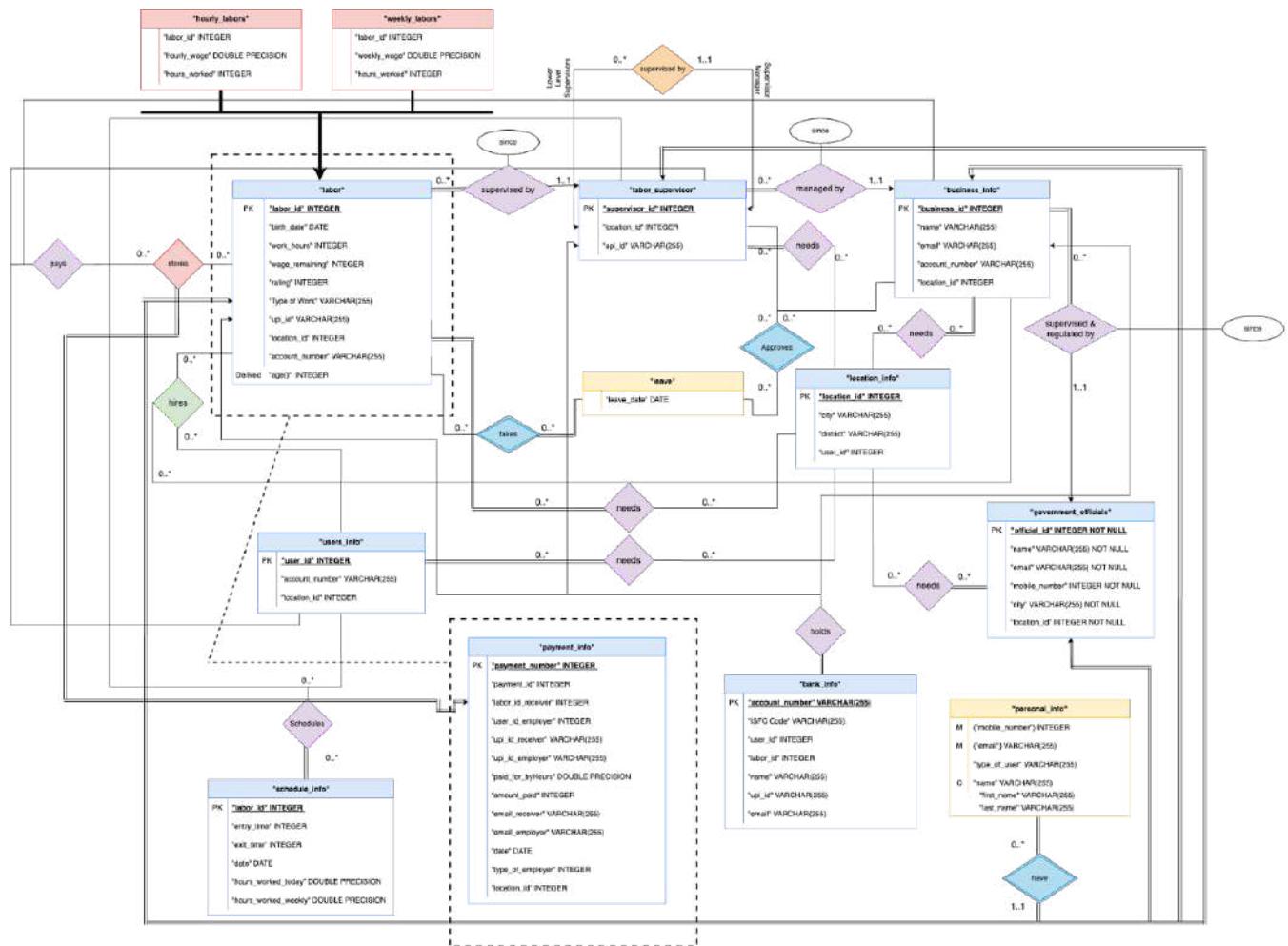
→ The E-R Diagram model used for this version is the Chen Model for database notation. It provides an abstract view of the E-R diagram with using different shapes which provide a good representation of Entities, Attributes and Relations for initial versions and brainstorming.



11.3. Final Version:

→ The final version of the E-R diagram has all of the features implemented in the diagram and are listed below. The previous version did not have types of entities and relationships defined in the diagram.

→ The following version consists of an E-R diagram that contains all the information regarding the entities and the relations.



Notation

- non binary relationship
- recursive relationship
- binary relationship
- Aggregation
- Identifying Relationship

Entity



Weak Entity



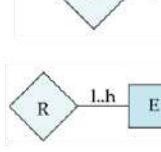
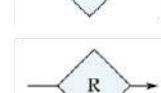
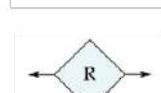
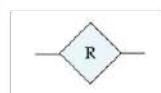
Strong Entity



Entities part of Aggregation



Specialised Entity



Attribute type is written on the left pane of the entity table and also follows standard notation

PK - Primary Key

C - Composite Attribute

D() - Derived Attribute

M {} - Multivalued Attribute

Descriptive Attribute

Total Participation

Partial Participation

The ER Diagram is also available at draw.io platform. The editable PNG file is attached in the zip file.

Entity Types:

The type of entities used in the E-R diagram are:

1] Strong Entity: The system consists of many strong entities which have a primary key defined and are not dependent on the attributes.

Strong Entities
labor
labor_supervisor
schedule_info
business_info
governmentOfficials
payment_info
user_info
location_info
bank_info

2] Weak Entity: There are two weak entities in the system which do not have a primary key i.e., a unique key to define and are dependent on other attributes.

Weak Entities
personal_info

leave

Extended Features of E-R Diagram:

The type of relationships used in the E-R diagram are:

1] Hierarchy (Generalization & Specialization): There are certain ISA hierarchies used in the LaborList and Wages System which are:

1) Hourly Laborers and Weekly Laborers → Labor

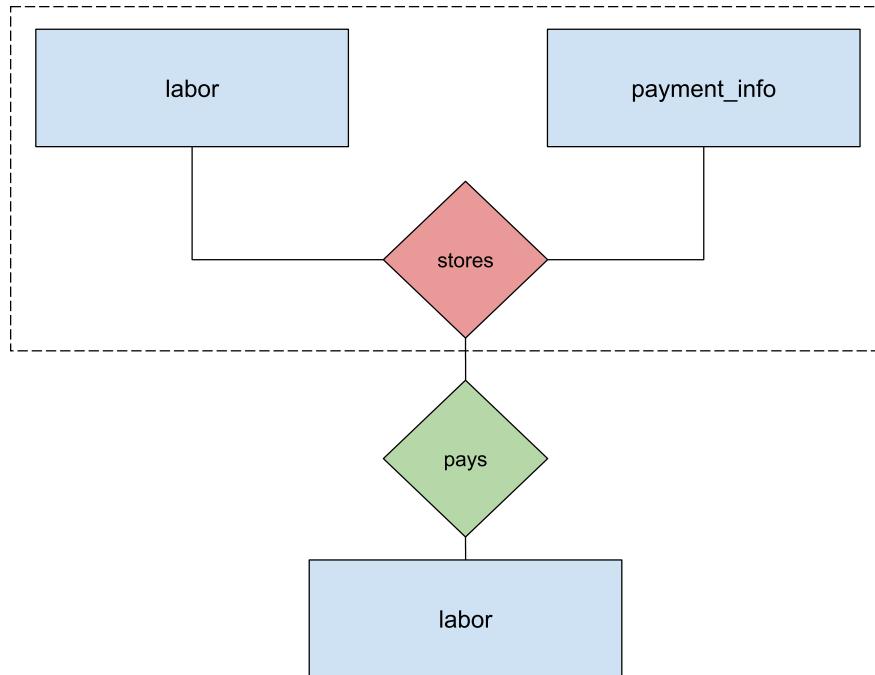
Used for the working of the Labor Management and to keep a track or record of its working. The users can utilize it to gain only the required data and thus function accordingly.

2) Normal Users & Labor Supervisors & Business → Employers

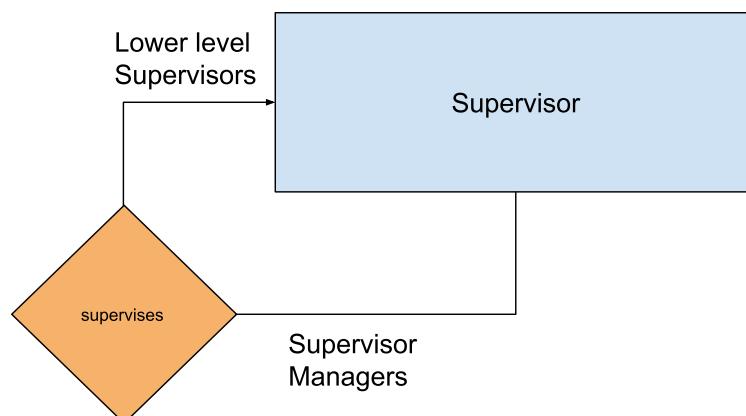
Used for generalizing the employers who are to hire a laborer. The employer can see which laborers are available for hiring and the laborers will have access to the information provided by their employer.

These hierarchies can be used to provide access to certain information to the user who should have the privilege to do so.

2] Aggregation: There is a simple usage of aggregation in the database between the entities **labor** and **payment_info**. The entities aggregate together to form a more meaningful entity which is used for storing the payment information of laborers used to make their wage payments.

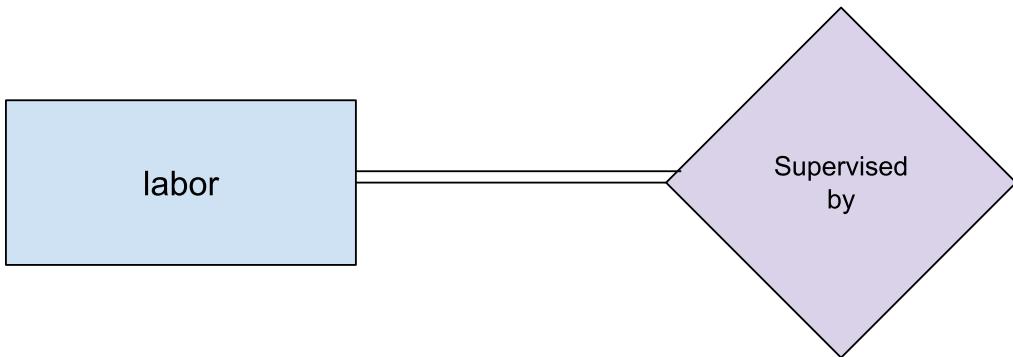


3] Recursive Relationship: There is one recursive relationship in the system which is to supervise. The supervisors on a lower level can be under a higher level supervisor whose task is to keep records of all the lower level supervisors under them. Thus, the supervisor relation becomes recursive in case there are multiple levels of supervisors in a labor management company.



Participation Constraints:

Total Participation: There is one relation in the system which has total participation. All the laborers are required to have a supervisor who would be assigned to them. Hence, the relationship of laborers being supervised on the entity labor requires total participation.



Partial participation: It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.

Example of partial participation in the system: All government officials might not be supervising or regulating the laborer system. So it's called partial participation of government officials.

Types of attributes used in the ER Diagram:

1] Single-valued Attribute: Attributes that have only one value per entity instance. There are multiple instances of single-valued attributes in the system since they are very common.

2] Multi-valued Attribute: Whenever an attribute has several values for a given entity instance, it is considered to be a multi-valued attribute. [Email](#) and [mobile_number](#) are examples of such attributes in the system since they can take up multiple values for a single user.

3] Derived Attribute: A derived attribute is an attribute that can be derived from another attribute. One such instance of derived attribute

used in the system is the [age](#) of a user which can be derived from their [birth_date](#).

4] Composite Attribute: The term composite attribute refers to an attribute whose components can be split. In the system, [name](#) is one such composite attribute in the [personal_info](#) entity. This attribute can be further split into [first_name](#) and [last_name](#).

5] Descriptive Attribute: The attributes which are used to describe a relationship in an E-R model are called descriptive attributes. For example, [since](#) is an attribute which is used multiple times to describe the relationships and their time.



IT214 - DBMS
(E-R Diagram to Relational Model, DDL Script)

Case Study Database: Laborlist and Wages Database

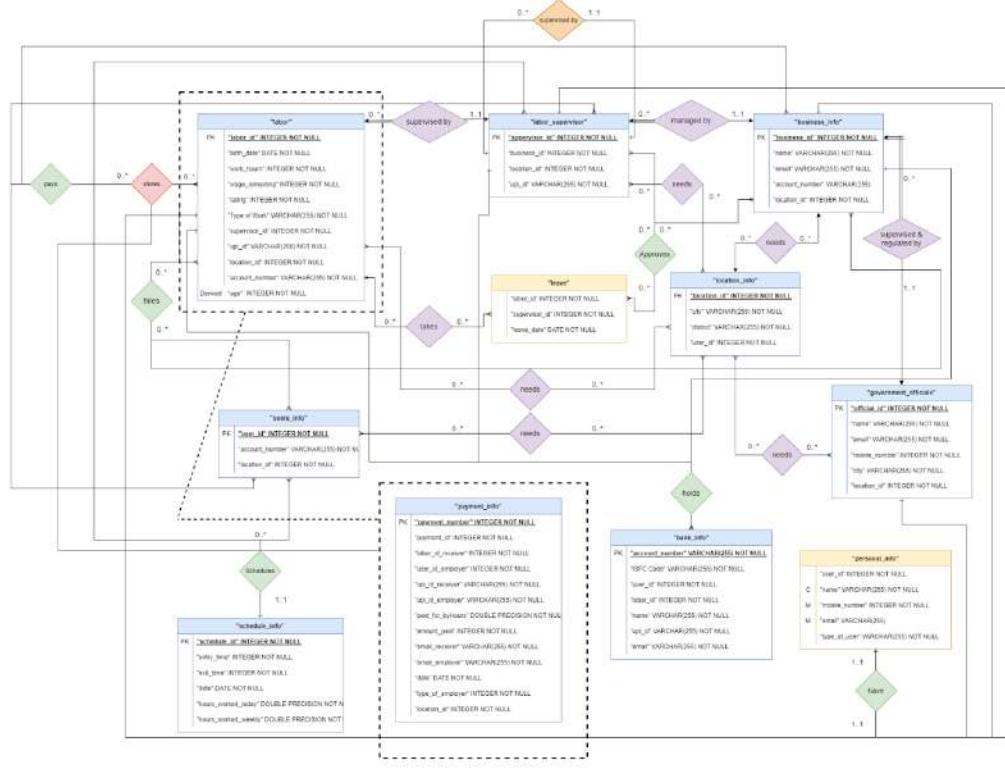
Prepared by
Team No. 17 | Group 6 | Section 10

Group Members
Dhruv Patel - 202001271
Jaykumar Navadiya - 202001465

Under the guidance of
Prof. Chhaya
Teaching Assistant - Vinay Maharaaj

Date
November 4, 2022

12. ER Diagram Final Version



Notation

n-ary relationship	Entity	Cardinality	Attribute type is written on the left pane of the entity table
recursive relationship	Entity	Indicates 0 to infinite instances	PK - Primary Key C - Composite Attribute D - Derived Attribute M - Multivalued Attribute
binary relationship	Weak Entity	Cardinality	Total Participation
Aggregation	Strong Entity	Indicates 1 instance	Partial Participation
	Entities part of Aggregation		

13. E-R Model to Relational Model Mapping

1] labor(labor_id, birth_date, work_hours, wage_remaining, rating, type_of_work, supervisor_id ,supervised_since, location_id, account_number, password)

2] labor_supervisor(supervisor_id, managed_since, business_id, location_id, account_number, password)

3] location_info(location_id, city, district, user_id)

4] governmentOfficials(official_id, location_id)

5] business_info(business_id, name, email, location_id, official_id, regulated_since, password)

6] users_info(user_id, account_number, location_id, password)

7] payment_info(payment_number, payment_id, labor_id_receiver, user_id_employer, account_number_receiver, account_number_employer, paid_for_byHours, amount_paid, email_receiver, email_employer, date, type_of_employer, location_id)

8] schedule_info(schedule_id, entry_time, exit_time, date, hours_worked_today, hours_worked_weekly)

9] bank_info(account_number, IFSC Code, user_id, labor_id, name, upi_id, email)

10] hourly_labors(labor_id, hourly_wage, hours_worked)

11] weekly_labors(labor_id, weekly_wage, hours_worked)

12] leave(leave_id, leave_date)

13] personal_info(user_id, labor_id, supervisor_id, business_id, official_id, mobile_number, type_of_user, name, email)

14] supervisor_manager(lowerlevel supervisor id, manager id,
supervised_since)

15] needs_location(user id, labor id, supervisor id, business id,
official id, location id)

16] schedules(user id, supervisor id, labor id)

17] hires(user id, business id, labor id)

18] takes_leave(user id, leave id, business id, supervisor id, labor id)

19] approves(user id, leave id, business id, supervisor id, labor id)

20] holds(user id, labor id, supervisor id, business id)

21] stores(labor id, payment number)

22] pays(business id, supervisor id, labor id, payment_number)

14. DDL Scripts

```
CREATE TABLE IF NOT EXISTS "labor"(
    "labor_id" INTEGER NOT NULL,
    "birth_date" DATE ,
    "work_hours" INTEGER ,
    "wage_remaining" INTEGER ,
    "rating" INTEGER ,
    "Type of Work" VARCHAR(255) ,
    "supervisor_id" INTEGER ,
    "age" INTEGER ,
    "supervised_since" INTEGER ,
```

```
"location_id" INTEGER ,  
"account_number" VARCHAR(255) ,  
PRIMARY KEY("labor_id"),  
FOREIGN KEY("supervisor_id") REFERENCES  
"labor_supervisor"("supervisor_id"),  
FOREIGN KEY("location_id") REFERENCES "location_info"("location_id")  
);
```

```
CREATE TABLE IF NOT EXISTS "labor_supervisor"(  
"supervisor_id" INTEGER NOT NULL,  
"managed_since" INTEGER ,  
"business_id" INTEGER ,  
"location_id" INTEGER ,  
"account_number" VARCHAR(255) ,  
PRIMARY KEY("supervisor_id"),  
FOREIGN KEY("business_id") REFERENCES "business_info"("business_id"),  
FOREIGN KEY("location_id") REFERENCES "location_info"("location_id")  
);
```

```
CREATE TABLE IF NOT EXISTS "location_info"(  
"location_id" INTEGER NOT NULL,  
"city" VARCHAR(255) ,  
"district" VARCHAR(255) ,  
"user_id" INTEGER ,  
PRIMARY KEY("location_id")  
);
```

```
CREATE TABLE IF NOT EXISTS "governmentOfficials"(  
"official_id" INTEGER NOT NULL,  
"city" VARCHAR(255) ,  
"location_id" INTEGER ,  
PRIMARY KEY("official_id"),  
FOREIGN KEY("location_id") REFERENCES "location_info"("location_id")  
);
```

```
CREATE TABLE IF NOT EXISTS "business_info"(  
"business_id" INTEGER NOT NULL ,  
"name" VARCHAR(255) ,
```

```
"email" VARCHAR(255) ,  
"account_number" VARCHAR(255) ,  
"location_id" INTEGER ,  
"official_id" INTEGER ,  
"regulated_since" DATE ,  
PRIMARY KEY("business_id"),  
FOREIGN KEY("official_id") REFERENCES "governmentOfficials"("official_id"),  
FOREIGN KEY("location_id") REFERENCES "location_info"("location_id")  
);
```

```
CREATE TABLE IF NOT EXISTS "users_info"(  
    "user_id" INTEGER NOT NULL,  
    "account_number" VARCHAR(255) ,  
    "location_id" INTEGER ,  
    PRIMARY KEY("user_id"),  
    FOREIGN KEY("location_id") REFERENCES "location_info"("location_id")  
);
```

```
CREATE TABLE IF NOT EXISTS "payment_info"(  
    "payment_number" INTEGER NOT NULL,  
    "payment_id" VARCHAR(255) ,  
    "labor_id_receiver" INTEGER ,  
    "user_id_employer" INTEGER ,  
    "account_number_receiver" VARCHAR(255) ,  
    "account_number_employer" VARCHAR(255) ,  
    "paid_for_byHours" DOUBLE PRECISION ,  
    "amount_paid" INTEGER ,  
    "email_receiver" VARCHAR(255) ,  
    "email_employer" VARCHAR(255) ,  
    "date" DATE ,  
    "type_of_employer" VARCHAR(255) ,  
    "location_id" INTEGER ,  
    PRIMARY KEY("payment_number")  
);
```

```
CREATE TABLE IF NOT EXISTS "schedule_info"(  
    "schedule_id" INTEGER NOT NULL,  
    "entry_time" TIME(0) WITHOUT TIME ZONE ,  
    "exit_time" TIME(0) WITHOUT TIME ZONE ,
```

```
        "date" DATE ,  
        "hours_worked_today" DOUBLE PRECISION ,  
        "hours_worked_weekly" DOUBLE PRECISION ,  
        PRIMARY KEY("schedule_id")  
);
```

```
CREATE TABLE IF NOT EXISTS "bank_info"(  
    "account_number" VARCHAR(255) NOT NULL,  
    "ISFC Code" VARCHAR(255) NOT NULL,  
    "user_id" INTEGER NOT NULL,  
    "labor_id" INTEGER NOT NULL,  
    "name" VARCHAR(255) ,  
    "upi_id" VARCHAR(255) NOT NULL,  
    "email" VARCHAR(255) ,  
    PRIMARY KEY("account_number")  
);
```

```
CREATE TABLE IF NOT EXISTS "hourly_labors"(  
    "labor_id" INTEGER NOT NULL,  
    "hourly_wage" INTEGER ,  
    "hours_worked" INTEGER ,  
    PRIMARY KEY("labor_id"),  
    FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id")  
);
```

```
CREATE TABLE IF NOT EXISTS "weekly_labors"(  
    "labor_id" INTEGER NOT NULL,  
    "weekly_wage" INTEGER ,  
    "hours_worked" INTEGER ,  
    PRIMARY KEY("labor_id"),  
    FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id")  
);
```

```
CREATE TABLE IF NOT EXISTS "leave"(  
    "leave_id" INTEGER NOT NULL,  
    "leave_date" DATE,  
    PRIMARY KEY("leave_id"),  
    ON DELETE CASCADE  
);
```

```
CREATE TABLE IF NOT EXISTS "personal_info"(
    "user_id" INTEGER ,
    "labor_id" INTEGER ,
    "supervisor_id" INTEGER ,
    "business_id" INTEGER ,
    "official_id" INTEGER ,
    "mobile_number" INTEGER ,
    "type_of_user" VARCHAR(255) NOT NULL ,
    "name" VARCHAR(255) ,
    "email" VARCHAR(255) ,
    UNIQUE("user_id", "labor_id", "supervisor_id", "business_id", "official_id", "mobile_number", "email"),
    FOREIGN KEY("user_id") REFERENCES "users_info"("user_id"),
    FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id"),
    FOREIGN KEY("supervisor_id") REFERENCES
        "labor_supervisor"("supervisor_id"),
    FOREIGN KEY("business_id") REFERENCES "business_info"("business_id"),
    FOREIGN KEY("official_id") REFERENCES "governmentOfficials"("official_id"),
    ON DELETE CASCADE
);
```

```
CREATE TABLE IF NOT EXISTS "supervisor_manager"(
    "lowerlevel_supervisor_id" INTEGER NOT NULL,
    "manager_id" INTEGER NOT NULL,
    "supervised_since" DATE ,
    PRIMARY KEY("lowerlevel_supervisor_id", "manager_id")
);
```

```
CREATE TABLE IF NOT EXISTS "needs_location"(
    "user_id" INTEGER ,
    "labor_id" INTEGER ,
    "supervisor_id" INTEGER ,
    "business_id" INTEGER ,
    "official_id" INTEGER ,
    "location_id" INTEGER ,
    UNIQUE("user_id", "labor_id", "supervisor_id", "business_id", "official_id", "location_id"),
    FOREIGN KEY("location_id") REFERENCES "location_info"("location_id"),
    FOREIGN KEY("user_id") REFERENCES "users_info"("user_id"),
```

```
FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id"),
FOREIGN KEY("supervisor_id") REFERENCES
"labor_supervisor"("supervisor_id"),
FOREIGN KEY("business_id") REFERENCES "business_info"("business_id"),
FOREIGN KEY("official_id") REFERENCES "governmentOfficials"("official_id")
);
```

```
CREATE TABLE IF NOT EXISTS "schedules"(
"user_id" INTEGER ,
"supervisor_id" INTEGER ,
"labor_id" INTEGER NOT NULL,
UNIQUE("user_id", "supervisor_id", "labor_id"),
FOREIGN KEY("user_id") REFERENCES "users_info"("user_id"),
FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id"),
FOREIGN KEY("supervisor_id") REFERENCES
"labor_supervisor"("supervisor_id"),
FOREIGN KEY("schedule_id") REFERENCES "schedule_info"("schedule_id")
);
```

```
CREATE TABLE IF NOT EXISTS "hires"(
"user_id" INTEGER ,
"business_id" INTEGER ,
"labor_id" INTEGER NOT NULL ,
UNIQUE("user_id", "business_id", "labor_id"),
FOREIGN KEY("user_id") REFERENCES "users_info"("user_id"),
FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id"),
FOREIGN KEY("business_id") REFERENCES "business_info"("business_id")
);
```

```
CREATE TABLE IF NOT EXISTS "takes_leave"(
"user_id" INTEGER ,
"leave_id" INTEGER ,
"business_id" INTEGER ,
"supervisor_id" INTEGER,
"labor_id" INTEGER NOT NULL,
UNIQUE("user_id", "business_id", "labor_id", "leave_id", "supervisor_id"),
FOREIGN KEY("user_id") REFERENCES "users_info"("user_id"),
FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id"),
FOREIGN KEY("supervisor_id") REFERENCES
```

```
"labor_supervisor"("supervisor_id"),
FOREIGN KEY("business_id") REFERENCES "business_info"("business_id"),
FOREIGN KEY("leave_id") REFERENCES "leave"("leave_id")
);
```

```
CREATE TABLE IF NOT EXISTS "approves"(
"user_id" INTEGER,
"leave_id" INTEGER ,
"business_id" INTEGER ,
"supervisor_id" INTEGER,
"labor_id" INTEGER NOT NULL,
UNIQUE("user_id", "business_id", "labor_id", "leave_id", "supervisor_id"),
FOREIGN KEY("user_id") REFERENCES "users_info"("user_id"),
FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id"),
FOREIGN KEY("supervisor_id") REFERENCES
"labor_supervisor"("supervisor_id"),
FOREIGN KEY("business_id") REFERENCES "business_info"("business_id"),
FOREIGN KEY("leave_id") REFERENCES "leave"("leave_id")
);
```

```
CREATE TABLE IF NOT EXISTS "holds"(
"user_id" INTEGER ,
"labor_id" INTEGER ,
"supervisor_id" INTEGER ,
"business_id" INTEGER ,
UNIQUE("user_id", "labor_id", "supervisor_id", "business_id"),
FOREIGN KEY("location_id") REFERENCES "location_info"("location_id"),
FOREIGN KEY("user_id") REFERENCES "users_info"("user_id"),
FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id"),
FOREIGN KEY("supervisor_id") REFERENCES
"labor_supervisor"("supervisor_id"),
FOREIGN KEY("business_id") REFERENCES "business_info"("business_id"),
FOREIGN KEY("official_id") REFERENCES "governmentOfficials"("official_id")
);
```

```
CREATE TABLE IF NOT EXISTS "stores"(
"labor_id" INTEGER NOT NULL,
"payment_number" INTEGER NOT NULL,
PRIMARY KEY("labor_id", "payment_number"),
```

```
FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id"),
FOREIGN KEY("payment_number") REFERENCES
"payment_info"("payment_number")
);
```

```
CREATE TABLE IF NOT EXISTS "pays"(
"business_id" INTEGER NOT NULL,
"supervisor_id" INTEGER NOT NULL,
"labor_id" INTEGER NOT NULL,
"payment_number" INTEGER NOT NULL,
PRIMARY KEY("labor_id", "business_id", "supervisor_id"),
FOREIGN KEY("labor_id") REFERENCES "labor"("labor_id"),
FOREIGN KEY("payment_number") REFERENCES
"payment_info"("payment_number")
FOREIGN KEY("supervisor_id") REFERENCES
"labor_supervisor"("supervisor_id"),
FOREIGN KEY("business_id") REFERENCES "business_info"("business_id"),
);

```



IT214 - DBMS
(Normalization and Schema Refinement)

Case Study Database: Laborlist and Wages Database

Prepared by
Team No. 17 | Group 6 | Section 10

Group Members
Dhruv Patel - 202001271
Jaykumar Navadiya - 202001465

Under the guidance of
Prof. Chhaya
Teaching Assistant - Vinay Maharaaj

Date
November 11, 2022

15. Normalization & Schema Refinement

15.1 Relations and Schemas (Initial)

- 1] labor(labor_id, birth_date, work_hours, wage_remaining, rating, Type of work, age, supervisor_id ,supervised_since, location_id, account_number)
- 2] labor_supervisor(supervisor_id, managed_since, business_id, location_id, account_number)
- 3] location_info(location_id, city, district, user_id)
- 4] governmentOfficials(official_id, city, location_id)
- 5] business_info(business_id, name, email, account_number, location_id, official_id, regulated_since)
- 6] users_info(user_id, account_number, location_id)
- 7] payment_info(payment_number, payment_id, labor_id_receiver, user_id_employer, account_number_receiver, account_number_employer, paid_for_byHours, amount_paid, email_receiver, email_employer, date, type_of_employer, location_id)
- 8] schedule_info(schedule_id, entry_time, exit_time, date, hours_worked_today, hours_worked_weekly)
- 9] bank_info(account_number, IFSC Code, user_id, labor_id, name, UPI_id, email)
- 10] hourly_labors(labor_id, hourly_wage, hours_worked)

- 11] weekly_labors(labor_id, weekly_wage, hours_worked)
- 12] leave(leave_id, leave_date)
- 13] personal_info(user_id, labor_id, supervisor_id, business_id, official_id,
mobile_number, type_of_user, name, email)
- 14] supervisor_manager(lowerlevel_supervisor_id, manager_id,
supervised_since)
- 15] needs_location(user_id, labor_id, supervisor_id, business_id,
official_id, location_id)
- 16] schedules(user_id, supervisor_id, labor_id)
- 17] hires(user_id, business_id, labor_id)
- 18] takes_leave(user_id, leave_id, business_id, supervisor_id, labor_id)
- 19] approves(user_id, leave_id, business_id, supervisor_id, labor_id)
- 20] holds(user_id, labor_id, supervisor_id, business_id)
- 21] stores(labor_id, payment_number)
- 22] pays(business_id, supervisor_id, labor_id, payment_number)

15.2 Types of Dependencies

labor

PK	labor_id
FK	supervisor_id, location_id
Functional Dependencies	$\text{labor_id} \rightarrow \text{birth_date}$, $\text{labor_id} \rightarrow \text{work_hours}$, $\text{labor_id} \rightarrow \text{wage_remaining}$, $\text{labor_id} \rightarrow \text{rating}$, $\text{labor_id} \rightarrow \text{Type of work}$, $\text{labor_id} \rightarrow \text{age}$, $\text{labor_id} \rightarrow \text{supervisor_id}$, $\text{labor_id} \rightarrow \text{supervised_since}$, $\text{labor_id} \rightarrow \text{location_id}$, $\text{labor_id} \rightarrow \text{account_number}$, $\text{birth_date} \rightarrow \text{age}$

labor_supervisor	
PK	supervisor_id
FK	business_id, location_id
Functional Dependencies	$\text{supervisor_id} \rightarrow \text{managed_since}$, $\text{supervisor_id} \rightarrow \text{business_id}$, $\text{supervisor_id} \rightarrow \text{location_id}$, $\text{supervisor_id} \rightarrow \text{account_number}$

location_info	
PK	location_id
FK	
Functional Dependencies	$\text{location_id} \rightarrow \text{city}$, $\text{location_id} \rightarrow \text{district}$, $\text{city} \rightarrow \text{district}$, $\text{location_id} \rightarrow \text{user_id}$

governmentOfficials	
PK	official_id
FK	location_id
Functional Dependencies	official_id → city, official_id → location_id

businessInfo	
PK	business_id
FK	official_id, location_id
Functional Dependencies	$\text{business_id} \rightarrow \text{name}$, $\text{business_id} \rightarrow \text{email}$, $\text{business_info} \rightarrow \text{account_number}$, $\text{email} \rightarrow \text{account_number}$, $\text{business_id} \rightarrow \text{location_id}$, $\text{business_id} \rightarrow \text{official_id}$, $\text{business_id} \rightarrow \text{regulated_since}$,

usersInfo	
PK	user_id
FK	location_id
Functional Dependencies	$\text{user_id} \rightarrow \text{account_number}$, $\text{user_id} \rightarrow \text{location_id}$

payment_info	
PK	payment_number
FK	
Functional Dependencies	$\text{payment_number} \rightarrow \text{payment_id}$, $\text{payment_number} \rightarrow \text{labor_id_receiver}$, $\text{payment_number} \rightarrow \text{user_id_employer}$, $\text{payment_number} \rightarrow \text{account_number_receiver}$, $\text{payment_number} \rightarrow \text{account_number_employer}$, $\text{payment_number} \rightarrow \text{paid_for_byHours}$, $\text{payment_number} \rightarrow \text{amount_paid}$, $\text{payment_number} \rightarrow \text{email_receiver}$, $\text{payment_number} \rightarrow \text{email_employer}$, $\text{payment_number} \rightarrow \text{date}$, $\text{payment_number} \rightarrow \text{type_of_employer}$, $\text{payment_number} \rightarrow \text{location_id}$

schedule_info	
PK	schedule_id
FK	
Functional Dependencies	$\text{schedule_id} \rightarrow \text{entry_time}$, $\text{schedule_id} \rightarrow \text{exit_time}$, $\text{schedule_id} \rightarrow \text{date}$, $\text{schedule_id} \rightarrow \text{hours_worked_today}$, $\text{schedule_id} \rightarrow \text{hours_worked_weekly}$

bank_info	
PK	account_number, user_id, upi_id
FK	
Functional Dependencies	(account_number, user_id, upi_id) → (IFSC Code, user_id, labor_id, name, upi_id, email)

hourly_labors	
PK	labor_id
FK	labor_id
Functional Dependencies	labor_id → (hourly_wage, hours_worked)

weekly_labors	
PK	labor_id
FK	labor_id
Functional Dependencies	labor_id → (weekly_wage, hours_worked)

leave	
-------	--

PK	leave_id
FK	
Functional Dependencies	leave_id → leave_date

personal_info	
PK	{ user_id, labor_id, supervisor_id, business_id, official_id, mobile_number, email }
FK	user_id, labor_id, supervisor_id, business_id, official_id
Functional Dependencies	(user_id, labor_id, supervisor_id, business_id, official_id, mobile_number, email) → (mobile_number, type_of_user, name, email)

supervisor_manager	
PK	{lowerlevel_supervisor_id, manager_id }
FK	
Functional Dependencies	(lowerlevel_supervisor_id, manager_id)→ supervised_since

schedules	
PK	{ user_id, supervisor_id, labor_id }
FK	user_id, labor_id, supervisor_id, schedule_id
Functional Dependencies	(user_id, supervisor_id, labor_id) → (user_id, supervisor_id, labor_id)

hires	
PK	{ user_id, business_id, labor_id }
FK	user_id, business_id, labor_id
Functional Dependencies	(user_id, business_id, labor_id) → (user_id, business_id, labor_id), user_id → labor_id, business_id → labor_id

takes_leave	
PK	{ user_id, business_id, labor_id, leave_id, supervisor_id }

FK	user_id, business_id, labor_id, leave_id, supervisor_id
Functional Dependencies	(user_id, business_id, labor_id, leave_id, supervisor_id) → (user_id, business_id, labor_id, leave_id, supervisor_id)

approves	
PK	{ user_id, business_id, labor_id, leave_id, supervisor_id }
FK	user_id, business_id, labor_id, leave_id, supervisor_id
Functional Dependencies	(user_id, business_id, labor_id, leave_id, supervisor_id) → (user_id, business_id, labor_id, leave_id, supervisor_id)

holds	
PK	{ user_id, business_id, labor_id, supervisor_id }
FK	user_id, business_id, labor_id, location_id, supervisor_id, official_id

Functional Dependencies	$(\text{user_id}, \text{business_id}, \text{labor_id}, \text{supervisor_id}) \rightarrow (\text{user_id}, \text{business_id}, \text{labor_id}, \text{supervisor_id})$
-------------------------	---

stores	
PK	{ labor_id, payment_number }
FK	labor_id, payment_number
Functional Dependencies	$(\text{labor_id}, \text{payment_number}) \rightarrow (\text{labor_id}, \text{payment_number})$

pays	
PK	{ labor_id, business_id, supervisor_id }
FK	labor_id, payment_number, supervisor_id, business_id
Functional Dependencies	$(\text{labor_id}, \text{business_id}, \text{supervisor_id}) \rightarrow \text{payment_number}$

15.3 Schema Investigation

15.3.1 Redundancy and General Analysis

1] labor(labor_id, birth_date, work_hours, wage_remaining, rating, Type of work, age, supervisor_id ,supervised_since, location_id, account_number)

- There is one transitive dependency i.e., birth_date → age (because labor_id → birth_date)
- This can be removed by making a new table with columns age(birth_date, age).
- Redundancy: for example, a type of work can be plumbing, now there can be many laborers who are plumbers.

2] labor_supervisor(supervisor_id, managed_since, business_id, location_id, account_number)

- There are no transitive dependencies.
- Redundancy: In 1 business there can be many supervisors hence making the business_id redundant.

3] location_info(location_id, city, district, user_id)

- There is one transitive dependency i.e., city → district (because location_id → city)
- This can be removed by making a new tables with columns (city, district) and location_info(location_id, city, user_id).

4] governmentOfficials(official_id, city, location_id)

- There are no transitive dependencies.
- Redundancy: In 1 city there can be many officials hence making the city redundant.

5] business_info(business_id, name, email, account_number, location_id, official_id, regulated_since)

- There are transitive dependencies, such as email can give the account_number and vice versa. Hence this table is in 2NF and needs to be converted into 3NF.
- Hence we can make tables like business_account(email, account_number) and business_info(business_id, name, email, location_id, official_id, regulated_since).

6] users_info(user_id, account_number, location_id)

- There are no transitive dependencies.

7] payment_info(payment_number, payment_id, labor_id_receiver, user_id_employer, account_number_receiver, account_number_employer, paid_for_byHours, amount_paid, email_receiver, email_employer, date, type_of_employer, location_id)

- There are no transitive dependencies.

8] schedule_info(schedule_id, entry_time, exit_time, date, hours_worked_today, hours_worked_weekly)

- There are no transitive dependencies.

9] bank_info(account_number, IFSC Code, user_id, labor_id, name, UPI_id, email)

- There can be many upi_id linked to 1 account. There are no non-prime to non-prime dependencies. But there is one dependency (account_number → upi_id) that can be removed.
- Making this into BCNF form we can decompose into 2 tables bank_info(account_number, IFSC Code, user_id, labor_id, name, email) and upi_info(account_number, UPI_id)

10] hourly_labors(labor_id, hourly_wage, hours_worked)

- There are no transitive dependencies.

11] weekly_labors(labor_id, weekly_wage, hours_worked)

- There are no transitive dependencies.

12] leave(leave_id, leave_date)

- There are no transitive dependencies.

13] personal_info(user_id, labor_id, supervisor_id, business_id, official_id,
mobile_number, type_of_user, name, email)

- There are no transitive dependencies.
- Redundancy: For any type of users, there will be many users of that type.

14] supervisor_manager(lowerlevel_supervisor_id, manager_id,
supervised_since)

- There are no transitive dependencies.

15] needs_location(user_id, labor_id, supervisor_id, business_id,
official_id, location_id)

- There are no transitive dependencies.

16] schedules(user_id, supervisor_id, labor_id)

- There are no transitive dependencies.

17] hires(user_id, business_id, labor_id)

- There are no transitive dependencies.

18] takes_leave(user_id, leave_id, business_id, supervisor_id, labor_id)

- There are no transitive dependencies.

19] approves(user_id, leave_id, business_id, supervisor_id, labor_id)

- There are no transitive dependencies nor partial dependencies.

20] holds(user_id, labor_id, supervisor_id, business_id)

- There are no transitive dependencies.

21] stores(labor_id, payment_number)

- There are no transitive dependencies.

22] pays(business_id, supervisor_id, labor_id, payment_number)

- There are no transitive dependencies.

15.3.2 List of update, delete and insert anomalies

1. Hires table ($\text{user_id} \rightarrow \text{labor_id}$, $\text{business_id} \rightarrow \text{labor_id}$)

Here combined ($\text{user_id}, \text{business_id} \rightarrow \text{labor_id}$) can give us labor_id .

Now, when there is an individual user hiring a laborer, business_id will be null. So, It creates an insertion anomaly as we are inserting NULL values in the business_id column and vice versa.

2. The personal_info table has personal information of all types of users. But while inserting information about an individual user, other types of users such as labor or supervisor will remain NULL, hence creating an insertion anomaly.

15.4 1NF Normalization

Each table is already in 1NF form. The table personal_info was already separated and mobile_number was a part of the primary key. Hence while making the final portal/website, users will have to login using their ID and mobile number. As mobile number is a part of the primary key, for 1 user multiple mobile numbers can be stored and 1NF is achieved.

15.5 2NF Normalization

All of the tables are in 2NF form as there are no Partial Functional Dependency in any table even though there might be

15.6 3NF/BCNF Normalization

The table/relation which has only 1 Candidate Key and if it is already in 3NF, it will be in BCNF (since they are equivalent).

For 3NF Normalization, we need to remove all the transitive dependencies from the 2NF form. For instance, in the relation business_info:

1] business_info(business_id, name, email, account_number, location_id, official_id, regulated_since).

There are transitive dependencies, such as email which can give the account_number and vice versa from the business_id. Hence this table is in 2NF and needs to be converted into 3NF.

To achieve this, we decompose business_info to get a new table which is business_account(email,account_number) and business_info(business_id, name, email, location_id, official_id, regulated_since).

2] bank_info(account_number, IFSC Code, user_id, labor_id, name, UPI_id, email)

- There can be many upi_id linked to 1 account. There are no non-prime to non-prime dependencies. But there is one dependency (account_number → upi_id) that can be removed.
- Making this into BCNF form we can decompose into 2 tables bank_info(account_number, IFSC Code, user_id, labor_id, name, email) and upi_info(account_number, UPI_id)

15.7 Final Relations

1] labor(labor_id, birth_date, work_hours, wage_remaining, rating, type_of_work, supervisor_id, supervised_since, location_id, account_number, password)

2] labor_supervisor(supervisor_id, managed_since, business_id, location_id, account_number, password)

3] location_info(location_id, city, district)

4] governmentOfficials(official_id, location_id)

5] business_info(business_id, location_id, official_id, name, email, regulated_since, password)

6] users_info(user_id, account_number, location_id, password)

- 7] payment_info(payment_number, payment_id, paid_for_byHours, amount_paid, date)
- 8] schedule_info(schedule_id, entry_time, exit_time, date)
- 9] bank_info(account_number, isfc_code, name, email, type_of_user, user_id)
- 10] hourly_labors(labor_id, hourly_wage, hours_worked)
- 11] weekly_labors(labor_id, weekly_wage, hours_worked)
- 12] leave(leave_id, leave_date)
- 13] personal_info(*user_id*, *labor_id*, *supervisor_id*, *official_id*, mobile_number, type_of_user, name, email) //Here unique key is in italics
- 14] supervisor_manager(lowerlevel supervisor id,manager id, supervised_since)
- 15] schedules(user_id, supervisor_id, labor_id, schedule id)
- 16] hires(user id, business id, labor id)
- 17] takes_leave(user_id, leave id, supervisor_id, labor_id)
- 18] approves(*user_id*, leave id, *supervisor_id*, *labor_id*) //Here unique key is in italics
- 19] pays(user_id, supervisor_id, labor_id, payment number)
- 20] business_account(email, account number)

21] upi_info(account_number, UPI_id)

→ District, holds, stores, needs_location tables were eventually dropped as they were not required. We allow only 1 location per user and they can update it afterwards. In one to one relationship we do not require a new table. These were removed since they were not required for any specific purpose and created redundancy in the database. All the relations associated with their attributes already had access to them.

→ Also, password attributes were added in relations labor, labor_supervisor, business_info and users_info. These are eventually required for authentication purposes for the web pages created for the system to recognize the users by the passwords.

→ user_id was removed from the primary key of the relation bank_info since account_number was sufficient for the primary key.

16. Re-write DDL Scripts

16.1 & 16.2 DDL Scripts for New Design (3NF/BCNF Form) and DDL Snapshots

1. Location_info Table

DDL Script:

```

-- Table: laborlist_and_wages_db.location_info

-- DROP TABLE IF EXISTS laborlist_and_wages_db.location_info;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.location_info
(
    location_id integer NOT NULL,
    city character varying COLLATE pg_catalog."default" NOT NULL,
    district character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT location_info_pkey PRIMARY KEY (location_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.location_info
OWNER to postgres;

```

DDL Snapshot:

The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database schema with 'Tables (21)' selected, showing various tables like 'approves', 'bank_info', etc., and 'location_info' highlighted. The main window has a 'Query' tab open containing the DDL script. The status bar at the bottom right shows 'Query complete 00:00:00.089'.

```

PgAdmin  File  Object  Tools  Help  PgAdmin  Dashboard  Properties  SQL  Statistics  Dependencies  Dependents  laborlist_and_wages/postgres@PostgreSQL 14*
Browser  FTS Templates  Foreign Tables  Functions  Materialized Views  Operators  Procedures  Sequences  Tables (21)  approves  bank_info  business_account  business_info  governmentOfficials  hires  hourlyLaborers  labor  laborSupervisor  leave  location_info  payment_info  pays  personal_info  schedule_info  schedules  supervisorManager  takesLeave  upInfo  users_info  weeklyLaborers  Trigger Functions  Types  Views  public  Subscriptions  postges  Login/Group Roles
Query  Query History
Query: laborlist_and_wages/postgres@PostgreSQL 14*
1 -- Table: laborlist_and_wages_db.location_info
2
3 -- DROP TABLE IF EXISTS laborlist_and_wages_db.location_info;
4
5 CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.location_info
6 (
7     location_id integer NOT NULL,
8     city character varying COLLATE pg_catalog."default" NOT NULL,
9     district character varying COLLATE pg_catalog."default" NOT NULL,
10    CONSTRAINT location_info_pkey PRIMARY KEY (location_id)
11 )
12
13 TABLESPACE pg_default;
14
15 ALTER TABLE IF EXISTS laborlist_and_wages_db.location_info
16    OWNER to postgres;

```

Data output Messages Notifications

NOTICE: relation "location_info" already exists, skipping
ALTER TABLE

Query returned successfully in 89 msec.

Total rows: 0 of 0 | Query complete 00:00:00.089 | Ln 16, Col 23

2. Schedule_info Table

DDL Script:

```
-- Table: laborlist_and_wages_db.schedule_info

-- DROP TABLE IF EXISTS laborlist_and_wages_db.schedule_info;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.schedule_info
(
    schedule_id integer NOT NULL,
    entry_time time without time zone NOT NULL,
    exit_time time without time zone NOT NULL,
    date date NOT NULL,
    CONSTRAINT schedule_info_pkey PRIMARY KEY (schedule_id),
    CONSTRAINT schedule_fkey FOREIGN KEY (schedule_id)
        REFERENCES laborlist_and_wages_db.schedules (schedule_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.schedule_info
OWNER to postgres;
```

DDL Snapshot:

The screenshot shows the PgAdmin 4 interface. The left sidebar is the 'Browser' pane, listing database objects like FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (21). The 'Tables' section is expanded, showing tables such as approves, bank_info, business_account, business_info, governmentOfficials, hires, hourlyJobs, labor, labor_supervisor, leave, location_info, payment_info, pays, personalInfo, schedule_info, schedules, supervisor_manager, takesLeave, upi_info, users_info, weeklyJobs, Trigger Functions, Types, Views, public, and postges. The right pane is the 'Query' editor, which contains the following SQL code:

```

2
3 -- DROP TABLE IF EXISTS laborlist_and_wages_db.schedule_info;
4
5 CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.schedule_info
6 (
7     schedule_id integer NOT NULL,
8     entry_time time without time zone NOT NULL,
9     exit_time time without time zone NOT NULL,
10    date_date NOT NULL,
11    CONSTRAINT schedule_info_pkey PRIMARY KEY (schedule_id),
12    CONSTRAINT schedule_fkey FOREIGN KEY (schedule_id)
13        REFERENCES laborlist_and_wages_db.schedules (schedule_id) MATCH SIMPLE
14        ON UPDATE NO ACTION
15        ON DELETE NO ACTION
16        NOT VALID
17 )
18
19 TABLESPACE pg_default;
20
21 ALTER TABLE IF EXISTS laborlist_and_wages_db.schedule_info
22 OWNER to postgres;

```

Below the code, the status bar shows 'Total rows: 0 of 0' and 'Query complete 00:00:00.030'. A green message at the bottom right says 'Query returned successfully in 30 msec. Ln 22, Col 23'.

3. Bank_info table

DDL Script:

```

-- Table: laborlist_and_wages_db.bank_info

-- DROP TABLE IF EXISTS laborlist_and_wages_db.bank_info;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.bank_info
(
    account_number character varying COLLATE pg_catalog."default" NOT NULL,
    isfc_code character varying COLLATE pg_catalog."default" NOT NULL,
    name character varying COLLATE pg_catalog."default" NOT NULL,
    email character varying COLLATE pg_catalog."default" NOT NULL,
    type_of_user character varying COLLATE pg_catalog."default",
    user_id integer NOT NULL,
    CONSTRAINT bank_info_pkey PRIMARY KEY (account_number)
)

TABLESPACE pg_default;

```

```
ALTER TABLE IF EXISTS laborlist_and_wages_db.bank_info
OWNER to postgres;
```

DDL Snapshot:

The screenshot shows the PgAdmin 4 interface. On the left is a tree view of database objects under 'laborlist_and_wages/postgres@PostgreSQL 14'. The 'Tables (21)' node is expanded, showing 'bank_info' as one of the tables. The main pane contains the SQL code for creating and altering the 'bank_info' table. The status bar at the bottom right indicates 'Query returned successfully in 26 msec'.

```
-- Table: laborlist_and_wages_db.bank_info
-- DROP TABLE IF EXISTS laborlist_and_wages_db.bank_info;
CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.bank_info
(
    account_number character varying COLLATE pg_catalog."default" NOT NULL,
    ifsc_code character varying COLLATE pg_catalog."default" NOT NULL,
    name character varying COLLATE pg_catalog."default" NOT NULL,
    email character varying COLLATE pg_catalog."default" NOT NULL,
    type_of_user character varying COLLATE pg_catalog."default",
    user_id integer NOT NULL,
    CONSTRAINT bank_info_pkey PRIMARY KEY (account_number)
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS laborlist_and_wages_db.bank_info
    OWNER to postgres;
```

4. Leave table

DDL Script:

```
-- Table: laborlist_and_wages_db.leave

-- DROP TABLE IF EXISTS laborlist_and_wages_db.leave;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.leave
(
    leave_id integer NOT NULL,
    leave_date date NOT NULL,
    CONSTRAINT leave_pkey PRIMARY KEY (leave_id),
```

```

CONSTRAINT leave_id_fkey FOREIGN KEY (leave_id)
    REFERENCES laborlist_and_wages_db.takes_leave (leave_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.leave
OWNER to postgres;

```

DDL Snapshot:

The screenshot shows the PgAdmin 4 interface with the 'laborlist_and_wages/postgres@PostgreSQL 14*' connection selected. The left sidebar displays the database schema with the 'Tables' node expanded, showing various tables like 'approves', 'bank_info', 'business_account', etc. The 'leave' table is currently selected. The main pane contains the DDL script for the 'leave' table, which includes creating the table if it doesn't exist, defining its columns (leave_id, leave_date), adding primary key constraints, and setting up a foreign key constraint (leave_id_fkey) that references the 'takes_leave' table. It also includes an ALTER TABLE command to set the owner to 'postgres'. The status bar at the bottom indicates 'Query returned successfully in 218 msec.'

```

-- Table: laborlist_and_wages_db.leave
-- DROP TABLE IF EXISTS laborlist_and_wages_db.leave;
CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.leave
(
    leave_id integer NOT NULL,
    leave_date date NOT NULL,
    CONSTRAINT leave_pk PRIMARY KEY (leave_id),
    CONSTRAINT leave_id_fkey FOREIGN KEY (leave_id) MATCH SIMPLE
        REFERENCES laborlist_and_wages_db.takes_leave (leave_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS laborlist_and_wages_db.leave
OWNER to postgres;

```

5. Upi_info table

DDL Script:

```

-- Table: laborlist_and_wages_db.upi_info

-- DROP TABLE IF EXISTS laborlist_and_wages_db.upi_info;

```

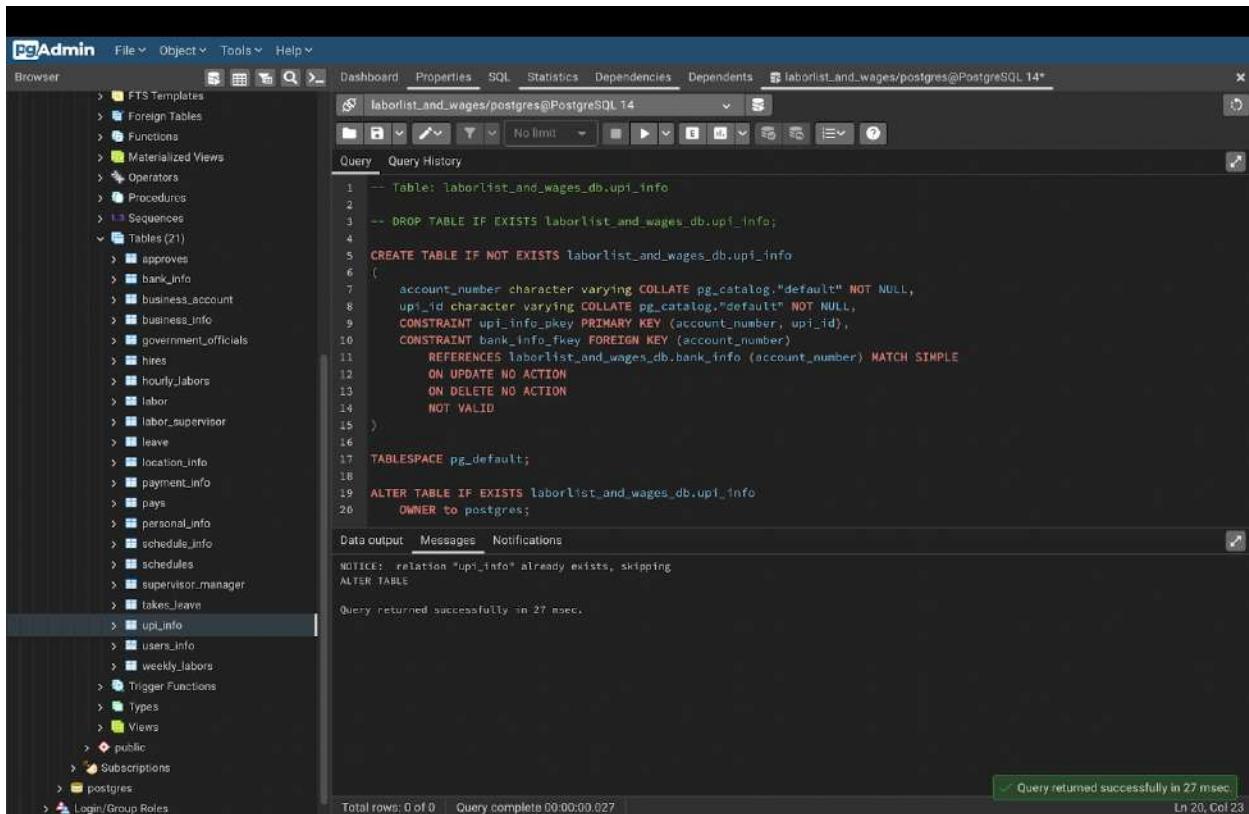
```

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.upi_info
(
    account_number character varying COLLATE pg_catalog."default" NOT NULL,
    upi_id character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT upi_info_pkey PRIMARY KEY (account_number, upi_id),
    CONSTRAINT bank_info_fkey FOREIGN KEY (account_number)
        REFERENCES laborlist_and_wages_db.bank_info (account_number) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.upi_info
OWNER to postgres;

```

DDL Snapshot:



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

- Toolbar:** File, Object, Tools, Help.
- Dashboard:** Shows the database connection: laborlist_and_wages/postgres@PostgreSQL 14*.
- Properties:** Tab selected.
- Dependencies:** Tab.
- Dependents:** Tab.
- Browser:** Shows the schema tree:
 - Tables (21):
 - approves
 - bank_info
 - business_account
 - business_info
 - governmentOfficials
 - hires
 - hourlyLaborers
 - labor
 - labor_supervisor
 - leave
 - location_info
 - payment_info
 - pays
 - personal_info
 - schedule_info
 - schedules
 - supervisor_manager
 - takesLeave
 - upi_info
 - users_info
 - weeklyLaborers
 - Trigger Functions
 - Types
 - Views
 - public
 - Subscriptions
 - Postgres
 - Login/Group Roles
- Query Editor:**
 - Query tab: Contains the DDL code for the upi_info table.
 - Message tab: Shows "NOTICE: relation \"upi_info\" already exists, skipping ALTER TABLE".
 - Notifications tab: Empty.
- Status Bar:** Total rows: 0 of 0 | Query complete 00:00:00.027 | Query returned successfully in 27 msec. | Ln 20, Col 23.

6. Business_account table

DDL Script:

```
-- Table: laborlist_and_wages_db.business_account

-- DROP TABLE IF EXISTS laborlist_and_wages_db.business_account;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.business_account
(
    email character varying COLLATE pg_catalog."default" NOT NULL,
    account_number character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT business_account_pkey PRIMARY KEY (email, account_number),
    CONSTRAINT business_info_fkey FOREIGN KEY (email)
        REFERENCES laborlist_and_wages_db.business_info (email) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.business_account
    OWNER to postgres;
```

DDL Snapshot:

```

-- Table: laborlist_and_wages_db.business_account
-- DROP TABLE IF EXISTS laborlist_and_wages_db.business_account;
CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.business_account
(
    email character varying COLLATE pg_catalog."default" NOT NULL,
    account_number character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT business_account_pkey PRIMARY KEY (email, account_number),
    CONSTRAINT business_info_fkey FOREIGN KEY (email)
        REFERENCES laborlist_and_wages_db.business_info (email) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS laborlist_and_wages_db.business_account
OWNER to postgres;

```

NOTICE: relation "business_account" already exists, skipping
ALTER TABLE

Query returned successfully in 205 msec.

Total rows: 0 of 0 Query complete 00:00:00.205 L11, Col 45

7. GovernmentOfficials table

DDL Script:

```

-- Table: laborlist_and_wages_db.governmentOfficials

-- DROP TABLE IF EXISTS laborlist_and_wages_db.governmentOfficials;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.governmentOfficials
(
    official_id integer NOT NULL,
    location_id integer NOT NULL,
    CONSTRAINT governmentOfficials_pkey PRIMARY KEY (official_id),
    CONSTRAINT governmentOfficial_uniquef UNIQUE (official_id),
    CONSTRAINT location_info_fkey FOREIGN KEY (location_id)
        REFERENCES laborlist_and_wages_db.location_info (location_id) MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID

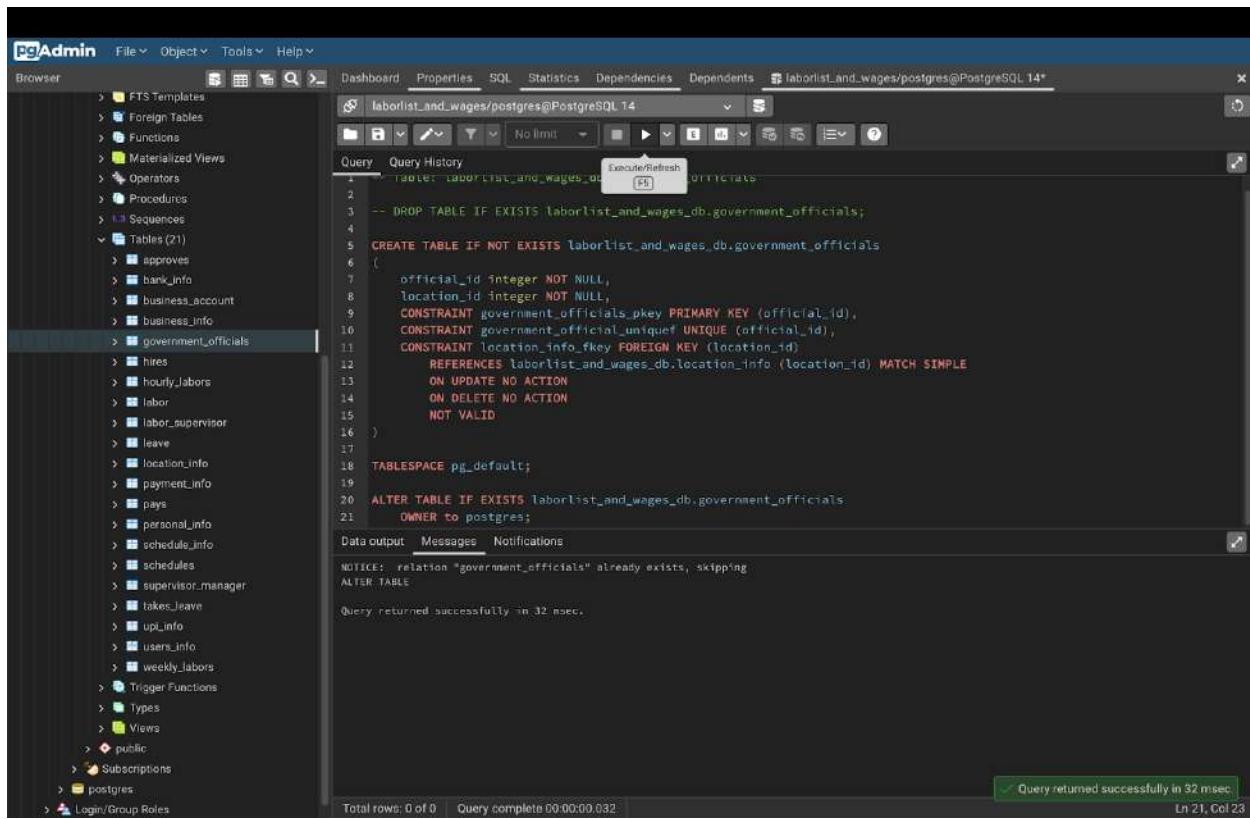
```

```
)
```

```
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.governmentOfficials
OWNER to postgres;
```

DDL Snapshot:



The screenshot shows the PgAdmin 4 interface. The left sidebar displays a tree view of database objects under the 'laborlist_and_wages/postgres@PostgreSQL 14*' connection. The 'Tables' node is expanded, showing 21 tables, with 'governmentOfficials' selected. The main pane contains the SQL DDL script for creating the 'governmentOfficials' table. The script includes a comment to drop the table if it exists, a CREATE TABLE statement with constraints (primary key on official_id and unique on official_id), a FOREIGN KEY constraint on location_info_id, and a NOT VALID constraint. It also specifies the TABLESPACE as pg_default and the OWNER as postgres. A notice message indicates that the relation already exists and is being skipped. The status bar at the bottom right shows 'Query returned successfully in 32 msec.' and 'Ln 21, Col 23'.

```
-- Table: laborlist_and_wages_db.governmentOfficials
-- DROP TABLE IF EXISTS laborlist_and_wages_db.governmentOfficials;
CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.governmentOfficials
(
    official_id integer NOT NULL,
    location_id integer NOT NULL,
    CONSTRAINT governmentOfficials_pkey PRIMARY KEY (official_id),
    CONSTRAINT governmentOfficial_unique UNIQUE (official_id),
    CONSTRAINT location_info_fkey FOREIGN KEY (location_id)
        REFERENCES laborlist_and_wages_db.location_info (location_id)
        MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS laborlist_and_wages_db.governmentOfficials
OWNER to postgres;
```

8. Business_info table

DDL Script:

```
-- Table: laborlist_and_wages_db.business_info
-- DROP TABLE IF EXISTS laborlist_and_wages_db.business_info;
CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.business_info
(
```

```

business_id integer NOT NULL,
location_id integer NOT NULL,
official_id integer NOT NULL,
name character varying COLLATE pg_catalog."default" NOT NULL,
email character varying COLLATE pg_catalog."default" NOT NULL,
regulated_since date,
password character varying DEFAULT 'busy'::character varying NOT NULL
CONSTRAINT business_info_pkey PRIMARY KEY (business_id, email),
CONSTRAINT unique_business UNIQUE (business_id),
CONSTRAINT unique_email UNIQUE (email),
CONSTRAINT governemnt_official_fkey FOREIGN KEY (official_id)
    REFERENCES laborlist_and_wages_db.governmentOfficials (official_id)

MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT location_info_fkey FOREIGN KEY (location_id)
    REFERENCES laborlist_and_wages_db.location_info (location_id) MATCH
SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.business_info
OWNER to postgres;

```

DDL Snapshot:

```

PgAdmin  File  Object  Tools  Help
Browser  Dashboard  Properties  SQL  Statistics  Dependencies  Dependents  laborlist_and_wages/postgres@PostgreSQL 14*
Query  Query History
10   name character varying COLLATE pg_catalog."default" NOT NULL;
11   email character varying COLLATE pg_catalog."default" NOT NULL;
12   CONSTRAINT business_info_pkey PRIMARY KEY (business_id, email);
13   CONSTRAINT unique_business UNIQUE (business_id);
14   CONSTRAINT unique_email UNIQUE (email);
15   CONSTRAINT government_official_fkey FOREIGN KEY (official_id)
16     REFERENCES laborlist_and_wages_db.governmentOfficials (official_id) MATCH SIMPLE
17     ON UPDATE NO ACTION
18     ON DELETE NO ACTION
19     NOT VALID;
20   CONSTRAINT location_info_fkey FOREIGN KEY (location_id)
21     REFERENCES laborlist_and_wages_db.location_info (location_id) MATCH SIMPLE
22     ON UPDATE NO ACTION
23     ON DELETE NO ACTION
24     NOT VALID;
25 )
26
27 TABLESPACE pg_default;
28
29 ALTER TABLE IF EXISTS laborlist_and_wages_db.business_info
30 OWNER to postgres;

```

NOTICE: relation "business_info" already exists, skipping
ALTER TABLE

Query returned successfully in 36 msec.

Total rows: 0 of 0 Query complete 00:00:00.036 Ln 30, Col 23

9. Labor_supervisor table

DDL Script:

```

-- Table: laborlist_and_wages_db.labor_supervisor

-- DROP TABLE IF EXISTS laborlist_and_wages_db.labor_supervisor;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.labor_supervisor
(
    supervisor_id integer NOT NULL,
    managed_since date NOT NULL,
    business_id integer NOT NULL,
    location_id integer NOT NULL,
    account_number character varying COLLATE pg_catalog."default" NOT NULL,
    password character varying DEFAULT 'super'::character varying NOT NULL
    CONSTRAINT labopr_supervisor_pkey PRIMARY KEY (supervisor_id),
    CONSTRAINT unique_supervisor UNIQUE (supervisor_id),
    CONSTRAINT business_fkey FOREIGN KEY (business_id)
)

```

```

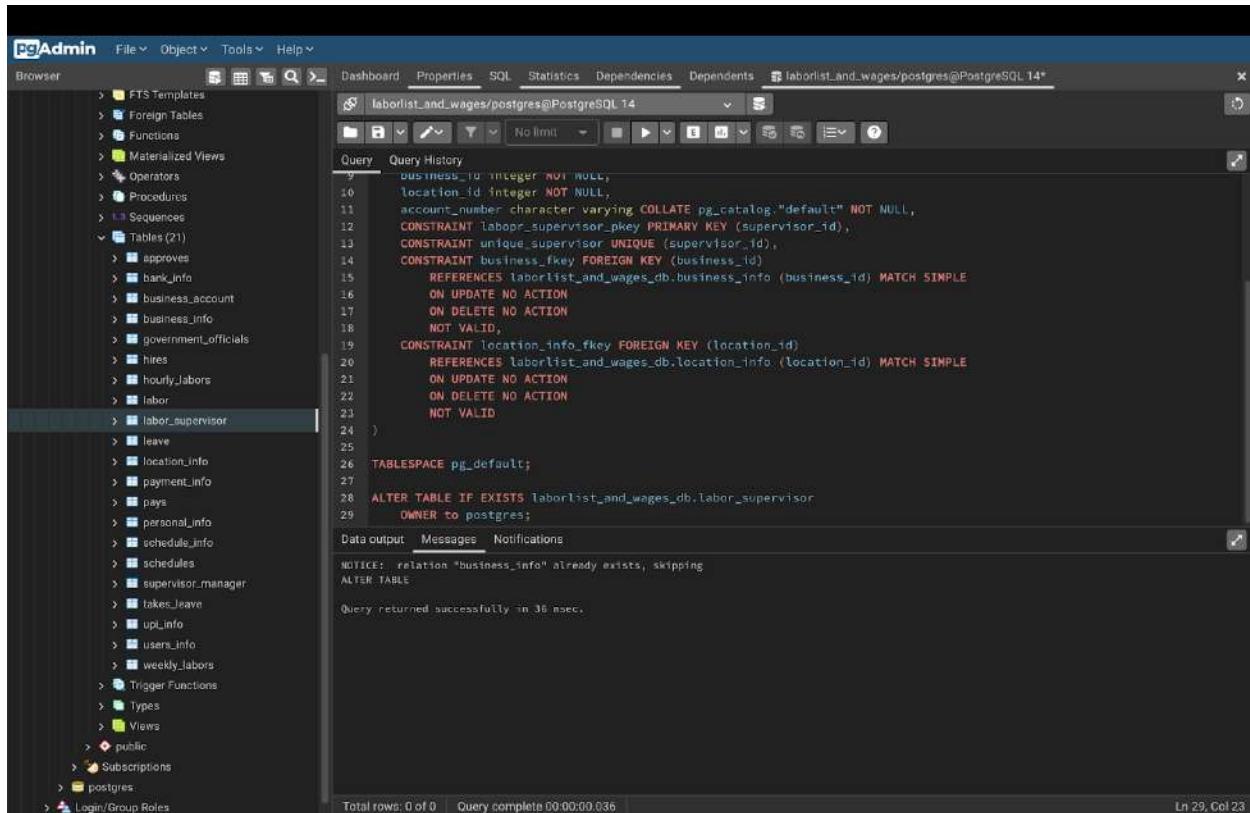
        REFERENCES laborlist_and_wages_db.business_info (business_id) MATCH
SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT location_info_fkey FOREIGN KEY (location_id)
    REFERENCES laborlist_and_wages_db.location_info (location_id) MATCH
SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.labor_supervisor
OWNER to postgres;

```

DDL Snapshot:



The screenshot shows the PgAdmin 4 interface with the 'laborlist_and_wages/postgres@PostgreSQL 14*' connection selected. The left sidebar displays the database schema with the 'Tables (21)' node expanded, showing tables like approves, bank_info, business_account, business_info, governmentOfficials, hires, hourlyJobs, labor, labor_supervisor, leave, location_info, payment_info, pays, personal_info, schedule_info, schedules, supervisorManager, takesLeave, up_info, users_info, weekly_labors, Trigger Functions, Types, Views, public, Subscriptions, and postges. The 'labor_supervisor' table is currently selected. The main pane contains the DDL code for the table, which includes constraints for primary key (supervisor_id), unique constraint (unique_supervisor), foreign key (business_id) referencing business_info, and foreign key (location_id) referencing location_info. The code also includes an ALTER TABLE command to change the owner to 'postgres'. The status bar at the bottom indicates 'Total rows: 0 of 0 | Query complete 00:00:00.036 | Ln 29, Col 23'.

```

PgAdmin  File  Object  Tools  Help  laborlist_and_wages/postgres@PostgreSQL 14*
Browser  Dashboard  Properties  SQL  Statistics  Dependencies  Dependents  laborlist_and_wages/postgres@PostgreSQL 14*
Query  Query History
9   business_id integer NOT NULL,
10  location_id integer NOT NULL,
11  account_number character varying COLLATE pg_catalog."default" NOT NULL,
12  CONSTRAINT laborpr_pkey PRIMARY KEY (supervisor_id),
13  CONSTRAINT unique_supervisor UNIQUE (supervisor_id),
14  CONSTRAINT business_fkey FOREIGN KEY (business_id) MATCH SIMPLE
15  ON UPDATE NO ACTION
16  ON DELETE NO ACTION
17  NOT VALID,
18  CONSTRAINT location_info_fkey FOREIGN KEY (location_id)
19  REFERENCES laborlist_and_wages_db.location_info (location_id) MATCH SIMPLE
20  ON UPDATE NO ACTION
21  ON DELETE NO ACTION
22  NOT VALID
23 )
24
25
26 TABLESPACE pg_default;
27
28 ALTER TABLE IF EXISTS laborlist_and_wages_db.labor_supervisor
29 OWNER to postgres;
Data output  Messages  Notifications
NOTICE: relation "business_info" already exists, skipping
ALTER TABLE
Query returned successfully in 36 msec.

```

10. Labor table

DDL Script:

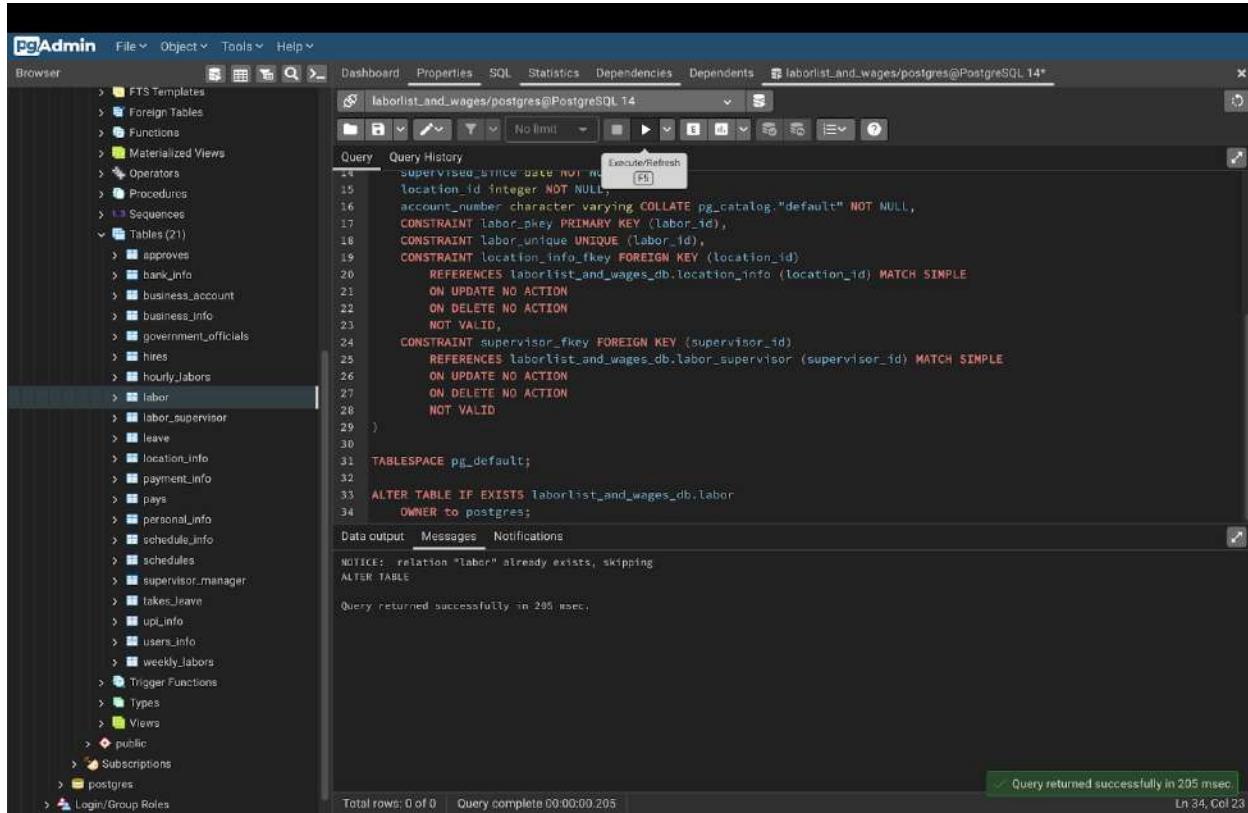
```
-- Table: laborlist_and_wages_db.labor

-- DROP TABLE IF EXISTS laborlist_and_wages_db.labor;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.labor
(
    labor_id integer NOT NULL,
    birth_date date NOT NULL,
    work_hours integer NOT NULL,
    wage_remaining integer NOT NULL,
    rating integer NOT NULL,
    type_of_work character varying COLLATE pg_catalog."default" NOT NULL,
    supervisor_id integer NOT NULL,
    supervised_since date NOT NULL,
    location_id integer NOT NULL,
    account_number character varying COLLATE pg_catalog."default" NOT NULL,
    password character varying DEFAULT 'labor' ::character varying NOT NULL
    CONSTRAINT labor_pkey PRIMARY KEY (labor_id),
    CONSTRAINT labor_unique UNIQUE (labor_id),
    CONSTRAINT location_info_fkey FOREIGN KEY (location_id)
        REFERENCES laborlist_and_wages_db.location_info (location_id)
        MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION
            NOT VALID,
    CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
        REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id)
        MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION
            NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.labor
OWNER to postgres;
```

DDL Snapshot:



The screenshot shows the PgAdmin 4 interface with the 'laborlist_and_wages/postgres@PostgreSQL 14*' connection selected. The left sidebar displays the database schema browser with various objects like FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (21). The 'Tables' section is expanded, and the 'labor' table is selected. The main pane shows the SQL query for creating the 'labor' table:

```
CREATE TABLE IF NOT EXISTS labor
(
    supervisor_since date NOT NULL,
    location_id integer NOT NULL,
    account_number character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT labor_pkey PRIMARY KEY (labor_id),
    CONSTRAINT labor_unique UNIQUE (labor_id),
    CONSTRAINT location_info_fkey FOREIGN KEY (location_id) MATCH SIMPLE
        REFERENCES laborlist_and_wages_db.location_info (location_id)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
        REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.labor
OWNER to postgres;
```

The status bar at the bottom indicates "Query returned successfully in 205 msec." and "Ln 34, Col 23".

11. Users_info table

DDL Script:

```
-- Table: laborlist_and_wages_db.users_info

-- DROP TABLE IF EXISTS laborlist_and_wages_db.users_info;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.users_info
(
    user_id integer NOT NULL,
    account_number integer NOT NULL,
    location_id integer NOT NULL,
    password character varying DEFAULT 'majoor'::character varying NOT NULL
    CONSTRAINT users_info_pkey PRIMARY KEY (user_id),
    CONSTRAINT location_info_fkey FOREIGN KEY (location_id)
        REFERENCES laborlist_and_wages_db.location_info (location_id) MATCH
        SIMPLE
```

```

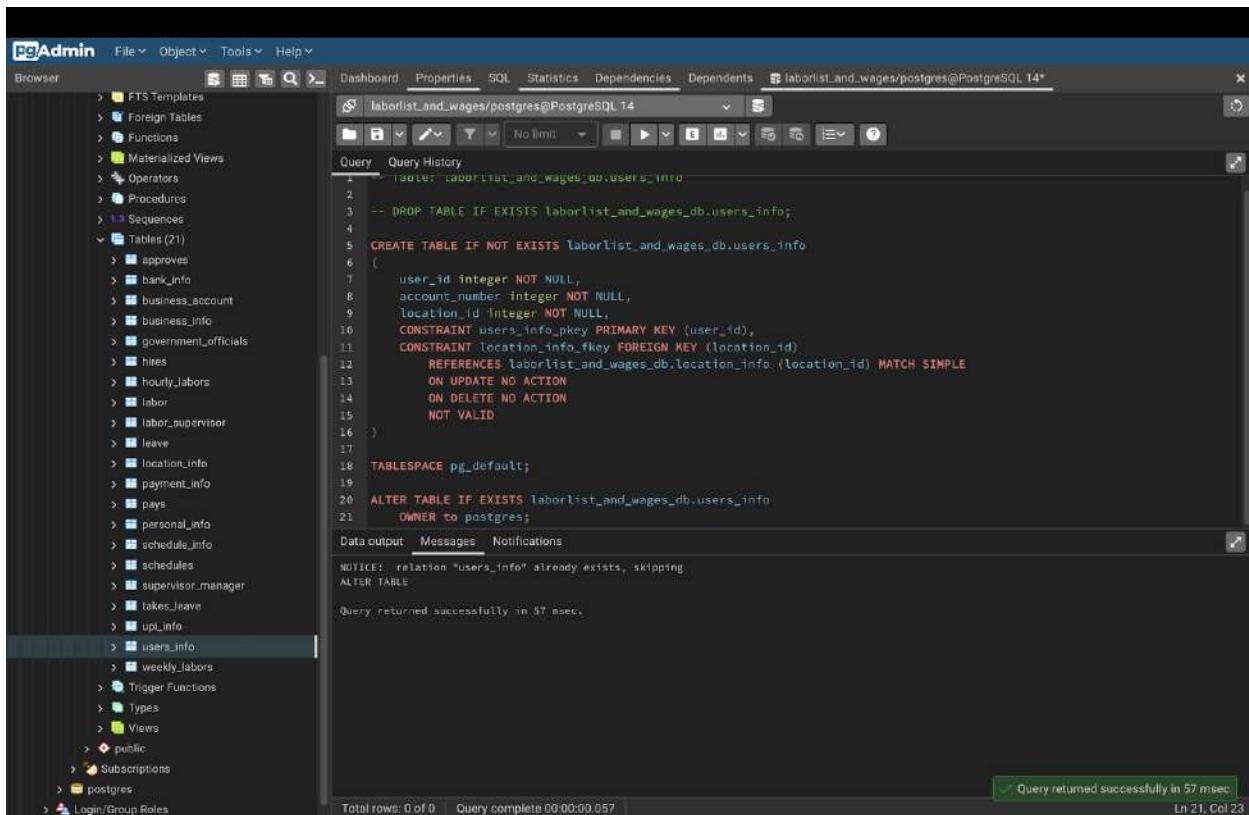
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
    )

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.users_info
OWNER to postgres;

```

DDL Snapshot:



The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database schema with various objects like FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures, Sequences, and Tables. The 'Tables' section is expanded, showing 21 entries including 'approves', 'bank_info', 'business_account', 'business_info', 'governmentOfficials', 'hires', 'hourly_labors', 'labor', 'labor_supervisor', 'leave', 'location_info', 'payment_info', 'pays', 'personal_info', 'schedule_info', 'schedules', 'supervisor_manager', 'takes_leave', 'upi_info', 'users_info', 'weekly_labors', 'Trigger Functions', 'Types', and 'Views'. The right pane contains a SQL query editor with the following DDL script:

```

-- Table: laborlist_and_wages_db.users_info
-- DROP TABLE IF EXISTS laborlist_and_wages_db.users_info;
CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.users_info
(
    user_id integer NOT NULL,
    account_number integer NOT NULL,
    location_id integer NOT NULL,
    CONSTRAINT users_info_pkey PRIMARY KEY (user_id),
    CONSTRAINT location_info_fkey FOREIGN KEY (location_id) MATCH SIMPLE
        REFERENCES laborlist_and_wages_db.location_info (location_id)
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS laborlist_and_wages_db.users_info
OWNER to postgres;

```

Below the query, a message states: "NOTICE: relation 'users_info' already exists, skipping". At the bottom of the screen, status bars indicate "Query returned successfully in 57 msec." and "Ln 21, Col 23".

12. Hourly_labors table

DDL Script:

```

-- Table: laborlist_and_wages_db.hourly_labors

-- DROP TABLE IF EXISTS laborlist_and_wages_db.hourly_labors;

```

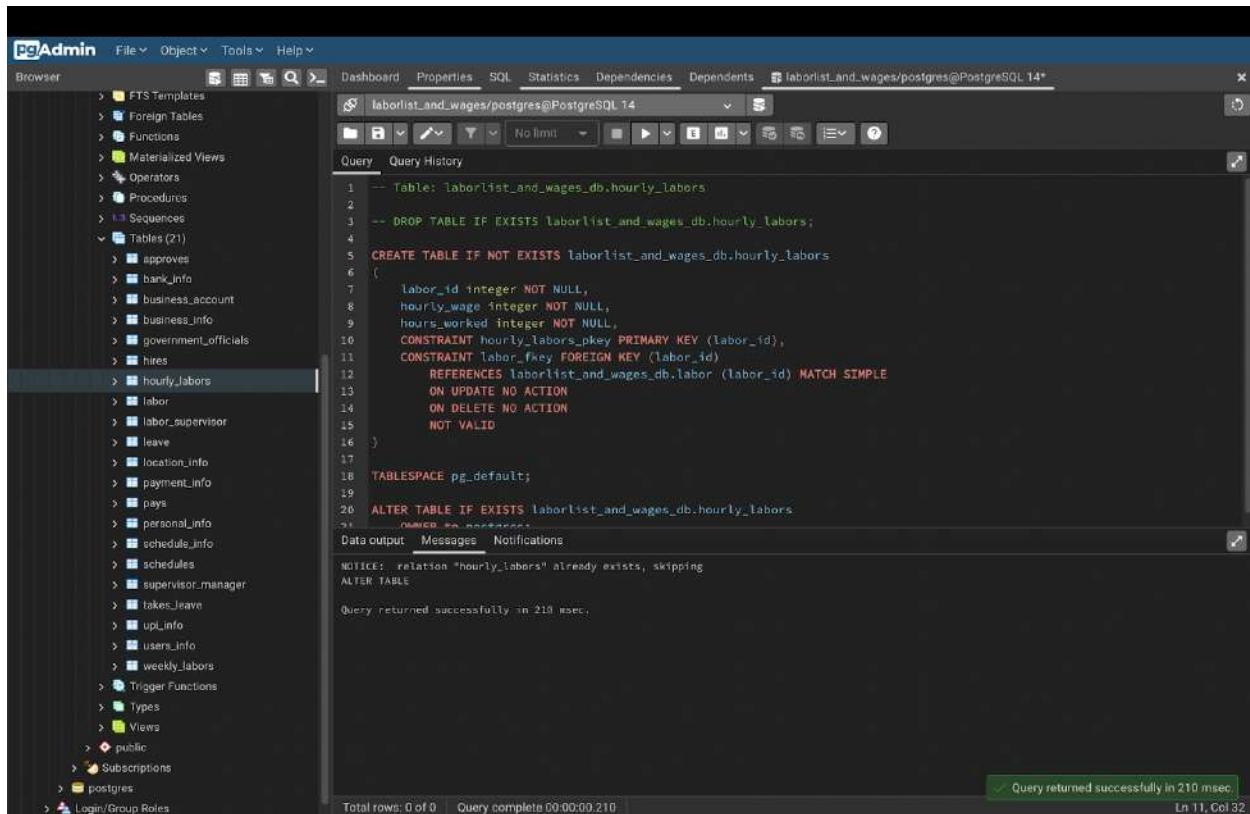
```

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.hourly_labors
(
    labor_id integer NOT NULL,
    hourly_wage integer NOT NULL,
    hours_worked integer NOT NULL,
    CONSTRAINT hourly_labors_pkey PRIMARY KEY (labor_id),
    CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
        REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.hourly_labors
OWNER to postgres;

```

DDL Snapshot:



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

- Toolbar:** File, Object, Tools, Help.
- Dashboard:** Shows the database connection: laborlist_and_wages/postgres@PostgreSQL 14*.
- Properties:** Tab showing table details.
- SQL:** Tab showing the DDL code.
- Statistics:** Tab showing performance metrics.
- Dependencies:** Tab showing dependencies.
- Dependents:** Tab showing dependents.
- Browser:** Left sidebar showing the schema tree:
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (21):
 - approves
 - bank_info
 - business_account
 - business_info
 - governmentOfficials
 - hires
 - hourlyLabors
 - labor
 - labor_supervisor
 - leave
 - location_info
 - payment_info
 - pays
 - personal_info
 - schedule_info
 - schedules
 - supervisor_manager
 - takesLeave
 - upInfo
 - users_info
 - weeklyLabors
 - Trigger Functions
 - Types
 - Views
 - public
 - Subscriptions
 - pg_statistic
 - Login/Group Roles
- Query:** Tab showing the executed DDL code.
- Output:** Tab showing the execution results and notices.

The DDL code in the Query tab is:

```

-- Table: laborlist_and_wages_db.hourly_labors
-- DROP TABLE IF EXISTS laborlist_and_wages_db.hourly_labors;
CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.hourly_labors
(
    labor_id integer NOT NULL,
    hourly_wage integer NOT NULL,
    hours_worked integer NOT NULL,
    CONSTRAINT hourly_labors_pkey PRIMARY KEY (labor_id),
    CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
        REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.hourly_labors
OWNER to postgres;

```

The Output tab shows the following message:

```

NOTICE: relation "hourly_labors" already exists, skipping
ALTER TABLE
Query returned successfully in 210 msec.

```

13. Weekly_labors table

DDL Script:

```
-- Table: laborlist_and_wages_db.weekly_labors

-- DROP TABLE IF EXISTS laborlist_and_wages_db.weekly_labors;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.weekly_labors
(
    labor_id integer NOT NULL,
    weekly_wage integer NOT NULL,
    hours_worked integer NOT NULL,
    CONSTRAINT weekly_labors_pkey PRIMARY KEY (labor_id),
    CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
        REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.weekly_labors
OWNER to postgres;
```

DDL Snapshot:

```

1 -- Table: laborlist_and_wages_db.weekly_labors
2
3 -- DROP TABLE IF EXISTS laborlist_and_wages_db.weekly_labors;
4
5 CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.weekly_labors
6 (
7     labor_id integer NOT NULL,
8     weekly_wage integer NOT NULL,
9     hours_worked integer NOT NULL,
10    CONSTRAINT weekly_labors_pkey PRIMARY KEY (labor_id),
11    CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
12        REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
13        ON UPDATE NO ACTION
14        ON DELETE NO ACTION
15        NOT VALID
16 )
17
18 TABLESPACE pg_default;
19
20 ALTER TABLE IF EXISTS laborlist_and_wages_db.weekly_labors
21 OWNER to postgres;

```

NOTICE: relation "weekly_labors" already exists, skipping
ALTER TABLE

Query returned successfully in 58 msec.

Total rows: 0 of 0 Query complete 00:00:00.058 Ln 21, Col 23

14. Supervisor_manager table

DDL Script:

```

-- Table: laborlist_and_wages_db.supervisor_manager

-- DROP TABLE IF EXISTS laborlist_and_wages_db.supervisor_manager;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.supervisor_manager
(
    lowerlevel_supervisor_id integer NOT NULL,
    manager_id integer NOT NULL,
    supervised_since date,
    CONSTRAINT supervisor_manager_pkey PRIMARY KEY (lowerlevel_supervisor_id,
manager_id),
    CONSTRAINT manager_fkey FOREIGN KEY (manager_id)
        REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id)
MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION

```

```

        NOT VALID,
CONSTRAINT supervisor_fkey FOREIGN KEY (lowerlevel_supervisor_id)
    REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id)
MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.supervisor_manager
OWNER to postgres;

```

DDL Snapshot:

The screenshot shows the PgAdmin 4 interface with the 'Query' tab selected. The query editor contains the DDL code for the supervisor_manager table. The code includes a primary key constraint (supervisor_manager_pkey) on the columns lowerlevel_supervisor_id and manager_id, a foreign key constraint (supervisor_fkey) referencing the supervisor_id column in the labor_supervisor table, and another foreign key constraint (manager_fkey) referencing the manager_id column in the same table. The table is defined with columns lowerlevel_supervisor_id, manager_id, and supervised_since_date. The table is created in the pg_default tablespace and owned by the postgres user.

```

CREATE TABLE supervisor_manager (
    lowerlevel_supervisor_id integer NOT NULL,
    manager_id integer NOT NULL,
    supervised_since date,
    CONSTRAINT supervisor_manager_pkey PRIMARY KEY (lowerlevel_supervisor_id, manager_id),
    CONSTRAINT manager_fkey FOREIGN KEY (manager_id) REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT supervisor_fkey FOREIGN KEY (lowerlevel_supervisor_id)
        REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
TABLESPACE pg_default;
ALTER TABLE IF EXISTS laborlist_and_wages_db.supervisor_manager
OWNER to postgres;

```

15. Personal_info table

DDL Script:

```

-- Table: laborlist_and_wages_db.personal_info

-- DROP TABLE IF EXISTS laborlist_and_wages_db.personal_info;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.personal_info
(
    user_id integer,
    labor_id integer,
    supervisor_id integer,
    official_id integer,
    mobile_number integer,
    type_of_user character varying COLLATE pg_catalog."default",
    name character varying COLLATE pg_catalog."default",
    email character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT personal_info_pkey PRIMARY KEY (email),
    CONSTRAINT unique_labor UNIQUE (labor_id),
    CONSTRAINT unique_official UNIQUE (official_id),
    CONSTRAINT unique_supervisor_pinfo UNIQUE (supervisor_id),
    CONSTRAINT unique_user UNIQUE (user_id),
    CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
        REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT official_fkey FOREIGN KEY (official_id)
        REFERENCES laborlist_and_wages_db.governmentOfficials (official_id)
    MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
        REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id)
    MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT user_fkey FOREIGN KEY (user_id)
        REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

```

```

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.personal_info
OWNER to postgres;

```

DDL Snapshots:

```

PgAdmin  File  Object  Tools  Help
Browser  Dashboard  Properties  SQL  Statistics  Dependencies  Dependents  laborlist_and_wages/postgres@PostgreSQL 14*
Query  Query History
25   CONSTRAINT official_fkey FOREIGN KEY (official_id)
26     REFERENCES laborlist_and_wages_db.governmentOfficials (official_id) MATCH SIMPLE
27     ON UPDATE NO ACTION
28     ON DELETE NO ACTION
29     NOT VALID,
30   CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
31     REFERENCES laborlist_and_wages_db.laborSupervisor (supervisor_id) MATCH SIMPLE
32     ON UPDATE NO ACTION
33     ON DELETE NO ACTION
34     NOT VALID,
35   CONSTRAINT user_fkey FOREIGN KEY (user_id)
36     REFERENCES laborlist_and_wages_db.usersInfo (user_id) MATCH SIMPLE
37     ON UPDATE NO ACTION
38     ON DELETE NO ACTION
39     NOT VALID,
40 )
41
42 TABLESPACE pg_default;
43
44 ALTER TABLE IF EXISTS laborlist_and_wages_db.personal_info
45   OWNER to postgres;
Data output  Messages  Notifications
NOTICE: relation "personal_info" already exists, skipping
ALTER TABLE
Query returned successfully in 59 msec.
Total rows: 0 of 0  Query complete 00:00:00.059  Ln 45, Col 23
Query returned successfully in 59 msec.
Ln 45, Col 23

```

16. Schedules table

DDL Script:

```

-- Table: laborlist_and_wages_db.schedules

-- DROP TABLE IF EXISTS laborlist_and_wages_db.schedules;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.schedules
(
    user_id integer,

```

```

supervisor_id integer,
labor_id integer NOT NULL,
schedule_id integer NOT NULL,
CONSTRAINT schedules_pkey PRIMARY KEY (schedule_id),
CONSTRAINT schedule_unique UNIQUE (schedule_id),
CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
    REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
    REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id)
MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT user_fkey FOREIGN KEY (user_id)
    REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.schedules
OWNER to postgres;

```

DDL Snapshots:

```

PgAdmin  File  Object  Tools  Help
Browser  Dashboard  Properties  SQL  Statistics  Dependencies  Dependents  laborlist_and_wages/postgres@PostgreSQL 14*
Query  Query History
13   CONSTRAINT Labor_fkey FOREIGN KEY (labor_id)
14     REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
15     ON UPDATE NO ACTION
16     ON DELETE NO ACTION
17     NOT VALID,
18   CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
19     REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id) MATCH SIMPLE
20     ON UPDATE NO ACTION
21     ON DELETE NO ACTION
22     NOT VALID,
23   CONSTRAINT user_fkey FOREIGN KEY (user_id)
24     REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
25     ON UPDATE NO ACTION
26     ON DELETE NO ACTION
27     NOT VALID,
28 )
29
30   TABLESPACE pg_default;
31
32 ALTER TABLE IF EXISTS laborlist_and_wages_db.schedules
33   OWNER to postgres;
Data output  Messages  Notifications
NOTICE: relation "schedules" already exists; skipping.
ALTER TABLE
Query returned successfully in 27 msec.
Ln 33, Col 23
Query returned successfully in 27 msec.
Ln 33, Col 23
Total rows: 0 of 0  Query complete 00:00:00.027

```

17. Hires table

DDL Script:

```

-- Table: laborlist_and_wages_db.hires

-- DROP TABLE IF EXISTS laborlist_and_wages_db.hires;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.hires
(
    user_id integer,
    business_id integer,
    labor_id integer NOT NULL,
    CONSTRAINT unique_hire UNIQUE (user_id, business_id, labor_id),
    CONSTRAINT business_fkey FOREIGN KEY (business_id)
        REFERENCES laborlist_and_wages_db.business_info (business_id) MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,

```

```

CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
    REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT user_fkey FOREIGN KEY (user_id)
    REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.hires
OWNER to postgres;

```

DDL Snapshots:

```

PgAdmin  File  Object  Tools  Help  Dashboard  Properties  SQL  Statistics  Dependencies  Dependents  laborlist_and_wages/postgres@PostgreSQL 14*
Browser  FTS Templates  Foreign Tables  Functions  Materialized Views  Operators  Procedures  Sequences  Tables (21)  approves  bank_info  business_account  business_info  governmentOfficials  hires  hourlyLabors  labor  labor_supervisor  leave  location_info  payment_info  pays  personal_info  schedule_info  schedules  supervisor_manager  bikesLeave  upInfo  usersInfo  weeklyLabors  Trigger Functions  Types  Views  public  Subscriptions  postgres  Login/Group Roles
Query  Query History
11  CONSTRAINT business_fkey FOREIGN KEY (business_id)
12      REFERENCES laborlist_and_wages_db.business_info (business_id) MATCH SIMPLE
13      ON UPDATE NO ACTION
14      ON DELETE NO ACTION
15      NOT VALID,
16  CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
17      REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
18      ON UPDATE NO ACTION
19      ON DELETE NO ACTION
20      NOT VALID,
21  CONSTRAINT user_fkey FOREIGN KEY (user_id)
22      REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
23      ON UPDATE NO ACTION
24      ON DELETE NO ACTION
25      NOT VALID
26 )
27
28 TABLESPACE pg_default;
29
30 ALTER TABLE IF EXISTS laborlist_and_wages_db.hires
31 OWNER to postgres;
Data output  Messages  Notifications
NOTICE: relation "hires" already exists, skipping
ALTER TABLE
Query returned successfully in 485 msec.
Total rows: 0 of 0  Query complete 00:00:00.485
Query returned successfully in 485 msec.
Ln 31, Col 23

```

18. Takes_leave table

DDL Script:

```
-- Table: laborlist_and_wages_db.takes_leave

-- DROP TABLE IF EXISTS laborlist_and_wages_db.takes_leave;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.takes_leave
(
    user_id integer,
    leave_id integer NOT NULL,
    supervisor_id integer,
    labor_id integer NOT NULL,
    CONSTRAINT takes_leave_pkey PRIMARY KEY (leave_id),
    CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
        REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
        REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id)
MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT user_fkey FOREIGN KEY (user_id)
        REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.takes_leave
    OWNER to postgres;
```

DDL Snapshots:

```

1 -- Table: laborlist_and_wages_db.takes_leave
2
3 -- DROP TABLE IF EXISTS laborlist_and_wages_db.takes_leave;
4
5 CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.takes_leave
6 (
7     user_id integer,
8     leave_id integer NOT NULL,
9     supervisor_id integer,
10    labor_id integer NOT NULL,
11    CONSTRAINT takes_leave_pkey PRIMARY KEY (leave_id),
12    CONSTRAINT labor_fkey FOREIGN KEY (labor_id) MATCH SIMPLE
13        REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
14        ON UPDATE NO ACTION
15        ON DELETE NO ACTION
16        NOT VALID,
17    CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
18        REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id) MATCH SIMPLE
19        ON UPDATE NO ACTION
20        ON DELETE NO ACTION
21        NOT VALID,
22    CONSTRAINT user_fkey FOREIGN KEY (user_id)
23        REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
24        ON UPDATE NO ACTION
25        ON DELETE NO ACTION
26        NOT VALID,
27 );

```

NOTICE: relation "takes_leave" already exists, skipping
ALTER TABLE

Query returned successfully in 34 msec.

Total rows: 0 of 0 Query complete 00:00:00.034 Ln 32, Col 23

19. Approves table

DDL Script:

```

-- Table: laborlist_and_wages_db.approves

-- DROP TABLE IF EXISTS laborlist_and_wages_db.approves;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.approves
(
    user_id integer,
    leave_id integer NOT NULL,
    supervisor_id integer,
    labor_id integer NOT NULL,
    CONSTRAINT approves_pkey PRIMARY KEY (leave_id),
    CONSTRAINT approve_unique UNIQUE (user_id, supervisor_id, labor_id,
    leave_id),
    CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
        REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
        ON UPDATE NO ACTION

```

```

        ON DELETE NO ACTION
        NOT VALID,
CONSTRAINT leave_fkey FOREIGN KEY (leave_id)
    REFERENCES laborlist_and_wages_db.takes_leave (leave_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
    REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id)
MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT user_fkey FOREIGN KEY (user_id)
    REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.approves
OWNER to postgres;

```

DDL Snapshots:

```

PgAdmin  File  Object  Tools  Help
Browser  Dashboard  Properties  SQL  Statistics  Dependencies  Dependents  laborlist_and_wages/postgres@PostgreSQL 14*
Tables (21)  Approves  Bank_info  Business_account  GovernmentOfficials  Hires  HourlyJabors  Labor  Labor_supervisor  Leave  Location_info  Payment_info  Pays  Personal_info  Schedule_info  Schedules  Supervisor_manager  Takes_leave  Up_info  Users_info  Weekly_jabors  Trigger Functions  Types  Views  public  Subscriptions  postges  Login/Group Roles
Approves
  user_id integer,
  leave_id integer NOT NULL,
  supervisor_id integer,
  labor_id integer NOT NULL,
  CONSTRAINT approves_pkey PRIMARY KEY (leave_id),
  CONSTRAINT approve_unique UNIQUE (user_id, supervisor_id, labor_id, leave_id),
  CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
    REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
  CONSTRAINT leave_fkey FOREIGN KEY (leave_id)
    REFERENCES laborlist_and_wages_db.takes_leave (leave_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
  CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
    REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
  CONSTRAINT user_fkey FOREIGN KEY (user_id)
    REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
ALTER TABLE
NOTICE: relation "approves" already exists, skipping
Query returned successfully in 34 msec.
Total rows: 0 of 0  Query complete 00:00:00.034  Ln 38, Col 23
  ✓ Query returned successfully in 34 msec.

```

20. Pays table

DDL Script:

```

-- Table: laborlist_and_wages_db.pays

-- DROP TABLE IF EXISTS laborlist_and_wages_db.pays;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.pays
(
    supervisor_id integer,
    labor_id integer NOT NULL,
    payment_number integer NOT NULL,
    user_id integer,
    CONSTRAINT pays_pkey PRIMARY KEY (payment_number),
    CONSTRAINT labor_fkey FOREIGN KEY (labor_id)
        REFERENCES laborlist_and_wages_db.labor (labor_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,

```

```

CONSTRAINT supervisor_fkey FOREIGN KEY (supervisor_id)
    REFERENCES laborlist_and_wages_db.labor_supervisor (supervisor_id)
MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT user_fkey FOREIGN KEY (user_id)
    REFERENCES laborlist_and_wages_db.users_info (user_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.pays
    OWNER to postgres;

```

DDL Snapshots:

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

- Browser:** Shows the database structure under "laborlist_and_wages/postgres@PostgreSQL 14*". The "Tables" node is expanded, showing tables like approves, bank_info, business_account, etc., and specifically highlighting the "pays" table.
- Query Editor:** Displays the SQL DDL code for creating the "pays" table. The code includes constraints for primary key (pays_pkey), foreign keys (supervisor_fkey and user_fkey), and various update/delete actions. It also specifies the table's owner as "postgres" and its tablespace as "pg_default".
- Data Output:** Shows the execution results, including a notice about the table already existing and being skipped, and a confirmation message indicating the query was executed successfully in 34 msec.
- Status Bar:** At the bottom right, it says "Query returned successfully in 34 msec." and "Ln 32, Col 23".

21. Payment_info table

DDL Script:

```
-- Table: laborlist_and_wages_db.payment_info

-- DROP TABLE IF EXISTS laborlist_and_wages_db.payment_info;

CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.payment_info
(
    payment_number integer NOT NULL,
    payment_id character varying COLLATE pg_catalog."default" NOT NULL,
    paid_for_byhours integer NOT NULL,
    amount_paid integer NOT NULL,
    date date NOT NULL,
    CONSTRAINT payment_info_pkey PRIMARY KEY (payment_number),
    CONSTRAINT payment_number_fkey FOREIGN KEY (payment_number)
        REFERENCES laborlist_and_wages_db.pays (payment_number) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS laborlist_and_wages_db.payment_info
    OWNER to postgres;
```

DDL Snapshots:

```

1 -- Table: laborlist_and_wages_db.payment_info
2
3 -- DROP TABLE IF EXISTS laborlist_and_wages_db.payment_info;
4
5 CREATE TABLE IF NOT EXISTS laborlist_and_wages_db.payment_info
6 (
7     payment_number integer NOT NULL,
8     payment_id character varying COLLATE pg_catalog."default" NOT NULL,
9     account_number_receiver character varying COLLATE pg_catalog."default" NOT NULL,
10    account_number_employer character varying COLLATE pg_catalog."default" NOT NULL,
11    "paid_for_byHours" integer NOT NULL,
12    amount_paid integer NOT NULL,
13    date_date NOT NULL,
14    type_of_employer character varying COLLATE pg_catalog."default" NOT NULL,
15    CONSTRAINT payment_info_pkey PRIMARY KEY (payment_number),
16    CONSTRAINT payment_number_fkey FOREIGN KEY (payment_number)
17        REFERENCES laborlist_and_wages_db.pays (payment_number) MATCH SIMPLE
18        ON UPDATE NO ACTION
19        ON DELETE NO ACTION
20        NOT VALID;
21 )
22
23 TABLESPACE pg_default;
24
25 ALTER TABLE IF EXISTS laborlist_and_wages_db.payment_info
26     OWNER to postgres;

```

NOTICE: relation "payment_info" already exists, skipping
ALTER TABLE

Query returned successfully in 62 msec.

Total rows: 0 of 0 Query complete 00:00:00.062 Ln 8, Col 38

Query returned successfully in 62 msec.

16.3 Data Snapshots

SQL Query for Showing Data:

```
SELECT * FROM laborlist_and_wages_db.location_info
```

```
SELECT * FROM laborlist_and_wages_db.schedule_info
```

```
SELECT * FROM laborlist_and_wages_db.bank_info
```

```
SELECT * FROM laborlist_and_wages_db.leave
```

```
SELECT * FROM laborlist_and_wages_db.upi_info
```

```
SELECT * FROM laborlist_and_wages_db.business_account
```

```
SELECT * FROM laborlist_and_wages_db.governmentOfficials
```

```
SELECT * FROM laborlist_and_wages_db.business_info
```

```
SELECT * FROM laborlist_and_wages_db.labor_supervisor  
  
SELECT * FROM laborlist_and_wages_db.labor  
  
SELECT * FROM laborlist_and_wages_db.users_info  
  
SELECT * FROM laborlist_and_wages_db.hourly_labors  
  
SELECT * FROM laborlist_and_wages_db.weekly_labors  
  
SELECT * FROM laborlist_and_wages_db.supervisor_manager  
  
SELECT * FROM laborlist_and_wages_db.personal_info  
  
SELECT * FROM laborlist_and_wages_db.schedules  
  
SELECT * FROM laborlist_and_wages_db.hires  
  
SELECT * FROM laborlist_and_wages_db.takes_leave  
  
SELECT * FROM laborlist_and_wages_db.approves  
  
SELECT * FROM laborlist_and_wages_db.pays  
  
SELECT * FROM laborlist_and_wages_db.payment_info
```

1. Location_info Table

The screenshot shows the PgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'Tables (21)'. The 'location_info' table is selected and highlighted with a blue border. The main area has a 'Query' tab containing the following SQL code:

```

1 SELECT * FROM laborlist_and_wages_db.location_info
2 ORDER BY location_id ASC

```

The results are displayed in a table titled 'Data output' with three columns: 'location_id', 'city', and 'district'. The data consists of 60 rows, each with a unique location ID, a city name, and a district name. A message at the bottom right indicates the query was successfully run.

location_id	city	district
1	Yablonovskiy	Linoan
2	Cansolungon	Wangbuzhueng
3	Ludivka	Putinci
4	Wudil	Melbourne
5	Diguuma	Uduka e Eperme
6	Bytkv	Santo Antônio do...
7	Iperó	Klippan
8	Yrea	Liutao
9	Santiago de Cac...	Londini
10	Alamnay	Ajila Yanvara
11	Sacomo	Dortmund
12	Ulaan-Ereg	Santo Tomás
13	Sarakhs	Zürich
14	Tolsto	Yataity del Norte
15	Sovetskiy	Pérmet
16	Ibrá	Nizhny Tagil
17	Rude	Zumiao
18	Pa Sang	Teberda
19	Krasiczyn	Rihimški

Total rows: 60 of 60 Query complete 00:00:00.806 Successfully run. Total query runtime: 806 msec. 60 rows affected.

2. Schedule_info Table

PgAdmin File Object Tools Help

Browser

- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (21)
 - > approves
 - > bank_info
 - > business_account
 - > business_info
 - > governmentOfficials
 - > hires
 - > hourlyLabors
 - > labor
 - > labor_supervisor
 - > leave
 - > location_info
 - > payment_info
 - > pays
 - > personal_info
 - > schedule_info
 - > schedules
 - > supervisor_manager
 - > takesLeave
 - > up_info
 - > users_info
 - > weekly_labors
- > Trigger Functions
- > Types
- > Views
- > public

Dashboard Properties SQL Statistics Dependencies Dependents [laborlist_and_wages_db.schedule_info/laborlist_and_wages/po...](#) [laborlist_and_wages_db.schedule_info/laborlist_and_wages/po...](#) [laborlist_and_wages_db.schedule_info/laborlist_and_wages/po...](#)

Query History

```
1 SELECT * FROM laborlist_and_wages_db.schedule_info
2 ORDER BY schedule_id ASC
```

Data output Messages Notifications

schedule_id	entry_time	time without time zone	exit_time	time without time zone	date	date
1	1	11:16:00	16:40:00		2022-10-21	
2	2	14:43:00	20:56:00		2022-05-28	
3	3	09:55:00	16:38:00		2022-09-07	
4	4	11:01:00	19:41:00		2022-01-11	
5	5	14:43:00	18:43:00		2022-01-03	
6	6	11:10:00	18:49:00		2022-05-08	
7	7	13:33:00	16:24:00		2022-01-20	
8	8	10:35:00	20:35:00		2022-01-29	
9	9	17:12:00	19:42:00		2021-12-27	
10	10	14:16:00	19:32:00		2022-11-09	
11	11	13:01:00	20:29:00		2022-08-19	
12	12	14:12:00	18:17:00		2022-06-25	
13	13	14:33:00	18:55:00		2022-04-10	
14	14	13:48:00	19:46:00		2022-05-31	
15	15	07:23:00	15:20:00		2022-07-27	
16	16	17:18:00	15:51:00		2022-03-01	
17	17	12:51:00	15:59:00		2022-06-14	
18	18	12:48:00	19:29:00		2021-12-26	
19	19	06:38:00	17:42:00		2022-01-08	

Total rows: 60 of 60 | Query complete 00:00:00.725 | Successfully run. Total query runtime: 725 msec. 60 rows affected.

3. Bank_info table

PgAdmin File Object Tools Help

Browser > FTS Configurations > FTS Dictionaries > FTS Parsers > FTS Templates > Foreign Tables > Functions > Materialized Views > Operators > Procedures > Sequences > Tables (21) > approves > bank_info > business_account > business_info > governmentOfficials > hires > hourlyLaborers > labor > labor_supervisor > leave > location_info > payment_info > pays > personal_info > schedule_info > schedules > supervisorManager > takesLeave > up_info > users_info > weeklyLaborers > Trigger Functions > Types > Views > public

Dashboard Properties SQL Statistics Dependencies Dependents laborlist_and... laborlist_and... laborlist_and... laborlist_and... laborlist_and... laborlist_and... laborlist_and...

Query Query History

```
1 SELECT * FROM Laborlist_and_wages_db.bank_info
2 ORDER BY account_number ASC
```

Data output Messages Notifications

account_number	fsc_code	name	email	user_id	type_of_user
104652962105	LZGZ0439145	Eryn Turville	eturville47@twinkl...	152	supervisor
106014000232	KVZG0405254	Larry Alp	la976@mcaas.gov	259	official
106796877612	BRYV0057046	Ora Weddell	oweddell21@om...	74	labor
107302617230	ZMWV0130256	Alix Skains	askains70@alege...	280	official
112579839396	BBJFD417877	Rollie Goodwill	rgoodwill12@pin...	39	user
119393631681	CAWV0221116	Maurise Pinock	mpinock83@hhs...	289	official
124029818183	PNPX0071689	Eleanor Culverhouse	eculverhouse21@o...	94	labor
124611301056	KOJC0577531	Meredith Kasper	mkasper29@un...	101	labor
131331128588	DEFRZ0919351	Rastian Lowthorpe	blowthorpe2n@ba...	96	labor
133233020359	JUCS0597686	Cameron Payn	cpayn49@sweth...	154	supervisor
13399417541	GUKP0247014	Vivienne Casterton	vcasterton7@t...	282	official
133982615930	CBAR0956323	Konstance Castle	kcastleton2q@d...	99	labor
139966889123	MZPI0061717	Mattheus Duebus	mduebusberry66@j...	295	official
140560588056	AGIUW0342446	Jacchim Pagan	jpagansc@devnu...	193	business
145810273412	CIHH0223307	Nikki Gerritzen	ngerritzen3u@di...	139	supervisor
148456509144	IQZD0178258	Maren Branford	mbanford2z@go...	3	user
150860783048	IIQ00105525	Adrien Dunes	adances8@cbene...	9	user
157893350626	VBFR0934143	Kelsy Matus	kmatus96@furlin...	218	business
161376655877	DRN60142808	Jaquith Giorgio	jgiorgio6n@ovh...	240	business

Total rows: 300 of 300 | Query complete 00:00:00.602 | Successfully run. Total query runtime: 602 msec. 300 rows affected.

4. Leave table

The screenshot shows the PgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'laborlist_and_wages_db.leave'. The 'Tables (21)' section is expanded, showing tables like approves, bank_info, business_account, business_info, governmentOfficials, hires, hourlyLabors, labor, labor_supervisor, leave, location_info, payment_info, pays, personalInfo, schedule_info, schedules, supervisorManager, takesLeave, up_info, users_info, weekly_labors, and trigger functions. The 'leave' table is currently selected.

The main area has tabs for 'Query' and 'Query History'. The 'Query' tab contains the following SQL code:

```

1 SELECT * FROM laborlist_and_wages_db.leave
2 ORDER BY leave_id ASC

```

The 'Data output' tab displays the results of the query as a table:

leave_id	leave_date
1	2022-08-17
2	2022-06-24
3	2022-06-07
4	2022-09-24
5	2022-07-17
6	2022-01-23
7	2022-04-10
8	2022-01-06
9	2022-07-08
10	2022-06-09
11	2022-09-29
12	2022-05-20
13	2022-10-18
14	2022-11-12
15	2022-07-21
16	2022-08-05
17	2022-07-04
18	2022-10-06
19	2022-02-23

At the bottom of the results pane, it says 'Total rows: 60 of 60' and 'Query complete 00:00:00.571'. A green status bar at the bottom right indicates 'Successfully run. Total query runtime: 571 msec. 60 rows affected.'

5. Upi_info table

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

- Toolbar:** File, Object, Tools, Help.
- Browser:** FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (21), Approves, Bank_Info, Business_Account, Business_Info, GovernmentOfficials, Hires, HourlyLabors, Labor, Labor_Supervisor, Leave, Location_Info, Payment_Info, Pays, Personal_Info, Schedule_Info, Schedules, SupervisorManager, TakesLeave, Upi_Info, Users_Info, Weekly_Labors, Trigger Functions, Types, Views, public.
- Query Editor:**劳动工资数据库中的upi_info表。SQL语句如下：


```
1 SELECT * FROM laborlist_and_wages_db.upi_info
2 ORDER BY account_number ASC, upi_id ASC
```
- Data Output:** 显示了300行数据，每行包含account_number和upi_id列。部分数据如下：

account_number	upi_id
104652962105	0722762067midjyed
106014000232	055462409fimrepnz
106796877612	8641209992dmalthe
107302617230	2008261110gqsmu
112579839396	5778102299pxzhl
119393631681	4092605797beedk
124029818183	8766969874gxmrp
124611301056	4976731522bpjhnt
131331128588	0008272803dvtbgm
133233020359	3156685728uxexn
133299417541	5955899287gzcwk
133982615930	6110845481tobtus
139966889123	4484162357hxifvg
140560588056	052915277beamnfo
145810273412	9394678791okuyuw
148456509144	4209524131ccwvwq
150860793048	5270362494wianfh
157893350626	5819392021oxbhlc
161376655877	6523759140bfddwn
- Status Bar:** Total rows: 300 of 300 | Query complete 00:00:00.671 | Successfully run. Total query runtime: 671 msec. 300 rows affected.

6. Business_account table

The screenshot shows the PgAdmin 4 interface. The left sidebar (Browser) lists various database objects under the schema 'laborlist_and_wages_db.business_account'. The central area shows a query window with the following SQL code:

```
1 SELECT * FROM laborlist_and_wages_db.business_account
2 ORDER BY email ASC, account_number ASC
```

The results pane displays the data from the 'business_account' table, which includes columns 'email' and 'account_number'. The data is sorted by 'email' in ascending order and 'account_number' in ascending order. The results show 60 rows of data.

	email	account_number
1	aalderson5o@wits.uk	969349800981
2	acuel5u@wsj.com	203593014709
3	adavid5n@pinteres...	932401608225
4	aealgrey50@discuz...	580452551897
5	ajosephson5a@ezine...	292403716483
6	anoye5t@shutterfly.c...	402716561802
7	asherebrooke5e@elpat...	869451373225
8	bhoogendorpb@care...	599807564273
9	bjunowicz58@ftc.gov	366436853358
10	cacuma50@51.la	568239797065
11	cchester5i@epa.gov	335501243015
12	cibson58@irs.gov	895261499952
13	elitchfield55@blogger...	642459436045
14	ctrex57@amazon.co...	724174518559
15	ewinifer55@biglobe...	988325658852
16	dbroomfield67@tcm...	230093194186
17	drudd5x@europa.eu	94689789663
18	dwestcottg@ramble...	899938309933
19	eryston5i@paginegl...	701870281052

Total rows: 60 of 60 Query complete 00:00:00.298 Ln 2, Col 40

7. GovernmentOfficials table

The screenshot shows the PgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'Tables (21)'. One table, 'governmentOfficials', is selected. The main area has a 'Query' tab containing the following SQL code:

```
1 SELECT * FROM laborlist_and_wages_db.governmentOfficials
2 ORDER BY official_id ASC
```

The results are displayed in a table with two columns: 'officialId' and 'locationId'. The data consists of 60 rows, each with a unique official ID and location ID. A message at the bottom right indicates the query was successfully run.

officialId	locationId
1	241
2	242
3	243
4	244
5	245
6	246
7	247
8	248
9	249
10	250
11	251
12	252
13	253
14	254
15	255
16	256
17	257
18	258
19	259
20	260
21	261
22	262
23	263
24	264
25	265
26	266
27	267
28	268
29	269
30	270
31	271
32	272
33	273
34	274
35	275
36	276
37	277
38	278
39	279
40	280
41	281
42	282
43	283
44	284
45	285
46	286
47	287
48	288
49	289
50	290
51	291
52	292
53	293
54	294
55	295
56	296
57	297
58	298
59	299
60	300

Total rows: 60 of 60 Query complete 00:00:00.385 Successfully run. Total query runtime: 385 msec. 60 rows affected

8. Business_info table

```

SELECT * FROM laborlist_and_wages_db.business_info
ORDER BY business_id ASC, email ASC

```

business_id	location_id	official_id	name	email	regulated_since
1	181	3152	Collins and Sons	rstolley50@newyork...	2021-12-30
2	182	3146	Wiegand-Koelpin	eskater51@nasa.gov	2022-10-21
3	183	3159	Quigley-Hudson	mfrancis52@reuters...	2022-08-11
4	184	3160	Turcotte-Weber	cswinffen53@tighe...	2022-05-20
5	185	3139	Muller, Leach and	jpoymon54@flavors...	2022-04-03
6	186	3124	Hodkiewicz, Fay...	clifchleif55@logger...	2022-03-23
7	187	3113	Ziemann-Hintz	hblackley56@cdc.gov	2022-02-19
8	188	3150	Fisher-Parisan	clorek57@amazon.co...	2022-09-06
9	189	3135	McClure-Dasper	bjanowicz58@itc.gov	2022-01-05
10	190	3155	Walker and Sons	lcoule59@state.gov	2022-07-16
11	191	3159	Nolan-Drady	ajosephson5a@xine...	2022-01-31
12	192	3140	Bauch-Larkin an...	aedgley5b@issuz...	2021-12-13
13	193	3157	Rebel-Schamber	jpagan5c@devhub.co...	2022-09-19
14	194	3137	Erdman-Dickinson	welgrub5d@so-net.ne.jp	2022-09-11
15	195	3105	Konopelski Inc	asherebrook5e@elpai...	2022-02-04
16	196	3116	Tillman Inc	gondirisco5f@osac.g...	2022-04-04
17	197	3159	Kub-Kuphal and	shakiet5g@ribu.com	2022-03-14
18	198	3104	Emmerich-Ratke	obolduc5h@nymdns.o...	2022-08-12
19	199	3109	Hill-Gorczany an...	eroyston5i@pagegl...	2022-08-01

Total rows: 60 of 60 Query complete 00:00:00.661 Successfully run. Total query runtime: 661 msec. 60 rows affected.

9. Labor_supervisor table

PgAdmin

File ▾ Object ▾ Tools ▾ Help ▾

Browser

- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- Tables (21)
 - > approves
 - > bank_info
 - > business_account
 - > business_info
 - > governmentOfficials
 - > hires
 - > hourlyLabors
 - > labor
 - > labor_supervisor
 - > leave
 - > location_info
 - > payment_info
 - > pays
 - > personalInfo
 - > schedule_info
 - > schedules
 - > supervisor_manager
 - > takesLeave
 - > up_info
 - > users_info
 - > weekly_labors
- > Trigger Functions
- > Types
- > Views
- > public

laborlist_and_wages_db.labor_supervisor/laborlist_and_wages_db.labor_supervisor

No limit ▾

Query History

```
1 SELECT * FROM laborlist_and_wages_db.labor_supervisor
2 ORDER BY supervisor_id ASC
```

Data output Messages Notifications

supervisor_id	managed_since	business_id	location_id	account_number
1	121	2020-04-25	182	3138 918850167327
2	122	2020-06-29	226	3144 514123985878
3	123	2022-09-08	187	3116 83321069641
4	124	2022-05-06	189	49334411423
5	125	2019-06-20	224	491223410475
6	126	2021-04-17	193	3143 729147815677
7	127	2020-09-21	189	3118 848314642376
8	128	2021-12-07	187	3146 665930567382
9	129	2019-07-28	199	3139 422972110819
10	130	2022-08-09	185	3103 855382782821
11	131	2021-02-13	221	3153 310771477107
12	132	2021-07-31	208	3160 24299694311
13	133	2019-09-09	200	3148 722641505880
14	134	2021-07-26	215	3160 235176531055
15	135	2021-03-07	181	3151 230251297309
16	136	2019-11-27	223	3120 85687000417
17	137	2020-06-23	221	3145 466879496372
18	138	2021-03-29	231	3109 85848437520
19	139	2021-05-13	196	3128 145810275412

Total rows: 60 of 60 Query complete 00:00:00.514

Successfully run. Total query runtime: 514 msec. 60 rows affected.

10. Labor table

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

- Toolbar:** File, Object, Tools, Help.
- Browser:** FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (21), approvers, bank_info, business_account, business_info, governmentOfficials, hires, hourlyLaborers, labor, labor_supervisor, leave, location_info, payment_info, pays, personalInfo, schedule_info, schedules, supervisor_manager, takesLeave, up_info, users_info, weekly_labors, Trigger Functions, Types, Views, public.
- Query Editor:**劳动表查询语句: `SELECT * FROM laborlist_and_wages_db.labor ORDER BY labor_id ASC`
- Data Output:** 显示了 'labor' 表的 60 行数据，每行包含 labor_id, birth_date, work_hours, wage_remaining, rating, type_of_work, supervisor_id, supervised_since, location_id, account_number 等字段。
- Message Bar:** Total rows: 60 of 60 | Query complete 00:00:00.457 | Successfully run. Total query runtime: 457 msec. 60 rows affected.

labor_id	birth_date	work_hours	wage_remaining	rating	type_of_work	supervisor_id	supervised_since	location_id	account_number
1	1986-05-29	95	\$35.00	1	Gardener	130	2022-03-25	3123	827356014688
2	1977-09-06	57	\$5.00	5	Carpenter	167	2022-08-12	3142	341357647565
3	1988-03-29	62	\$50.00	1	Carpenter	124	2022-07-23	3148	213792854742
4	1986-12-02	51	\$18.00	3	Painter	145	2022-07-31	3109	831866853390
5	1984-03-09	77	\$41.00	4	Housekeeper	180	2022-06-28	3148	706015395842
6	1988-06-16	74	\$11.00	1	Barber	159	2022-06-22	3159	7713976959693
7	1977-03-20	22	\$31.00	3	Construction Wor...	125	2022-01-25	3120	181976498102
8	1991-03-17	54	\$4.00	5	Carpenter	155	2022-09-22	3126	74403052323
9	1986-10-16	55	\$23.00	2	Plumber	148	2022-08-12	3119	520484572504
10	1984-06-03	65	\$48.00	5	Electrician	130	2022-08-09	3139	864316055778
11	1961-05-23	81	\$14.00	2	Housekeeper	164	2022-04-12	3146	542765434017
12	1986-06-18	38	\$24.00	2	Construction Wor...	128	2022-07-22	3104	926811920358
13	1984-03-16	91	\$5.00	3	Painter	177	2023-09-10	3127	44063000275
14	1990-03-24	46	\$27.00	2	Carpenter	171	2022-08-27	3157	106796877612
15	1973-03-26	94	\$19.00	2	Gardener	144	2022-06-24	3144	655114191614
16	1984-03-28	37	\$8.00	4	Plumber	156	2022-06-24	3154	889831183161
17	1986-05-02	41	\$22.00	5	Carpenter	139	2022-07-10	3101	162944334608
18	1980-11-01	85	\$16.00	1	Gardener	187	2022-09-12	3156	711875587690
19	1981-10-30	92	\$41.00	2	Construction Wor...	124	2022-02-15	3105	215290900210

11. Users_info table

```

1: SELECT * FROM LaborList_and_Wages_db.users_info
2: ORDER BY user_id ASC

```

	user_id	account_number	location_id
1	1	298537795750	3152
2	2	544028694207	3145
3	3	148456509144	3160
4	4	867476259	3106
5	5	233313250426	3143
6	6	883907249600	3104
7	7	498375389985	3128
8	8	488790545998	3136
9	9	150860783048	3143
10	10	400874478344	3146
11	11	505369945455	3158
12	12	26790092820	3157
13	13	457324950031	3111
14	14	57934205593	3132
15	15	79193647990	3105
16	16	922549739603	3146
17	17	7279396949	3138
18	18	59403921618	3125
19	19	286872016277	3113

Total rows: 60 of 60 Query complete 00:00:00.455 Successfully run. Total query runtime: 455 msec. 60 rows affected.

12. Hourly_labors table

The screenshot shows the PgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'Tables (21)'. The 'hourly_Jabors' table is selected and highlighted in blue. The main area contains a query editor with the following SQL code:

```
1: SELECT * FROM laborlist_and_wages_db.hourly_Jabors
2: ORDER BY labor_id ASC
```

The results are displayed in a data grid with the following columns: labor_id, hourly_wage, and hours_worked. The data consists of 27 rows:

labor_id	hourly_wage	hours_worked
1	62	12
2	64	3
3	66	6
4	67	11
5	69	10
6	70	5
7	71	10
8	77	7
9	84	3
10	86	10
11	87	3
12	95	12
13	97	12
14	98	4
15	102	5
16	103	2
17	104	6
18	105	2
19	106	7

At the bottom of the interface, a message indicates: 'Total rows: 27 of 27 | Query complete 00:00:00.554 | Successfully run. Total query runtime: 554 msec. 27 rows affected.'

13. Weekly_labors table

The screenshot shows the PgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'laborlist_and_wages_db'. The 'Tables (21)' section is expanded, showing tables like approves, bank_info, business_account, business_info, governmentOfficials, hires, hourlyLabors, labor, labor_supervisor, leave, location_info, payment_info, pays, personalInfo, schedule_info, schedules, supervisor_manager, takesLeave, up_info, users_info, and weekly_labors. The 'weekly_labors' table is selected. The main area has tabs for 'Query' and 'Query History'. The 'Query' tab contains the following SQL code:

```
1 SELECT * FROM laborlist_and_wages_db.weekly_labors
2 ORDER BY labor_id ASC
```

The results are displayed in a table titled 'Data output' with three columns: labor_id, weekly_wage, and hours_worked. The data is as follows:

labor_id	weekly_wage	hours_worked
1	63	31
2	67	71
3	68	90
4	69	77
5	72	29
6	73	91
7	79	70
8	84	58
9	85	65
10	93	74
11	108	25
12	110	25
13	113	94
14	114	85
15	116	76

At the bottom, it says 'Total rows: 15 of 15' and 'Query complete 00:00:00.462'. A green status bar indicates 'Successfully run. Total query runtime: 462 msec. 15 rows affected.'

14. Supervisor_manager table

PgAdmin File Object Tools Help

Browser

- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (21)
 - > approves
 - > bank_info
 - > business_account
 - > business_info
 - > governmentOfficials
 - > hires
 - > hourly_labors
 - > labor
 - > labor_supervisor
 - > leave
 - > location_info
 - > payment_info
 - > pays
 - > personal_info
 - > schedule_info
 - > schedules
 - > supervisor_manager
 - > takes_leave
 - > upl_info
 - > users_info
 - > weekly_labors
- > Trigger Functions
- > Types
- > Views
- > public

Dashboard Properties SQL Statistics Dependencies Dependents [laborlist_and_wages_db.supervisor_manager/laborlist_and_wages_db.supervisor_manager](#) [laborlist_and_wages_db.supervisor_manager/laborlist_and_wages_db.supervisor_manager](#) [laborlist_and_wages_db.supervisor_manager/laborlist_and_wages_db.supervisor_manager](#) [laborlist_and_wages_db.supervisor_manager/laborlist_and_wages_db.supervisor_manager](#)

Query History

```
1 SELECT * FROM laborlist_and_wages_db.supervisor_manager
2 ORDER BY lowerlevel_supervisor_id ASC, manager_id ASC
```

Data output Messages Notifications

lowerlevel_supervisor_id	[PK] integer	manager_id	[PK] integer	supervised_since	date
1	122	131		2022-04-29	
2	140	158		2022-05-11	
3	152	177		2022-03-15	
4	177	180		2022-08-11	
5	180	170		2022-05-25	

Total rows: 5 of 5 Query complete 00:00:00.455 Ln 2, Col 55

The screenshot shows the PgAdmin interface with a query results table. The table has five rows and six columns. The columns are labeled: lowerlevel_supervisor_id, [PK] integer, manager_id, [PK] integer, supervised_since, and date. The data in the table is as follows:

lowerlevel_supervisor_id	[PK] integer	manager_id	[PK] integer	supervised_since	date
1	122	131		2022-04-29	
2	140	158		2022-05-11	
3	152	177		2022-03-15	
4	177	180		2022-08-11	
5	180	170		2022-05-25	

15. Personal_info table

PgAdmin File Object Tools Help

Browser

- > FTS Configurations
- > FTS Dictionaries
- > All FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (21)
 - > approves
 - > bank_info
 - > business_account
 - > business_info
 - > governmentOfficials
 - > hires
 - > hourlyLaborers
 - > labor
 - > labor_supervisor
 - > leave
 - > location_info
 - > payment_info
 - > pays
 - > personal_info
 - > schedule_info
 - > schedules
 - > supervisor_manager
 - > takesLeave
 - > upl_info
 - > users_info
 - > weekly_laborers
- > Trigger Functions
- > Types
- > Views
- > public

Dashboard Properties SQL Statistics Dependencies Dependents [laborlist_and_wages_db.personalInfo.laborlist_and_wages_db.personal_info](#) [laborlist_and_wages_db.personalInfo.laborlist_and_wages_db.personal_info](#) [laborlist_and_wages_db.personalInfo.laborlist_and_wages_db.personal_info](#)

Query History

```
1 SELECT * FROM LaborList_and_wages_db.personalInfo.laborlist_and_wages_db.personal_info
2 ORDER BY email ASC
```

Data output Messages Notifications

	user_id	labor_id	supervisor_id	official_id	mobile_number	type_of_user	name	email
1	[null]	[null]	148	[null]	7887054919	supervisor	Aleta Bendifx	abendifx43@nsw.gov.au
2	[null]	[null]	[null]	297	626575981	official	Anett Berntsen	abertson88@bduhos.com
3	[null]	113	[null]	[null]	9878026728	labor	Arvy Coggill	acoggill32@chicago.gov
4	[null]	9	[null]	[null]	6588214596	user	Adrien Daines	adaines81@cbsnews.com
5	[null]	[null]	151	[null]	7445033627	supervisor	Andonis Grice	agreco46@state.tx.us
6	[null]	16	[null]	[null]	631934716	user	Aldin Greenhalgh	agreenhalgh@atatos.com
7	[null]	53	[null]	[null]	6511502661	user	Armin Heinschke	aheinschke1@gsbwhi.com
8	[null]	[null]	125	[null]	7677073996	supervisor	Aprillete Lamph	alamotoishi@peu.edu
9	[null]	[null]	144	[null]	8408672397	supervisor	Aurora Lupson	alupson32@mysql.com
10	[null]	[null]	[null]	261	1699728106	official	Akim Meeus	ameeus78@bloga.com
11	[null]	[null]	[null]	130	1860843469	supervisor	Alejandro Messier	amessier3@kickstart.com
12	[null]	[null]	125	[null]	5888758820	supervisor	Avril Ninnoli	aninnoli3g@studiodreams.com
13	[null]	[null]	[null]	255	8928076472	official	Aerell Orchard	aorchard72@reuters.com
14	[null]	[null]	124	[null]	1582644650	supervisor	Amie Redholes	aredholes3f@eventbrite.com
15	[null]	57	[null]	[null]	8971224913	user	Andrej Rigosford	anglesford1@eckblog.com
16	[null]	[null]	[null]	280	408504681	official	Alix Skains	askains3@elopartth.com
17	[null]	50	[null]	[null]	7694093635	user	Angel Steptowe	asteptowe1@mitreypark.com
18	[null]	[null]	[null]	246	4524837056	official	Bridgette Andino	bandino61@aristher.com
19	[null]	[null]	89	[null]	3332539670	labor	Bethany Angrave	bangrave7g@yale.edu
20	[null]	[null]	[null]	163	3955113228	supervisor	Reh Banton	rbanton4b@virginia.edu

Total rows: 240 of 240 Query complete 00:00:00.333 Ln 1, Col 1

16. Schedules table

PgAdmin File Object Tools Help

Browser > FTS Configurations > FTS Dictionaries > FTS Parsers > FTS Templates > Foreign Tables > Functions > Materialized Views > Operators > Procedures > Sequences > Tables (21) > approves > bank_info > business_account > business_info > governmentOfficials > hires > hourly_labors > labor > labor_supervisor > leave > location_info > payment_info > pays > personalInfo > schedule_info > schedules > supervisor_manager > takesLeave > up_info > users_info > weekly_labors > Trigger Functions > Types > Views > public

laborlist_and_wages_db.schedules.laborlist_and_wages/postgr... No limit

Query History

```
1. SELECT * FROM laborlist_and_wages_db.schedules
2. ORDER BY schedule_id ASC
```

Data output Messages Notifications

user_id	supervisor_id	labor_id	schedule_id
1	47	154	87
2	5	180	91
3	33	135	111
4	38	177	105
5	11	124	116
6	53	123	85
7	33	145	120
8	56	174	85
9	10	161	66
10	31	168	107
11	28	123	99
12	26	146	93
13	45	135	85
14	4	156	90
15	39	179	114
16	6	142	63
17	46	147	93
18	59	139	82
19	42	131	98

Total rows: 60 of 60 | Query complete 00:00:00.320 | Successfully run. Total query runtime: 320 msec. 60 rows affected

17. Hires table

PgAdmin File Object Tools Help

Browser

- > FTS Configurations
- > FTS Dictionaries
- > FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (21)
 - > approves
 - > bank_info
 - > business_account
 - > business_info
 - > governmentOfficials
 - > hires
 - > hourlyLabors
 - > labor
 - > labor_supervisor
 - > leave
 - > location_info
 - > payment_info
 - > pays
 - > personal_info
 - > schedule_info
 - > schedules
 - > supervisor_manager
 - > takesLeave
 - > up_info
 - > users_info
 - > weekly_labors
- > Trigger Functions
- > Types
- > Views
- > public

laborlist_and_wages_db hires/laborlist_and_wages/postgres@P... No limit

Query History

```
1 SELECT * FROM laborlist_and_wages_db.hires
2
```

Data output Messages Notifications

	user_id	business_id	labor_id
1	27	222	62
2	31	220	117
3	13	236	98
4	44	233	111
5	17	225	82
6	30	218	73
7	26	181	61
8	11	206	111
9	32	218	109
10	55	200	82
11	47	223	105
12	56	218	65
13	4	223	76
14	20	224	64
15	19	197	83
16	37	210	76
17	50	187	115
18	18	205	108
19	56	188	106

Total rows: 60 of 60 | Query complete 00:00:00.208 | Successfully run. Total query runtime: 208 msec. 60 rows affected

18. Takes_leave table

The screenshot shows the PgAdmin interface with the 'laborlist_and_wages_db' database selected. In the left sidebar, under 'Tables (21)', the 'takes_leave' table is highlighted. The main pane displays the table's schema and data. The schema includes columns: user_id, leave_id, supervisor_id, and labor_id. The data pane shows 60 rows of data, each with values for these four columns. A status bar at the bottom right indicates the query was successfully run with a runtime of 325 msec and 60 rows affected.

	user_id	leave_id	supervisor_id	labor_id
1	31	1	128	119
2	58	2	140	62
3	16	3	125	75
4	44	4	138	97
5	21	5	139	76
6	31	6	141	65
7	41	7	155	86
8	39	8	180	73
9	20	9	130	66
10	49	10	175	85
11	27	11	143	98
12	33	12	172	119
13	54	13	163	107
14	14	14	169	61
15	4	15	132	79
16	37	16	152	62
17	10	17	148	115
18	29	18	140	98
19	30	19	155	88

19. Approves table

PgAdmin File Object Tools Help

Browser

- FTS Configurations
- FTS Dictionaries
- All FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (21)
 - approves
 - bank_info
 - business_account
 - business_info
 - governmentOfficials
 - hires
 - hourlyLaborers
 - labor
 - labor_supervisor
 - leave
 - location_info
 - payment_info
 - pays
 - personalInfo
 - schedule_info
 - schedules
 - supervisor_manager
 - takesLeave
 - upl_info
 - users_info
 - weekly_labors
- Trigger Functions
- Types
- Views
- public

laborlist_and_wages_db approves/laborlist_and_wages_db/postgres@P... No limit

Query History

```
1 SELECT * FROM laborlist_and_wages_db.approves
2 ORDER BY leave_id ASC
```

Data output Messages Notifications

	user_id	leave_id	supervisor_id	labor_id
1	31	1	128	119
2	58	2	140	62
3	16	3	125	75
4	44	4	138	97
5	21	5	139	76
6	31	6	141	65
7	41	7	155	86
8	39	8	180	73
9	20	9	130	66
10	49	10	175	85
11	27	11	143	98
12	33	12	172	119
13	54	13	163	107
14	14	14	169	61
15	4	15	132	79
16	37	16	152	62
17	10	17	148	115
18	29	18	140	98
19	30	19	155	88

Total rows: 40 of 40 Query complete 00:00:00.214 Successfully run. Total query runtime: 214 msec, 40 rows affected

20. Pays table

PgAdmin File Object Tools Help

Browser

- FTS Configurations
- FTS Dictionaries
- All FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (21)
 - approves
 - bank_info
 - business_account
 - business_info
 - governmentOfficials
 - hires
 - hourlyLaborers
 - labor
 - labor_supervisor
 - leave
 - location_info
 - payment_info
 - pays
 - personalInfo
 - schedule_info
 - schedules
 - supervisor_manager
 - takesLeave
 - upl_info
 - users_info
 - weekly_labors
- Trigger Functions
- Types
- Views
- public

laborlist_and_wages_db pays/laborlist_and_wages_db/postgres@P... No limit

Query History

```
1 SELECT * FROM laborlist_and_wages_db.pays
2 ORDER BY payment_number ASC
```

Data output Messages Notifications

	supervisor_id	labor_id	payment_number	user_id
1	165	75	1	27
2	132	65	2	57
3	169	110	3	11
4	129	69	4	29
5	168	61	5	58
6	180	77	6	2
7	150	117	7	22
8	151	67	8	1
9	161	86	9	5
10	176	88	10	2
11	143	115	11	56
12	154	71	12	51
13	122	62	13	27
14	165	65	14	32
15	123	112	15	6
16	134	114	16	22
17	175	93	17	2
18	141	67	18	48
19	134	83	19	6

Total rows: 60 of 60 Query complete 00:00:00.151 Ln 1, Col 1

21. Payment_info table

The screenshot shows the pgAdmin interface with the 'laborlist_and_wages_db' database selected. In the left sidebar, under 'Tables (21)', the 'payment_info' table is highlighted. The main pane displays a query window with the following SQL code:

```
1 SELECT * FROM laborlist_and_wages_db.payment_info
2 ORDER BY payment_number ASC
```

The results are shown in a table with the following data:

payment_number	payment_id	paid_for_byhours	amount_paid	date
1	1	85781	29	674
2	2	79743	69	708
3	3	29605	65	871
4	4	57911	85	855
5	5	59755	57	982
6	6	21827	99	827
7	7	93752	22	951
8	8	65050	22	530
9	9	57619	93	801
10	10	25630	48	629
11	11	51048	65	914
12	12	23624	46	564
13	13	24022	96	539
14	14	68044	14	929
15	15	84892	54	524
16	16	80234	29	872
17	17	47315	97	948
18	18	59147	86	657
19	19	81049	44	755

Total rows: 60 of 60 Query complete 00:00:00 156 Ln 1, Col 1

17. SQL Queries

~20 SQL queries and their outputs for the Case Study. (Consists of mostly complex queries and some simple queries because of the database's high complexity)

1st Query: Find out the labor id all the labors living in "city_name"

```
SELECT labor_id, city, district FROM laborlist_and_wages_db.labor NATURAL JOIN
laborlist_and_wages_db.location_info WHERE location_info.city='Ludvika';
```

The screenshot shows the PgAdmin 4 interface. On the left, the database browser tree is visible, showing a connection to 'laborlist_and_wages/postgres@PostgreSQL 14*' and a list of databases, tables, and other objects. The main area contains a query editor with the following SQL code:

```

1 SET SEARCH_PATH TO laborlist_and_wages_db;
2
3 -- 1st Query: Find out the labor_id all the labors living in "city_name"
4 SELECT Labor_Id,city,district FROM laborlist_and_wages_db.labor NATURAL JOIN laborlist_and_wages_db.location_info WHERE location_info.city='Ludvika';
5
6 -- 2nd Query: Find out the age of all of the labors
7 SELECT personal_info.name AS Name,date_part('year',age(labor.birth_date)) AS Age
8 FROM personal_info NATURAL JOIN labor;
9
10 -- 3rd Query: Find out the labors whose hourly_wage is less than 100 rupees.
11 SELECT personal_info.name, hourly_wage FROM hourly_labors NATURAL JOIN labor NATURAL JOIN personal_info WHERE (hourly_wage<100);
12
13 -- 4th Query: Find out all the labours hired by the business "business_name"
14 SELECT personal_info.name, labor_id FROM hires NATURAL JOIN personal_info NATURAL JOIN business_info WHERE (business_info.name = "business");
15
16 -- 5th Query: Count the number of Hourly Wage taking labors are there
17 SELECT COUNT(*) FROM hourly_labors;
18
19 -- 6th Query:

```

Below the query editor, a data output grid displays the results of the second query:

	labor_id	city	district
1	96	Ludvika	Putnol
2	107	Ludvika	Putnol

Total rows: 2 of 2 | Query complete 00:00:00.055 | Ln 4, Col 150

2nd Query: Find out the age of all of the labors

```
SELECT name,age(current_date,labor.birth_date) as Age FROM personal_info INNER
JOIN labor ON personal_info.labor_id = labor.labor_id;
```

The screenshot shows the PgAdmin 4 interface. On the left is a tree view of the database structure under 'Servers (1)'. The selected database is 'laborlist_and_wages' on 'PostgreSQL 14'. The main area has a 'Query History' tab open with several SQL queries numbered 1 through 19. Query 19 is the one being run, showing the results of a query to find laborers whose hourly wage is less than 100 rupees. The results table has columns 'name', 'character varying', 'age', and 'interval'. The data shows 10 rows of laborer names and their ages.

```

1 SET SEARCH_PATH TO laborlist_and_wages_db;
2
3 -- 1st Query: Find out the labor_id all the labors living in "city_name"
4 SELECT labor_id,city,district FROM laborlist_and_wages_db.labor NATURAL JOIN laborlist_and_wages_db.location_info WHERE location_info.cit
5
6 -- 2nd Query: Find out the age of all of the labors
7 SELECT name,age(current_date,labor.birth_date) as Age FROM personal_info INNER JOIN labor ON personal_info.labor_id = labor.labor_id;
8
9 -- 3rd Query: Find out the labors whose hourly wage is less than 100 rupees.
10 SELECT personal_info.name, hourly_wage FROM hourly_labors NATURAL JOIN labor NATURAL JOIN personal_info WHERE (hourly_wage<100) ;
11
12 -- 4th Query: Find out all the labours hired by the business "business_name"
13 SELECT personal_info.name, labor_id FROM hires NATURAL JOIN personal_info NATURAL JOIN business_info WHERE (business_info.name = "busines
14
15 -- 5th Query: Count the number of Hourly Wage taking labors are there
16 SELECT COUNT (*) FROM hourly_labors;
17
18 -- 6th Query:
19

```

	name	character varying	age	interval
1	Ella Yandrey		36	years ..
2	Dougie Georgeson		45	years ..
3	Janyae Sealby		34	years ..
4	Othilia Firman		35	years ..
5	Hirsch Charlotte		38	years ..
6	Elbertino Delake		34	years ..
7	Etel Hamill		45	years ..
8	Sargent Placksto...		31	years ..
9	Dian Dmitenko		58	years ..
10	Lexine Gantner		38	years ..

Total rows: 60 of 60 | Query complete 00:00:00.085 | Ln 8, Col 1

3rd Query: Find out the labors whose hourly wage is less than 100 rupees.

```
SELECT name,age(current_date,labor.birth_date) as Age FROM personal_info INNER
JOIN labor ON personal_info.labor_id = labor.labor_id;
```

The screenshot shows the PgAdmin 4 interface. On the left, the database browser displays a tree structure of servers, databases, tables, and other objects. In the center, a query editor window is open with a tab titled 'laborlist_and_wages/postgres@PostgreSQL 14*'. The query pane contains the following SQL code:

```

1 -- 1st Query: Find out the labor_id of all the labors living in "city_name"
2 SELECT labor_id, city, district FROM laborlist_and_wages_db.labor NATURAL JOIN laborlist_and_wages_db.location_info WHERE location_info.city = 'Mumbai';
3
4 -- 2nd Query: Find out the age of all of the labors
5 SELECT name, age(current_date, labor.birth_date) AS Age FROM personal_info INNER JOIN labor ON personal_info.labor_id = labor.labor_id;
6
7 -- 3rd Query: Find out the labors whose hourly wage is less than 100 rupees.
8 SELECT personal_info.name, hourly_wage FROM hourly_labors NATURAL JOIN labor INNER JOIN personal_info ON labor.labor_id = personal_info.labor_id WHERE hourly_wage < 100;
9
10 -- 4th Query: Find out all the labours hired by the business "business_name"
11 SELECT personal_info.name, labor_id FROM hires NATURAL JOIN personal_info NATURAL JOIN business_info WHERE (business_info.name = "business_name");
12
13 -- 5th Query: Count the number of Hourly Wage taking labors are there
14 SELECT COUNT(*) FROM hourly_labors;
15
16 -- 6th Query:
17
18 -- 7th Query: Show Labors in Company "business_name" have worked less than specified working hours
19
20 --with open_id show labors in company "business_name" have worked less than specified working hours

```

The results pane shows a table named 'hourly_labors' with columns 'name' (character varying) and 'hourly_wage' (integer). The data is as follows:

	name	hourly_wage
1	Douglas Georgeson	12
2	Othilia Firman	3
3	Elberina Delake	6
4	Etele Hamill	11
5	Bian Dmitenko	10
6	Lexine Gantner	5
7	Bernardina Sidsaff	10
8	Trumann May	7
9	Quentin Phaezey	3
10	Lorin Oney	10

Total rows: 27 of 27 | Query complete 00:00:00.045 | Ln 11, Col 1

4th Query: Find out all the labors hired by the business "business_name"

```

SELECT personal_info.name, hires.labor_id, business_info.name FROM
business_info NATURAL JOIN hires INNER JOIN personal_info ON hires.labor_id =
personal_info.labor_id WHERE (business_info.name = 'Zemlak Inc');

```

The screenshot shows the PgAdmin 4 interface. On the left is a tree view of the database schema under 'laborlist_and_wages/postgres@PostgreSQL 14'. The schema includes tables like approves, bank_info, business_account, business_info, governmentOfficials, hires, hourly_labors, labor, labor_supervisor, leave, location_info, personal_info, payment_info, pays, schedule_info, schedules, supervisor_manager, takes_leave, up_info, users_info, weekly_labors, and various system objects. The central area contains a query editor with the following SQL code:

```
1 SET SEARCH_PATH TO laborlist_and_wages_db;
2
3 -- 1st Query: Find out the labor id all the labors living in "city_name"
4 SELECT labor_id, city, district FROM laborlist_and_wages_db.labor NATURAL JOIN laborlist_and_wages_db.location_info WHERE location_info.city = 'Mumbai';
5
6 -- 2nd Query: Find out the age of all of the labors
7 SELECT name, age(current_date, labor.birth_date) AS Age FROM personal_info INNER JOIN labor ON personal_info.labor_id = labor.labor_id;
8
9 -- 3rd Query: Find out the labors whose hourly wage is less than 180 rupees.
10 SELECT personal_info.name, hourly_wage FROM hourly_labors NATURAL JOIN labor INNER JOIN personal_info ON labor.labor_id = personal_info.labor_id WHERE hourly_wage < 180;
11
12 -- 4th Query: Find out all the labors hired by the business "business_name"
13 SELECT personal_info.name, hires.labor_id, business_info.name FROM business_info NATURAL JOIN hires INNER JOIN personal_info ON hires.labor_id = personal_info.labor_id WHERE business_info.name = 'Zembla Inc';
14
15 -- 5th Query: Count the number of Hourly Wage taking labors are there
16 SELECT COUNT(*) FROM hourly_labors;
17
18 -- 6th Query:
19
```

The 'Data output' tab is selected, showing a table with one row of data:

	name	labor_id	name
	character varying	integer	character varying
1	Dacy Irreson	113	Zembla Inc

At the bottom, it says 'Total rows: 1 of 1' and 'Query complete 00:00:00.047'. The status bar indicates 'Ln 12, Col 76'.

5th Query: Count the number of Hourly Wage taking labors are there

```
SELECT COUNT(*) FROM hourly_labors;
```

```

PgAdmin  File  Object  Tools  Help
Browser  Dashboard  Properties  SQL  Statistics  Dependencies  Dependents  laborlist_and_wages/postgres@PostgreSQL 14*
Query  Query History
2
3 -- 1st Query: Find out the labor id all the labors living in "city_name"
4 SELECT Labor_id,city,district FROM laborlist_and_wages_db.labor NATURAL JOIN laborlist_and_wages_db.location_info WHERE location_info.city
5
6 -- 2nd Query: Find out the age of all of the labors
7 SELECT name,age(current_date),labor.birth_date) as Age FROM personal_info INNER JOIN labor ON personal_info.labor_id = labor.labor_id;
8
9 -- 3rd Query: Find out the labors whose hourly wage is less than 100 rupees.
10 SELECT personal_info.name, hourly_wage FROM hourly_labors NATURAL JOIN labor INNER JOIN personal_info ON labor.labor_id = personal_info.labor_id;
11
12 -- 4th Query: Find out all the labors hired by the business "business_name"
13 SELECT personal_info.name, hires.labor_id, business_info.name FROM business_info NATURAL JOIN hires INNER JOIN personal_info ON hires.labor_id = personal_info.labor_id;
14
15 -- 5th Query: Count the number of Hourly Wage taking labors are there
16 SELECT COUNT (*) FROM hourly_labors;
17
18 -- 6th Query:
19
20 -- 13th Query: Show Labors in company "business_name" have worked less than average working hours

```

Total rows: 1 of 1 Query complete 00:00:00.047 Successfully run. Total query runtime: 47 msec. 1 rows affected.

count	bigint
1	27

Ln 16, Col 1

6th Query: Find out the details of labor who has highest hourly wage among others

```

SELECT hourly_labors.labor_id, personal_info.name, hourly_labors.hourly_wage
FROM hourly_labors NATURAL JOIN personal_info WHERE hourly_wage in (SELECT
MAX(hourly_wage) FROM hourly_labors);

```

The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database schema with tables like labor, personal_info, hourly_labor, business_info, and location_info. The main area contains a query editor with several SQL statements and a results table.

```

7 SELECT name, age(current_date, labor.birth_date) as Age FROM personal_info INNER JOIN labor ON personal_info.labor_id = labor.labor_id;
8
9 -- 3rd Query: Find out the labors whose hourly wage is less than 100 rupees.
10 SELECT personal_info.name, hourly_wage FROM hourly_labors NATURAL JOIN labor INNER JOIN personal_info ON labor.labor_id = personal_info.labor_id;
11
12 -- 4th Query: Find out all the labors hired by the business "business_name"
13 SELECT personal_info.name, hires.labor_id, business_info.name FROM business_info NATURAL JOIN hires INNER JOIN personal_info ON hires.labor_id = personal_info.labor_id;
14
15 -- 5th Query: Count the number of Hourly Wage taking labors are there
16 SELECT COUNT(*) FROM hourly_labors;
17
18 -- 6th Query: Find out the details of labor who has highest hourly wage among others
19 SELECT hourly_labors.labor_id, personal_info.name, hourly_labors.hourly_wage FROM hourly_labors NATURAL JOIN personal_info WHERE hourly_wage > ALL (SELECT hourly_wage FROM hourly_labors WHERE labor_id <> hourly_labors.labor_id);
20
21 -- 12th Query: Show labors in company "business_name" have worked less than average working hours
22 SELECT labor_id, personal_info.name FROM labor NATURAL JOIN personal_info NATURAL JOIN business_info
23 WHERE (business_info.business_id = 186) AND labor.work_hours < (SELECT AVG(work_hours) FROM labor);
24
25

```

Data output:

labor_id	name	hourly_wage	
1	82	Dougie Georges	12
2	95	Efrem Oldnall	12
3	97	Franky Spooner	12
4	118	Robyn Bazache	12

Total rows: 4 of 4 Query complete 00:00:00.062 Ln 20; Col 1

7th Query: find out the name of all supervisors whose name starts from S.

```
SELECT supervisor_id, name FROM personal_info WHERE supervisor_id IS NOT NULL
AND personal_info.name LIKE 'S%';
```

```

PgAdmin  File  Object  Tools  Help
Browser  Dashboard  Properties  SQL  Statistics  Dependencies  Dependents  laborlist_and_wages/postgres@PostgreSQL 14*
Query  Query History
13 -- 4th Query: Find out all the labors hired by the business "business_name"
14 SELECT personal_info.name, hires.labor_id, business_info.name
15 FROM business_info NATURAL JOIN hires INNER JOIN personal_info ON hires.labor_id = personal_info.labor_id WHERE (business_info.name = 'Business A' OR business_info.name = 'Business B')
16
17 -- 5th Query: Count the number of Hourly Wage taking labors are there
18 SELECT COUNT (*) FROM hourly_labors;
19
20 -- 6th Query: Find out the details of labor who has highest hourly wage among others
21 SELECT hourly_labors.labor_id, personal_info.name, hourly_labors.hourly_wage
22 FROM hourly_labors NATURAL JOIN personal_info WHERE hourly_wage IN (SELECT MAX(hourly_wage) FROM hourly_labors);
23
24 -- 7th Query: find out name of all supervisors whose name starts from S.
25 SELECT supervisor_id, name FROM personal_info WHERE supervisor_id IS NOT NULL AND personal_info.name LIKE 'S%';
26
27 -- 12th Query: Show labors in company "business_name" have worked less than average working hours
28 SELECT labor_id, personal_info.name FROM labor NATURAL JOIN personal_info NATURAL JOIN business_info
29 WHERE (business_info.business_id = 186) AND labor.work_hours < (SELECT AVG(work_hours) FROM Labor);
30
31
Data output  Messages  Notifications
Columns (3)
location_id integer
city character varying
district character varying
Total rows: 5 of 5  Query complete 00:00:00.616  Ln 25, Col 1

```

The screenshot shows the PgAdmin 4 interface. The left sidebar displays a tree view of the database schema, including domains, FTS configurations, FTS dictionaries, FTS parsers, FTS templates, foreign tables, functions, materialized views, operators, procedures, sequences, and various tables like business_info, hourly_labors, and personal_info. The main window contains a query history pane with several numbered SQL queries. Below the queries is a data output pane showing the results of a query that selects supervisor_id and name from personal_info where supervisor_id is not null and name starts with 'S'. The results are as follows:

	supervisor_id	name
1	137	Steph Hiddle
2	141	Shanen Freer
3	167	Stuart Vanners
4	168	Samson Amphlett
5	173	Sarouann Firness

8th Query: Find out all laborers whose hourly wage is greater than 5 and whose age is higher than 30.

```

SELECT
hourly_labors.labor_id, personal_info.name, age(current_date, labor.birth_date), hourly_labors.hourly_wage
FROM hourly_labors NATURAL JOIN labor INNER JOIN personal_info ON
labor.labor_id = personal_info.labor_id
WHERE hourly_wage > 5 AND age(current_date, labor.birth_date) > INTERVAL '30Y';

```

The screenshot shows the PgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects under the schema 'laborlist_and_wages'. The 'Query History' tab in the center contains several SQL queries numbered 1 through 36. Query 8 is highlighted in red. Below the queries, the 'Data output' pane shows the results of Query 8, which lists 16 laborers with their supervisor ID, name, age, and hourly wage. The results are as follows:

	labor_id	name	age	hourly_wage
1	62	Dougie Georgeson	45 years ..	12
2	66	Elbertine Dulake	34 years ..	6
3	67	Estall Harill	45 years ..	11
4	69	Bian Dimitriko	58 years ..	10
5	71	Bernardina Sidsall	61 years ..	10
6	72	Trumann May	36 years ..	7
7	86	Lorin Quiry	56 years ..	10
8	95	Efren Oldhall	48 years ..	12
9	97	Franky Spooner	62 years ..	12
10	104	Standislaus Eagle	39 years ..	6

Total rows: 16 of 16 Query complete 00:00:00.567 Ln 28, Col 1

9th Query: Grouped by supervisor, printing sum of their labor's hourly wage so that the company can see their hourly expense on laborers per supervisor.

```

SELECT labor_supervisor.supervisor_id, personal_info.name,
sum(hourly_labors.hourly_wage)
FROM hourly_labors NATURAL JOIN labor INNER JOIN labor_supervisor ON
labor.supervisor_id = labor_supervisor.supervisor_id
INNER JOIN personal_info ON labor_supervisor.supervisor_id =
personal_info.supervisor_id
GROUP BY labor_supervisor.supervisor_id, personal_info.name;
    
```

```

PgAdmin  File  Object  Tools  Help
Browser  Servers (1)  Databases (3)  202001465.db  laborlist_and_wages  PostgreSQL 14*
laborlist_and_wages/postgres@PostgreSQL 14*
Query  Query History
22 FROM hourly_labors NATURAL JOIN personal_info WHERE hourly_wage IN (SELECT MAX(hourly_wage) FROM hourly_labors);
23
24 -- 7th Query: find out name of all supervisors whose name starts from S.
25 SELECT supervisor_id, name FROM personal_info WHERE supervisor_id IS NOT NULL AND personal_info.name LIKE 'S%';
26
27 -- 8th Query: Find out all labors whose hourly wage is greater than 5 and whose age is higher than 30.
28 SELECT hourly_labors.labor_id, personal_info.name, age(current_date, labor.birth_date), hourly_labors.hourly_wage
29 FROM hourly_labors NATURAL JOIN labor INNER JOIN personal_info ON labor.labor_id = personal_info.labor_id
30 WHERE hourly_wage > 5 AND age(current_date, labor.birth_date) > INTERVAL '30Y';
31
32 -- 9th Query: Grouped by supervisor, printing sum of their labor's hourly wage so that the company can see their hourly expense.
33 SELECT labor_supervisor.supervisor_id, personal_info.name, sum(hourly_labors.hourly_wage)
34 FROM hourly_labors NATURAL JOIN labor INNER JOIN labor_supervisor ON labor.supervisor_id = labor_supervisor.supervisor_id
35 INNER JOIN personal_info ON labor_supervisor.supervisor_id = personal_info.supervisor_id
36 GROUP BY labor_supervisor.supervisor_id, personal_info.name;
37
38 --10th Query: Find out all the supervisors who have paid laborers till now.
39
40 SELECT supervisor_id, personal_info.name FROM personal_info
41 WHERE supervisor_id NOT IN (SELECT supervisor_id FROM pays);

```

Total rows: 24 of 24 Query complete 00:00:00.068 Ln 33, Col 1

10th Query: Find out all the supervisors who have paid laborers till now.

```

SELECT supervisor_id, personal_info.name FROM personal_info
WHERE supervisor_id NOT IN (SELECT supervisor_id FROM pays);

```

The screenshot shows the PgAdmin 4 interface. The left sidebar displays a tree view of database objects: bank_info, business_account, business_info, governmentOfficials, hires, hourlyJobs, labor, laborSupervisor, leave, location_info (with columns location_id, city, district), payment_info, and pays (with columns supervisor_id, labor_id, payment_number, user_id). The right pane contains a query history window with the following SQL code:

```

30 WHERE hourly_wage > 5 AND age(current_date, labor.birth_date) > INTERVAL '30Y';
31
32 -- 9th Query: Grouped by supervisor, printing sum of their labor's hourly wage so that the company can see their hourly expense
33 SELECT labor_supervisor.supervisor_id, sum(hourly_labor.hourly_wage)
34 FROM labor_supervisor NATURAL JOIN labor INNER JOIN hourly_labor ON hourly_labor.labor_id = labor.labor_id
35 GROUP BY labor_supervisor.supervisor_id;
36
37 -- 10th Query: Find out all the supervisors who have paid labors till now.
38
39 SELECT supervisor_id, personal_info.name FROM personal_info
40 WHERE supervisor_id NOT IN (SELECT supervisor_id FROM pays);
41
42 -- 12th Query: Show labors in company "business_name" have worked less than average working hours
43 SELECT labor_id, personal_info.name FROM labor NATURAL JOIN personal_info NATURAL JOIN business_info
44 WHERE (business_info.business_id = 186) AND labor.work_hours > (SELECT AVG(work_hours) FROM labor);
45
46
47

```

The results of the last query are displayed in a table:

	supervisor_id	name
1	121	Mendo Morales
2	125	Avril Minoli
3	127	Gabrielle Stide
4	133	Vanya Ragless
5	137	Steph Hobble
6	138	Harriette Burniston
7	139	Nikki Gentzen
8	140	Trude Husthwaite
9	147	Esther Sehorsch
10	148	Aleta Bendix

Total rows: 10 of 19 Query complete 00:00:00.056 Ln 42, Col 50

11th Query: Find out all users who have hired a labor sometime in their life.

```

SELECT hires.user_id, personal_info.name, hires.labor_id FROM hires INNER JOIN
personal_info
ON hires.user_id = personal_info.user_id ;

```

```

12 -- 12th Query: Show labors in company who have worked less than average
13   work_hours < (SELECT AVG(work_hours) FROM labor);
14
15 WHERE hourly_wage > 5 AND age(current_date, labor.birth_date) > INTERVAL '30Y';
16
17 -- 9th Query: Grouped by supervisor, printing sum of their labor's hourly wage so that the company can see their hourly expense
18 SELECT labor.supervisor_id, personal_info.name, sum(hourly_labor.hourly_wage)
19 FROM hourly_labor NATURAL JOIN labor INNER JOIN labor.supervisor ON labor.supervisor_id = labor.supervisor.supervisor_id
20 INNER JOIN personal_info ON labor.supervisor.supervisor_id = personal_info.supervisor_id
21 GROUP BY labor.supervisor.supervisor_id, personal_info.name;
22
23 -- 10th Query: Find out all the supervisors who have paid labors till now.
24
25 SELECT supervisor_id, personal_info.name FROM personal_info
26 WHERE supervisor_id NOT IN (SELECT supervisor_id FROM pays);
27
28 -- 11th Query: Find out all users who have hired a labor sometime in their life.
29 SELECT hires.user_id, personal_info.name, hires.labor_id FROM hires INNER JOIN personal_info
30 ON hires.user_id = personal_info.user_id ;
31
32 -- 12th Query: Show labors in company who have worked less than average working hours

```

Total rows: 60 of 60 Query complete 00:00:00.084 Successfully run. Total query runtime: 84 msec. 60 rows affected.

12th Query: Show labors in company who have worked less than average working hours

```

SELECT labor.labor_id, personal_info.name FROM labor INNER JOIN personal_info
ON labor.labor_id = personal_info.labor_id
WHERE labor.work_hours < (SELECT AVG(work_hours) FROM labor);

```

```

30 WHERE hourly_wage > 5 AND age(current_date, labor.birth_date) > INTERVAL '30Y';
31
32 -- 9th Query: Grouped by supervisor, printing sum of their labor's hourly wage so that the company can see their hourly expense
33 SELECT labor.supervisor.supervisor_id, sum(hourly_laborers.hourly_wage)
34 FROM labor_supervisor NATURAL JOIN labor INNER JOIN hourly_laborers ON hourly_laborers.labor_id = labor.labor_id
35 GROUP BY labor.supervisor.supervisor_id;
36
37 -- 10th Query: Find out all the supervisors who have paid labors till now.
38
39 SELECT supervisor_id, personal_info.name FROM personal_info
40 WHERE supervisor_id NOT IN (SELECT supervisor_id FROM pays);
41
42 -- 12th Query: Show labors in company who have worked less than average working hours
43 SELECT labor.labor_id, personal_info.name FROM labor INNER JOIN personal_info ON labor.labor_id = personal_info.labor_id
44 WHERE labor.work_hours < (SELECT AVG(work_hours) FROM labor);
45
46
47
48

```

labor_id	name
1	Ophilia Firman
2	Esel Harill
3	Sargent Placksto...
4	Jacinta Giffin
5	Ora Weddell
6	Bordy Baty
7	Rheffie Jebb
8	Trumann May
9	Kakalina Adshad
10	Stewar Tournell

Total rows: 30 of 30 Query complete 00:00:00.050 Ln 43, Col 1

13th Query: Trigger to check and preventing inserting into the labor table if duplicate labor is found

SQL Query:

```

CREATE OR REPLACE FUNCTION check_labor_insert()
RETURNS trigger AS
$$
BEGIN
IF EXISTS(SELECT COUNT(*) FROM labor WHERE labor_id = NEW.labor_id) THEN
    RAISE NOTICE'Duplicate Found (%)', NEW.labor_id;
    RETURN OLD;
    --IF DUPLICATE WILL BE FOUND THEN IT WILL RAISE NOTICE AND THEN RETURN
OLD TABLE
END IF;
--ELSE BY DEFAULT IT WILL RETURN NEW TABLE
RETURN NEW;
END;

$$

```

```

LANGUAGE 'plpgsql';

DROP TRIGGER IF EXISTS check_labor_before_insert ON labor;
CREATE TRIGGER check_labor_before_insert
    BEFORE INSERT --BEFORE INSERTION THIS TRIGGER WILL RUN
    ON labor
    FOR EACH ROW --FOR EACH AFFECTED COLUMN THIS WILL RUN
    EXECUTE PROCEDURE check_labor_insert();

```

Checked with the insert query:

```

INSERT INTO
labor(labor_id,birth_date,work_hours,wage_remaining,rating,Type_of_work,super
visor_id,supervised_since,location_id,account_number)
VALUES
(61,'08-Nov-1961',49,'50',4,'Electrician',121,'29-Jun-2022',3114,827356014688)
;

```

The screenshot shows the PgAdmin 4 interface. The left sidebar displays the database schema with tables like labor, hourly_jobs, and location_info. The main window shows a SQL query editor with the following content:

```

60 END;
61
62 $$;
63 LANGUAGE 'plpgsql';
64
65 DROP TRIGGER IF EXISTS check_labor_before_insert ON labor;
66 CREATE TRIGGER check_labor_before_insert
67     BEFORE INSERT --BEFORE INSERTION THIS TRIGGER WILL RUN
68     ON labor
69     FOR EACH ROW --FOR EACH AFFECTED COLUMN THIS WILL RUN
70     EXECUTE PROCEDURE check_labor_insert();
71
72 -----
73 INSERT INTO labor(labor_id,birth_date,work_hours,wage_remaining,rating,Type_of_work,super
74 visor_id,supervised_since,location_id,account_number)
75 VALUES (61,'08-Nov-1961',49,'50',4,'Electrician',121,'29-Jun-2022',3114,827356014688);
76
77

```

The message area at the bottom shows:

- Data output: Messages: Notifications
- NOTICE: Duplicate Found (61)
- INSERT 0 0
- Query returned successfully in 541 msec.

Total rows: 30 of 30 | Query complete 00:00:00.541 | Ln 73, Col 1

As we can see it prevented inserting these and duplicate labor id 61 was found.

14th Query: Function to return 10 labors with maximum work hours per week

Function Query:

```
CREATE OR REPLACE FUNCTION max_work_hours()
RETURNS TABLE(
    laborID integer,
    workHours integer
)
LANGUAGE 'plpgsql'
AS $BODY$
declare f record;
begin
    CREATE TEMP TABLE test1(
        laborID integer,
        workHours integer
    ) ON COMMIT DROP;

    for f in select labor_id, work_hours
        from labor
        order by work_hours DESC LIMIT 10
    loop
        INSERT INTO test1(laborID,workHours)
            VALUES(f.labor_id,f.work_hours); --inserting those tuples
    into temp table
    end loop;
    return query table test1;
end;
$BODY$;
```

Checked with the query:

```
SELECT * FROM max_work_hours();
```

The screenshot shows the PgAdmin 4 interface. On the left, the 'Browser' pane displays the database structure under 'PostgreSQL 14'. The 'laborlist_and_wages' database is selected, showing tables like 'Casts', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', 'Schemas', 'Aggregates', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', and 'Foreign Tables'. Below these are 'Functions', 'Materialized Views', 'Operators', 'Procedures', 'Sequences', 'Tables (21)', 'Trigger Functions', 'Types', 'Views', and 'public' and 'postgres' schemas. The 'Login/Group Roles' and 'Tablespaces' sections are also visible.

The main window contains a 'Query' tab with the following SQL code:

```
88    CREATE TEMP TABLE test1
89        laborID integer,
90        workHours integer
91    ) ON COMMIT DROP;
92
93    for f in select labor_id, work_hours
94        from labor
95        order by work_hours DESC LIMIT 10
96    loop
97        INSERT INTO test1(laborID,workHours)
98            VALUES(f.labor_id,f.work_hours); --inserting those tuples into temp table
99    end loop;
100   return query_table test1;
101
102 $BODY$;
103
104
105 SELECT * FROM max_work_hours();
106
107
```

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

	laborID	workhours
1	95	99
2	102	97
3	82	96
4	61	95
5	115	93
6	79	92
7	73	91
8	88	87
9	105	86
10	108	86

Total rows: 10 of 10 | Query complete 00:00:01.064 | Ln 105, Col 1

15th Query: find out bank information of all the business.

```
SELECT * FROM bank_info WHERE type_of_user = 'business';
```

```

110  SELECT * FROM max_work_hours();
111
112
113 -- 15th Query: find out bank information of all the business.
114  SELECT * FROM bank_info WHERE type_of_user = 'business';
115
116 -- 16th Query:
117
118 -- 17th Query:
119
120 -- 18th Query:
121
122 -- 19th Query:
123
124 -- 20th Query:
125
126
127

```

	account_number	ifsc_code	name	email	user_id	type_of_user
1	246423547831	UGRW9W794353	Ruth Stolley	rstolley50@new...	181	business
2	395705930861	IOTH0798000	Evyn Skate	eskate51@nasa...	182	business
3	833157994886	WQUXK0637010	Tessa Fransson	tfransson52@re...	183	business
4	988925850852	HAILDA31708	Cesario Swinfin	cswinfin53@bgl...	184	business
5	999643202382	ZKOPD195257	Jimmy Poynton	jpoynon54@flav...	185	business
6	642469436045	ORGB0301005	Calypso Litchfield	clitchfield55@blo...	186	business
7	710981170695	WKJQ0258779	Homere Blackley	hblackley56@ed...	187	business
8	72417451859	MNRD0461635	Charly Lorek	clorek57@amszo...	188	business
9	366438665339	DR100101152	Bailee Janowicz	bjanowicz58@ftc...	189	business
10	546391035835	PNYS0353937	Franky Coule	fcoule59@state...	190	business

Total rows: 60 of 60 Query complete 00:00:36.870 Ln 113, Col 4

16th Query: find out all the government officials who have been monitoring the businesses.

```

SELECT DISTINCT business_info.official_id, personal_info.name FROM
business_info INNER JOIN personal_info ON business_info.official_id =
personal_info.official_id;

```

The screenshot shows the PgAdmin 4 interface. On the left, the 'Browser' panel displays a tree view of database objects. Under 'Databases', there are two entries: '202001465_db' and 'laborlist_and_wages'. The 'laborlist_and_wages' database is expanded, showing its contents: Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (2), and Tables. The 'Tables' node is further expanded to show 'laborlist_and_wages_db', which contains Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables, Trigger Functions, Types, and Views. The 'public' schema is also listed. The 'Tables' node is currently selected. On the right, the 'Query' tab of the main window contains a series of SQL queries numbered 121 through 138. Query 121 is a comment. Queries 123, 127, and 130 are SELECT statements. Other queries are comments or continuation of previous ones. Below the queries, the 'Data output' tab is active, showing a table with two columns: 'official_id' (integer) and 'name' (character varying). The data consists of 10 rows:

official_id	name
1	244 Claudell Skill
2	298 NIKI Mellabund
3	293 Leo Floyed
4	296 Lani Heintze
5	282 Vivene Casterton
6	252 Gimmi Nichol
7	257 Pancho Trim
8	281 Etta's Shelger
9	253 Hubo Nellsen

Total rows: 41 of 41 | Query complete 00:00:00.088 | Ln 131, Col 1

17th Query: Show all the cities and districts in which the users live

```
SELECT city,district FROM location_info;
```

```

122
123 --- 12th Query: Show all weekly wage taking laborers in ascending order of the
124 SELECT * FROM weekly_labors ORDER BY weekly_wage ASC;
125
126
127 --- 13th Query: find out bank information of all the business.
128 SELECT * FROM bank_info WHERE type_of_user = 'business';
129
130 --- 14th Query: find out all the government officials who have been monitoring the businesses.
131 SELECT DISTINCT business_info.official_id, personal_info.name FROM business_info INNER JOIN personal_info ON business_info.official_id = personal_info.id;
132
133 --- 15th Query: Show all the cities and districts in which the users live
134 SELECT city, district FROM location_info;
135
136 --- 16th Query:
137
138 --- 17th Query:
139
140 --- 18th Query:

```

	city	district
1	Yablonovskiy	Liman
2	Censolungon	Weigbuzhuang
3	Luzvika	Putaci
4	Wadil	Melbourne
5	Dujumma	Lluka e Eperme
6	Bytkiv	Santo Antônio do... ...
7	Iperó	Klippan
8	Yles	Lluto
9	Santiago do Cac...	Lendah
10

Total rows: 60 of 60 Query complete 00:00:00.050 Successfully run. Total query runtime: 50 msec. 60 rows affected.

18th Query: Show all weekly wage taking laborers in asceding order of the weekly wage.

```
SELECT * FROM weekly_labors ORDER BY weekly_wage ASC;
```

The screenshot shows the PgAdmin 4 interface. The left sidebar is a tree view of the database schema, including Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (21). The 'Tables' node is expanded, showing various tables like approves, bank_info, business_account, business_info, governmentOfficials, hires, hourlyLaborers, labor, laborSupervisor, leave, location_info, payment_info, pays, personal_info, schedule_info, schedules, supervisorManager, takesLeave, upl_info, users_info, and weeklyLaborers. The 'weeklyLaborers' table is currently selected, showing its columns: labor_id (PK), weekly_wage, and hours_worked. The main area contains a query history window with several numbered queries. Query 19 is highlighted, which is the 19th query from the start. The data output tab shows the results of the 19th query:

labor_id	weekly_wage	hours_worked
1	108	25
2	110	25
3	72	29
4	68	31
5	84	58
6	85	65
7	79	70
8	87	71
9	93	74

Total rows: 15 of 15 | Query complete 00:00:00.045 | Ln 136, Col 4

19th Query: count the number of laborers under supervision by supervisors.

```
SELECT labor.supervisor_id, personal_info.name, COUNT(*) FROM labor INNER JOIN personal_info ON labor.supervisor_id = personal_info.supervisor_id GROUP BY labor.supervisor_id, personal_info.name;
```

```

126
127 -- 15th Query: find out bank information of all the business.
128 SELECT * FROM bank_info WHERE type_of_user = 'business';
129
130 -- 16th Query: find out all the government officials who have been monitoring the businesses.
131 SELECT DISTINCT business_info.official_id, personal_info.name FROM business_info INNER JOIN personal_info ON business_info.official_id = personal_info.id;
132
133 -- 17th Query: Show all the cities and districts in which the users live
134 SELECT city,district FROM location_info;
135
136 -- 18th Query: Show all weekly wage taking laborers in ascending order of the weekly wage
137 SELECT * FROM weekly_labors ORDER BY weekly_wage ASC;
138
139 -- 19th Query: count the number of labors under supervision by supervisors
140 SELECT labor.supervisor_id, personal_info.name, COUNT(*) FROM labor INNER JOIN personal_info ON labor.supervisor_id = personal_info.id GROUP BY labor.supervisor_id, personal_info.name;
141
142
143 -- 20th Query:

```

	supervisor_id	name	count
1	145	Bryan Carete	1
2	137	Steph Hobble	1
3	171	Randie Robbie	2
4	146	Laird Pywell	1
5	133	Vanya Ragless	1
6	149	Broni Simancu	1
7	154	Carmon Payn	1
8	127	Gabrielle Slide	1
9	159	Tiffy Rivers	1

Total rows: 43 of 43 Query complete 00:00:00.060 Ln 139, Col 4

20th Query: -- 20th Query: find out labors labor id whose leave is not yet approved.

```
SELECT leave_id, supervisor_id, labor_id FROM takes_leave WHERE leave_id NOT IN (SELECT leave_id FROM approves);
```

PgAdmin File Object Tools Help

Browser

- > FTS Configurations
- > FTS Dictionaries
- > All FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > Sequences
- > Tables (21)
 - > approves
 - > bank_info
 - > business_account
 - > business_info
 - > governmentOfficials
 - > hires
 - > hourlyLabors
 - > labor
 - > labor_supervisor
 - > leave
 - > location_info
 - > payment_info
 - > pays
 - > personal_info
 - > schedule_info
 - > schedules
 - > supervisor_manager
 - > takes_leave
 - > up_info
 - > users_info
 - > weekly_labors
- > Trigger Functions
- > Types
- > Views
- > public

laborlist_and_wages/postgres@PostgreSQL 14*

Query History

```
127 -- 15th Query: find out bank information of all the business.
128 SELECT * FROM bank_info WHERE type_of_user = 'business';
129
130 -- 16th Query: find out all the government officials who have been monitoring the businesses.
131 SELECT DISTINCT business_info.supervisor_id, personal_info.name FROM business_info INNER JOIN personal_info ON business_info.supervisor_id = personal_info.id;
132
133 -- 17th Query: Show all the cities and districts in which the users live.
134 SELECT city,district FROM location_info;
135
136 -- 18th Query: Show all weekly wage taking laborers in ascending order of the weekly wage
137 SELECT * FROM weekly_labors ORDER BY weekly_wage ASC;
138
139 -- 19th Query: count the number of labors under supervision by supervisors
140 SELECT labor.supervisor_id, personal_info.name, COUNT(*) FROM labor INNER JOIN personal_info ON labor.supervisor_id = personal_info.id GROUP BY labor.supervisor_id, personal_info.name;
141
142
143 -- 20th Query: find out labors labor_id whose leave is not yet approved.
144 SELECT leave_id, supervisor_id, labor_id FROM takes_leave WHERE leave_id NOT IN (SELECT leave_id FROM approves);
```

Data output Messages Notifications

leave_id	supervisor_id	labor_id
1	41	134
2	42	127
3	43	128
4	44	147
5	45	157
6	46	149
7	47	128
8	48	161
9	49	145
...

Total rows: 20 of 20 | Query complete 00:00:00.028 | Ln 143, Col 1



**IT214 - DBMS
(Front-end Development)**

Case Study Database: Laborlist and Wages Database

**Prepared by
Team No. 17 | Group 6 | Section 10**

**Group Members
Dhruv Patel - 202001271
Jaykumar Navadiya - 202001465**

**Under the guidance of
Prof. Chhaya
Teaching Assistant - Vinay Maharaaj**

**Date
November 18, 2022**

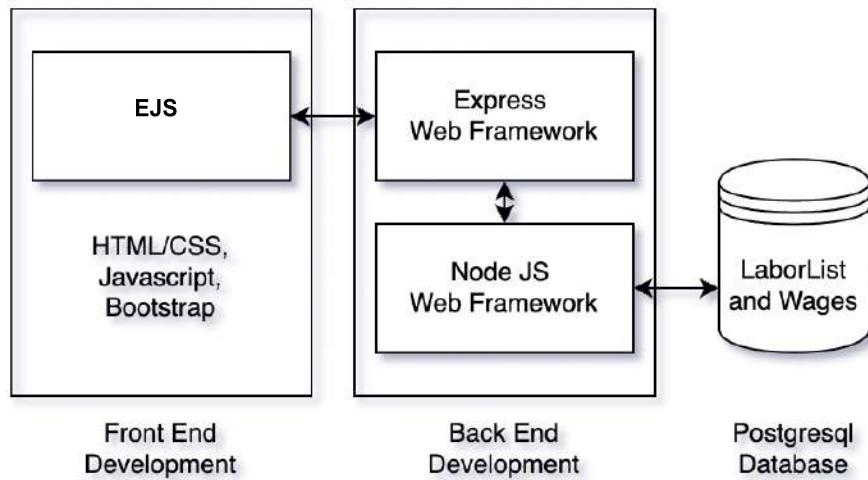
18. Website with CRUD functionalities and Other Features:

We have used Embedded Javascript (an alternative of HTML5 as we can write both HTML and Javascript code in one file itself in Embedded Javascript file, extension: .ejs)

For designing purposes we have used CSS and Javascript.

For the backend of the website, we have used Node.js and Express Module which are written in Javascript as well.

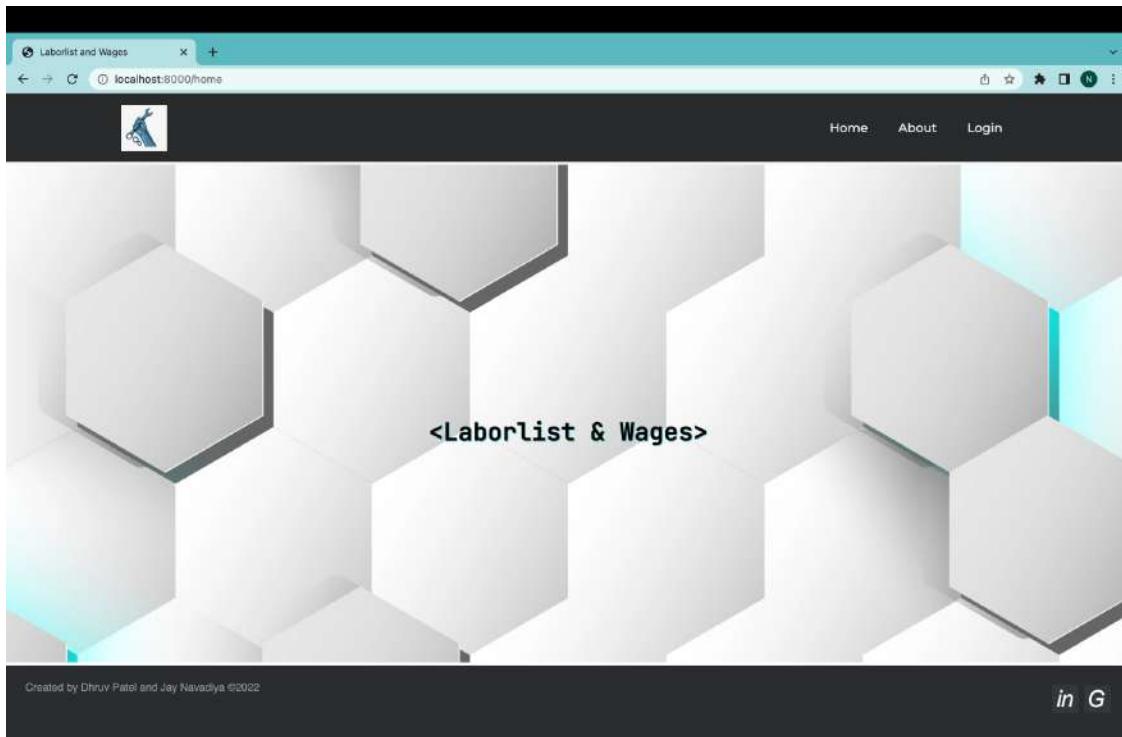
Below is the diagram explaining the PEEN Stack we have used for this website:



All programming files (Website Code Files) are there in the website folder included in the zip file.

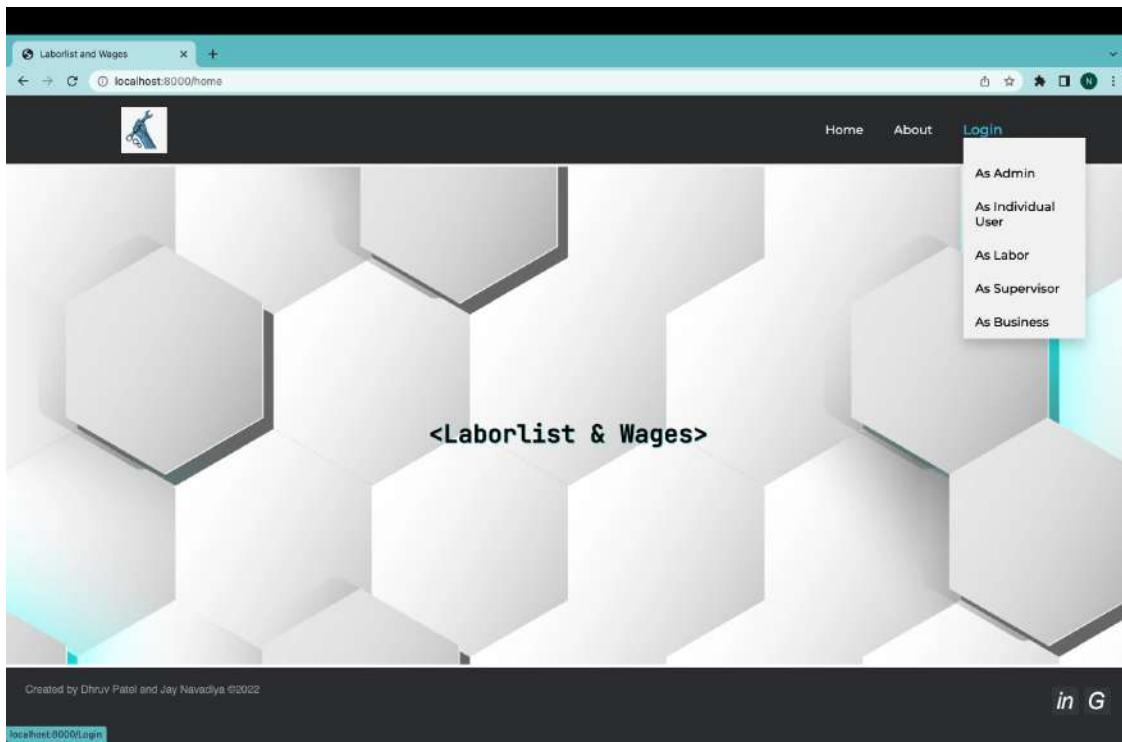
18.1 Functionalities:

1. Homepage



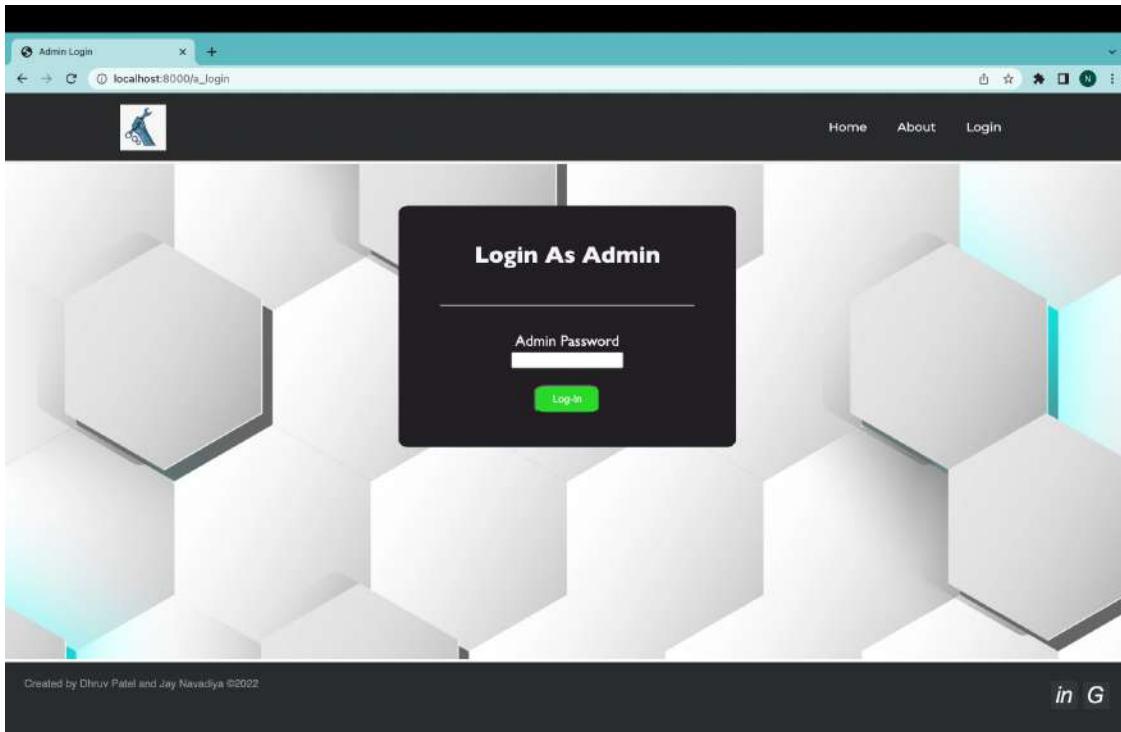
There is an animation in the Laborlist & Wages text which can be explored in detail with the code given.

2. Login Function



This opens a hidden dropdown menu which contains various types of users that our database supports.

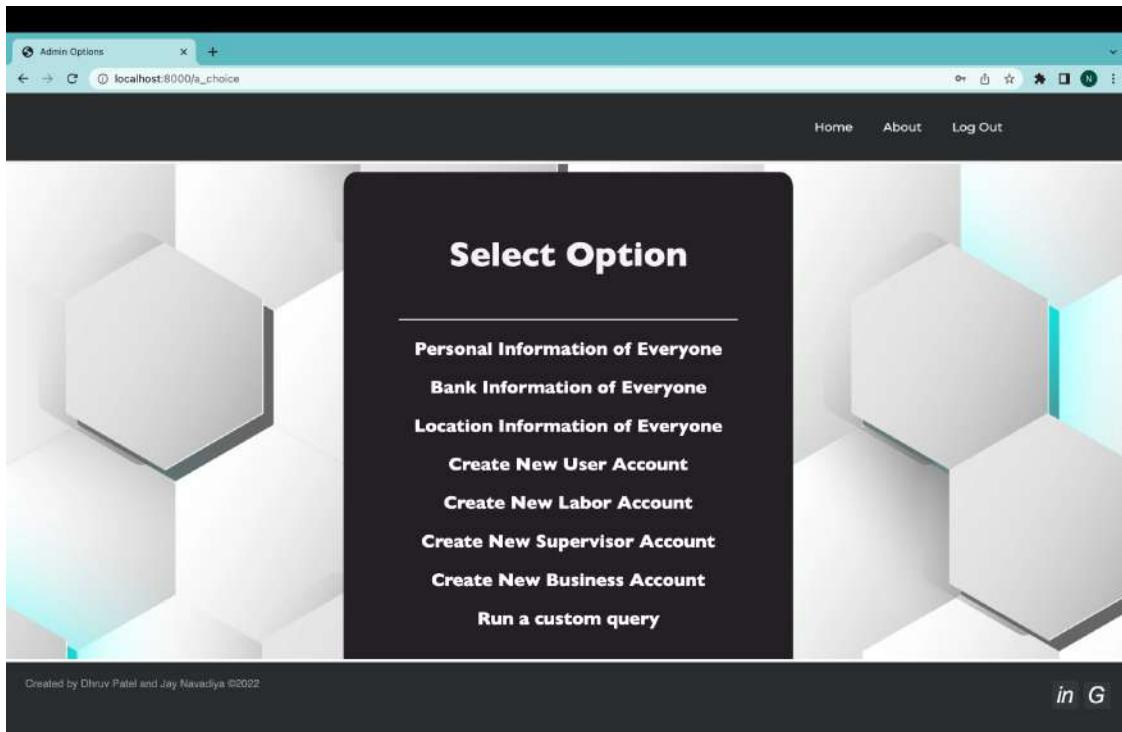
3. Admin Login



This only contains a password field. The password is admin.

After logging into the Admin, there are various options to choose from for the admin.

4. Admin Choices



5. Option 1: Personal Information of Everyone

It shows the table of personal_info of our database with pages and prev and next options.

Employee Info

localhost:8000/personal_info?page=1&limit=20

Home About Log Out

User ID	Labor ID	Supervisor ID	Government Official ID	Mobile Number	Type of User	Name	Email
2			8055157589	user	Gabbi Yardley	gycardley1@slashdot.org	
3			835386732	user	Maren Branford	mbranford2@google.com.br	
4			346200409	user	Marven Sissland	msissland3@amazon.com	
5			5229203792	user	Camia Butler	chutter1@usa.gov	
6			4728880509	user	Madeline Pompilotti	mpompilotti5@bloglovin.com	
7			7752229709	user	Bradan Wickham	bwickham6@webnode.com	
8			9962022911	user	Candide Prosch	cprosch7@geocities.jp	
9			6588214596	user	Adrien Danes	adanes8@cbnews.com	
10			292166951	user	Ingeborg Murdoch	imurdoch9@nifty.com	
11			4644975144	user	Domenico Sherbrook	dsherbrook@noaa.gov	
12			1587651787	user	Lilias Conrait	lconfairb@google.de	
13			1569402053	user	Gasper Davis	gdavisc@altervista.org	
14			7934460030	user	Matti Luton	mlutond@cmu.edu	
15			8265048110	user	Hughie Curnok	hcurnokc@booking.com	
16			6331934716	user	Aldin Greenhalgh	agreenhalgh@statcounter.com	
17			8817052207	user	Ben Schimpke	bschimpke@statcounter.com	
18			894566101	user	Dru Giffaut	dgiffauth@wisc.edu	
19			960289850	user	Granny Peterkin	gpeterkin@mpg.org	
20			8872615835	user	Gothart O'Keavan	gokeavanj@cbc.ca	
21			19196628	user	Barbi Knapper	bknapperk@freewebs.com	

NEXT >

Created by Dhiruv Patel and Jay Navadiya ©2022

in G

Employee Info

localhost:8000/personal_info?page=2&limit=20

Home About Log Out

User ID	Labor ID	Supervisor ID	Government Official ID	Mobile Number	Type of User	Name	Email
22			7844217276	user	Christine Beckenham	ebeckenham1@creativecommons.org	
23			3735511865	user	Emmalyn Le Grand	elem@hibu.com	
24			9840364548	user	Berti Muzzini	bmuzzini@medialife.com	
25			236916427	user	Rheta Jenken	rjenkeno@pagesperso-orange.fr	
26			5369916083	user	Montgomery Wride	mwridep@fda.gov	
27			9120402488	user	Desmond Godbehere	dgodbehereq@jiathis.com	
28			6366223283	user	Evy Bennis	ebennisr@cisco.com	
29			6810924926	user	Katya Bruhnicke	kbruhnicke@iuc.edu	
30			580140490	user	Nanni Collett	ncollicutti@mpquest.com	
31			6048132896	user	Ester O'Heddersoll	eheddersoll@pagesperso-orange.fr	
32			8018110836	user	Ursz Robardley	urobardley@smh.com.au	
33			3306391503	user	Rickie Kelland	rkelandw@washington.edu	
34			5577504233	user	Julissa Muscat	jmuscat@free.fr	
35			2929809569	user	Billye Vairow	bvairow@360.cn	
36			51888964	user	Lane Puckrin	lpuckrinz@wordpress.com	
37			4836973398	user	Torrin Cleugh	tcleugh10@reverberation.com	
38			3030979912	user	Sydel Perford	sperford11@time.com	
39			1971300481	user	Rolie Goodwill	rgoodwill12@pinterest.com	
40			5325216992	user	Ludwig Royston	lroyston13@weather.com	
41			3440013433	user	Vinnie Southey	vssouthey14@bing.com	

<< PREV

NEXT >

Created by Dhiruv Patel and Jay Navadiya ©2022

in G

6. Option 2: Shows the bank information for every account in various pages as before.

Account Number	ISFC Code	Name	Email	User ID	Type of User
298537795750		Dannie Symondson	dsymondson0@slate.com	1	user
544028694207		Gabbi Yearday	gyearday1@slashdot.org	2	user
148456509144		Maren Branford	mbranford2@google.com.br	3	user
867476259		Marven Sissland	msissland3@amazon.com	4	user
23313250426		Camila Buttler	cbuttler4@usa.gov	5	user
883007249600		Madelaine Pompiliet	mpompiliet5@bloglovin.com	6	user
498375389985		Bradan Wickham	bwickham6@webnode.com	7	user
488790545998		Candide Prosch	cprosch7@geocities.jp	8	user
150860783048		Adrien Danes	adanes8@cbsnews.com	9	user
400874478344		Ingeborg Murdoch	imurdoch9@nifty.com	10	user
5053893934545		Domenico Sherbrook	dsherbrook@noaa.gov	11	user
267908092820		Lilias Confair	lconfair@google.de	12	user
457324950031		Gaspar Davis	gdavis@altervista.org	13	user
57934205593		Matti Luton	mluton1@cmu.edu	14	user
79193647990		Hughie Cumrook	hcumrook@booking.com	15	user
922549739503		Aldin Greenhalgh	agreenhalgh@staticcounter.com	16	user
7279396949		Ben Schimpke	bschimpke9@statecounter.com	17	user
59403621618		Dru Giffaut	dgiffaut@wisc.edu	18	user
286872016277		Granny Peterkin	g peterkin@gmpg.org	19	user
602207375039		Gothart O'Keefan	gokeefan@cbc.ca	20	user

7. Option 3: Shows the location table.

The pagination logic is given in the code and 20 entries will be shown at once. Also, by using the blur filter we made the table information more readable.

Hovering is also implemented in the table as well. Have a look at the screenshot given below. The row turns blue while hovering over it.

Location Info

localhost:8000/location_info?page=1&limit=20

Home About Log Out

Location ID	City	District
3101	Yablonovskiy	Linoan
3102	Cansolungon	Wangbuzhuang
3103	Ludvika	Putinci
3104	Wudil	Melbourne
3105	Dujumna	Ljuka e Eperme
3106	Bytkiv	Santo Antônio do Monte
3107	Iperó	Klippan
3108	Yisa	Liaotao
3109	Santiago do Cacém	Londiani
3110	Alannay	Agia Varvára
3111	Socorro	Dortmund
3112	Ulaan-Ereg	Santo Tomás
3113	Sarakhs	Zürich
3114	Tolosa	Yataity del Norte
3115	Sovetskiy	Pérmek
3116	Ibra'	Nizhniy Tagil
3117	Rude	Zumiao
3118	Pa Sang	Teberda
3120	Yangjingzhan	Esperanza
3121	Shijing	Makurazaki

NEXT >>

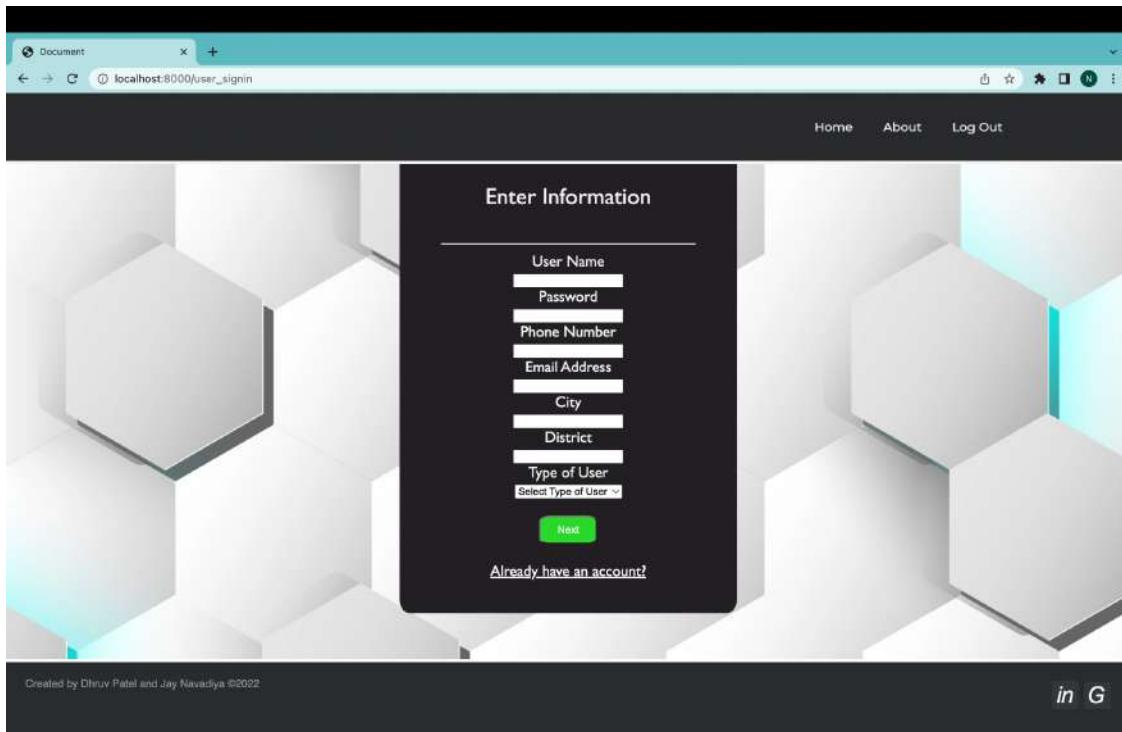
Created by Dhruv Patel and Jay Navadiya ©2022

in G

8. Option 4: Create new user account

This option lands you to the /user_signin page. Similarly for other options given after, such as creating a new supervisor, business and labor account does the same.

After clicking, Create a new user account it lands you to the /user_signin page as shown below.

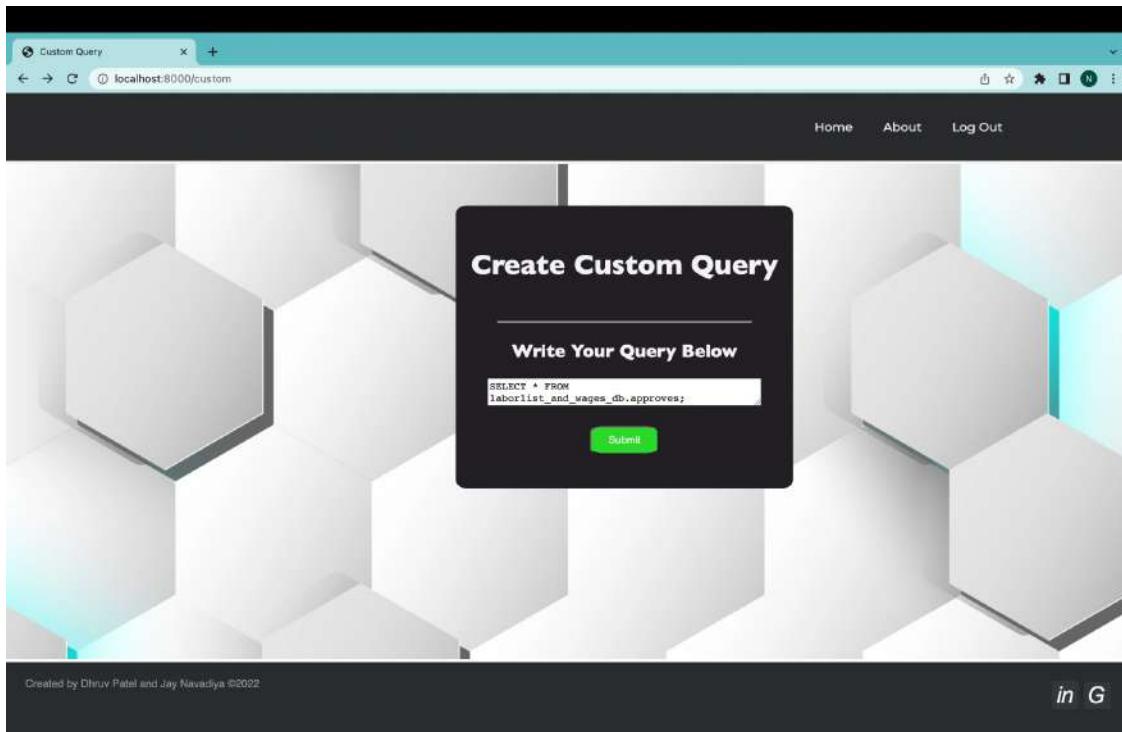


Here, if the user already has an account and the admin wants to access it, the Already have an account? option will land you on the login page and they can login into the account.

9. Option 5: Run a Custom Query

It allows the admin to run a custom query directly to the database. It is implemented with the help of Javascript and JSON.

Example: Query for seeing the leaves that are approved by the higher authorities.



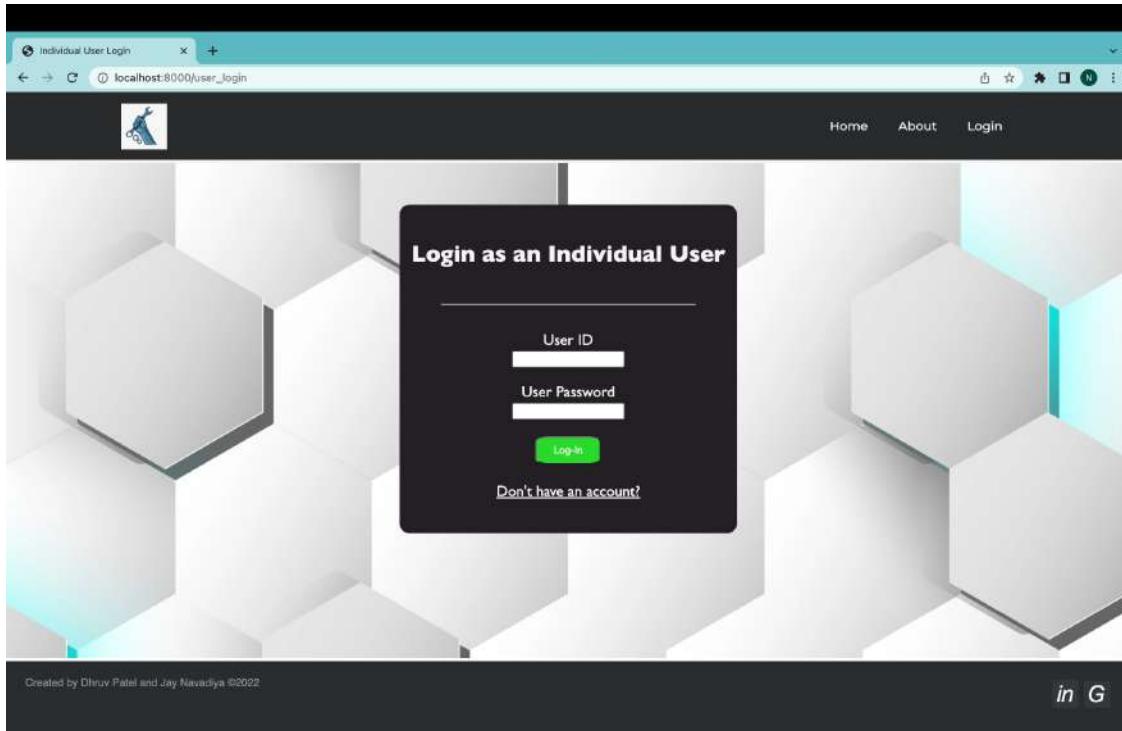
Result:

The scrolling feature is also implemented here as you can see on the right of the screenshot given below.

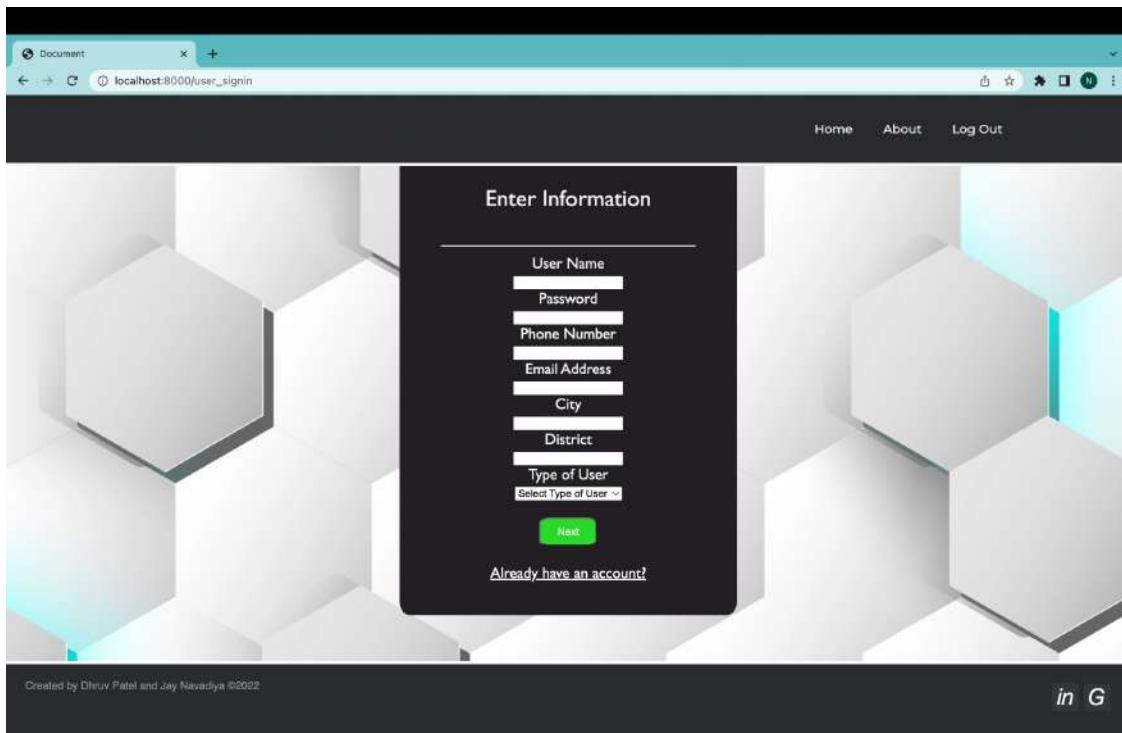
A screenshot of a web browser window titled "Student Info". The URL is "localhost:8000/custom". The page has a dark header with "Home", "About", and "Log Out" links. The main content area displays a large table with four columns: "user_id", "leave_id", "supervisor_id", and "labor_id". The table contains numerous rows of data. On the right side of the table, there are vertical scroll bars, indicating that the table is scrollable. The background of the page is a light gray with a hexagonal pattern.

Now, let's have a look at the login as an individual user feature.

10. Login Window

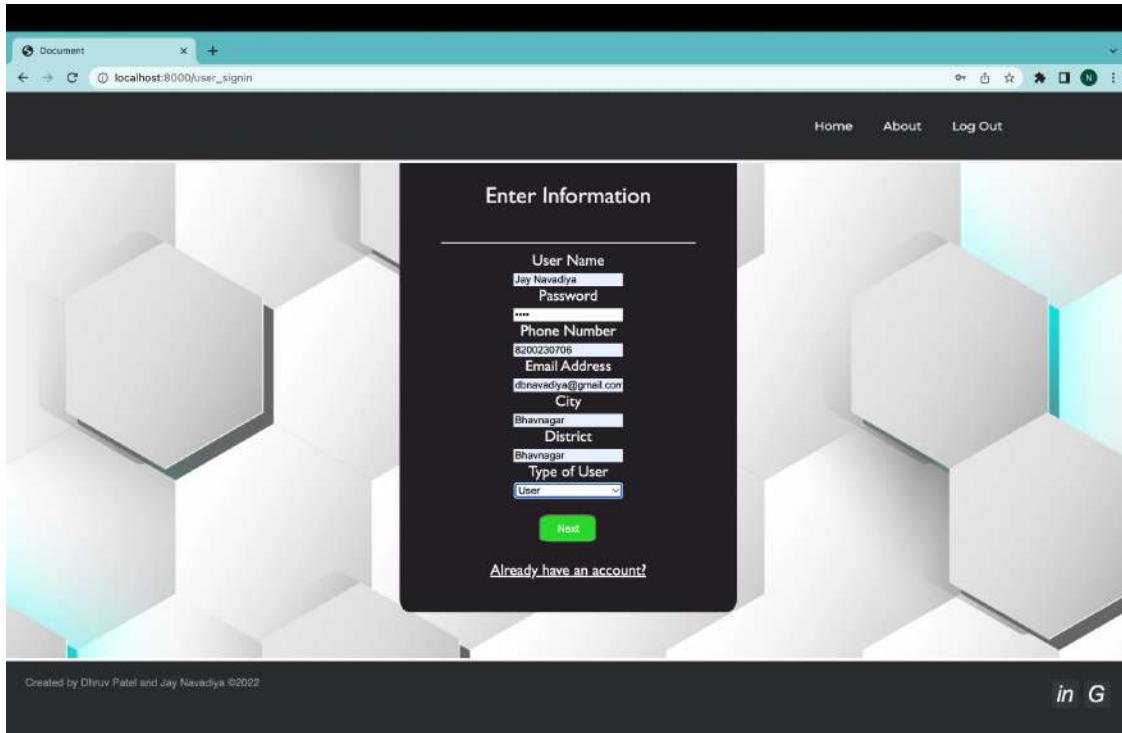


In case of no account, it can be created using [Don't have an account?](#)



A new user account can be created from here.

Let's create an account by filling up the details shown in the screenshot given below.



If the user account has been created successfully, the website will take you to the homepage.

Let's check if the user has been created or not.

Pgadmin screenshot of users_info table:

The screenshot shows the pgAdmin interface with a query editor and a results grid. The query is:

```

1: SELECT * FROM laborlist_and_wages_db.users_info
2: ORDER BY user_id ASC

```

The results grid shows the following data:

	user_id	account_number	location_id	password
58	58	227870859249	3116	major
59	59	891644541071	3151	major
60	60	284245012079	3154	major
61	301	8543914	2161	III
62	542	3983320	3162	III
63	783	6076996	3163	1
64	1024	6660981	3164	12
65	1265	9311504	3165	12
66	1506	8560300	3187	1234

Total rows: 66 of 66 Query complete 00:00:00.165

As we can see that user id 1506 has been created in the users_info table.
Let's see the personal_info of user_id 1506.

The screenshot shows the pgAdmin interface with a query editor and a results grid. The query is:

```

1: SELECT * FROM laborlist_and_wages_db.personal_info WHERE user_id=1506;

```

The results grid shows the following data:

	user_id	labor_id	supervisor_id	official_id	mobile_number	type_of_user	name	email
1	1506	[null]	[null]	[null]	8230236706	user	Jay Navadya	dnavadya@gmail.com

Total rows: 1 of 1 Query complete 00:00:00.066

Successfully run. Total query runtime: 66 msec. 1 rows affected.

As we can see, the name, type-of-user and every other field have been successfully created.

The location_info table:

As we can see, The city, Bhavnagar and the district, Bhavnagar has been successfully created with location_id 3187. Ignore other entries as they were used for testing purposes.

The screenshot shows the pgAdmin interface with the 'laborlist_and_wages_db' database selected. In the left sidebar, under 'Tables (21)', the 'location_info' table is highlighted. The main pane displays a SQL query and its results. The SQL query is:

```
1 : SELECT * FROM laborlist_and_wages_db.location_info
2 : ORDER BY location_id ASC
```

The results table has three columns: 'locationId' (PK integer), 'city' (character varying), and 'district' (character varying). The data is as follows:

locationId	city	district
78	Bhavnagar	Bhavnagar
79	Bhavnagar	Bhavnagar
80	Bhavnagar	Bhavnagar
91	Bhavnagar	Bhavnagar
82	Bhavnagar	Bhavnagar
83	Bhavnagar	Bhavnagar
84	Bhavnagar	Bhavnagar
85	Bhavnagar	Bhavnagar14
86	Bhavnagar	Bhavnagar5
87	Bhavnagar	Bhavnagar
Total rows: 87 of 87 Query complete 00:00:00.261		

Let's look at the bank_info entry for user_id = 1506.

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' pane displays a tree view of database objects under 'laborlist_and_wages'. A node for 'bank_info' is expanded, showing its columns: account_number, ifsc_code, name, email, user_id, and type_of_user. The 'Tables (21)' node is also visible. On the right, the 'Query' tab contains the following SQL query:

```
1 | SELECT * FROM laborlist_and_wages.bank_info WHERE user_id=1506;
```

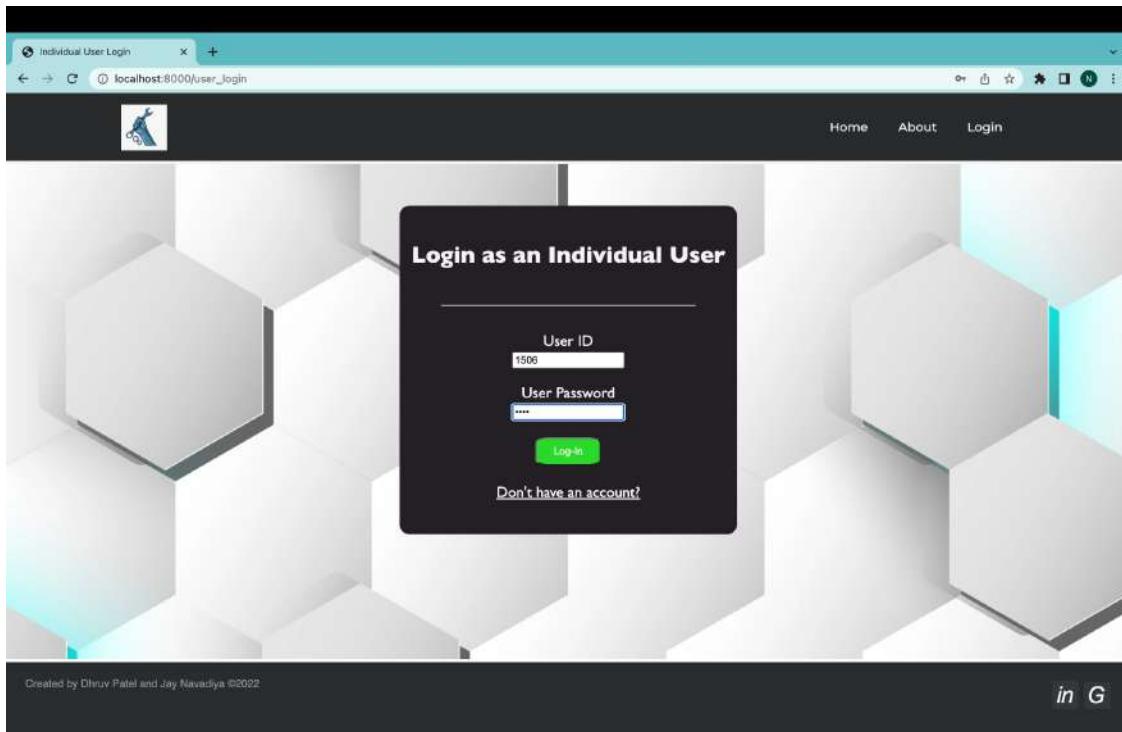
The results are displayed in a table titled 'Data output' with the following data:

	account_number	ifsc_code	name	email	user_id	type_of_user
1	8860300	T15723140	Jay Navadiya	dnnavadiya@gmail.co...	1506	user

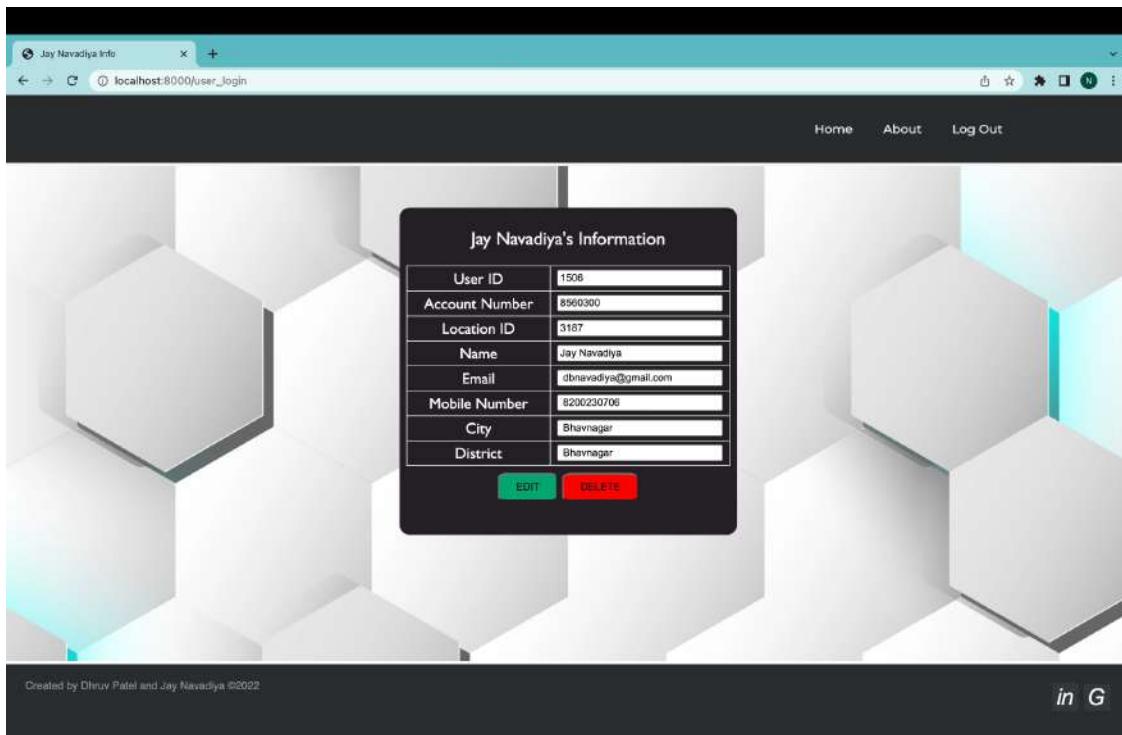
Below the table, the status bar shows: 'Total rows: 1 of 1 Query complete 00:00:00.041'. At the bottom right, a green message box indicates: 'Successfully run Total query runtime: 41 msec. 1 rows affected Ln 1, Col 67'.

As we can see that the account_number and the ifsc code have been inserted successfully randomly and they are unique too.

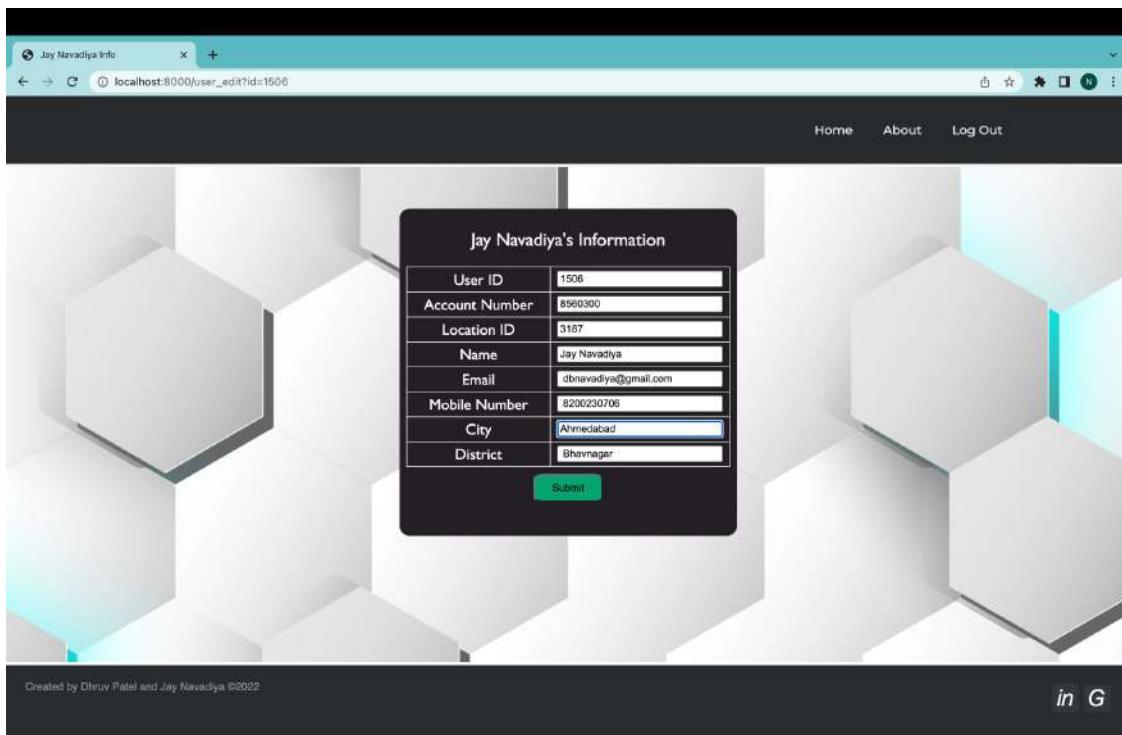
Now, let's login into this account.



It shows their information and gives an option to delete their account or edit the editable allowed information.



Let's edit the location and change the city to Ahmedabad.



As we can see, the edit button lands the user on user_edit page.

Let's check if the changes have been made in the database or not.

The screenshot shows the pgAdmin interface. On the left, the 'Browser' panel displays a tree view of database objects under 'Tables (21)'. One of the tables, 'location_info', is selected. The 'Query' tab in the center contains the following SQL code:

```

1: SELECT * FROM laborlist_and_wages_db.location_info
2: ORDER BY location_id ASC

```

The 'Data output' tab on the right shows the results of the query as a table:

location_id	city	district
78	Bhavnagar	Bhavnagar
79	Bhavnagar	Bhavnagar
80	Bhavnagar	Bhavnagar
91	Bhavnagar	Bhavnagar
82	Bhavnagar	Bhavnagar
83	Bhavnagar	Bhavnagar
84	Bhavnagar	Bhavnagar
85	Bhavnagar	Bhavnagar14
86	Bhavnagar	Bhavnagar5
87	Ahmedabad	Bhavnagar

At the bottom of the pgAdmin window, a status bar indicates: 'Successfully run. Total query runtime: 213 msec. 87 rows affected.'

As we can see, the entry is updated as the location_id remains the same and the city has been changed to Ahmedabad.

Now, let's delete this account by again logging into the account and clicking the red delete button.

If any action has been taken and it's successful, you will be landed on the homepage.

After deletion, let's check if the user exists in the users_info table or not.

```

SELECT * FROM laborlist_and_wages_db.users_info WHERE user_id=1506;

```

user_id	account_number	location_id	password
1506	1234567890	1	password123

Successfully run. Total query runtime: 41 msec. 0 rows affected.
Ln 1, Col 43

As we can see, there are no users with user_id=1506. By the foreign key constraints and the way we have run the queries it will be deleted from all the tables. I.e. the location_info, bank_info and personal_info tables.

The same can be seen within the code file of server.js.

The same functionalities have been implemented for all types of users.

Labor registration form:

Document x +

localhost:8000/labor_signin

Home About Log Out

Enter Information

Labor Name
Password
Phone Number
Email Address
City
District
Birth Date
dd/mm/yyyy
Type of Work
Type of User
Select Type of User

[Already have an account?](#)

Created by Dhiruv Patel and Jay Navadiya ©2022

in G

Supervisor Registration Form:

Document x +

localhost:8000/supervisor_signin

Home About Log Out

Enter Information

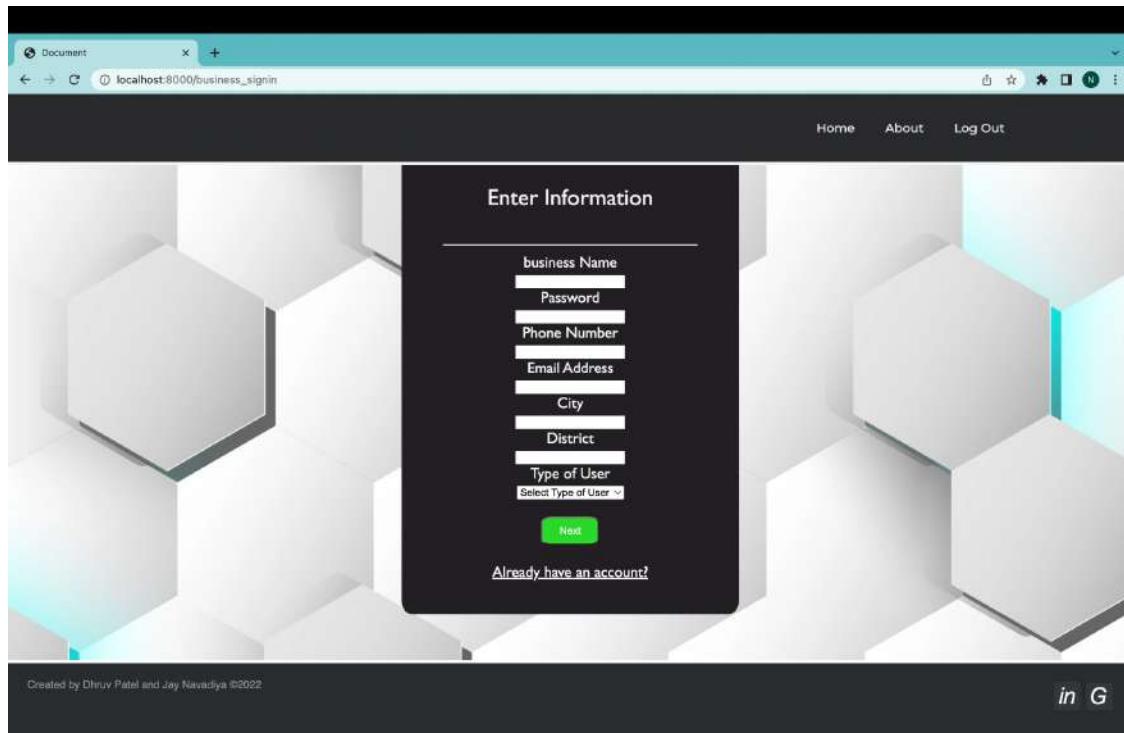
Supervisor Name
Password
Phone Number
Email Address
City
District
Type of User
Select Type of User

[Already have an account?](#)

Created by Dhiruv Patel and Jay Navadiya ©2022

in G

Business Registration Form:



Also, if someone enters the wrong password or any information it will be displayed as a text and he/she will not be able to log in.

These features are available for all types of users as previously explained. The website can be accessed when the database has been properly set in the local machine and node has been installed. Node server.js will allow the link localhost:8000 to be run in the browser.

The database can be made easily by using the DDL Scripts, database.sql for the whole database or schema.sql for making the schema in the database only.

Thank You!

Teaching Assistant Vinay Sir and Prof. Rachit Chhaya,
For the guidance and support throughout the project.