B.TECH. (CSE)

V SEMESTER

UE19CS301 – DATABASE MANAGEMENT SYSTEM

ASSIGNMENT-4

SUBMITTED BY

TEAM ID-3

NAME	SRN
T. LOHITH SRINIVAS	PES2UG19CS203
LALITHA SRAVANTI DASU	PES2UG19CS201
MEENAKSHI SURESH	PES2UG19CS228

AUGUST – DECEMBER 2021

DEPARTMENT OF COMPUTER

SCIENCE & ENGINEERING

ELECTRONIC CITY CAMPUS,

BENGALURU – 560100,
KARNATAKA, INDIA

TASK 1- Simple User interface design for front end

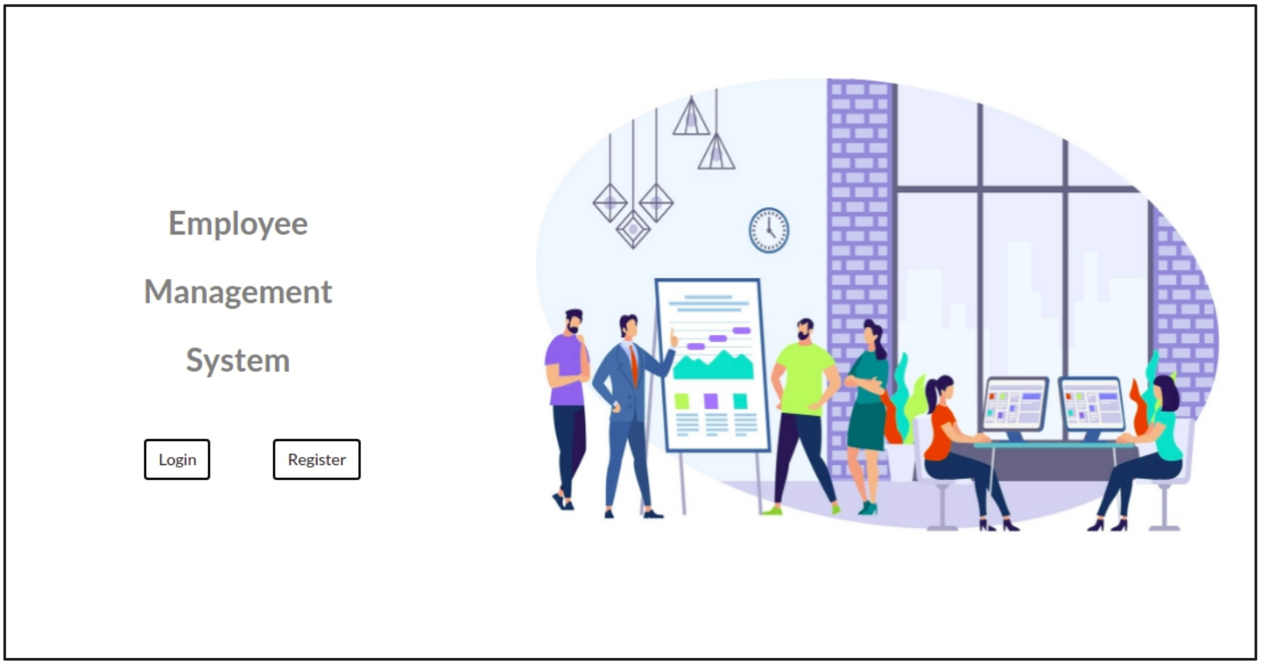
Technologies used: Postgres as database, React JS for Front End, Node JS for back-end, and Express JS as a back-end web framework.

Reasons for using PERN stack

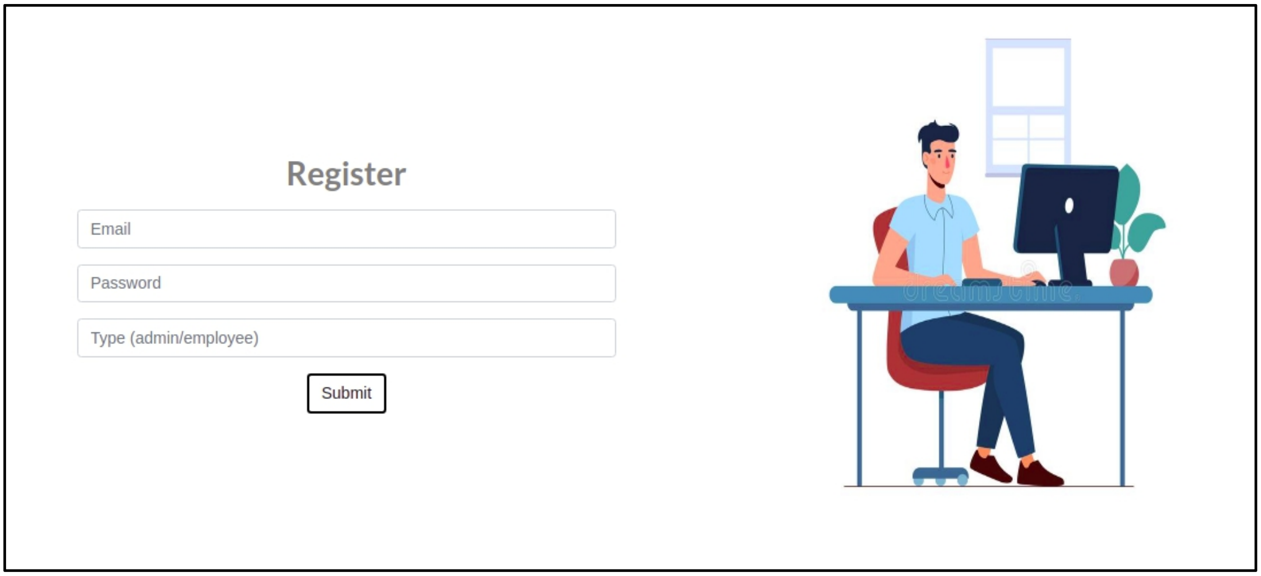
- React is used for handling the view layer for web and mobile apps and it also allows us to create reusable UI components.
- React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in the application. This corresponds to the view in the MVC template.
- NodeJS is the greatest tool for building real-time web applications. It provides cross-platform applications which run easily on any web. And hence, you don’t need anything extra for running up a node application. You only need for making one.
- NodeJS is a light, scalable, and open-source language platform that makes it very easy to build apps even at the enterprise level also.
- NodeJS increases the efficiency of the development process as it fills the gap between frontend and backend applications.

Demonstration Screenshots

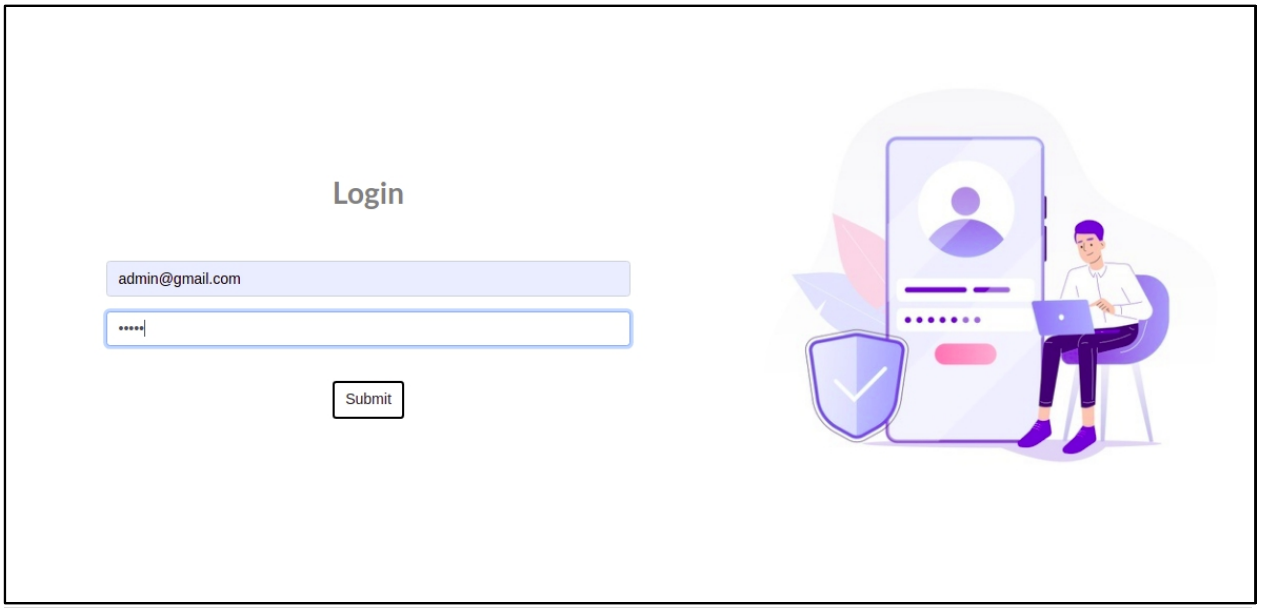
A home page having options to register or login



Register page inputting the email ID, password, and type. Based on the account type, the user would be given different options upon logging in.



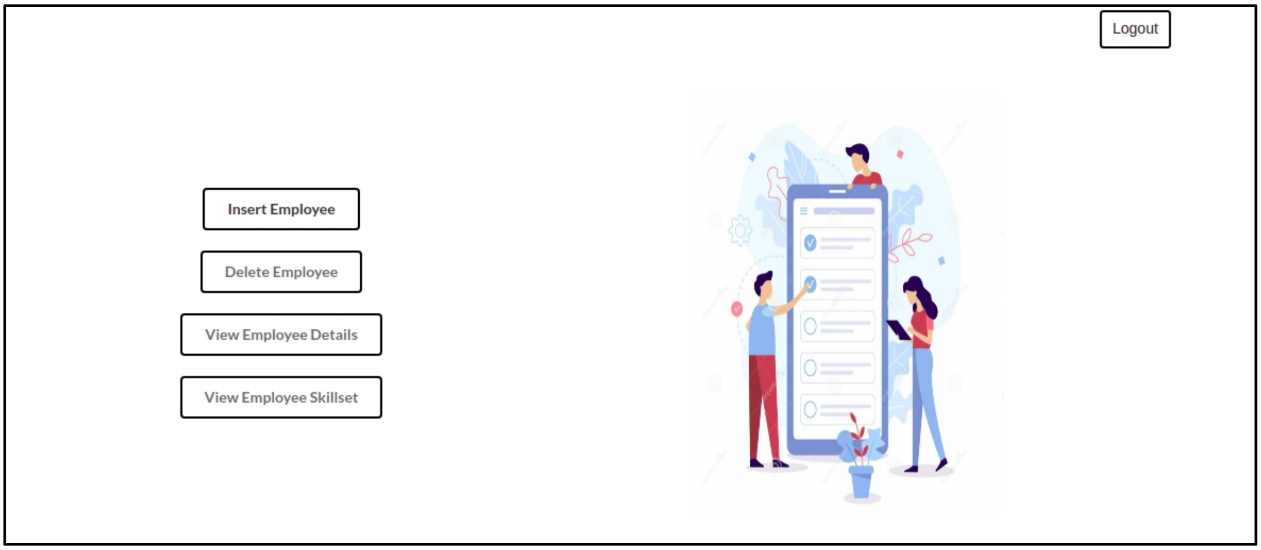
Login as account type admin



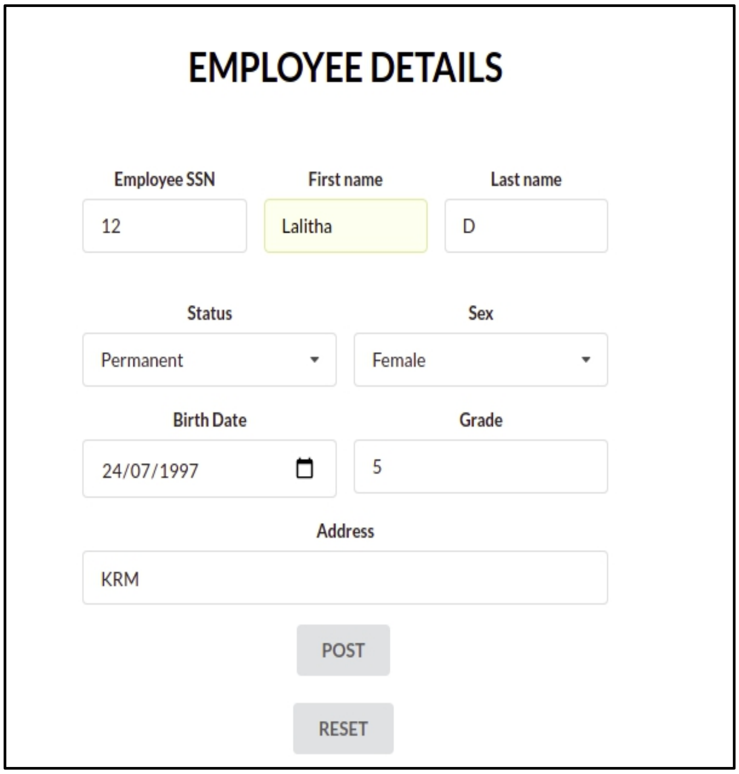
admin@gmail.com

•••••

Options available for account of type admin



Add new employee



EMPLOYEE DETAILS

Employee SSN: 12

First name: Lalitha

Last name: D

Status: Permanent

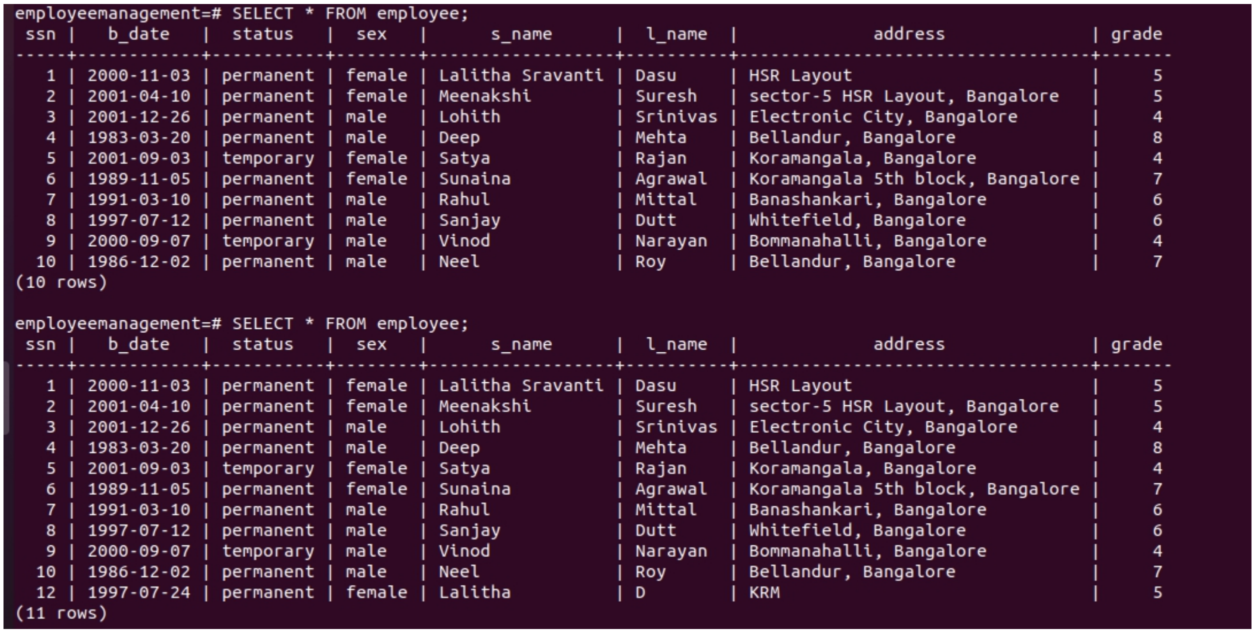
Sex: Female

Birth Date: 24/07/1997

Grade: 5

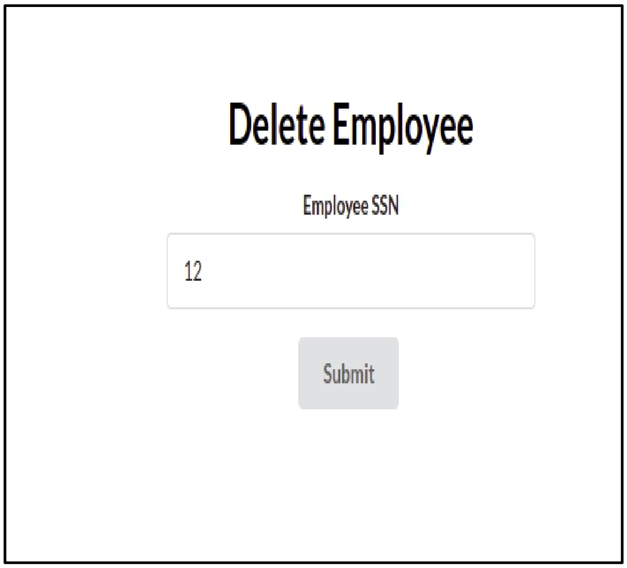
Address: KRM

Employee table before and after insertion



```
employeemanagement=# SELECT * FROM employee;
 ssn |   b_date   |  status   |  sex   |      s_name       |  l_name  |              address               | grade
-----+------------+-----------+--------+-------------------+----------+------------------------------------+-------
   1 | 2000-11-03 | permanent | female | Lalitha Sravanti | Dasu     | HSR Layout                         |     5
   2 | 2001-04-10 | permanent | female | Meenakshi         | Suresh   | sector-5 HSR Layout, Bangalore     |     5
   3 | 2001-12-26 | permanent | male   | Lohith            | Srinivas | Electronic City, Bangalore         |     4
   4 | 1983-03-20 | permanent | male   | Deep              | Mehta    | Bellandur, Bangalore               |     8
   5 | 2001-09-03 | temporary | female | Satya             | Rajan    | Koramangala, Bangalore             |     4
   6 | 1989-11-05 | permanent | female | Sunaina           | Agrawal  | Koramangala 5th block, Bangalore |     7
   7 | 1991-03-10 | permanent | male   | Rahul             | Mittal   | Banashankari, Bangalore            |     6
   8 | 1997-07-12 | permanent | male   | Sanjay            | Dutt     | Whitefield, Bangalore              |     6
   9 | 2000-09-07 | temporary | male   | Vinod             | Narayan  | Bommanahalli, Bangalore            |     4
  10 | 1986-12-02 | permanent | male   | Neel              | Roy      | Bellandur, Bangalore               |     7
(10 rows)

employeemanagement=# SELECT * FROM employee;
 ssn |   b_date   |  status   |  sex   |      s_name       |  l_name  |              address               | grade
-----+------------+-----------+--------+-------------------+----------+------------------------------------+-------
   1 | 2000-11-03 | permanent | female | Lalitha Sravanti | Dasu     | HSR Layout                         |     5
   2 | 2001-04-10 | permanent | female | Meenakshi         | Suresh   | sector-5 HSR Layout, Bangalore     |     5
   3 | 2001-12-26 | permanent | male   | Lohith            | Srinivas | Electronic City, Bangalore         |     4
   4 | 1983-03-20 | permanent | male   | Deep              | Mehta    | Bellandur, Bangalore               |     8
   5 | 2001-09-03 | temporary | female | Satya             | Rajan    | Koramangala, Bangalore             |     4
   6 | 1989-11-05 | permanent | female | Sunaina           | Agrawal  | Koramangala 5th block, Bangalore |     7
   7 | 1991-03-10 | permanent | male   | Rahul             | Mittal   | Banashankari, Bangalore            |     6
   8 | 1997-07-12 | permanent | male   | Sanjay            | Dutt     | Whitefield, Bangalore              |     6
   9 | 2000-09-07 | temporary | male   | Vinod             | Narayan  | Bommanahalli, Bangalore            |     4
  10 | 1986-12-02 | permanent | male   | Neel              | Roy      | Bellandur, Bangalore               |     7
  12 | 1997-07-24 | permanent | female | Lalitha           | D        | KRM                                |     5
(11 rows)
```

Delete an employee

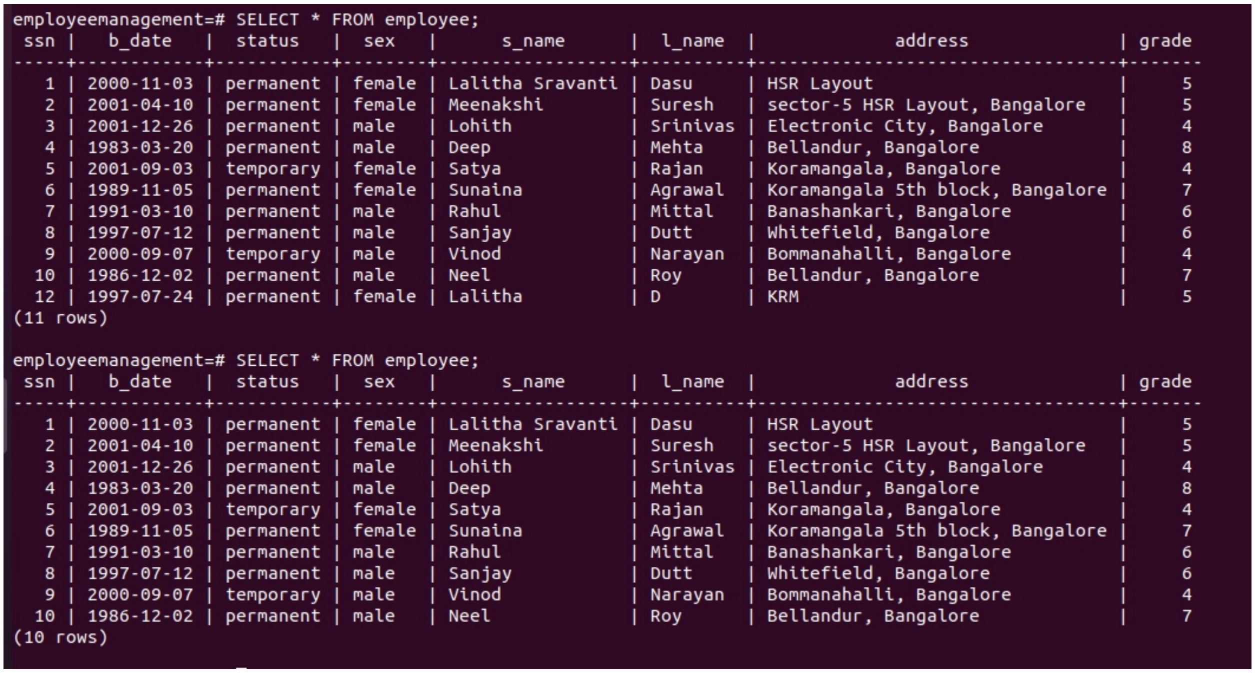


Delete Employee

Employee SSN

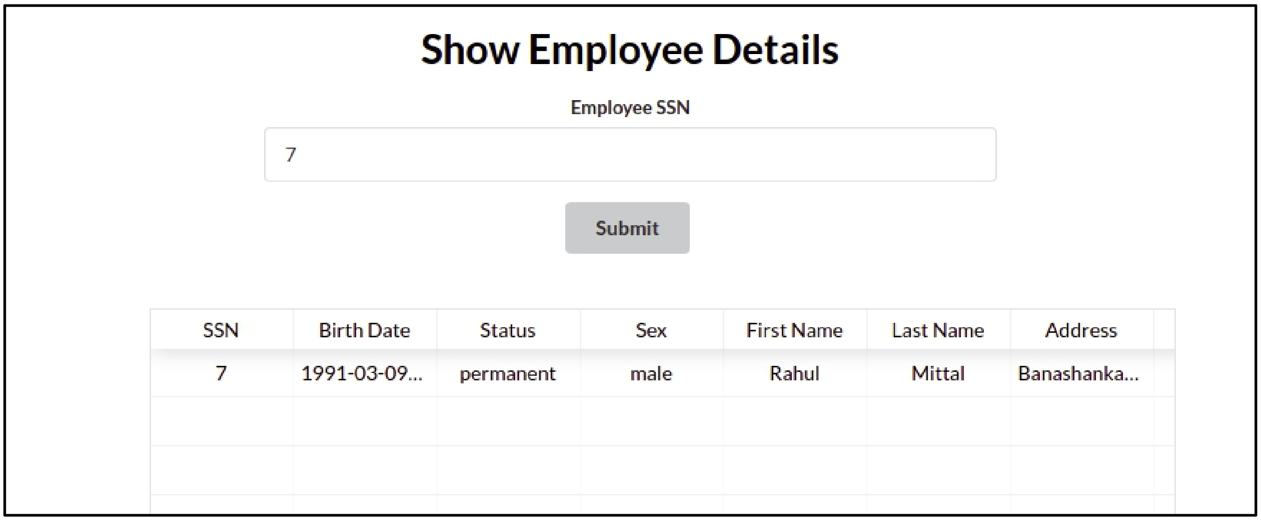
12

Employee table before and after deletion



```
employeemanagement=# SELECT * FROM employee;
 ssn |   b_date   |  status   |  sex   |      s_name      |  l_name  |             address              | grade
-----+------------+-----------+--------+------------------+----------+----------------------------------+-------
   1 | 2000-11-03 | permanent | female | Lalitha Sravanti | Dasu     | HSR Layout                       |     5
   2 | 2001-04-10 | permanent | female | Meenakshi        | Suresh   | sector-5 HSR Layout, Bangalore   |     5
   3 | 2001-12-26 | permanent | male   | Lohith           | Srinivas | Electronic City, Bangalore       |     4
   4 | 1983-03-20 | permanent | male   | Deep             | Mehta    | Bellandur, Bangalore             |     8
   5 | 2001-09-03 | temporary | female | Satya            | Rajan    | Koramangala, Bangalore           |     4
   6 | 1989-11-05 | permanent | female | Sunaina          | Agrawal  | Koramangala 5th block, Bangalore |     7
   7 | 1991-03-10 | permanent | male   | Rahul            | Mittal   | Banashankari, Bangalore          |     6
   8 | 1997-07-12 | permanent | male   | Sanjay           | Dutt     | Whitefield, Bangalore            |     6
   9 | 2000-09-07 | temporary | male   | Vinod            | Narayan  | Bommanahalli, Bangalore          |     4
  10 | 1986-12-02 | permanent | male   | Neel             | Roy      | Bellandur, Bangalore             |     7
  12 | 1997-07-24 | permanent | female | Lalitha          | D        | KRM                              |     5
(11 rows)

employeemanagement=# SELECT * FROM employee;
 ssn |   b_date   |  status   |  sex   |      s_name      |  l_name  |             address              | grade
-----+------------+-----------+--------+------------------+----------+----------------------------------+-------
   1 | 2000-11-03 | permanent | female | Lalitha Sravanti | Dasu     | HSR Layout                       |     5
   2 | 2001-04-10 | permanent | female | Meenakshi        | Suresh   | sector-5 HSR Layout, Bangalore   |     5
   3 | 2001-12-26 | permanent | male   | Lohith           | Srinivas | Electronic City, Bangalore       |     4
   4 | 1983-03-20 | permanent | male   | Deep             | Mehta    | Bellandur, Bangalore             |     8
   5 | 2001-09-03 | temporary | female | Satya            | Rajan    | Koramangala, Bangalore           |     4
   6 | 1989-11-05 | permanent | female | Sunaina          | Agrawal  | Koramangala 5th block, Bangalore |     7
   7 | 1991-03-10 | permanent | male   | Rahul            | Mittal   | Banashankari, Bangalore          |     6
   8 | 1997-07-12 | permanent | male   | Sanjay           | Dutt     | Whitefield, Bangalore            |     6
   9 | 2000-09-07 | temporary | male   | Vinod            | Narayan  | Bommanahalli, Bangalore          |     4
  10 | 1986-12-02 | permanent | male   | Neel             | Roy      | Bellandur, Bangalore             |     7
(10 rows)
```

View employee details



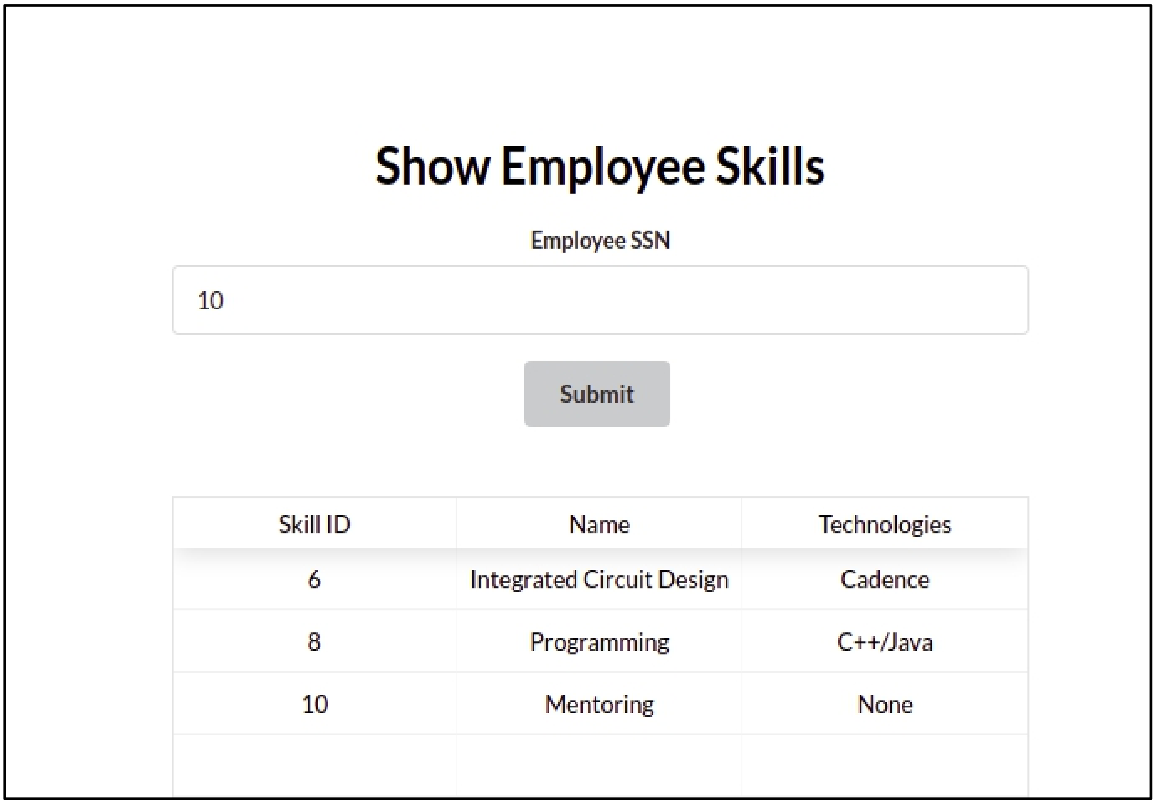
Show Employee Details

Employee SSN

7

SSN	Birth Date	Status	Sex	First Name	Last Name	Address
7	1991-03-09...	permanent	male	Rahul	Mittal	Banashanka...

View skills of an employee



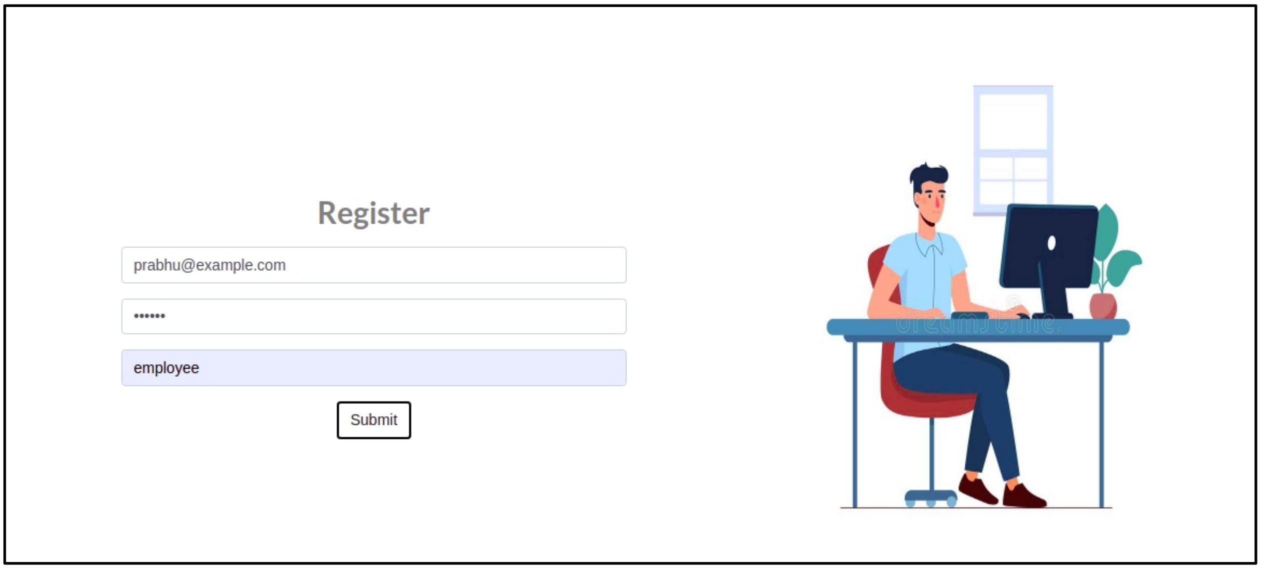
Show Employee Skills

Employee SSN

10

Skill ID	Name	Technologies
6	Integrated Circuit Design	Cadence
8	Programming	C++/Java
10	Mentoring	None

User registering an account of type employee



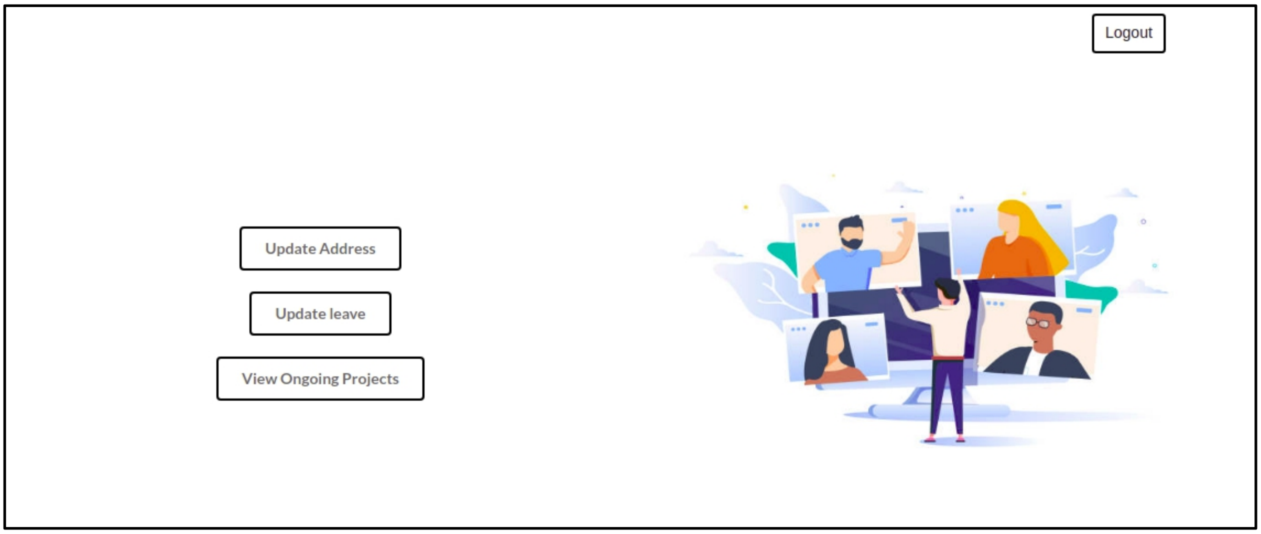
Register

prabhu@example.com

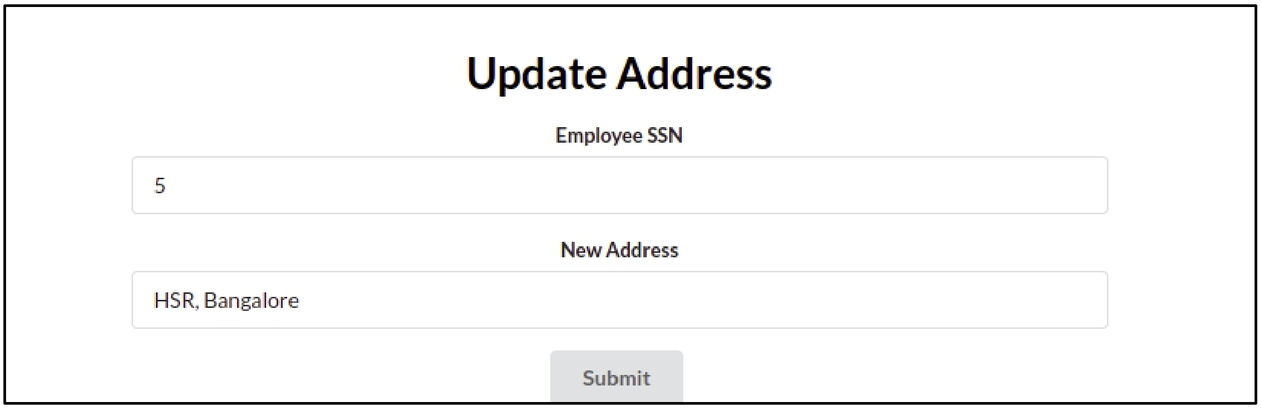
••••••

employee

Employee account options



Update employee address



Update Address

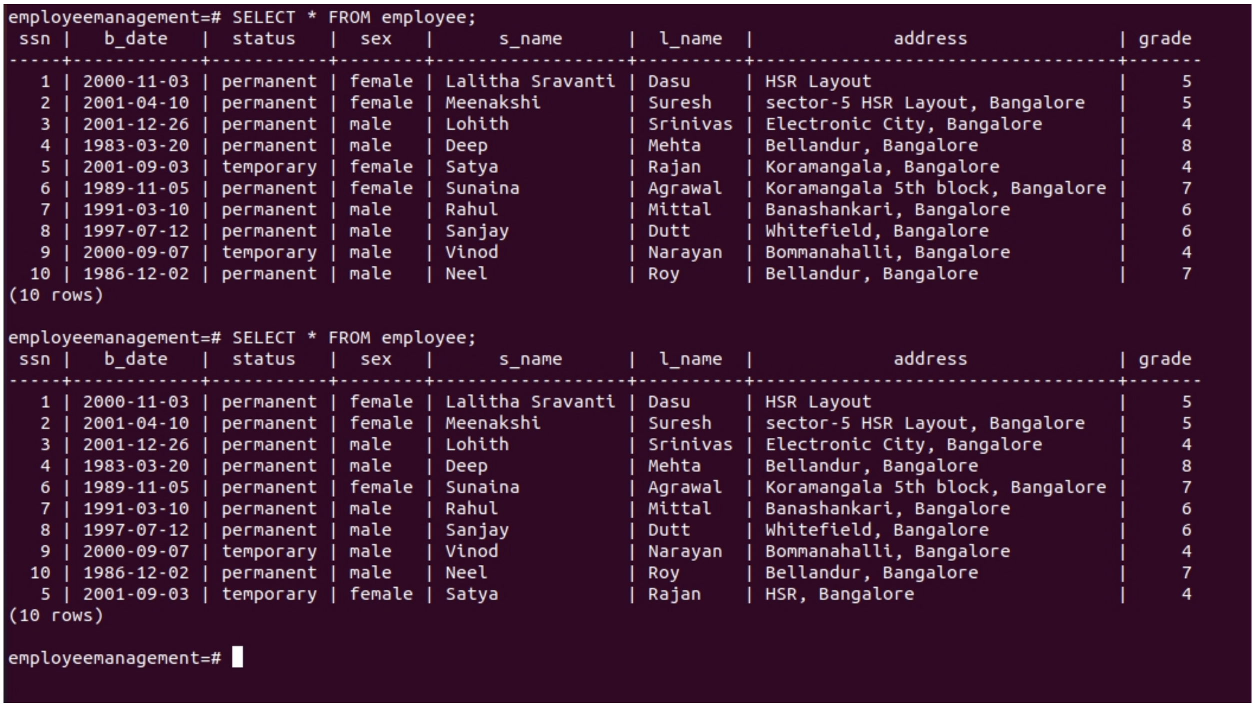
Employee SSN

5

New Address

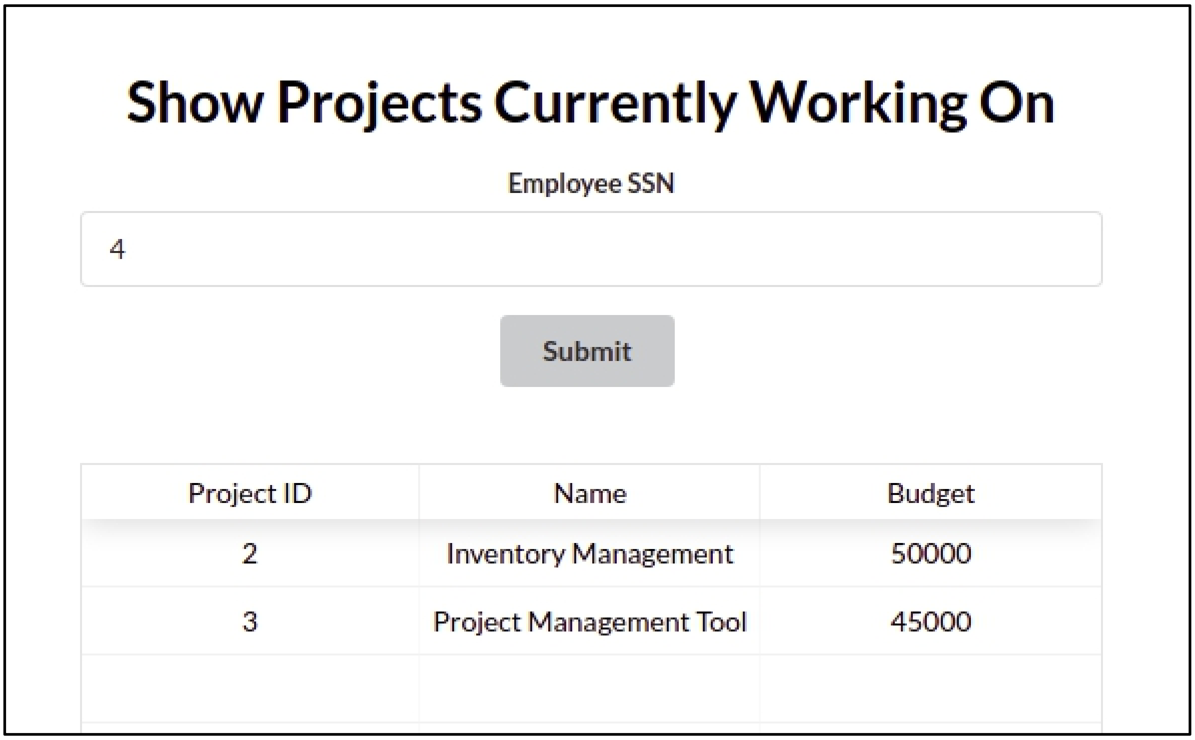
HSR, Bangalore

Before and after updating employee address (We can see the address of employee with ssn 5 has been updated)



```
employeemanagement=# SELECT * FROM employee;
 ssn |   b_date   |  status   |  sex   |      s_name      |  l_name  |             address              | grade
-----+------------+-----------+--------+------------------+----------+----------------------------------+-------
   1 | 2000-11-03 | permanent | female | Lalitha Sravanti | Dasu     | HSR Layout                       |     5
   2 | 2001-04-10 | permanent | female | Meenakshi        | Suresh   | sector-5 HSR Layout, Bangalore   |     5
   3 | 2001-12-26 | permanent | male   | Lohith           | Srinivas | Electronic City, Bangalore       |     4
   4 | 1983-03-20 | permanent | male   | Deep             | Mehta    | Bellandur, Bangalore             |     8
   5 | 2001-09-03 | temporary | female | Satya            | Rajan    | Koramangala, Bangalore           |     4
   6 | 1989-11-05 | permanent | female | Sunaina          | Agrawal  | Koramangala 5th block, Bangalore |     7
   7 | 1991-03-10 | permanent | male   | Rahul            | Mittal   | Banashankari, Bangalore          |     6
   8 | 1997-07-12 | permanent | male   | Sanjay           | Dutt     | Whitefield, Bangalore            |     6
   9 | 2000-09-07 | temporary | male   | Vinod            | Narayan  | Bommanahalli, Bangalore          |     4
  10 | 1986-12-02 | permanent | male   | Neel             | Roy      | Bellandur, Bangalore             |     7
(10 rows)

employeemanagement=# SELECT * FROM employee;
 ssn |   b_date   |  status   |  sex   |      s_name      |  l_name  |             address              | grade
-----+------------+-----------+--------+------------------+----------+----------------------------------+-------
   1 | 2000-11-03 | permanent | female | Lalitha Sravanti | Dasu     | HSR Layout                       |     5
   2 | 2001-04-10 | permanent | female | Meenakshi        | Suresh   | sector-5 HSR Layout, Bangalore   |     5
   3 | 2001-12-26 | permanent | male   | Lohith           | Srinivas | Electronic City, Bangalore       |     4
   4 | 1983-03-20 | permanent | male   | Deep             | Mehta    | Bellandur, Bangalore             |     8
   6 | 1989-11-05 | permanent | female | Sunaina          | Agrawal  | Koramangala 5th block, Bangalore |     7
   7 | 1991-03-10 | permanent | male   | Rahul            | Mittal   | Banashankari, Bangalore          |     6
   8 | 1997-07-12 | permanent | male   | Sanjay           | Dutt     | Whitefield, Bangalore            |     6
   9 | 2000-09-07 | temporary | male   | Vinod            | Narayan  | Bommanahalli, Bangalore          |     4
  10 | 1986-12-02 | permanent | male   | Neel             | Roy      | Bellandur, Bangalore             |     7
   5 | 2001-09-03 | temporary | female | Satya            | Rajan    | HSR, Bangalore                   |     4
(10 rows)

employeemanagement=# 
```

View ongoing projects

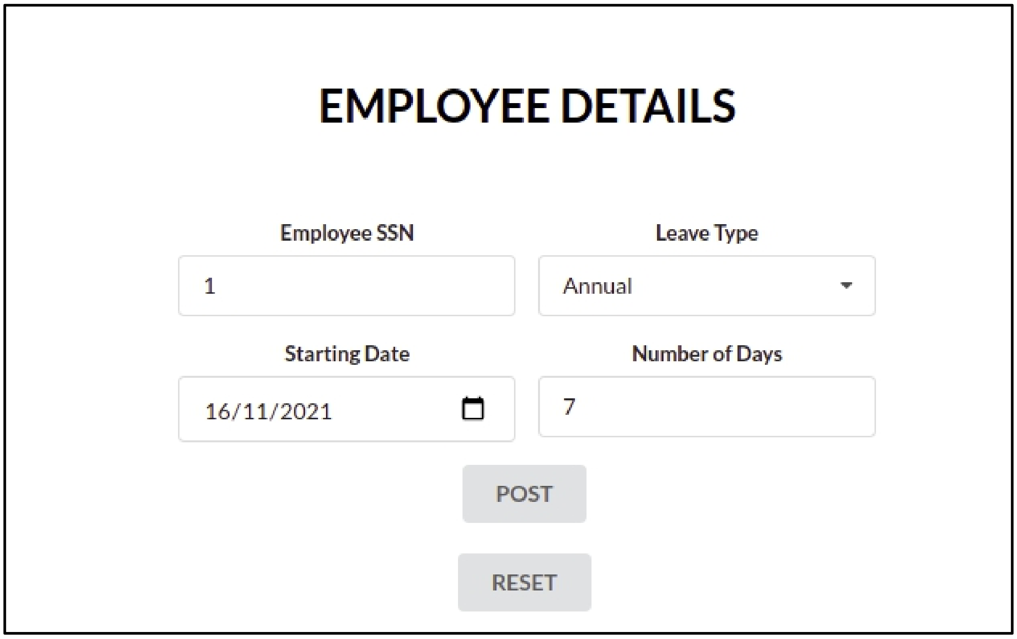


Show Projects Currently Working On

Employee SSN: 4

Project ID	Name	Budget
2	Inventory Management	50000
3	Project Management Tool	45000

Update leave



EMPLOYEE DETAILS

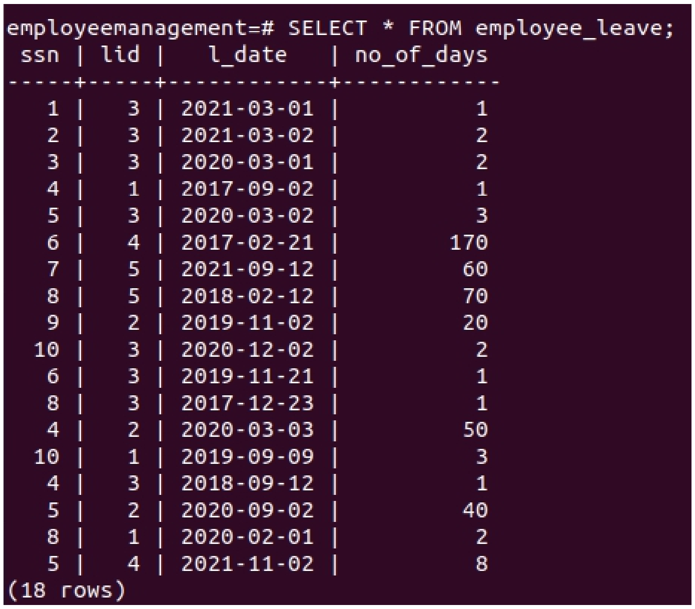
Employee SSN: 1

Leave Type: Annual

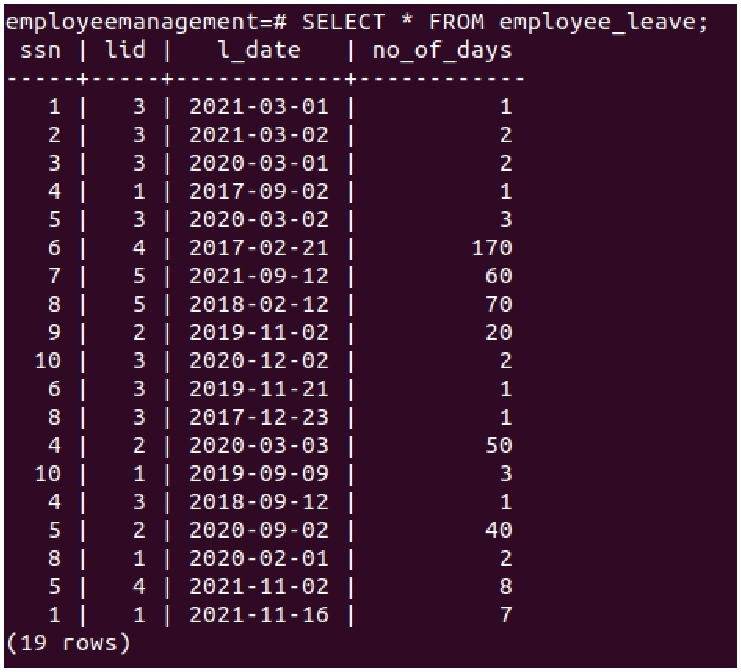
Starting Date: 16/11/2021

Number of Days: 7

Before and after updating leave



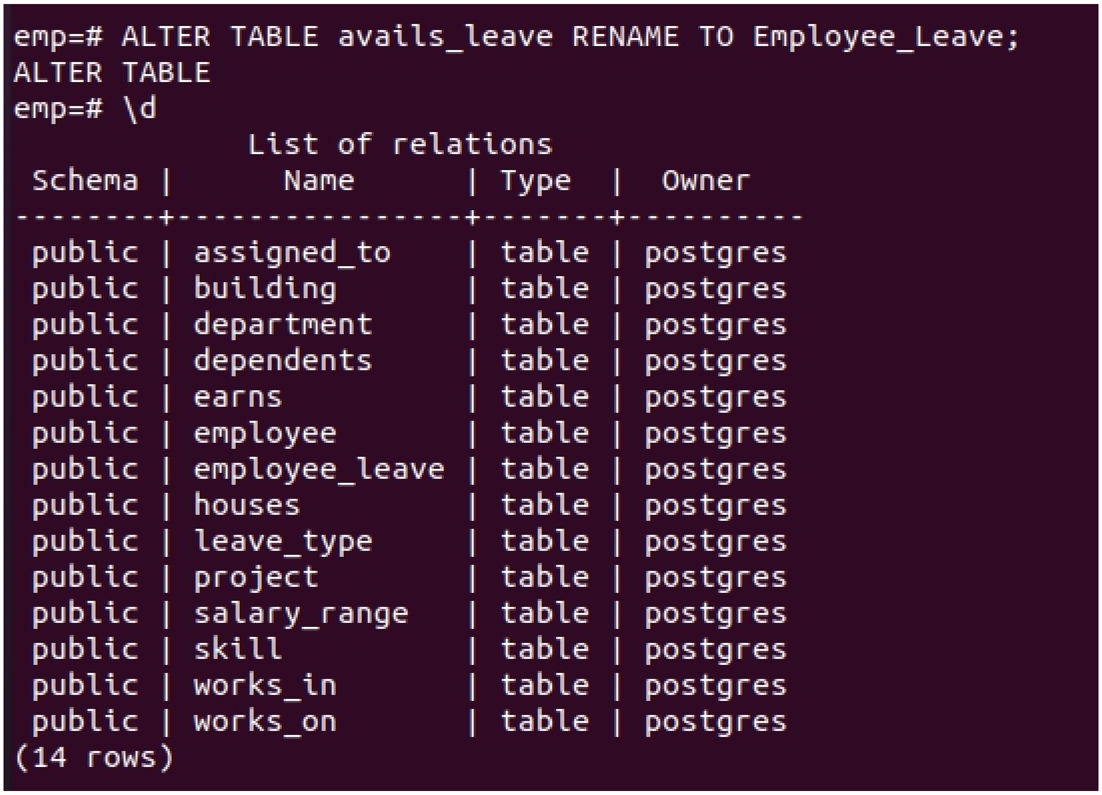
```
employeemanagement=# SELECT * FROM employee_leave;
 ssn | lid |   l_date   | no_of_days
-----+-----+------------+------------
   1 |   3 | 2021-03-01 |          1
   2 |   3 | 2021-03-02 |          2
   3 |   3 | 2020-03-01 |          2
   4 |   1 | 2017-09-02 |          1
   5 |   3 | 2020-03-02 |          3
   6 |   4 | 2017-02-21 |        170
   7 |   5 | 2021-09-12 |         60
   8 |   5 | 2018-02-12 |         70
   9 |   2 | 2019-11-02 |         20
  10 |   3 | 2020-12-02 |          2
   6 |   3 | 2019-11-21 |          1
   8 |   3 | 2017-12-23 |          1
   4 |   2 | 2020-03-03 |         50
  10 |   1 | 2019-09-09 |          3
   4 |   3 | 2018-09-12 |          1
   5 |   2 | 2020-09-02 |         40
   8 |   1 | 2020-02-01 |          2
   5 |   4 | 2021-11-02 |          8
(18 rows)
```



```
employeemanagement=# SELECT * FROM employee_leave;
 ssn | lid |   l_date   | no_of_days
-----+-----+------------+------------
   1 |   3 | 2021-03-01 |          1
   2 |   3 | 2021-03-02 |          2
   3 |   3 | 2020-03-01 |          2
   4 |   1 | 2017-09-02 |          1
   5 |   3 | 2020-03-02 |          3
   6 |   4 | 2017-02-21 |        170
   7 |   5 | 2021-09-12 |         60
   8 |   5 | 2018-02-12 |         70
   9 |   2 | 2019-11-02 |         20
  10 |   3 | 2020-12-02 |          2
   6 |   3 | 2019-11-21 |          1
   8 |   3 | 2017-12-23 |          1
   4 |   2 | 2020-03-03 |         50
  10 |   1 | 2019-09-09 |          3
   4 |   3 | 2018-09-12 |          1
   5 |   2 | 2020-09-02 |         40
   8 |   1 | 2020-02-01 |          2
   5 |   4 | 2021-11-02 |          8
   1 |   1 | 2021-11-16 |          7
(19 rows)
```

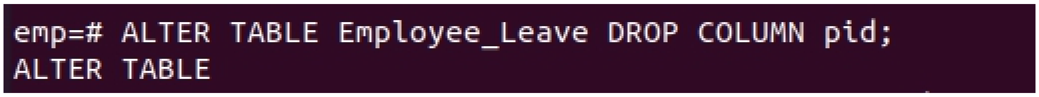
Task 2- Schema Changes

Quite a lot of tables names were changed for better readability. The screenshots of some such tables have been attached below:



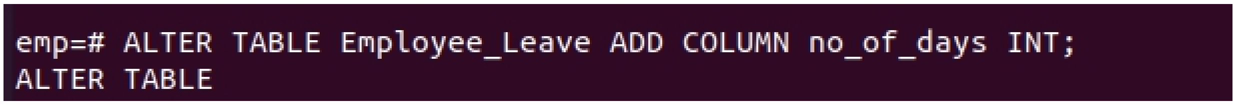
```
emp=# ALTER TABLE avails_leave RENAME TO Employee_Leave;
ALTER TABLE
emp=# \d
             List of relations
 Schema |      Name      | Type  |  Owner
--------+----------------+-------+----------
 public | assigned_to    | table | postgres
 public | building       | table | postgres
 public | department     | table | postgres
 public | dependents     | table | postgres
 public | earns          | table | postgres
 public | employee       | table | postgres
 public | employee_leave | table | postgres
 public | houses         | table | postgres
 public | leave_type     | table | postgres
 public | project        | table | postgres
 public | salary_range   | table | postgres
 public | skill          | table | postgres
 public | works_in       | table | postgres
 public | works_on       | table | postgres
(14 rows)
```

The column pid is dropped to make the new primary key the combination of department ID and project ID.



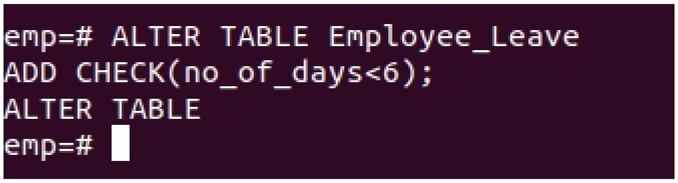
```
emp=# ALTER TABLE Employee_Leave DROP COLUMN pid;
ALTER TABLE
```

A column called no_of_days is added to indicate the number of leave days.



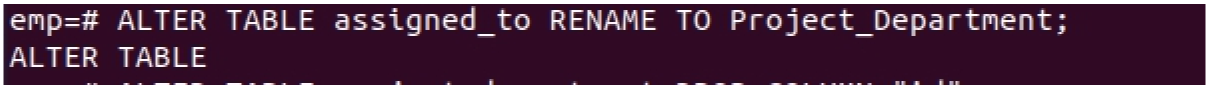
```
emp=# ALTER TABLE Employee_Leave ADD COLUMN no_of_days INT;
ALTER TABLE
```

A constraint was added on the number of days of leave availed.



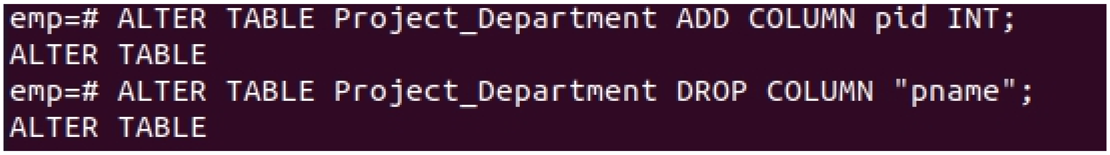
```
emp=# ALTER TABLE Employee_Leave
ADD CHECK(no_of_days<6);
ALTER TABLE
emp=# 
```

Table assigned_to renamed to Project_Department



```
emp=# ALTER TABLE assigned_to RENAME TO Project_Department;
ALTER TABLE
```

An integer value is preferred as a primary key over a string value. The primary key was changed from pname to pid.



```
emp=# ALTER TABLE Project_Department ADD COLUMN pid INT;
ALTER TABLE
emp=# ALTER TABLE Project_Department DROP COLUMN "pname";
ALTER TABLE
```

Task 3- Data Migration and Support

Reason for migration to an alternative:

The RDBMS which we have used is PostgreSQL. We now have a clear and concise predefined schema and we do not see the need for more schema changes as ours is an Employee Management system.

On the other hand, the main purpose of NoSQL DBMS is to mainly support schema-less data and for distributed data stores with humongous data storage needs.

If not for the performance issues, there is not an actual need to migrate from an RDBMS flavor to a NoSQL flavor as an Employee management system does not require the functionality of real-time analytics.

So, therefore, only taking into account the performance issues with PostgreSQL, we feel that migration to a document-based NoSQL database like MongoDB could contribute to improved performance.

Steps to follow to migrate from PostgreSQL to MongoDB:

While switching from PostgreSQL to MongoDB is not difficult, the process often involves more than just extracting and migrating data. You'll also need to examine the details of the applications that access your database.

For example, if you're using an ORM that does *not* support both relational and document databases, you’ll need to find a new library that can connect to MongoDB.

Once you’ve considered any changes needed to your application, the next step is to migrate the data. The migration for some of your tables might be simple. However, you might want to restructure your data to fit better within a MongoDB schema design. In that case, you should become familiar with best practices for MongoDB schema design, including anti-patterns.

The process for transferring data from PostgreSQL to MongoDB is clear-cut. Ultimately, the ease of your task depends on the complexity of the PostgreSQL database and the structure of document collections needed in the new MongoDB database.

To migrate data, you’ll extract it from PostgreSQL and then import it to MongoDB using the mongoimport tool.

Using JSON to transfer data:

Using JSON for data migration is preferable if your PostgreSQL schema is complex and you want to nest arrays of related records inside of a MongoDB document.

To return the results of a PostgreSQL query as JSON, you will need three functions:

1. row_to_json: Returns a row as a JSON object with column names as keys and the values from the row
2. array_to_json: Returns an array of data as a JSON array
3. array_agg: Accepts a set of values and returns an array where each value in the set becomes an element in the array

Let’s look at a dummy orders table (not relevant to our employee management system) which in our relational schema keeps a record for every product ordered by the user:

(This table is used solely for demonstration purposes)

```
 id | userid | product | quantity | price
----+--------+---------+----------+-------
  1 |      1 | shoes   |        4 | 50.75
  2 |      1 | razer   |       20 |  1.75
```

Here is an example query using all three functions:

```
COPY (SELECT row_to_json(results)
FROM (
  SELECT userid,first_name, last_name,
         (
         SELECT array_to_json(array_agg(o))
         FROM (
         SELECT id, product, quantity, price
         FROM orders
         WHERE products.userid = users.userid
         ) o
         ) AS orders
  FROM users
) results) TO '/tmp/orders.json' WITH (FORMAT text, HEADER FALSE);
```

The query above will create a file orders.json with JSON documents for each user from the users table:

```
{
        id: 1,
        first_name: "Bob",
        last_name: "Smith",
        "orders" : [
        {
        "id" : 1,
        "product" : "shoes",
        "quantity" : 4,
        "price" : 50.75
        },
        {
        "id" : 2,
        "product" : "razer",
        "quantity" : 20,
        "price" : 1.75
        }
        ]
}
```

Once you have written the query and saved it, you can use mongoimport to import the file:

```
mongoimport --uri
mongodb+srv://<mongodb_user>:<mongodb_password>@<atlas-cluster>.mognodb.net/<DATABASE>

--collection orders --jsonArray orders.json
```

Summing up the steps:

1. Prepare your application for connecting to MongoDB., MongoDB has support for all of the major programming languages as well as many popular frameworks.

2. Consider the schema changes that would be best for your data, while keeping in mind MongoDB schema best practices and avoiding anti-patterns.

3. Export the data from your PostgreSQL databases by piping the result of an SQL query into a COPY command, outputting the result either as JSON or TSV (Only if our relational schema is non-complex).

4. Restructure the data to fit your MongoDB schema by using mongoimport (or as an alternative: use bulkwrite operations to load the data).

Reasons for migrating to MongoDB among the available NoSQL databases:

1. MongoDB gives us the flexibility to insert data into mongo collection as per our need.

2. In MongoDB users can insert heterogeneous data.

3.MongoDB provides high scalability, availability, and performance.

4. With the help of the sharding feature we can save data to multiple servers without worrying about storage failure.

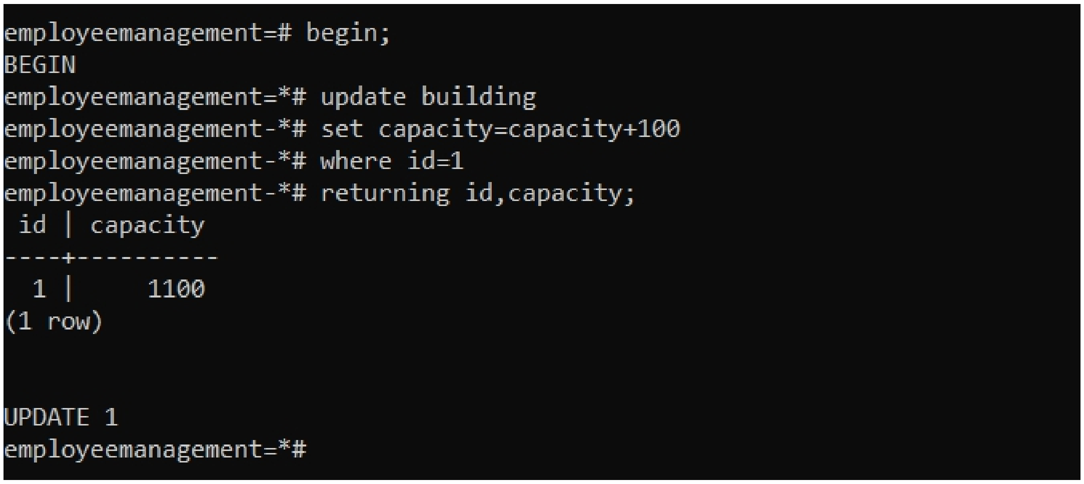
5. MongoDB has support for many drivers which help users to interact with MongoDB using multiple languages.

6. Searching a document in MongoDB is pretty fast as the documents are indexed.

Initiation of Concurrent transactions and demonstration of Concurrency control

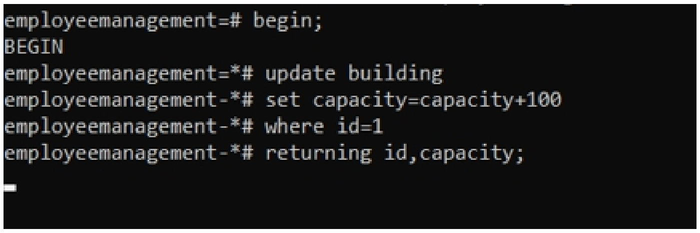
Isolation level ***read committed***

User 1



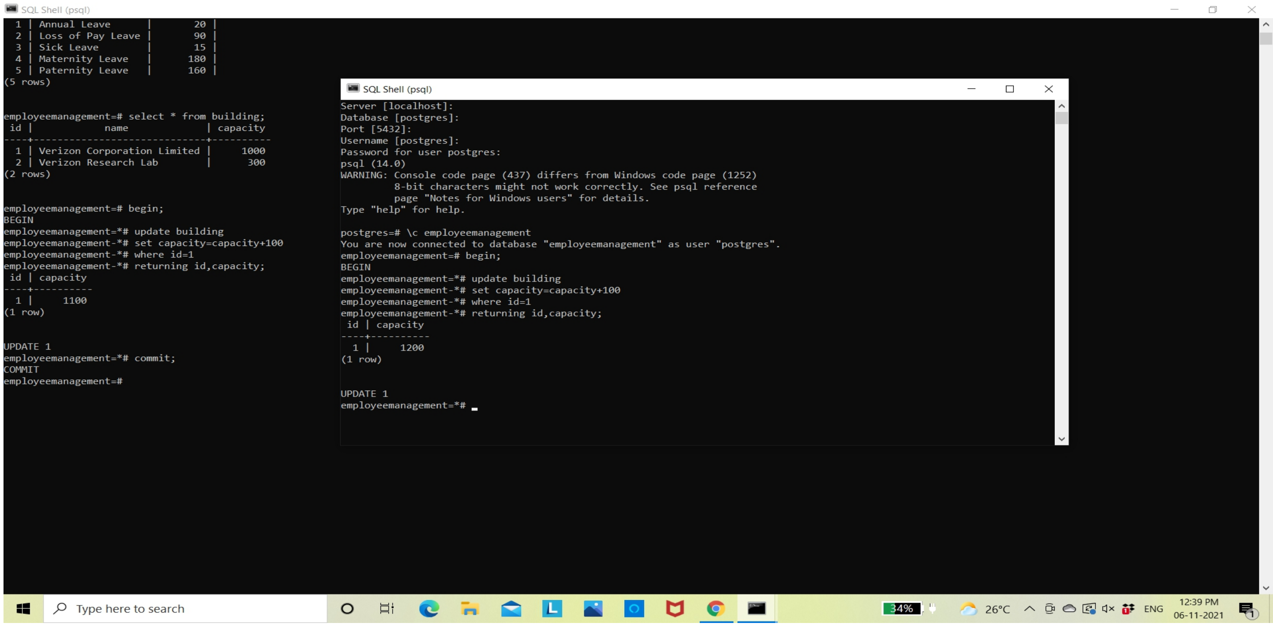
```
employeemanagement=# begin;
BEGIN
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
 id | capacity
----+----------
  1 |     1100
(1 row)


UPDATE 1
employeemanagement=*#
```

User 2 (Transaction paused)



```
employeemanagement=# begin;
BEGIN
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
```

User 1 and User 2(After commit in user1 transaction)

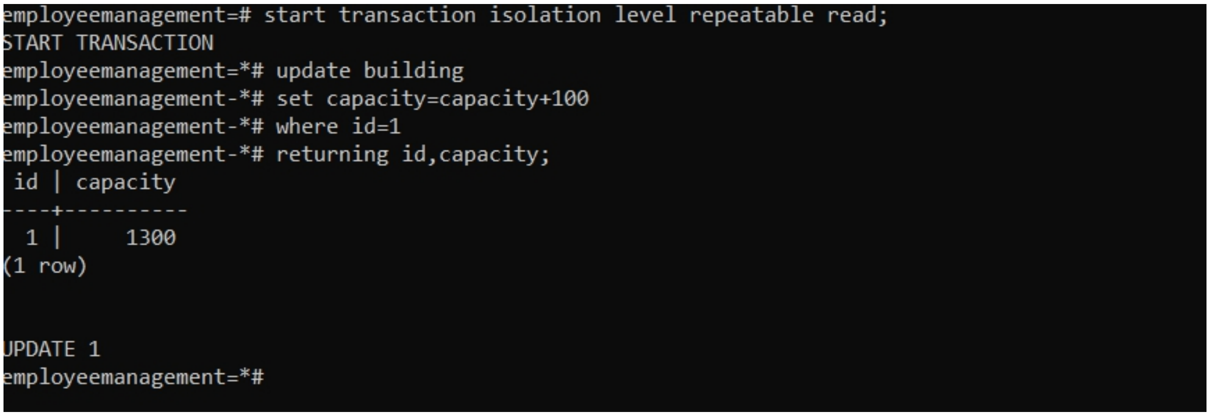


```
  1 | Annual Leave       |       20 |
  2 | Loss of Pay Leave |       90 |
  3 | Sick Leave         |       15 |
  4 | Maternity Leave    |      180 |
  5 | Paternity Leave    |      160 |
(5 rows)


employeemanagement=# select * from building;
 id |             name              | capacity
----+-------------------------------+----------
  1 | Verizon Corporation Limited |     1000
  2 | Verizon Research Lab          |      300
(2 rows)


employeemanagement=# begin;
BEGIN
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
 id | capacity
----+----------
  1 |     1100
(1 row)


UPDATE 1
employeemanagement=*# commit;
COMMIT
employeemanagement=#
```

```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (14.0)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# \c employeemanagement
You are now connected to database "employeemanagement" as user "postgres".
employeemanagement=# begin;
BEGIN
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
 id | capacity
----+----------
  1 |     1200
(1 row)


UPDATE 1
employeemanagement=*#
```

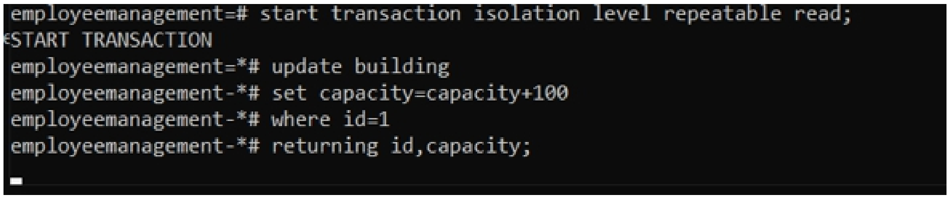
Isolation level *repeatable read*

User 1



```
employeemanagement=# start transaction isolation level repeatable read;
START TRANSACTION
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
 id | capacity
----+----------
  1 |     1300
(1 row)


UPDATE 1
employeemanagement=*#
```

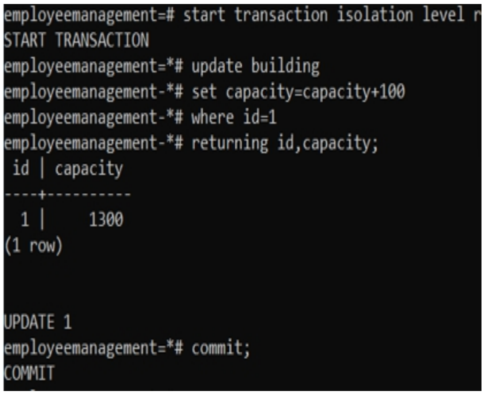
User 2 (Transaction paused)



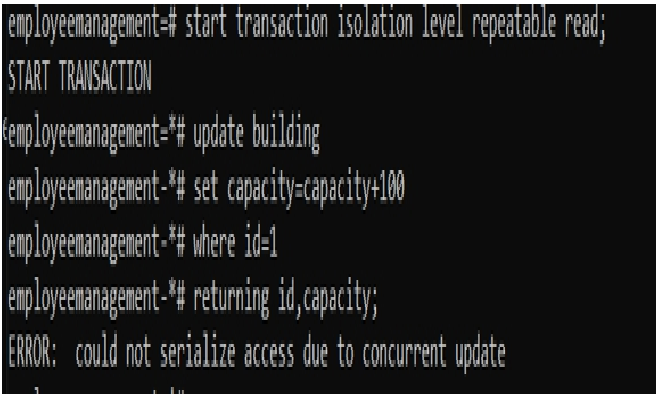
```
employeemanagement=# start transaction isolation level repeatable read;
START TRANSACTION
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
```

User1 and User2 (After commit in user1 transaction)

User 1 User 2



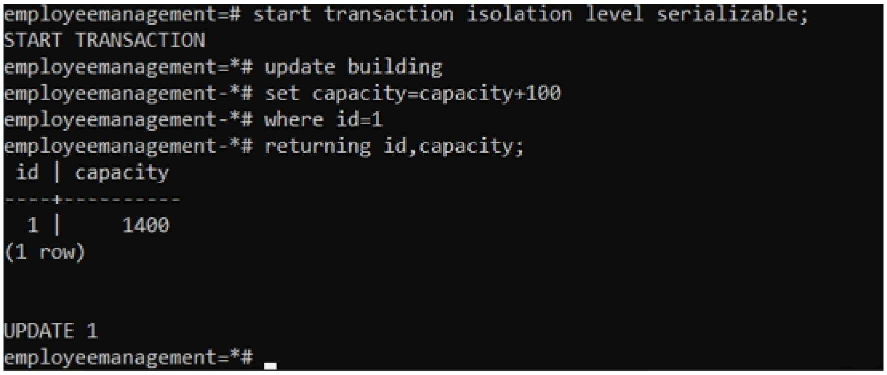
```
employeemanagement=# start transaction isolation level r
START TRANSACTION
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
 id | capacity
----+----------
  1 |     1300
(1 row)


UPDATE 1
employeemanagement=*# commit;
COMMIT
```



```
employeemanagement=# start transaction isolation level repeatable read;
START TRANSACTION
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
ERROR:  could not serialize access due to concurrent update
```

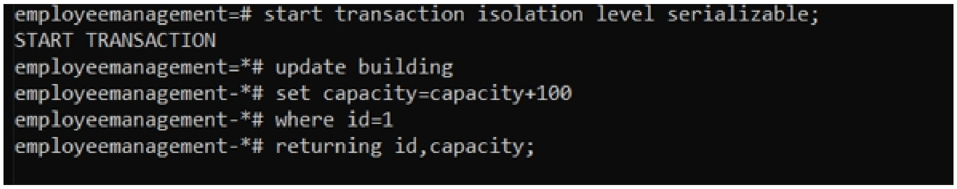
Isolation level *serializable*

User1



```
employeemanagement=# start transaction isolation level serializable;
START TRANSACTION
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
 id | capacity
----+----------
  1 |     1400
(1 row)


UPDATE 1
employeemanagement=*# _
```

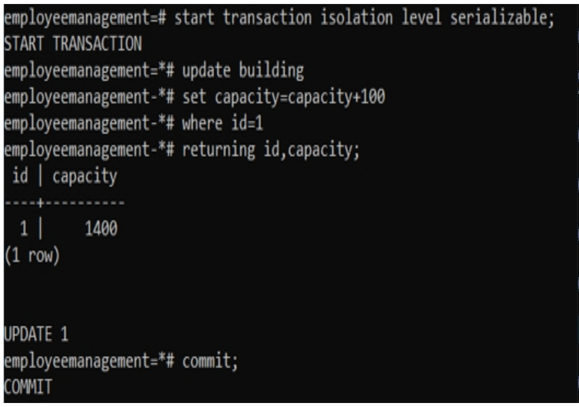
User 2 (Transaction paused)



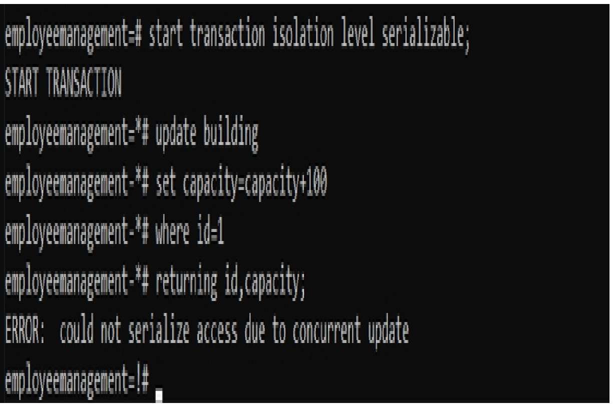
```
employeemanagement=# start transaction isolation level serializable;
START TRANSACTION
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
```

User1 and User2 (After commit in user1 transaction)

User 1 User 2



```
employeemanagement=# start transaction isolation level serializable;
START TRANSACTION
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
 id | capacity
----+----------
  1 |     1400
(1 row)


UPDATE 1
employeemanagement=*# commit;
COMMIT
```



```
employeemanagement=# start transaction isolation level serializable;
START TRANSACTION
employeemanagement=*# update building
employeemanagement-*# set capacity=capacity+100
employeemanagement-*# where id=1
employeemanagement-*# returning id,capacity;
ERROR:  could not serialize access due to concurrent update
employeemanagement=!#
```

CONTRIBUTIONS

Name	SRN	TASK	TIME SPENT
Lalitha Sravanti Dasu	PES2UG19CS201	User Interface using PERN, Schema changes	6 days
Lohith Srinivas	PES2UG19CS203	Data Migration and Support, Transaction and Demo of Concurrency control	3 days
Meenakshi Suresh	PES2UG19CS228	User Interface using PERN, Schema changes	6 days