

1. Installed PostgreSQL

- You don't need any kind of GUI
- Very fast as compared to other dbs
- Free , Open Source
- World's most advanced open source relational dbs
- Port : 5432
- pgAdmin4 is the web based GUI client
- Tables are under the pg server >> schemas

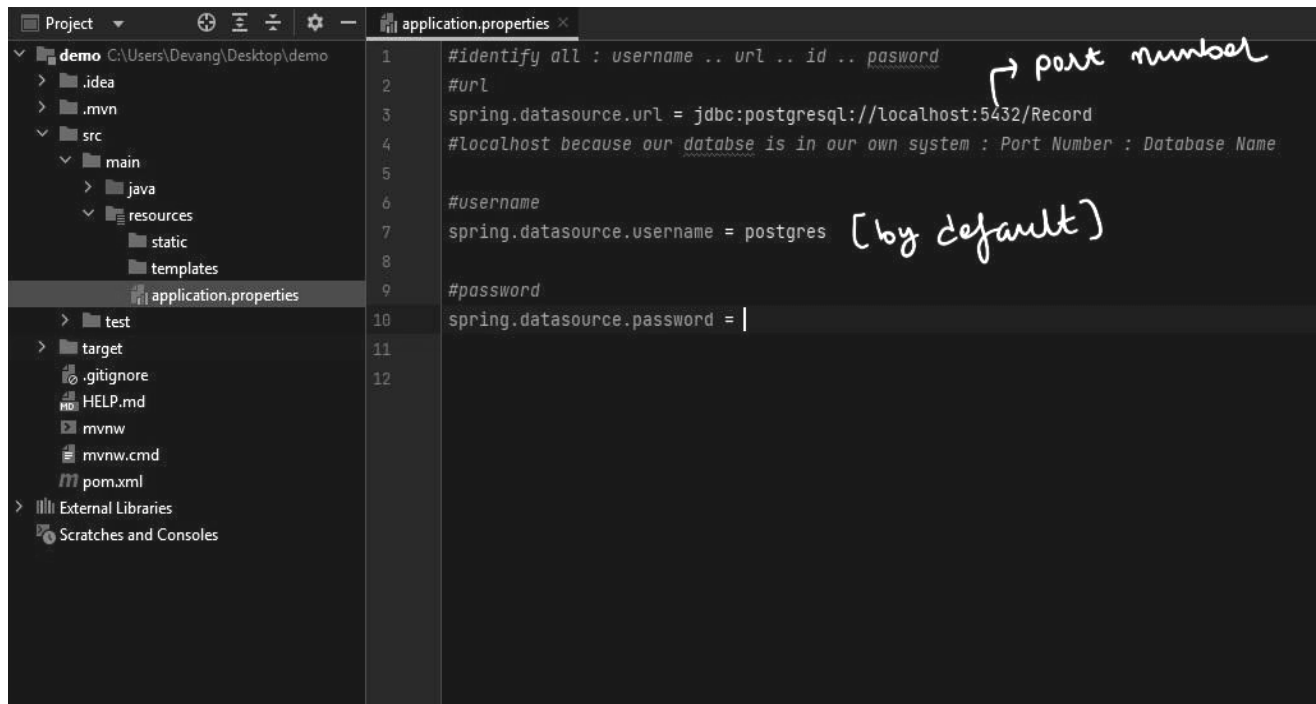
2. IDE used : IntelliJ

3. start.spring.io : Filled in the details and other things to initialize my spring boot project.

- Project : Maven
- Language : Java
- Dependencies : Postgres Driver , JDBC Driver

4. In the project

- src >> main >> resources >> application.properties
- In this file filled in the postgres credentials



```
1 #identify all : username .. url .. id .. password
2 #url
3 spring.datasource.url = jdbc:postgresql://localhost:5432/Record
4 #localhost because our database is in our own system : Port Number : Database Name
5
6 #username
7 spring.datasource.username = postgres [by default]
8
9 #password
10 spring.datasource.password = |
11
12
```

- src >> main >> java

- Created a class person.java
- In this initially importing all modules ... by initializer and by editor
- Autowiring the object of JdbcTemplate in our class person
- Creating the table and inserting the data

```

import ...
1 usage
@Repository
public class Person {

    3 usages
    @Autowired
    //Autowiring : Feature of spring framework in
    //which spring container inject the dependancies automatically
    //Works with objects only
    private JdbcTemplate jdbcTemplate;

    //Creating the table
    public void createTable(){
        var query = "CREATE TABLE person(id SERIAL PRIMARY KEY , " +
            "First_Name varchar(255) NOT NULL , Last_Name varchar(255) NOT NULL , " +
            "Email_id varchar(255))";
        int update = this.jdbcTemplate.update(query);
        System.out.println(update);
    }

    //Inserting the data
    public void insertData(String fname,String lname, String id){
        String q = "INSERT INTO person(first_name,last_name,email_id) values(?,?,?)";
        int update = this.jdbcTemplate.update(q,fname,lname,id);
        System.out.println(update + "rows added");
    }
}

```

importing the modules

System → *Can be used in new Java versions*

defining the structure of table

It will return the no. of rows affected

Inserting

- Inserting With Input from user

```

//Inserting the data with input
1 usage
public void insertDataWithInput(){
    Scanner s = new Scanner(System.in);

    System.out.println("Enter First Name");
    String f_name = s.nextLine();

    System.out.println("Enter Last Name");
    String l_name = s.nextLine();

    System.out.println("Enter Email Id");
    String e_id = s.nextLine();

    String q = "INSERT INTO person(first_name,last_name,email_id) values(?,?,?)";
    int update = this.jdbcTemplate.update(q,f_name,l_name,e_id);

    System.out.println(update + "rows added");
}

```

Inputting data from user

Scanner

System

insertions via jdbc template

- To execute a query, you use one of the following methods of the Statement object:
 - execute: returns true if the first object of the query is a ResultSet object. You can get the ResultSet by calling the method getResultSet.
 - executeQuery: returns only one ResultSet object.
 - executeUpdate: returns the number of rows affected by the statement. You use this method for the INSERT, DELETE, or UPDATE statement.

5. After this inside main >> java >> com.postgre >> Creation >> DemoApplication.java.
6. Learnt to use CLI (Command Line Interface)

```
package com.postgraes.demo;

import ...

1 usage
@SpringBootApplication
public class DemoApplication implements CommandLineRunner{

    1 usage
    @Autowired
    private Person person;

    public static void main(String[] args){ SpringApplication.run(DemoApplication.class, args); }

    @Override
    public void run(String... args) throws Exception {
        |      this.person.insertDataWithInput();
        |
    }

}

//Command Line Runner : Interface
//It helps to run and test the Spring Boot Applications using command prompt
```

learn CLI

autowiring the object of person in our class.

using the methods of person class

7. Outputs and results :

pgAdmin 4

Admin File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.person/... public.pe < > x

public.person/Record/postgres@PostgreSQL 15

Query Query History Scratch Pad x

```
1 SELECT * FROM public.person
2 ORDER BY id ASC
```

Data Output Messages Notifications

	id [PK] integer	first_name character varying (255)	last_name character varying (255)	email_id character varying (255)
1	1	Devang	Chopra	devangchopra@gmail.com
2	2	Quick	Snap	quicksnap@gmail.com
3	5	Hello	World	helloworld@gmail.com
4	6	Hi	World	Hi@world.com
5	7	HP	Laptop	hp@gmail.com
6	8	Apple	Laptop	apple@gmail.com

