
Name: **Zahra Soukhtedel**

Student Id: **98105138**

Assignment Number: **1**

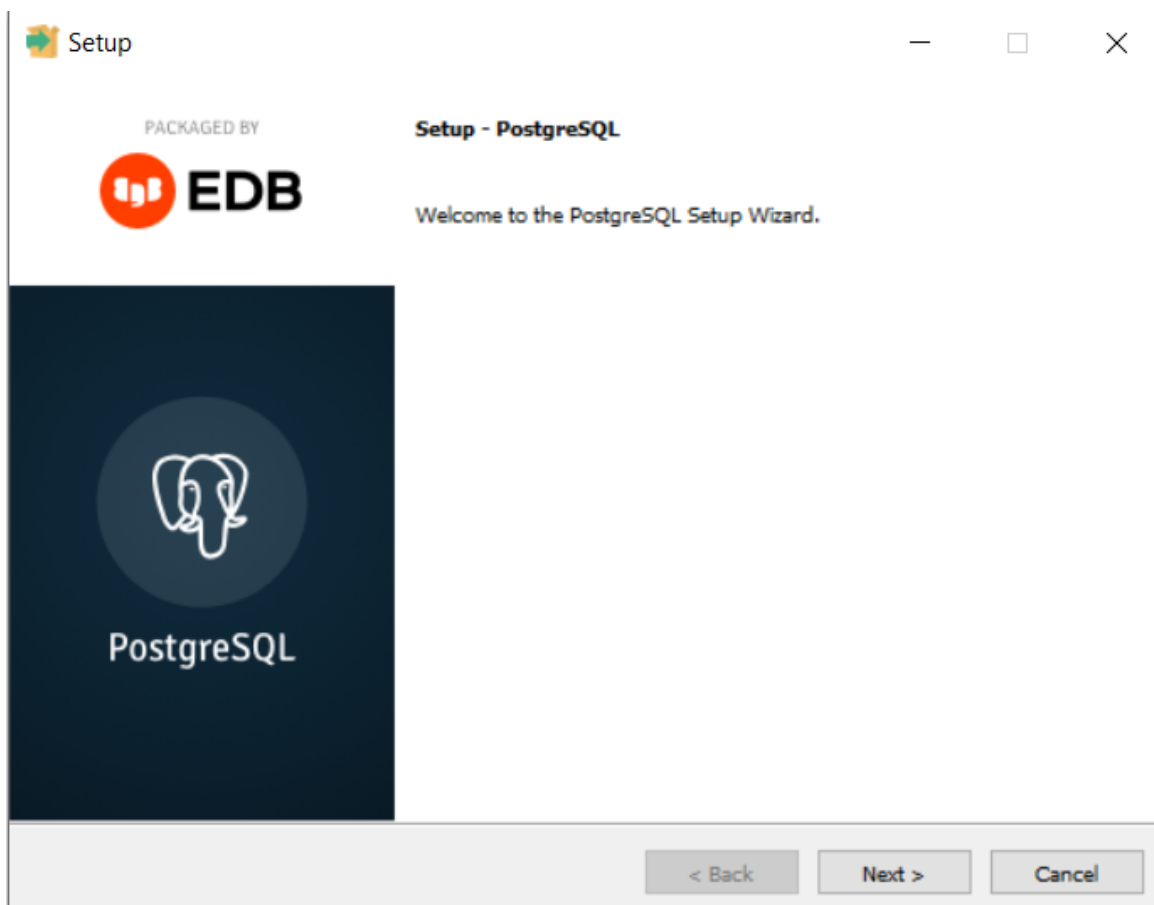
Course: **Database**

Problem 1

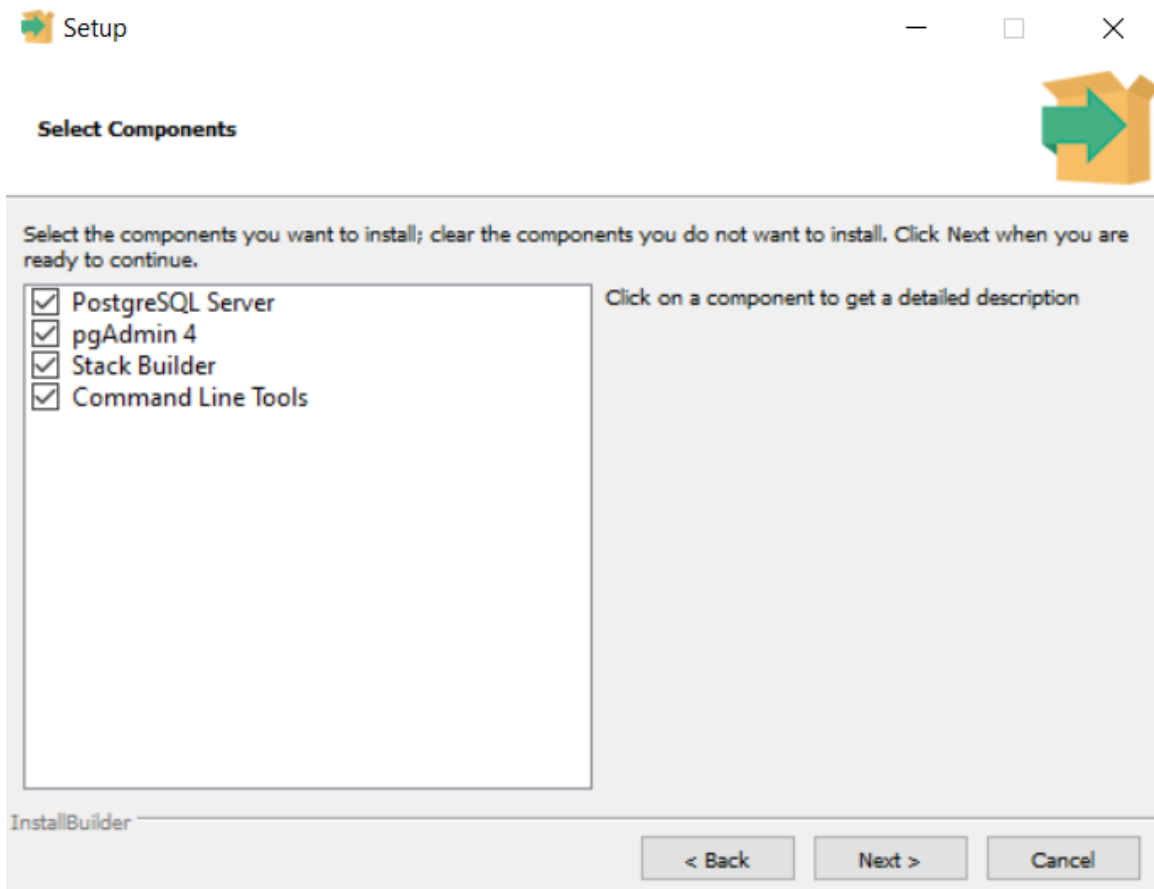
Select a DBMS and install it:

Following is a step-by-step process on How to Install PostgreSQL on Windows:

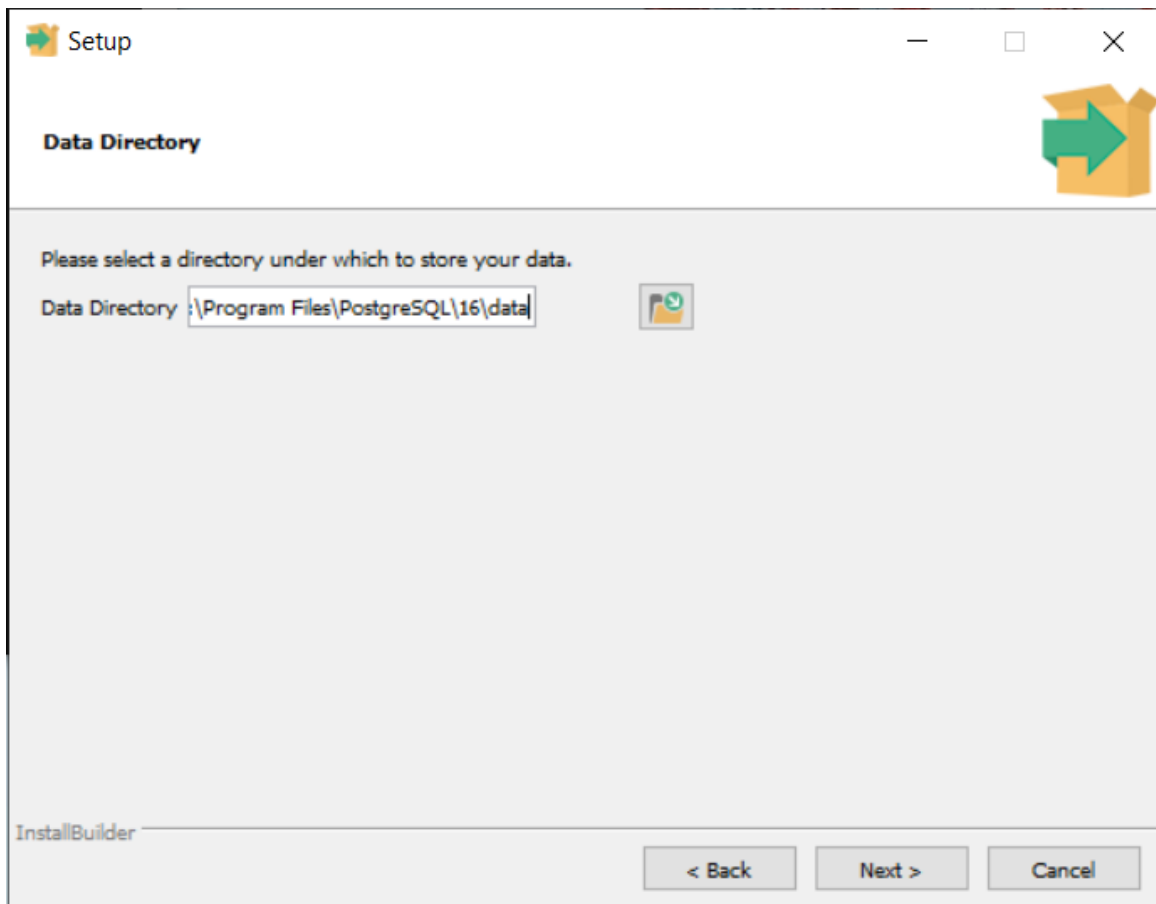
1. Visit <https://www.postgresql.org/download/> and download the appropriate version for your operating system.
2. Once Download PostgreSQL, open the downloaded exe and Click next on the install welcome screen:



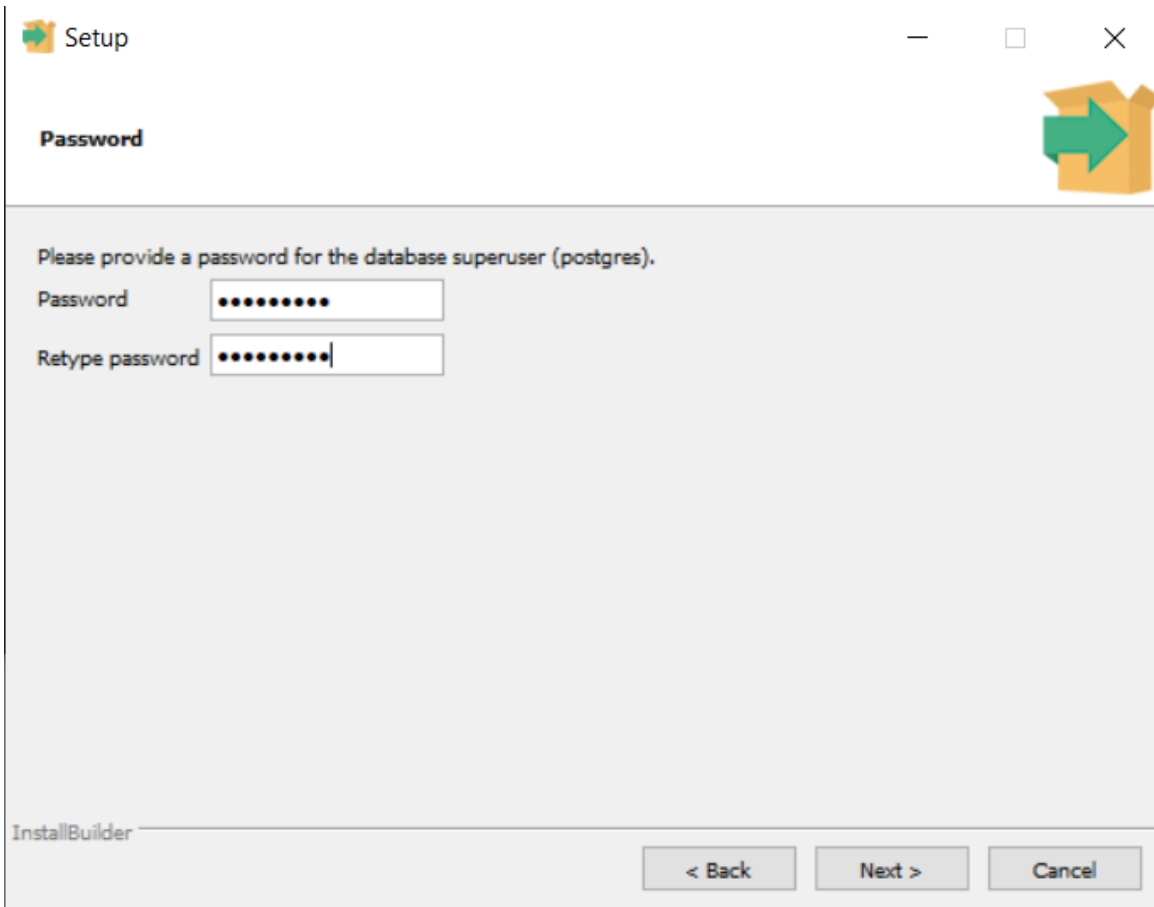
3. You may choose the components you want to install in your system. You may uncheck Stack Builder, and click Next.



4. You may change the data location, and click Next.



5. Enter the superuser password. Make a note of it, click Next.



Setup

Password

Please provide a password for the database superuser (postgres).

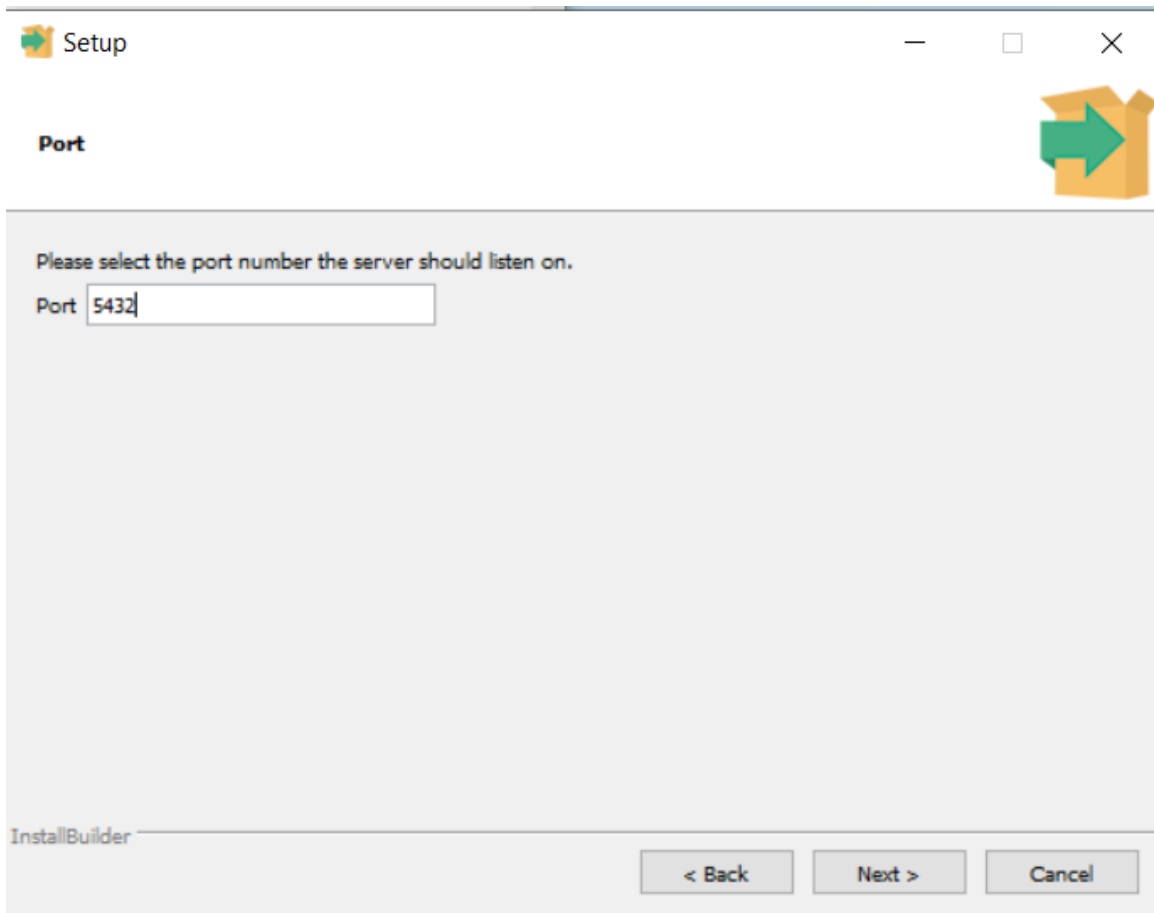
Password

Retype password

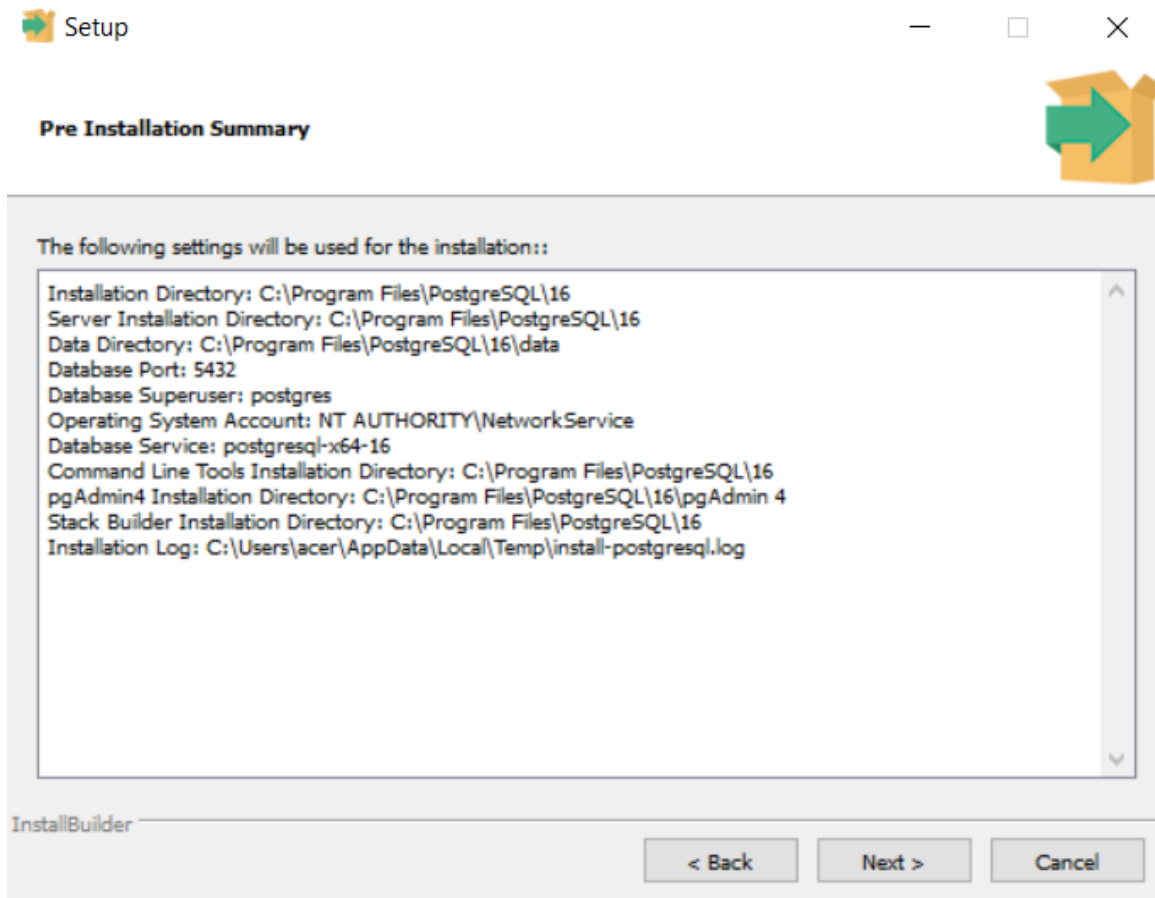
InstallBuilder

< Back Next > Cancel

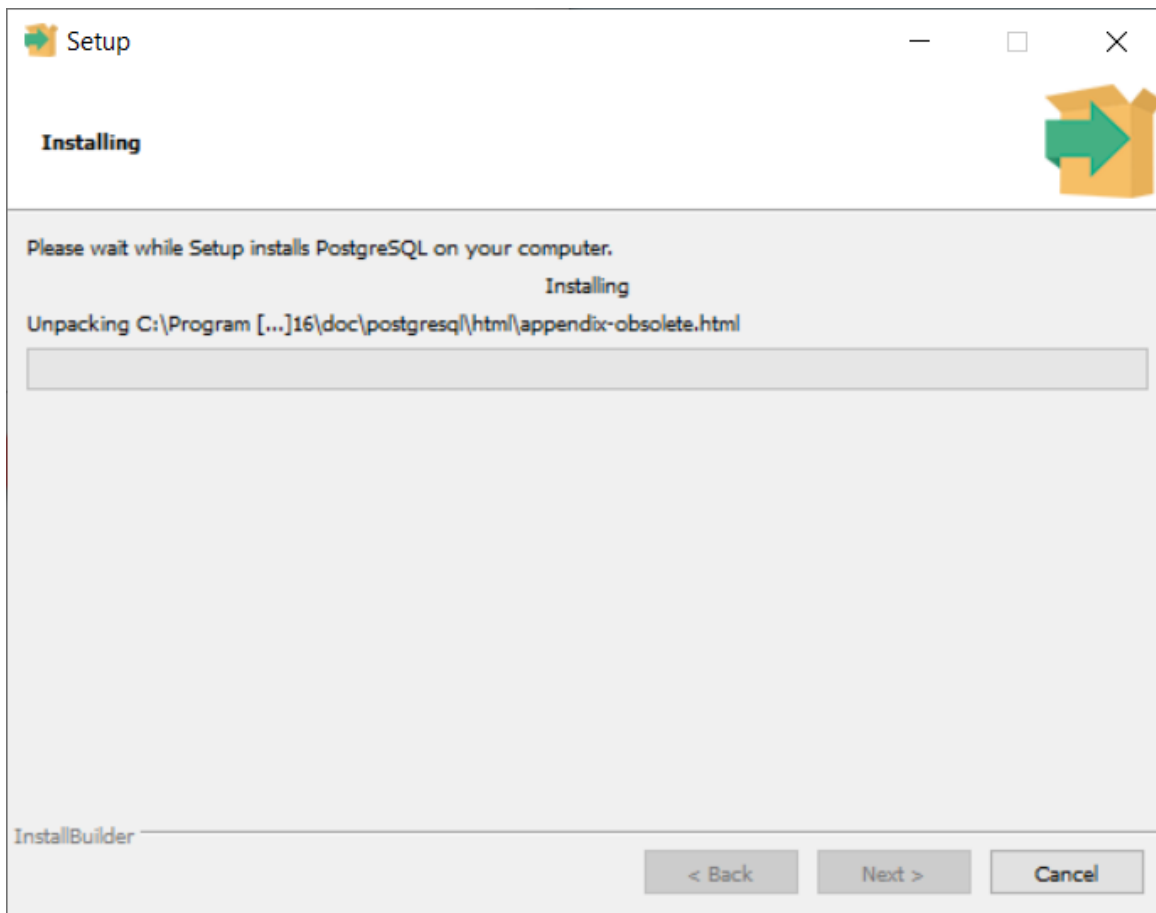
6. Leave the port number default, and click Next.



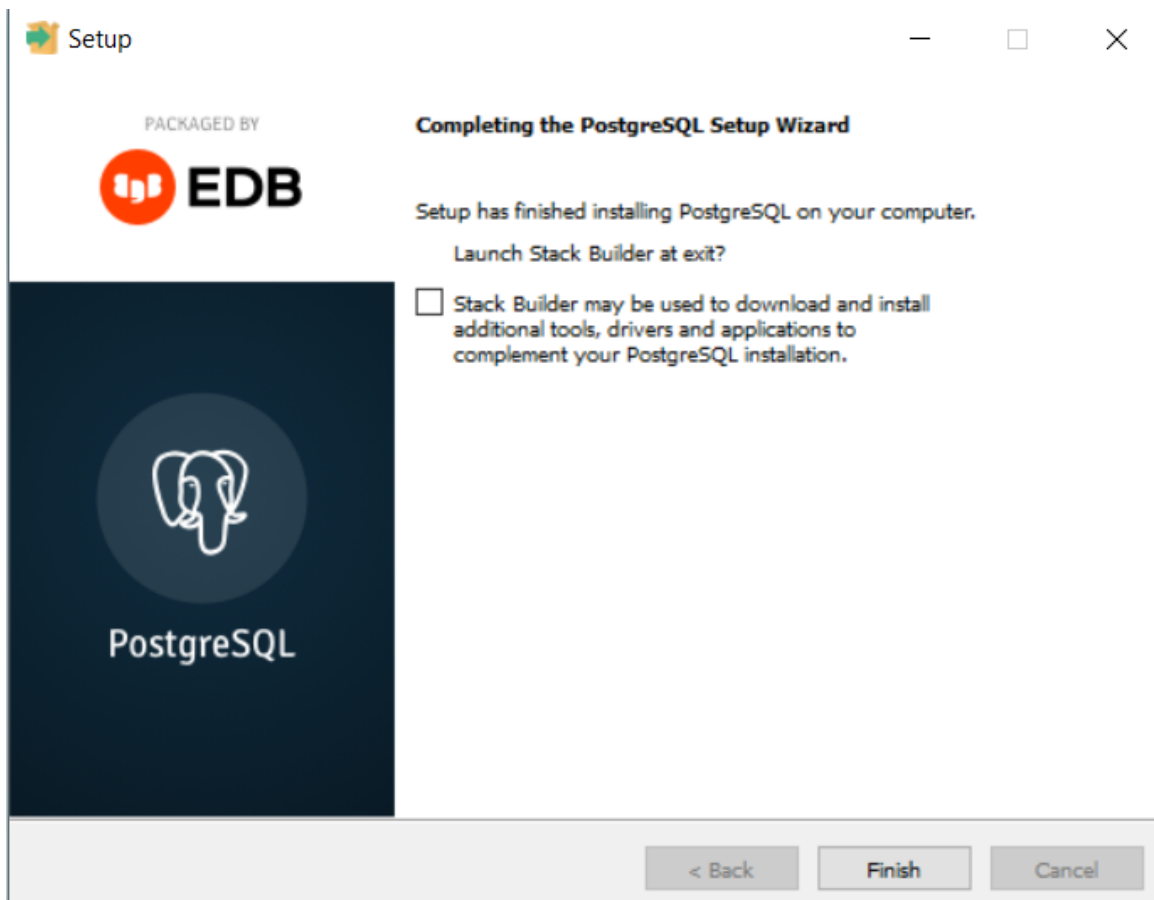
7. Check the pre-installation summary, and click Next.



8. Installation begins.



9. Uncheck the stack builder option. We will use Stack Builder in more advanced tutorials, click Finish.



Problem 2

State where and how to store information in postgresQL DBSM?

article

Storing Information in PostgreSQL DBMS

In PostgreSQL DBMS, information is stored in a hierarchical structure consisting of databases, schemas, tables, and columns. Here's a breakdown of how and where information is stored:

Databases

PostgreSQL allows you to create multiple databases within a single PostgreSQL server instance. Each database acts as a separate container for data. You can create, delete, and manage databases using SQL commands or graphical tools.

Schemas

Within each database, you can create multiple schemas. Schemas are like namespaces that organize database objects such as tables, views, functions, and more. They provide a way to logically group related objects and avoid naming conflicts.

Tables

Tables are the primary storage objects in PostgreSQL. They consist of rows and columns, similar to a spreadsheet. Each table represents a specific entity or concept in your application. You define the structure of a table by specifying column names, data types, constraints, and indexes.

Columns

Columns define the individual fields or attributes within a table. Each column has a name and a data type that determines the kind of data it can store, such as text, numbers, dates, or binary data. You can also specify constraints on columns to enforce data integrity rules.

Rows

Rows represent individual records or instances within a table. Each row contains data values corresponding to the columns defined in the table's structure. You can insert, update, and delete rows to manipulate the data stored in a table.

Indexes

PostgreSQL allows you to create indexes on columns to improve query performance. Indexes are data structures that enable faster data retrieval by creating a sorted copy of selected columns. They speed up search operations but require additional storage space and maintenance overhead.

PostgreSQL provides a flexible and extensible storage model, allowing you to define complex data structures, relationships, and constraints. It also supports advanced features like transactions, views, triggers, and more. The exact location of the data files and configuration settings may vary depending on your PostgreSQL installation and operating system.

With the code below we can check the location of data in which it is saved.

```
postgres=# show data_directory;
           data_directory
-----
C:/Program Files/PostgreSQL/16/data
(1 row)
```

Problem 3

on PostgreSQL DBSM Create a database. Describe the creation commands and default settings of this database and its contents.

This command creates a new database named "homework1".

```
1 CREATE DATABASE homework1;
```

Listing 1:

Default settings and contents of the newly created database:

1. Character Set and Collation: By default, the database inherits the character set and collation settings from the template database used during the creation. The template database is typically "template1". These settings determine the encoding and sorting rules for text data in the database.

2. Access Privileges: The creator of the database becomes the owner and is granted all privileges on the database. Other users may need to be granted specific privileges to access or modify the database objects.

3. Schema: The newly created database contains a default schema called "public". This schema is created automatically and serves as the default namespace for database objects. We can create additional schemas within the database to organize our objects further.

4. Tables and Data: Initially, the database does not contain any tables or data. We can create tables and populate them with data using the 'CREATE TABLE' and 'INSERT' commands, respectively.

5. Extensions: PostgreSQL allows us to install extensions that provide additional functionality to the database. By default, the newly created database does not have any extensions installed. We can install extensions using the 'CREATE EXTENSION' command.

6. Configuration: The configuration settings of the new database are inherited from the template database. These settings control various aspects of the database's behavior, such as memory allocation, connection limits, and query optimization. We can modify these settings as per our requirements.

Problem 4

In this database, create two arbitrary tables with the following conditions: each table has 5 columns, one of which is numeric and non-repeating, one column with a text value, one column with a logical value, and one column with a date value. In one table, create a column with the possibility of saving the file and in the other, create a column with a numerical value in which the values that can be recorded must exist in the numerical column of the other table. Mention the commands to create these tables in the report.

Commands to create this dataset with required tables are written in the "script.sql" file, which is attached in the uploaded zip file.

```

postgres=# \l
                                List of
Name | Owner | Encoding | Locale Provider | Collate
-----+-----+-----+-----+-----
postgres | postgres | UTF8 | libc | English_United States.12
template0 | postgres | UTF8 | libc | English_United States.12
template1 | postgres | UTF8 | libc | English_United States.12
(3 rows)

postgres=# CREATE DATABASE homework1;
CREATE DATABASE
postgres=# \connect homework1
You are now connected to database "homework1" as user "postgres".
homework1=# CREATE TABLE IF NOT EXISTS authors (
homework1(#      id integer UNIQUE,
homework1(#      full_name varchar(255),
homework1(#      sex boolean,
homework1(#      birth_date date,
homework1(#      avatar bytea
homework1(# );
CREATE TABLE
homework1=#
homework1=# CREATE TABLE IF NOT EXISTS books (
homework1(#      id integer UNIQUE,
homework1(#      title varchar(255),
homework1(#      is_published boolean,
homework1(#      created_at date,
homework1(#      author_id integer REFERENCES authors(id)
homework1(# );
CREATE TABLE
homework1=# \dt
                List of relations
Schema | Name  | Type  | Owner
-----+-----+-----+-----
public | authors | table | postgres
public | books  | table | postgres
(2 rows)

homework1=#

```

```

1 CREATE DATABASE homework1;

```

```

2

```

```
3 CREATE TABLE IF NOT EXISTS authors (  
4     id integer UNIQUE,  
5     full_name varchar(255),  
6     sex boolean,  
7     birth_date date,  
8     avatar bytea  
9 );  
10  
11 CREATE TABLE IF NOT EXISTS books (  
12     id integer UNIQUE,  
13     title varchar(255),  
14     is_published boolean,  
15     created_at date,  
16     author_id integer REFERENCES authors(id)  
17 );
```

Listing 2:

Problem 5

Insert ten million rows in each of these tables. The insertion method should be mentioned.

I have inserted rows like those below using Python:

```
1 import psycopg2  
2 from faker import Faker  
3  
4  
5 def _create_authors_instances(count):  
6     fake = Faker()  
7     with open("temp.txt", "rb") as file:  
8         binary_data = file.read()  
9     instances = [  
10         (  
11             i,  
12             fake.name(),  
13             fake.boolean(),  
14             fake.date(),  
15             binary_data  
16         )  
17         for i in range(1, 1 + count)  
18     ]  
19     return instances  
20  
21  
22 def _create_books_instances(count):
```

```

23     fake = Faker()
24     instances = [
25         (
26             i,
27             fake.catch_phrase(),
28             fake.boolean(),
29             fake.date(),
30             10 ** 6 - i + 1
31         )
32         for i in range(1, 1 + count)
33     ]
34
35     return instances
36
37
38 def main():
39     sample_count = 10 ** 6
40     conn = psycopg2.connect(database="homework1", user="postgres"
41                             , password="*****", host="localhost", port="5432")
42     cursor = conn.cursor()
43
44     authors = _create_authors_instances(sample_count)
45     cursor.executemany("INSERT INTO authors (id, full_name, sex,
46                         birth_date, avatar) VALUES (%s, %s, %s, %s, %s)", authors)
47     conn.commit()
48     print("authors don!")
49     books = _create_books_instances(sample_count)
50     cursor.executemany("INSERT INTO books (id, title,
51                         is_published, created_at, author_id) VALUES (%s, %s, %s, %s, %s)", books)
52     conn.commit()
53     print("books don!")
54     conn.close()
55
56 if __name__ == '__main__':
57     main()

```

Listing 3:

Following psql commands shows that all rows have been added correctly to the tables.

```

postgres=# \connect homework1
You are now connected to database "homework1" as user "postgres".
homework1=# SELECT COUNT(*) FROM books;
count
-----
1000000
(1 row)

homework1=# SELECT COUNT(*) FROM authors;
count
-----
1000000
(1 row)

homework1=# SELECT * FROM authors;
 id |          full_name          | sex | birth_date |
-----+-----+-----+-----+
  1 | Jason Moore                 | t   | 2000-11-15 |
  2 | Kaitlin Burke               | f   | 2003-08-13 |
  3 | Anthony Dixon               | f   | 1982-09-24 |
  4 | Thomas Henry                | f   | 1995-07-21 |
  5 | Laura Williams              | t   | 2017-01-04 |
  6 | Matthew Weaver              | t   | 1980-11-19 |
  7 | Brent Scott                 | t   | 1986-07-03 |
  8 | Tracy Barton                | t   | 1972-04-04 |
  9 | Brenda Price                | t   | 1980-11-22 |
 10 | James Cantu                 | t   | 1979-11-25 |
 11 | Jeffery Brooks              | t   | 1985-01-21 |
 12 | Lisa Bell                   | t   | 1988-03-30 |
^CCancel request sent
homework1=# SELECT * FROM books;
 id | title | is_published | created_at | author_id |
-----+-----+-----+-----+
  1 | Sharable solution-oriented software | t | 1970-03-15 | 1000000 |
  2 | User-centric content-based solution | f | 1991-11-11 | 999999 |
  3 | Organized dynamic collaboration | t | 1985-12-22 | 999998 |
  4 | Exclusive mobile matrices | t | 1970-09-24 | 999997 |
  5 | Inverse didactic installation | f | 1987-02-16 | 999996 |
  6 | Digitized mission-critical approach | t | 1981-11-15 | 999995 |
  7 | Synergized neutral task-force | f | 2011-04-15 | 999994 |
  8 | Public-key optimal leverage | t | 1993-03-07 | 999993 |
  9 | Realigned mobile standardization | t | 2013-01-14 | 999992 |
 10 | Versatile content-based focus group | f | 1985-12-15 | 999991 |
 11 | Persistent fresh-thinking challenge | f | 2017-02-18 | 999990 |

```

Problem 6

Define a separate user to access each of these tables and ensure the correct operation of this access definition. Mention the necessary instructions and the method of obtaining assurance.

Connect to our database using an account with administrative privileges.

Then create two users to access each table, using the command below:

```

1 CREATE USER user01 WITH PASSWORD 'p1';
2 CREATE USER user02 WITH PASSWORD 'p2';

```

Listing 4:

And now grant the appropriate privileges to each user for their respective tables using the GRANT command. We can do as follows:

```

1 GRANT SELECT, INSERT, UPDATE, DELETE ON authors TO
  user01;

```

```
2 | GRANT SELECT, INSERT, UPDATE, DELETE ON books TO user02  
  | ;
```

Listing 5:

I have done it on psql shell and it is as follows:

```
postgres=# \connect homework1  
You are now connected to database "homework1" as user "postgres".  
homework1=# create user user01 with password 'p1';  
CREATE ROLE  
homework1=# create user user02 with password 'p2';  
CREATE ROLE  
  
homework1=# grant select, insert, update, delete on authors to user01;  
GRANT  
homework1=# grant select, insert, update, delete on books to user02;  
GRANT  
homework1=#
```

Finally, to ensure the correct operation of the access definition for each user, we can test the access for each user by connecting to the database using their credentials and performing the allowed operations on their respective tables.

First, connect to the database with user1 using the command below:

```
1 | psql -U user1 -d homework1;
```

Listing 6:

Once connected, we can perform the allowed operations on the respective table. I have done it and see that The created accesses are established.

```
C:\Program Files\PostgreSQL\16\pgAdmin 4\web\pgadmin\tools\psql>cd C:\Program Files\PostgreSQL\16\bin
C:\Program Files\PostgreSQL\16\bin>psql -U user01 -d homework1 -W
Password:
psql (16.0)
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.

homework1=> select * from authors;
 id | full_name | sex | birth_date |
-----+-----+-----+-----+
  1 | Jason Moore | t | 2000-11-15 |
  2 | Kaitlin Burke | f | 2003-08-13 |
  3 | Anthony Dixon | f | 1982-09-24 |
  4 | Thomas Henry | f | 1995-07-21 |
  5 | Laura Williams | t | 2017-01-04 |
  6 | Matthew Weaver | t | 1980-11-19 |
  7 | Brent Scott | t | 1986-07-03 |
  8 | Tracy Barton | t | 1972-04-04 |
  9 | Brenda Price | t | 1980-11-22 |
 10 | James Cantu | t | 1979-11-25 |
 11 | Jeffery Brooks | t | 1985-01-21 |
 12 | Lisa Bell | t | 1988-03-30 |
 13 | Martin Morgan | f | 1997-10-03 |
homework1=> select * from books;
ERROR: permission denied for table books
homework1=>
```

```
C:\Program Files\PostgreSQL\16\bin>psql -U user02 -d homework1 -W
Password:
psql (16.0)
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.

homework1=> select * from authors;
ERROR: permission denied for table authors
homework1=> select * from books;
 id | title | is_published | created_at | author_id |
-----+-----+-----+-----+-----+
  1 | Sharable solution-oriented software | t | 1970-03-15 | 1000000 |
  2 | User-centric content-based solution | f | 1991-11-11 | 999999 |
  3 | Organized dynamic collaboration | t | 1985-12-22 | 999998 |
  4 | Exclusive mobile matrices | t | 1970-09-24 | 999997 |
  5 | Inverse didactic installation | f | 1987-02-16 | 999996 |
  6 | Digitized mission-critical approach | t | 1981-11-15 | 999995 |
  7 | Synergized neutral task-force | f | 2011-04-15 | 999994 |
  8 | Public-key optimal leverage | t | 1993-03-07 | 999993 |
  9 | Realigned mobile standardization | t | 2013-01-14 | 999992 |
 10 | Versatile content-based focus group | f | 1985-12-15 | 999991 |
 11 | Persistent fresh-thinking challenge | f | 2017-02-18 | 999990 |
 12 | Operative multimedia frame | t | 1999-10-17 | 999989 |
 13 | Fundamental foreground paradigm | f | 1971-08-13 | 999988 |
 14 | Innovative global support | f | 1981-03-19 | 999987 |
 15 | Optimized impactful structure | t | 2019-04-21 | 999986 |
 16 | Focused composite application | f | 2013-05-30 | 999985 |
```

Problem 7

Backup the created database and restore it under another database name. Mention the necessary instructions.

First, using command below we can make a back up file, that stores our database:


```
1 pg_dump -U postgres -d homework1 -f C:\Users\acer\Desktop\db\
  dump_file.sql
```

Listing 7:

```
C:\Program Files\PostgreSQL\16\bin>pg_dump -U postgres -d homework1 -f C:\Users\acer\Desktop\db\dump_file.sql
Password:
```

Second, create new database using command below:

```
1 create database back_up_homework1;
```

Listing 8:

```
homework1=# create database back_up_homework1;
CREATE DATABASE
homework1=# \q
```

Finally, using command below we can restore data saves in backed up file:

```
1 psql -U postgres -d back_up_homework1 -f C:\Users\acer\
  Desktop\db\dump_file.sql
```

Listing 9:

```
C:\Program Files\PostgreSQL\16\bin>psql -U postgres -d back_up_homework1 -f C:\Users\acer\Desktop\db\dump_file.sql
Password for user postgres:
SET
SET
SET
SET
SET
set_config
-----
(1 row)

SET
SET
SET
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
COPY 1000000
COPY 1000000
ALTER TABLE
ALTER TABLE
ALTER TABLE
GRANT
GRANT
GRANT
GRANT
```

Now, we can check whether it does correctly added to the new database.

```

C:\Program Files\PostgreSQL\16\bin>psql -U postgres -d back_up_homework1 -W
Password:
psql (16.0)
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.

back_up_homework1=# \dt
               List of relations
 Schema | Name  | Type  | Owner
-----+-----+-----+-----
 public | authors | table | postgres
 public | books  | table | postgres
(2 rows)

back_up_homework1=# select count(*) from authors;
 count
-----
 1000000
(1 row)

back_up_homework1=# select * from authors;
 id | full_name | sex | birth_date |
-----+-----+-----+-----+-----
  1 | Jason Moore | t   | 2000-11-15 | \x6869696969696969
  2 | Kaitlin Burke | f   | 2003-08-13 | \x6869696969696969
  3 | Anthony Dixon | f   | 1982-09-24 | \x6869696969696969
  4 | Thomas Henry | f   | 1995-07-21 | \x6869696969696969
  5 | Laura Williams | t   | 2017-01-04 | \x6869696969696969
  6 | Matthew Weaver | t   | 1980-11-19 | \x6869696969696969
  7 | Brent Scott | t   | 1986-07-03 | \x6869696969696969
  8 | Tracy Barton | t   | 1972-04-04 | \x6869696969696969
  9 | Brenda Price | t   | 1980-11-22 | \x6869696969696969
 10 | James Cantu | t   | 1979-11-25 | \x6869696969696969
 11 | Jeffery Brooks | t   | 1985-01-21 | \x6869696969696969
 12 | Lisa Bell | t   | 1988-03-30 | \x6869696969696969
 13 | Martin Morgan | f   | 1997-10-03 | \x6869696969696969
 14 | Megan Sullivan | t   | 2023-05-11 | \x6869696969696969
 15 | Shane Coleman | t   | 1972-09-06 | \x6869696969696969
 16 | Nancy Gallagher | f   | 1972-06-10 | \x6869696969696969
 17 | Lisa Johnson | f   | 1992-07-10 | \x6869696969696969
 18 | Melissa Martin | f   | 1981-01-02 | \x6869696969696969

```