**ASSIGNMENT-4**

**ART GALLERY MANAGEMENT DATABASE MANAGEMENT SYSTEM**

SUBMITTED BY:

-DEEKSHITHA R PES2UG19CS104

-DEEPA SHREE C V PES2UG19CS105

**Department of Computer Science and Engineering**

**Front End for the execution of queries:**

* Python is used to implement the front end.
* It is easy to use and understand and given that we are familiar with the language, we decided to opt python.
* Python also supports an API to work on psql databases, psycopg2. The library is very convenient to use and has a wide variety of functions which makes it easier to work with.
* Psycopg2 is a [PostgreSQL](https://www.postgresql.org/) adapter for the [Python](https://www.python.org/) programming language. It is a wrapper for the [libpq](https://www.postgresql.org/docs/current/static/libpq.html), the official PostgreSQL client library.  The quickest way to install Psycopg is using the [wheel](https://pythonwheels.com/) package available on [PyPI](https://pypi.org/project/psycopg2-binary/). ‘pip install pyscopg2’ is also one of the ways to install the package.

**Code for the front-end:**

import psycopg2

def users\_choice():

    # the user can enter a number from 1-6 to get the required data

    print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

    print("1.Obtain the last name and the email of all the customers who have bought at least one painting")

    print("2.To get the details of all the paintings that have been sold in auction")

    print("3. To get the number of instalments and the total amount of of each customer")

    print("4. To get the paintings that have been exhibited atleast once")

    print("5.To get artists with paintings in the gallery")

    print("6.To get employees from a given department number")

    print("Press 0 to exit from the terminal")

    print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

    #input from the terminal

    ch = int(input("Enter your query of choice: "))

    print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

    return ch

#try except block to connect to the db

try:

    connection = psycopg2.connect (host="localhost",

    database="gallery",

    user="postgres",

    password="dee!!828")

    cursor = connection.cursor()

    #executing the queries according to the users choice

    ch=1

    while(ch!=0):

        ch = users\_choice()

        if ch==1:

            query = "select distinct c.c\_lname,c.c\_email from customer as c where exists (select p.c\_id from painting as p where c.c\_id = p.c\_id);"

            cursor.execute(query)

            records = cursor.fetchall()

            print("Required output:")

            print("------------------")

            for row in records:

                print("Last name:" ,row[0],end=" || ")

                print("Email: ",row[1])

                print("------------------")

        elif ch==2:

            query = "select distinct p.p\_name,p.p\_price from painting as p, auction as a where exists (select a.p\_id from auction as a where a.p\_id=p.p\_id and a.price\_fetched >0);"

            cursor.execute(query)

            records = cursor.fetchall()

            print("Required output:")

            print("------------------")

            for row in records:

                print("Painting Name:",row[0],end=" || ")

                print("Price:",row[1])

                print("------------------")

        elif ch==3:

            query = "select c\_id,p\_id, count(c\_id),sum(amount) from instalments group by c\_id,p\_id order by c\_id;"

            cursor.execute(query)

            records = cursor.fetchall()

            print("Required output:")

            print("------------------")

            for row in records:

                print("Customer\_id:",row[0],end=" || ")

                print("Painting\_id:",row[1],end=" || ")

                print("Count:",row[2],end=" || ")

                print("Total:",row[3])

                print("------------------")

        elif ch==4:

            query = "select p.p\_id,p.p\_name,p.p\_price,ex.ex\_id from painting as p inner join exhibited\_in as ex on ex.p\_id = p.p\_id;"

            cursor.execute(query)

            records = cursor.fetchall()

            print("Required output:")

            print("------------------")

            for row in records:

                print("Painting\_id:",row[0],end=" || ")

                print("Painting\_name:",row[1],end=" || ")

                print("Painting Price:",row[2],end=" || ")

                print("Exhibition\_id:",row[3])

                print("------------------")

        #5 and 6 are cursors implemented as a part of the previous assignment

        elif ch==5:

            query = "select \* from artist as a where exists (select p.a\_id from painting as p where p.a\_id = a.a\_id);"

            cursor.execute(query)

            records = cursor.fetchall()

            for r1 in records:

                #print(r1)

                print("Artist\_fname:",r1[1],end=" || ")

                print("Artist\_lname:",r1[3],end=" || ")

                print("Email\_id:",r1[5])

                print("------------------")

        elif ch==6:

            dep\_no = int(input("Enter department number:"))

            cursor.execute("SELECT get\_employee\_by\_dept(%s)",[dep\_no])

            rows = cursor.fetchall()

            print("(Employee\_fname,Employee\_lname,Salary)")

            print("------------------")

            for r1 in rows:

                print(r1)

    #incase of an incorrect entry, prompt the user to enter the

        else:

            if(ch==0):

                break

            else:

                print("Please enter the correct choice")

#if there's an error throw the exception

except (Exception, psycopg2.Error) as error:

    print("Error while fetching data from PostgreSQL", error)

finally:

    # closing database connection.

    if connection:

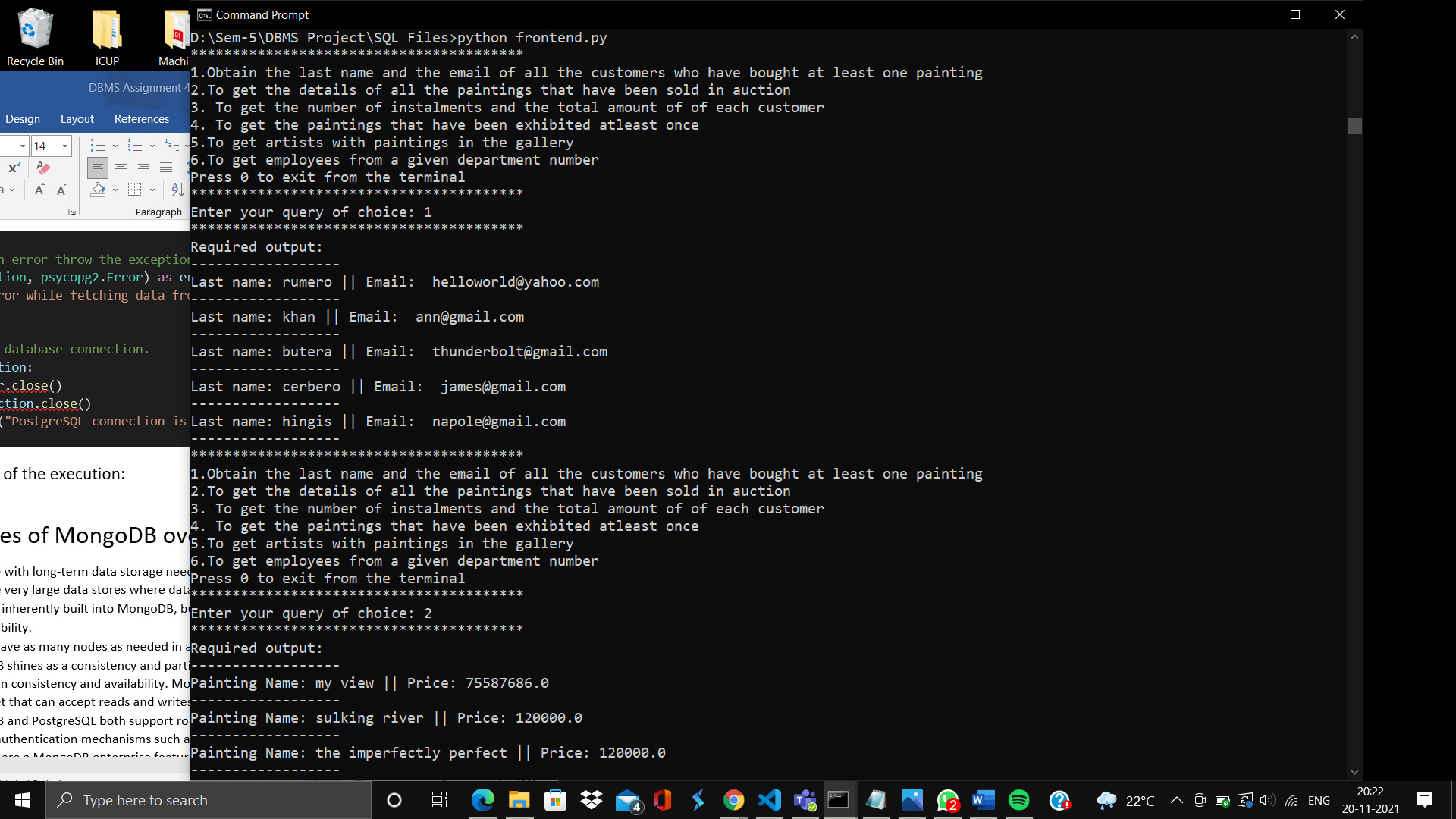
        cursor.close()

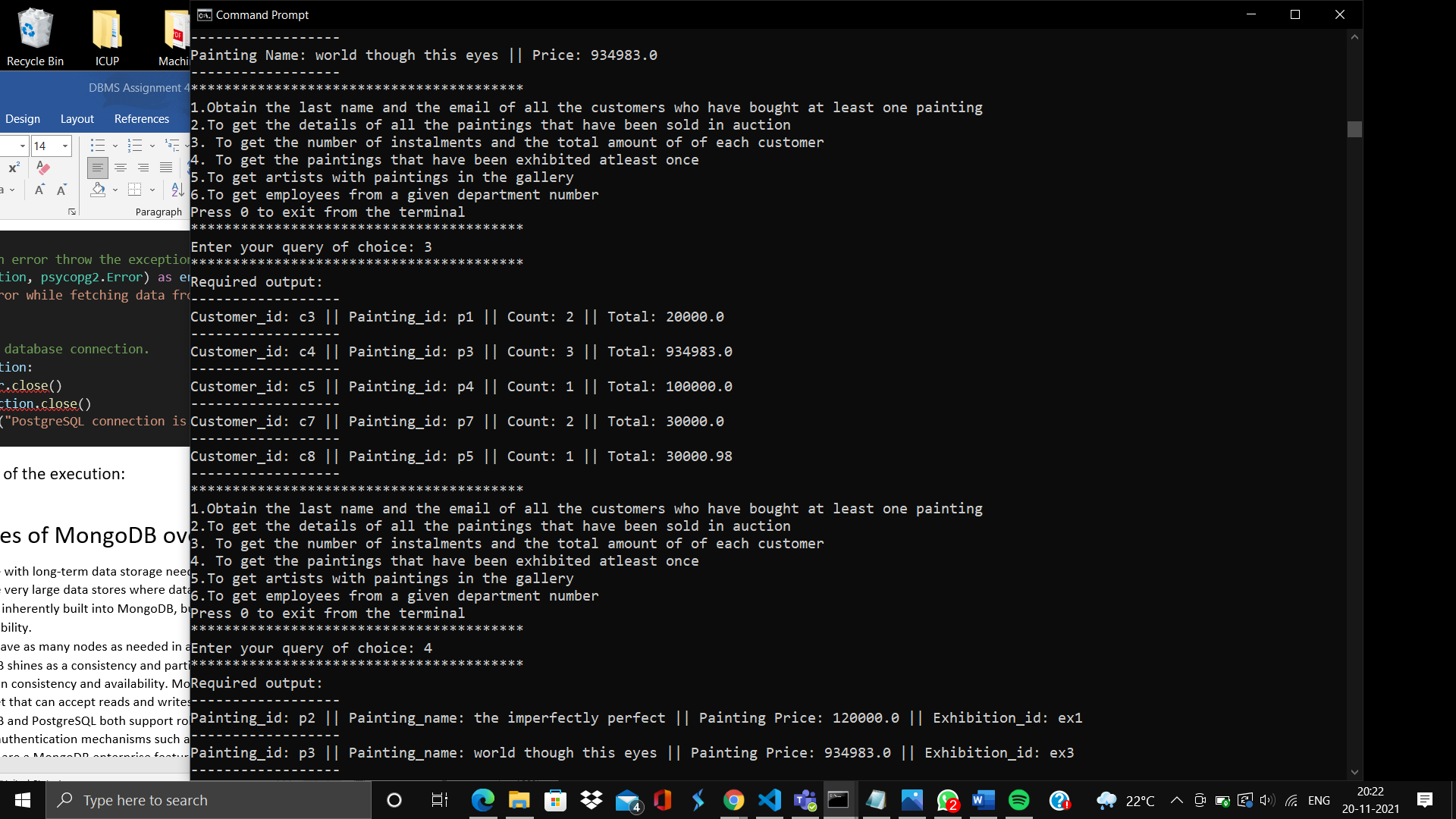
        connection.close()

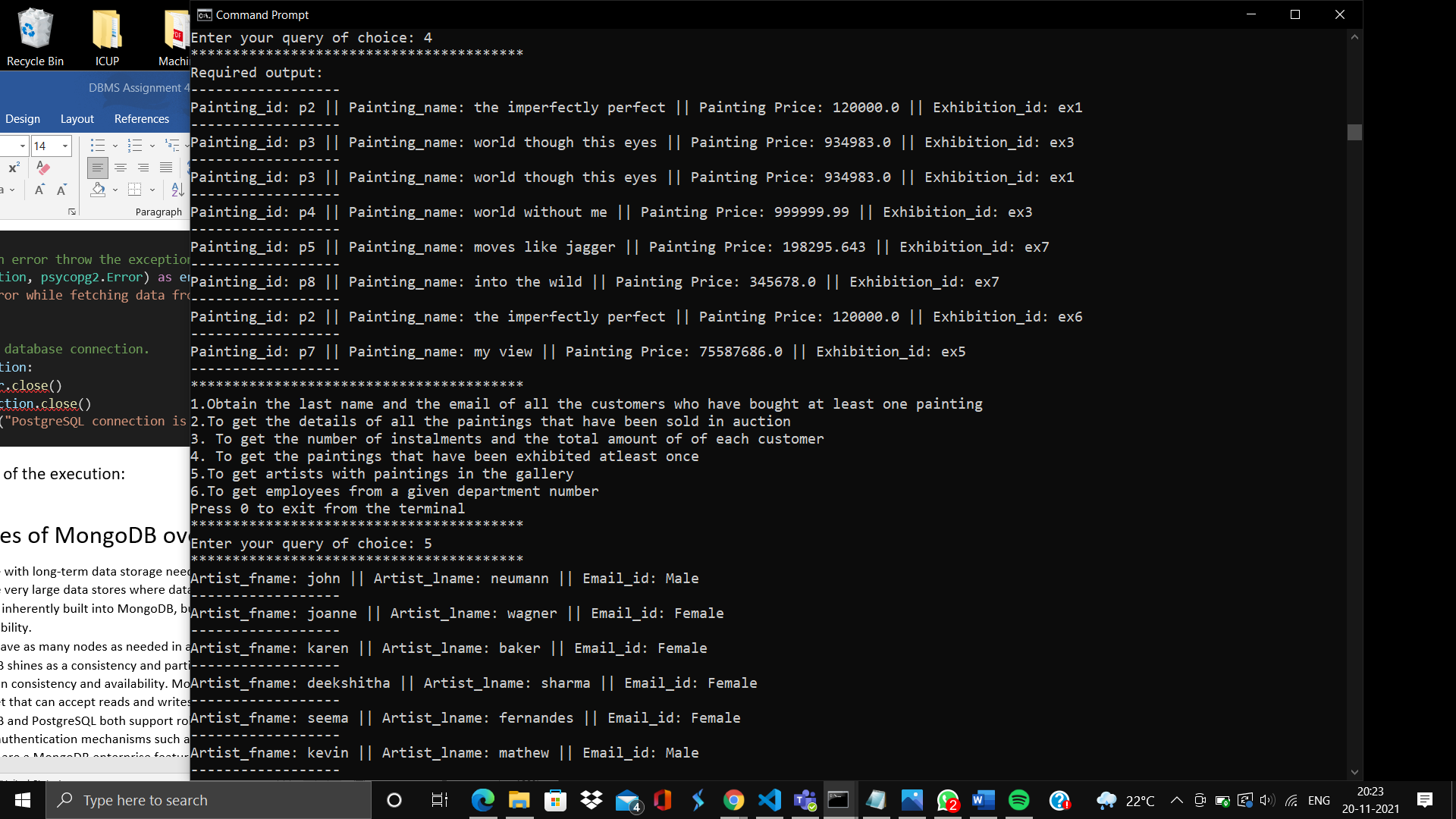
        print("PostgreSQL connection is closed")

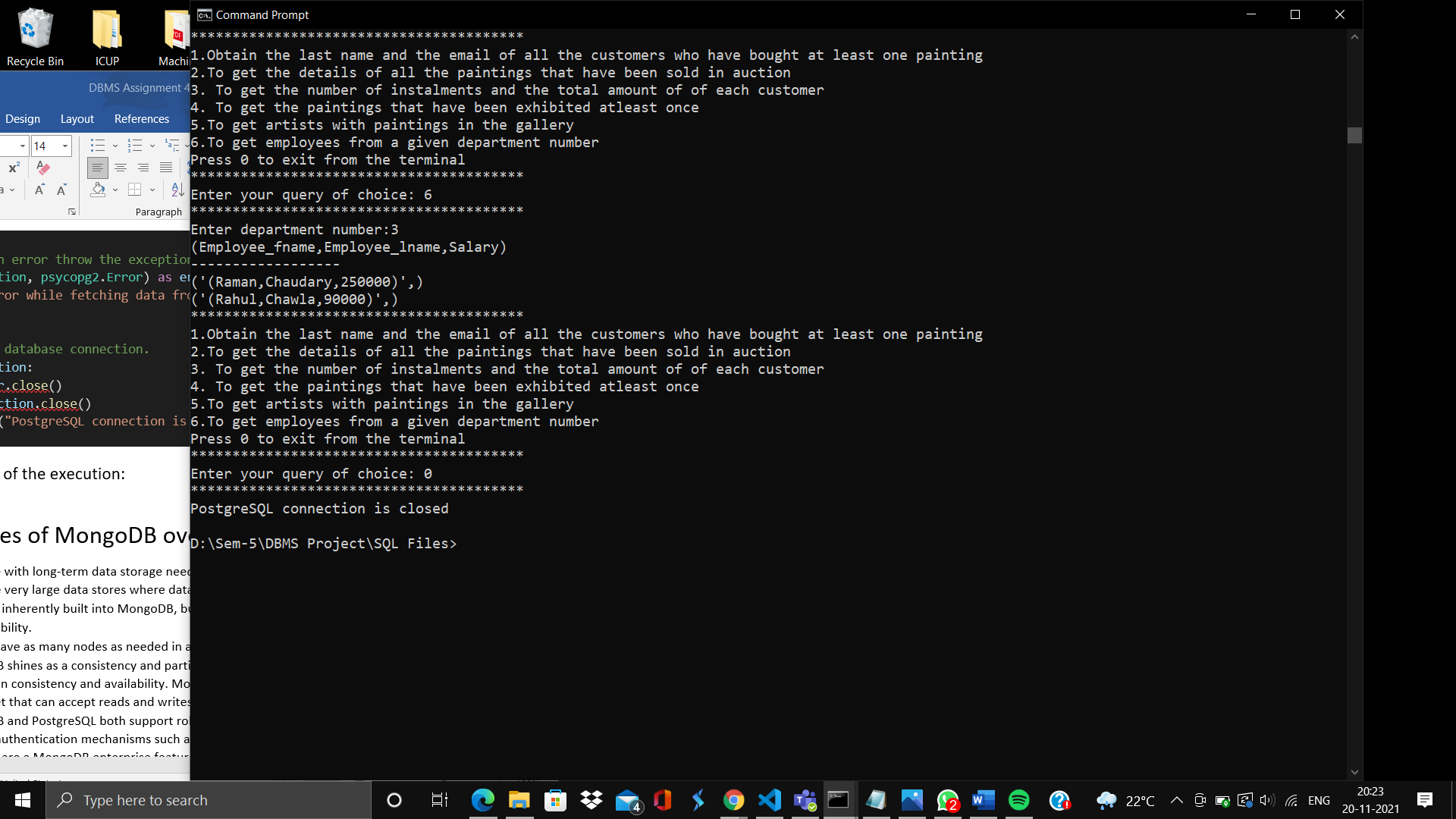
**Output of the execution:**

All the implemented queries are executed here. On entering a number between 1 and 6, the desired query is executed. On entering an invalid number other than these, the terminal prompts the user to enter a valid number. If zero is entered, then the program is terminated and the connection to the database is closed.





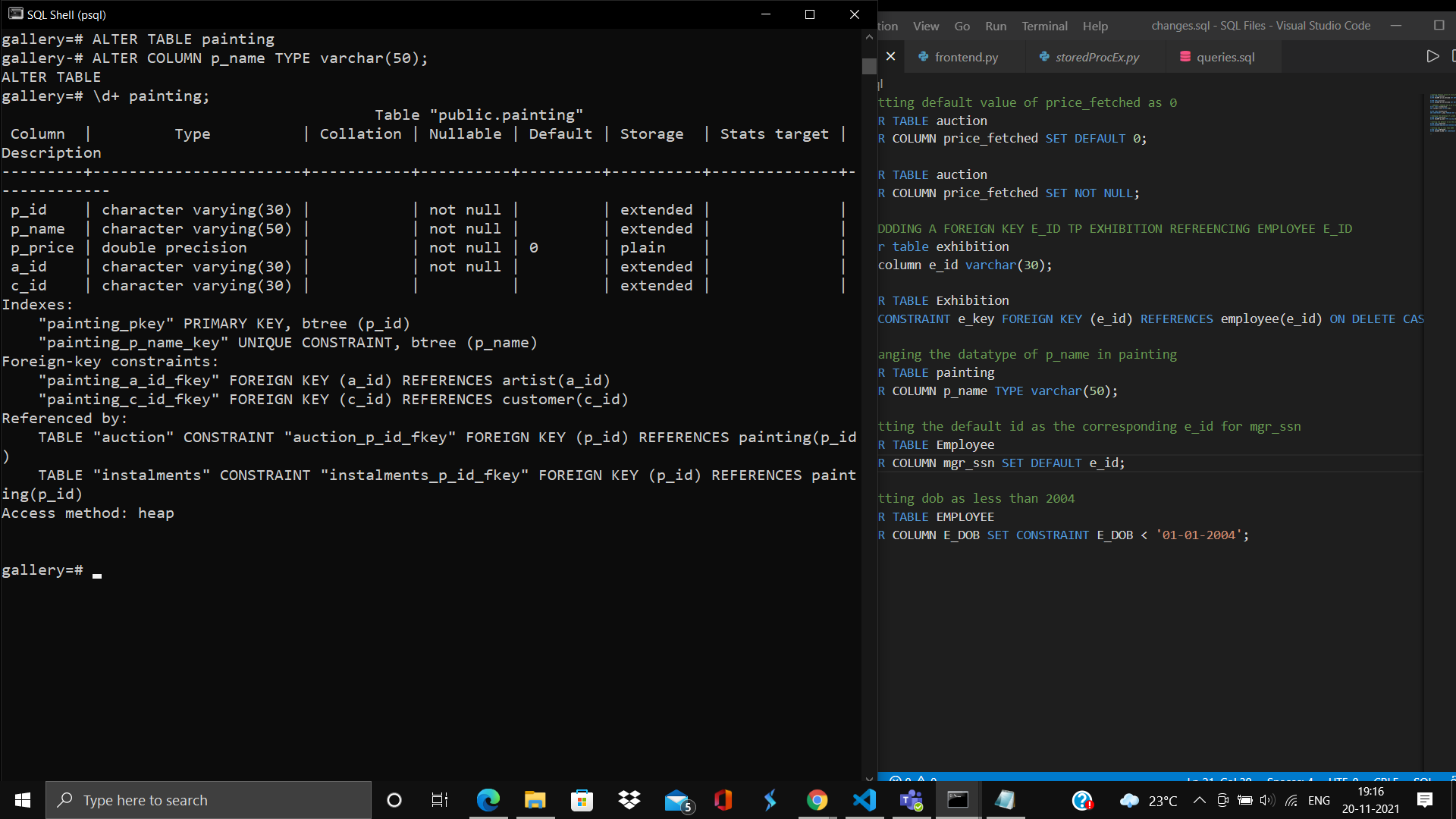




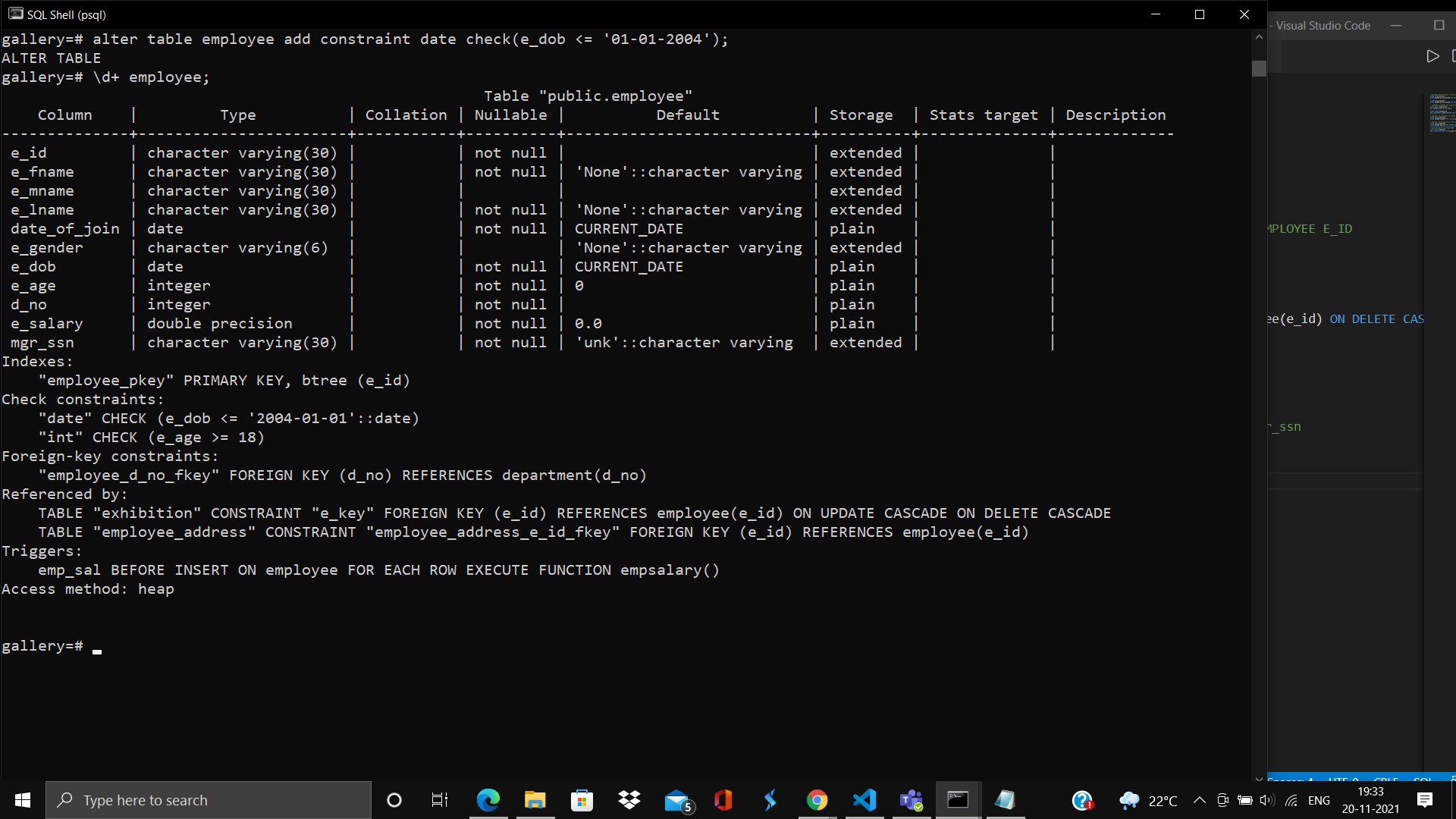
**Changes to the database:**

1. Altering the datatype of the painting\_name of painting relation.

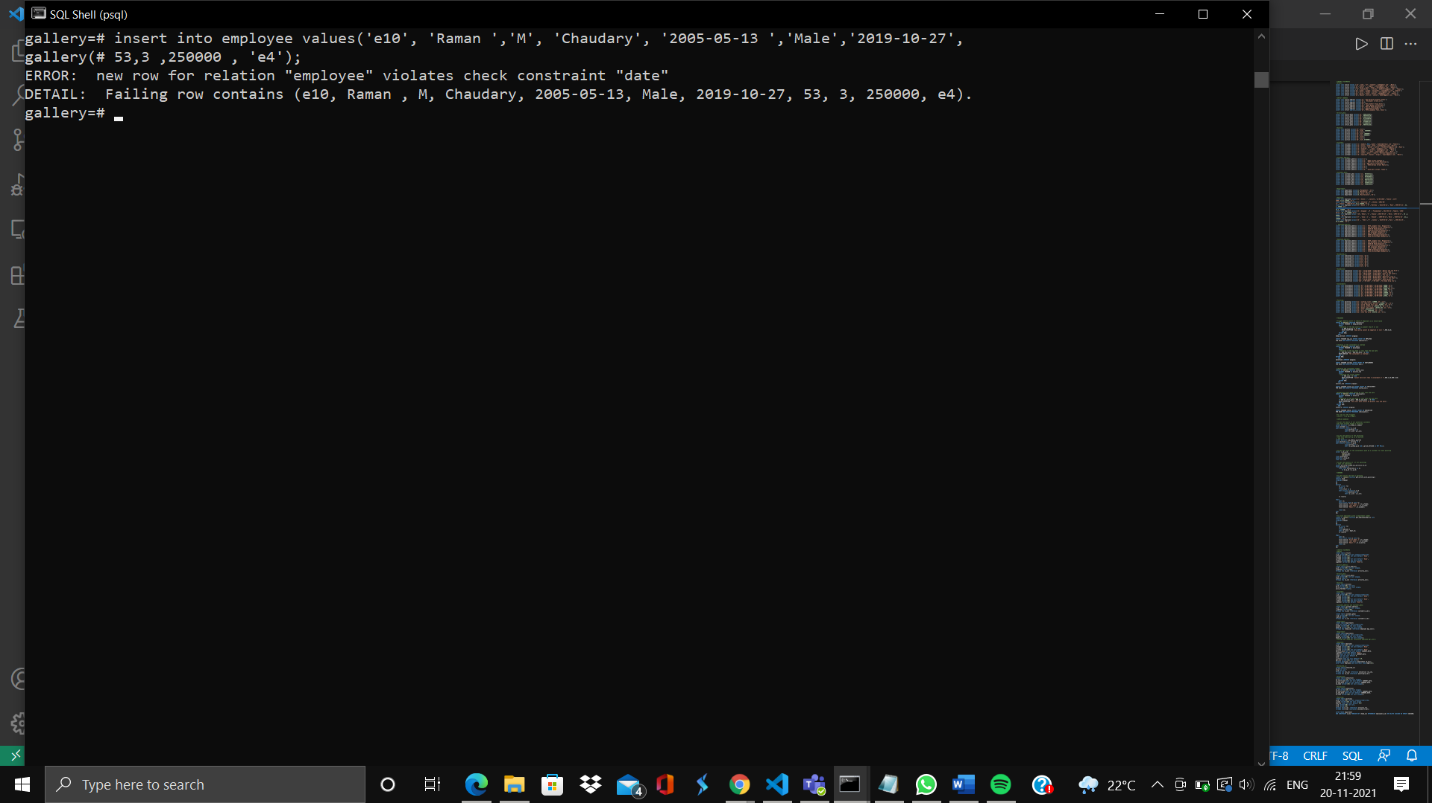
The length of the field is changed from 30 to 50. This changes the constraints on the relation painting of the database.



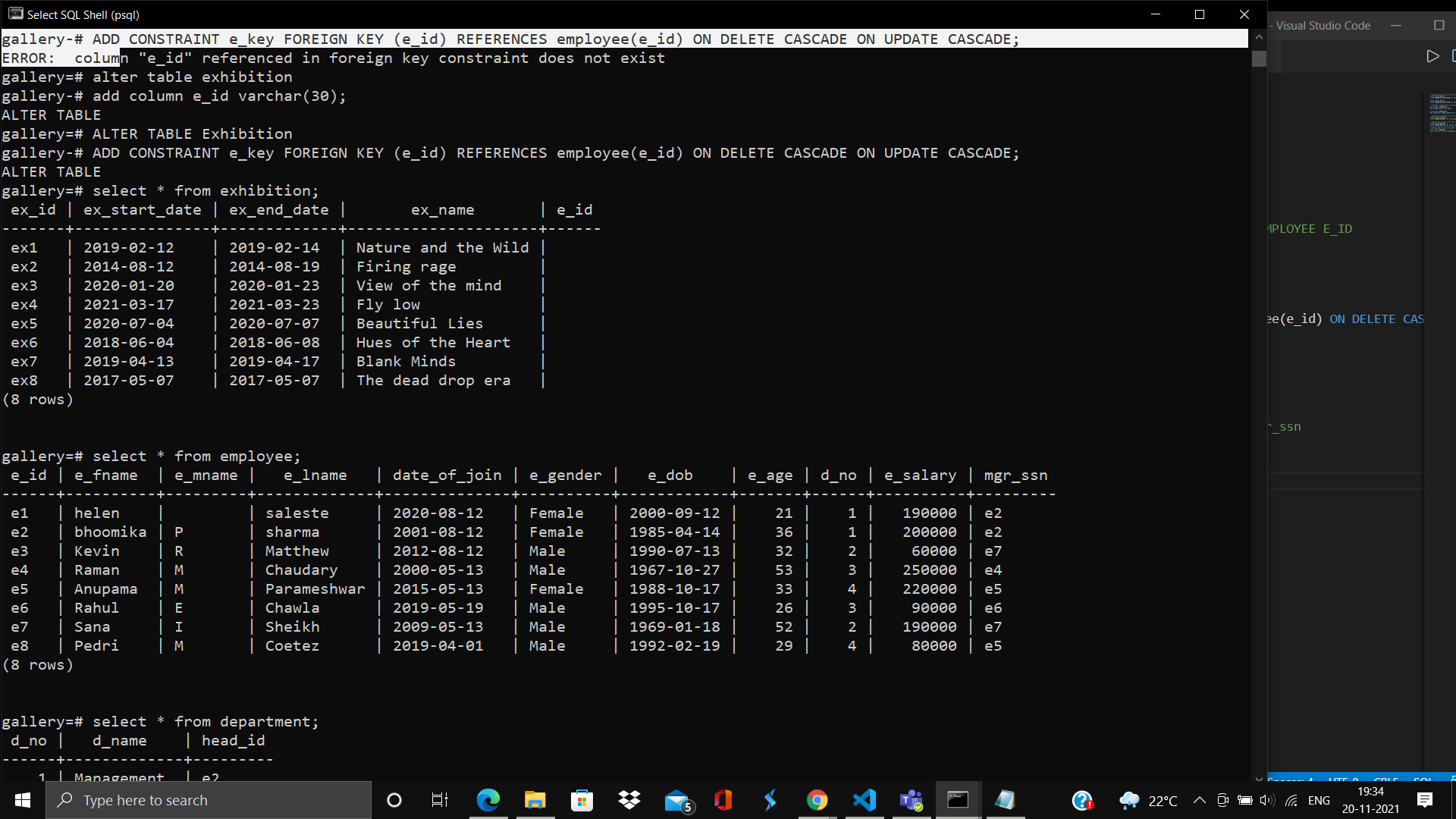
1. Setting a constraint on the e\_dob of employee relation. The dob cannot contain a year greater than 2004.



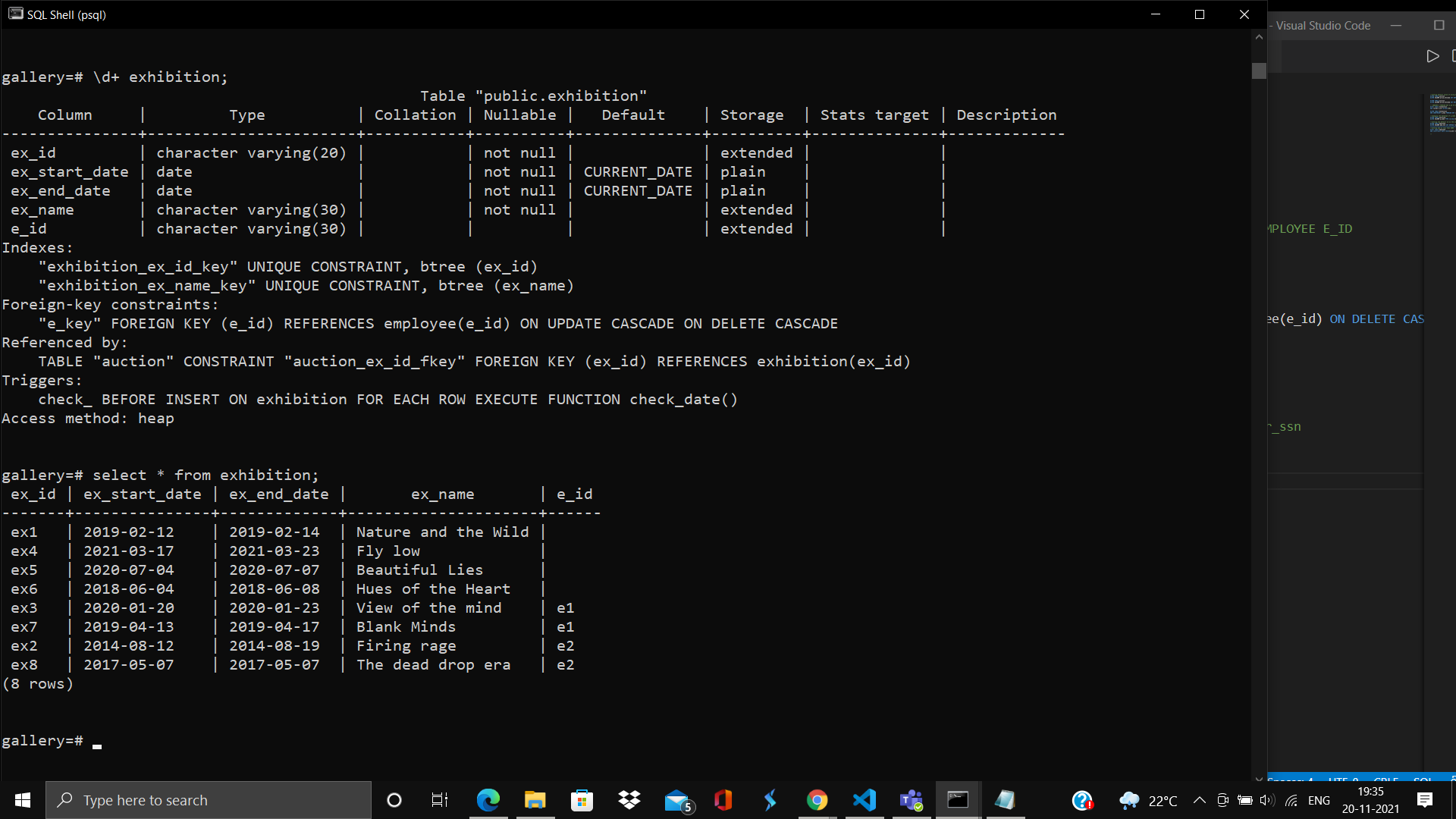
Inserting an invalid value throws an exception.



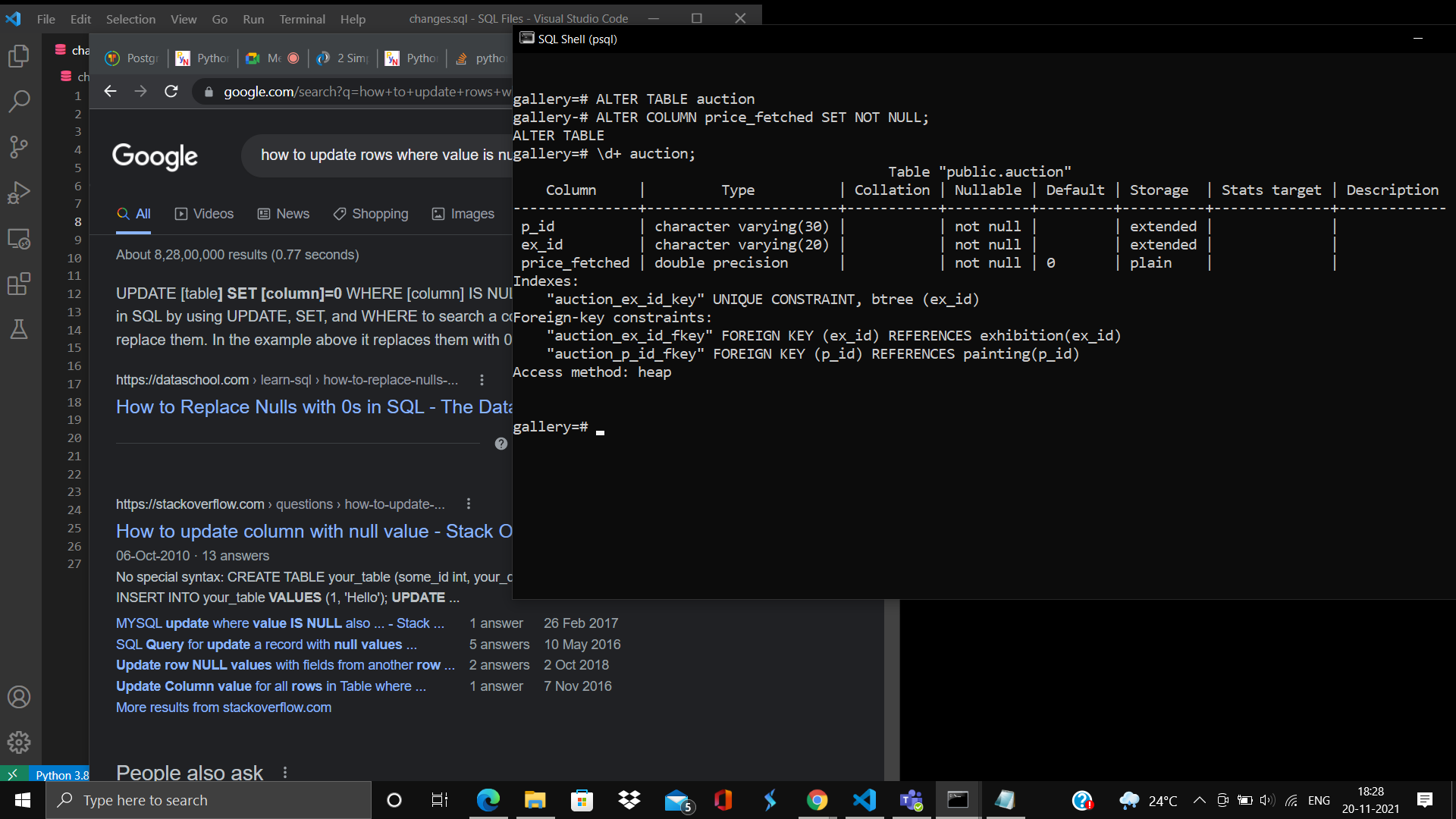
3 . Setting a manager for the exhibition. A foreign key e\_id is set from exhibition to employee e\_id. This changes the schema of exhibition. A new foreign key is inserted into the table.



Inserting a few records to ensure the validity of the constraint. The employees from management department are added as exhibition managers.

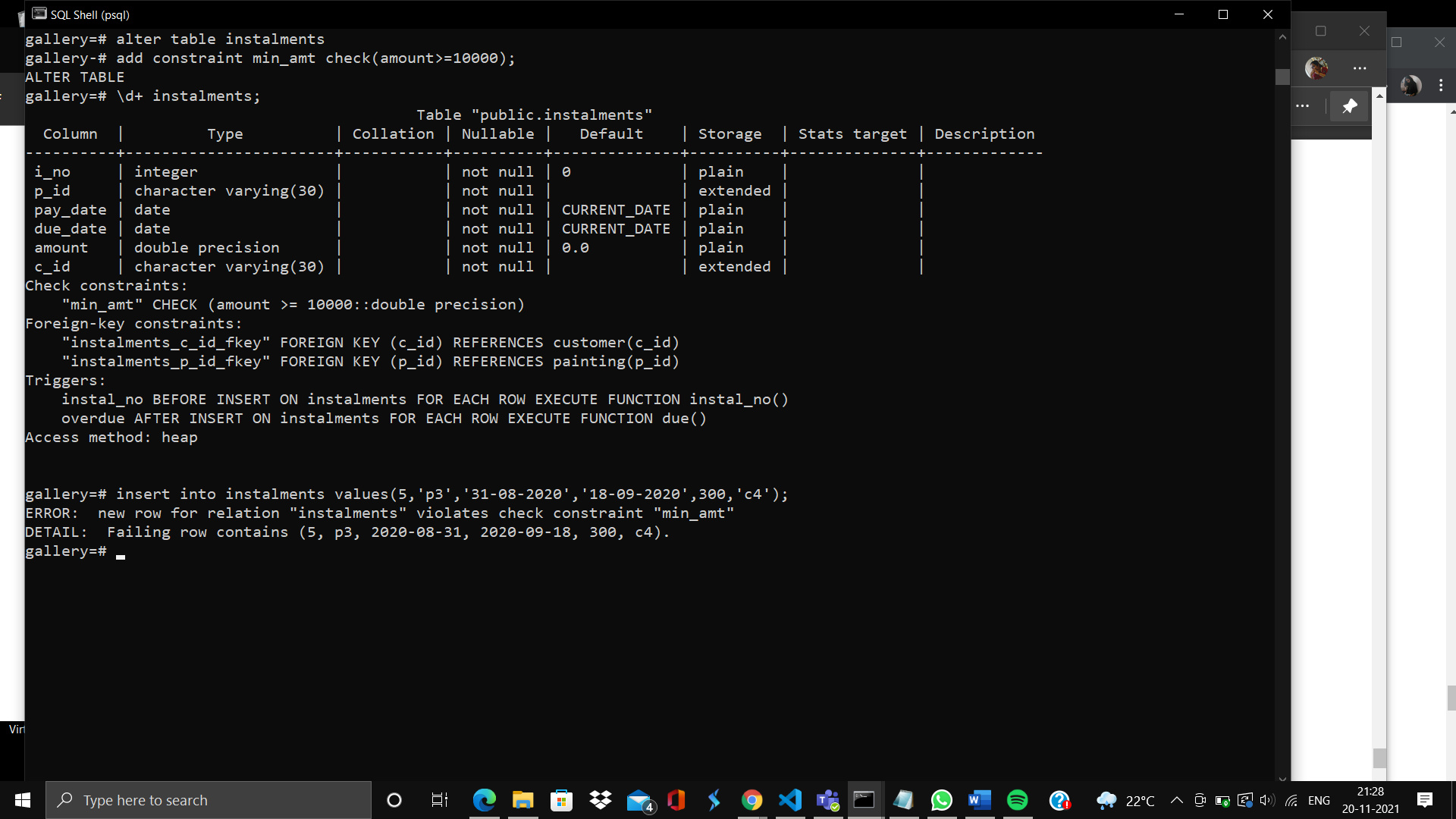


1. Setting a not null constraint and default value of 0 to the price\_fetched attribute of auction relation. Two additional constraints are added to the auction relation.



1. A constraint on the amount attribute of instalments column to ensure the amount is no less than 10000.

On entering an amount less than 10000 as the amount to the relation, an exception is thrown.



**Migration of the Database:**

The current database has a lot of multivalued attributes, like customer phone number, customer address, employee address etc. All these have to be implemented as separate relations. It can get really tedious while inserting values. But using a document-based database like MongoDB can simplify our task.

As the database used JSON format to store records, it is easier even for a naïve user to understand the records. Unlike PSQL, there isn’t a need to create separate relations to store these values. MongoDB allows the usage of lists to store multiple values for a single attribute.

Another advantage of MongoDB over PSQL is, there isn’t a need to specify a separate unique key to identify a record table. The database provides an unique id on insertion of a record into the table. But in PSQL it’s the user’s responsibility to ensure a unique id is specified to each record.

Creating relational data models take time where a document database such as MongoDB can be more fluid and works well with developers. Scaling is inherently built into MongoDB, but with PostgreSQL an extension is required to add that capability.

For those with long-term data storage needs, MongoDB performs well with online applications that have very large data stores where data is required to be kept for years.

**Individual Contributions:**

Deekshitha: 2 additional queries, Migration of the Database.

Deepa: Front-end, 3 additional queries, Report write-up.