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 adapter for the Python programming language. It is a wrapper for the libpq, the official PostgreSQL client library. The quickest way to install Psycopg is using the wheel package available on PyPI. ‘pip install pyscopg2’ is also one of the ways to install the package.

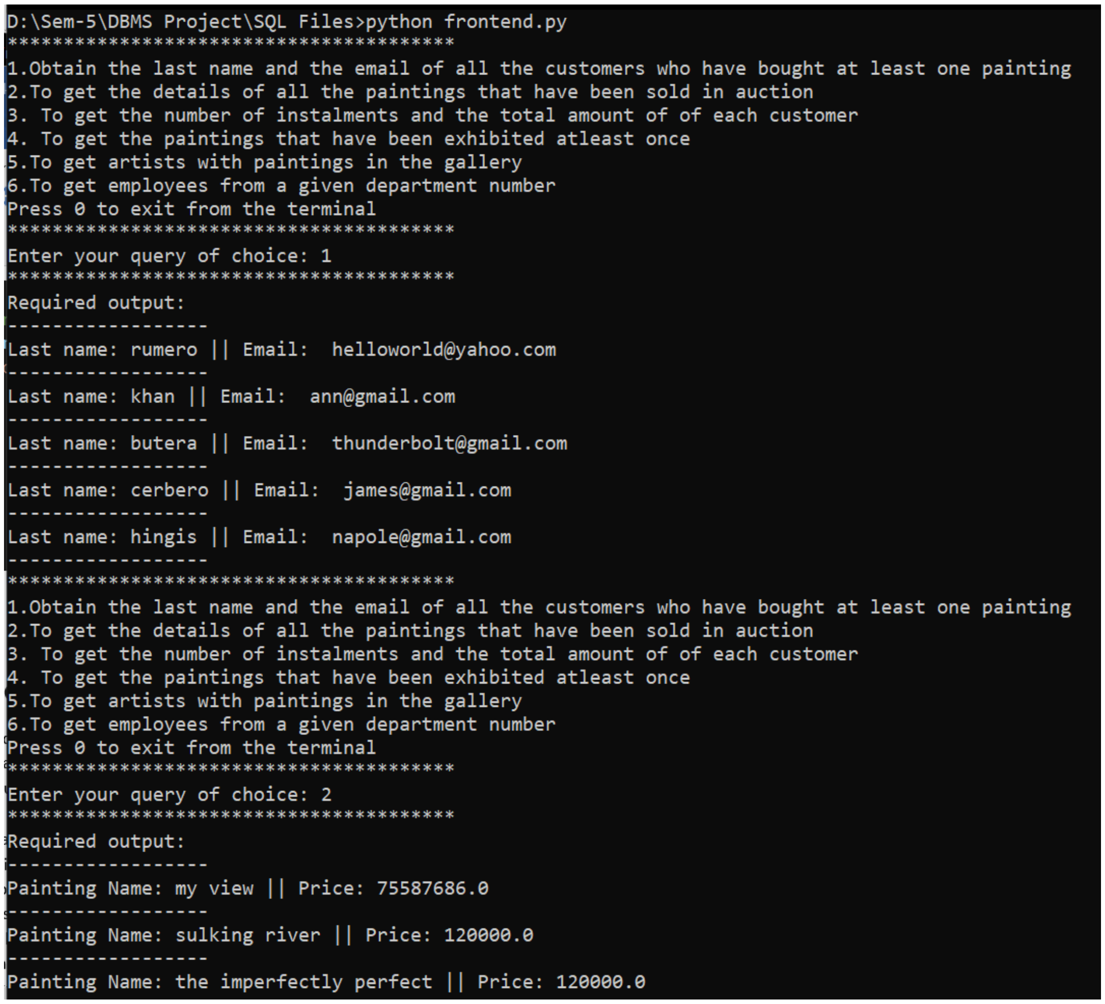
Code for the front-end:

```
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=" || ")
```

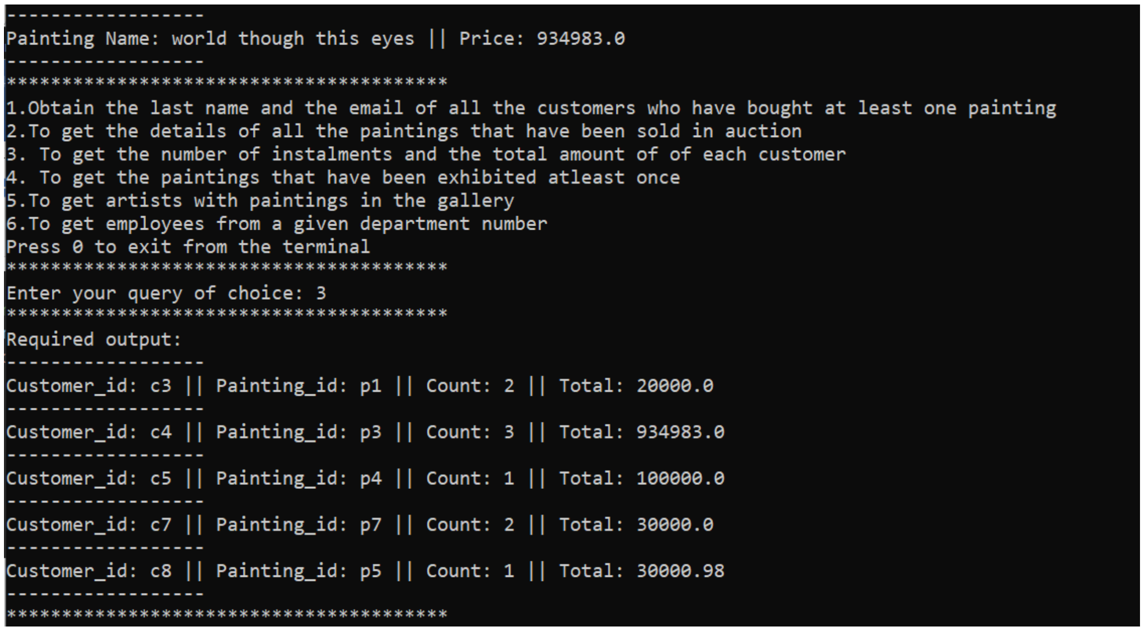
```python
                print("Painting_id:",row[1],end=" || ")
                print("Count:",row[2],end=" || ")
                print("Total:",row[3])
                print("------------------")


        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)
```

```python
    #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")
```

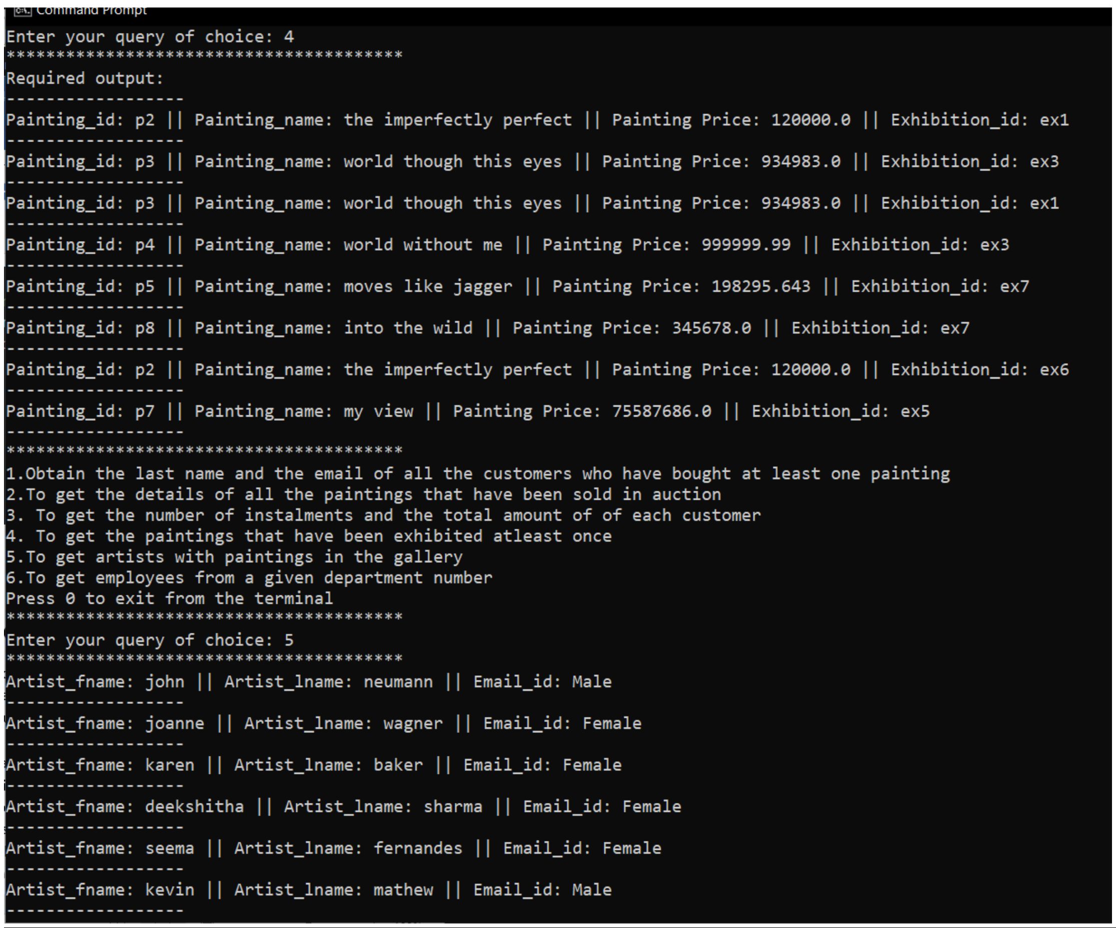
Output of the execution:
All the implemented queries are executed here. On entering a number between 1 and 6, the desired query is executed. On entering an invalid number other than these, the terminal prompts the user to enter a valid number. If zero is entered, then the program is terminated and the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



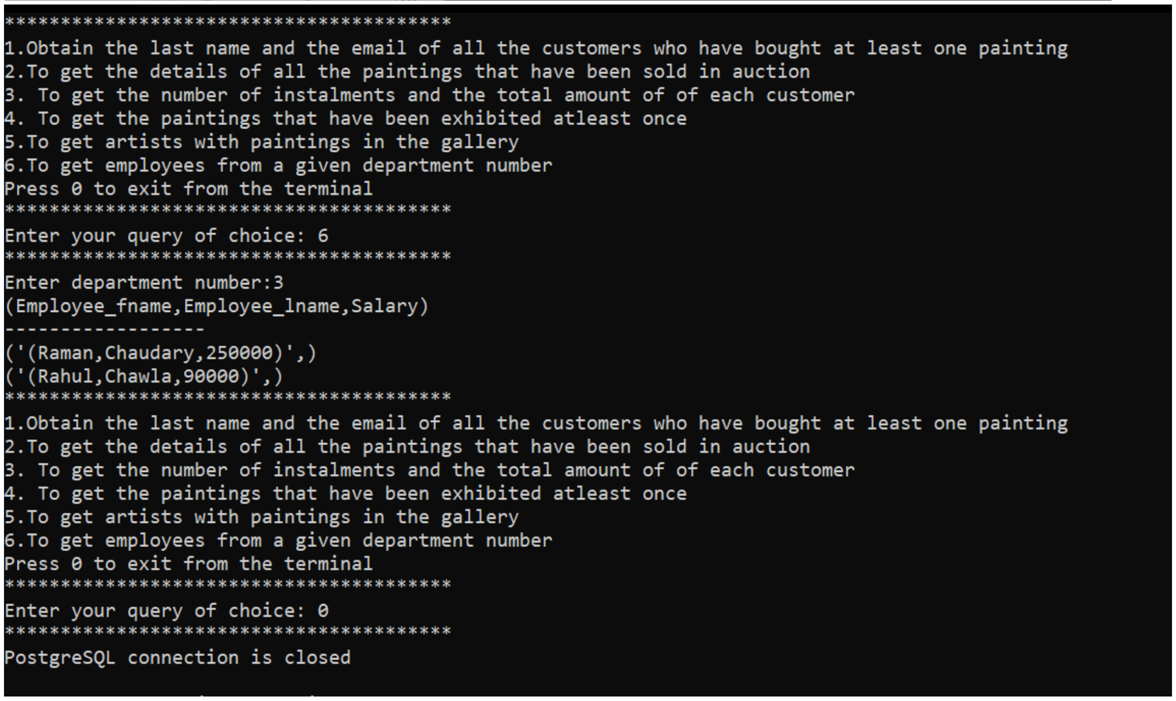
```
D:\Sem-5\DBMS Project\SQL Files>python frontend.py
****************************************
1.Obtain the last name and the email of all the customers who have bought at least one painting
2.To get the details of all the paintings that have been sold in auction
3. To get the number of instalments and the total amount of of each customer
4. To get the paintings that have been exhibited atleast once
5.To get artists with paintings in the gallery
6.To get employees from a given department number
Press 0 to exit from the terminal
****************************************
Enter your query of choice: 1
****************************************
Required output:
------------------
Last name: rumero || Email:  helloworld@yahoo.com
------------------
Last name: khan || Email:  ann@gmail.com
------------------
Last name: butera || Email:  thunderbolt@gmail.com
------------------
Last name: cerbero || Email:  james@gmail.com
------------------
Last name: hingis || Email:  napole@gmail.com
------------------
****************************************
1.Obtain the last name and the email of all the customers who have bought at least one painting
2.To get the details of all the paintings that have been sold in auction
3. To get the number of instalments and the total amount of of each customer
4. To get the paintings that have been exhibited atleast once
5.To get artists with paintings in the gallery
6.To get employees from a given department number
Press 0 to exit from the terminal
****************************************
Enter your query of choice: 2
****************************************
Required output:
------------------
Painting Name: my view || Price: 75587686.0
------------------
Painting Name: sulking river || Price: 120000.0
------------------
Painting Name: the imperfectly perfect || Price: 120000.0
```



```
------------------
Painting Name: world though this eyes || Price: 934983.0
------------------
****************************************
1.Obtain the last name and the email of all the customers who have bought at least one painting
2.To get the details of all the paintings that have been sold in auction
3. To get the number of instalments and the total amount of of each customer
4. To get the paintings that have been exhibited atleast once
5.To get artists with paintings in the gallery
6.To get employees from a given department number
Press 0 to exit from the terminal
****************************************
Enter your query of choice: 3
****************************************
Required output:
------------------
Customer_id: c3 || Painting_id: p1 || Count: 2 || Total: 20000.0
------------------
Customer_id: c4 || Painting_id: p3 || Count: 3 || Total: 934983.0
------------------
Customer_id: c5 || Painting_id: p4 || Count: 1 || Total: 100000.0
------------------
Customer_id: c7 || Painting_id: p7 || Count: 2 || Total: 30000.0
------------------
Customer_id: c8 || Painting_id: p5 || Count: 1 || Total: 30000.98
------------------
****************************************
```



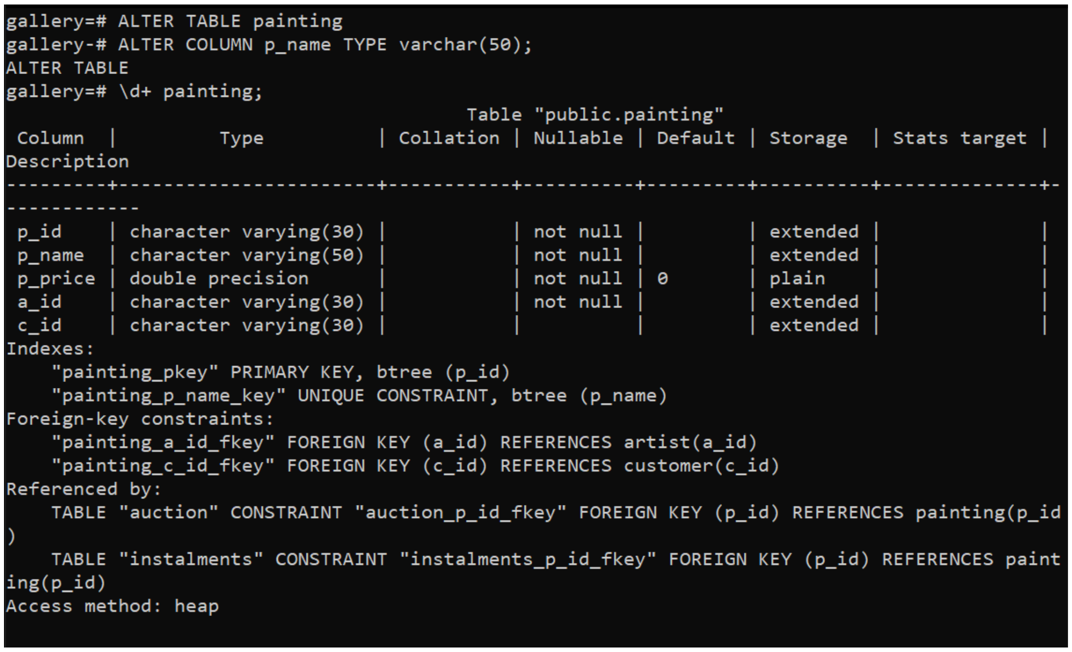
```
Command Prompt
Enter your query of choice: 4
****************************************
Required output:
------------------
Painting_id: p2 || Painting_name: the imperfectly perfect || Painting Price: 120000.0 || Exhibition_id: ex1
------------------
Painting_id: p3 || Painting_name: world though this eyes || Painting Price: 934983.0 || Exhibition_id: ex3
------------------
Painting_id: p3 || Painting_name: world though this eyes || Painting Price: 934983.0 || Exhibition_id: ex1
------------------
Painting_id: p4 || Painting_name: world without me || Painting Price: 999999.99 || Exhibition_id: ex3
------------------
Painting_id: p5 || Painting_name: moves like jagger || Painting Price: 198295.643 || Exhibition_id: ex7
------------------
Painting_id: p8 || Painting_name: into the wild || Painting Price: 345678.0 || Exhibition_id: ex7
------------------
Painting_id: p2 || Painting_name: the imperfectly perfect || Painting Price: 120000.0 || Exhibition_id: ex6
------------------
Painting_id: p7 || Painting_name: my view || Painting Price: 75587686.0 || Exhibition_id: ex5
------------------
****************************************
1.Obtain the last name and the email of all the customers who have bought at least one painting
2.To get the details of all the paintings that have been sold in auction
3. To get the number of instalments and the total amount of of each customer
4. To get the paintings that have been exhibited atleast once
5.To get artists with paintings in the gallery
6.To get employees from a given department number
Press 0 to exit from the terminal
****************************************
Enter your query of choice: 5
****************************************
Artist_fname: john || Artist_lname: neumann || Email_id: Male
------------------
Artist_fname: joanne || Artist_lname: wagner || Email_id: Female
------------------
Artist_fname: karen || Artist_lname: baker || Email_id: Female
------------------
Artist_fname: deekshitha || Artist_lname: sharma || Email_id: Female
------------------
Artist_fname: seema || Artist_lname: fernandes || Email_id: Female
------------------
Artist_fname: kevin || Artist_lname: mathew || Email_id: Male
------------------
```



```
****************************************
1.Obtain the last name and the email of all the customers who have bought at least one painting
2.To get the details of all the paintings that have been sold in auction
3. To get the number of instalments and the total amount of of each customer
4. To get the paintings that have been exhibited atleast once
5.To get artists with paintings in the gallery
6.To get employees from a given department number
Press 0 to exit from the terminal
****************************************
Enter your query of choice: 6
****************************************
Enter department number:3
(Employee_fname,Employee_lname,Salary)
------------------
('(Raman,Chaudary,250000)',)
('(Rahul,Chawla,90000)',)
****************************************
1.Obtain the last name and the email of all the customers who have bought at least one painting
2.To get the details of all the paintings that have been sold in auction
3. To get the number of instalments and the total amount of of each customer
4. To get the paintings that have been exhibited atleast once
5.To get artists with paintings in the gallery
6.To get employees from a given department number
Press 0 to exit from the terminal
****************************************
Enter your query of choice: 0
****************************************
PostgreSQL connection is closed
```

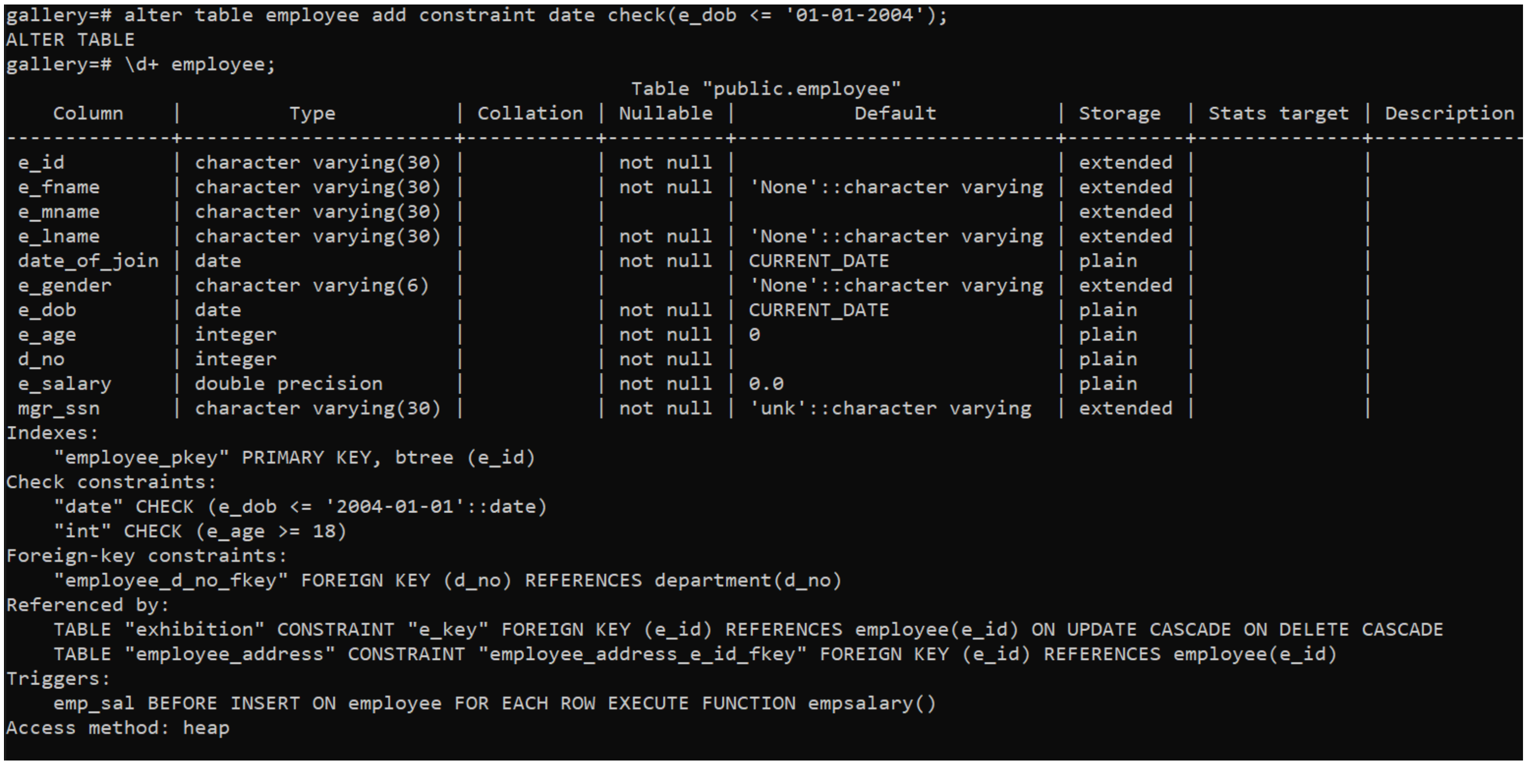
Changes to the database:

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.



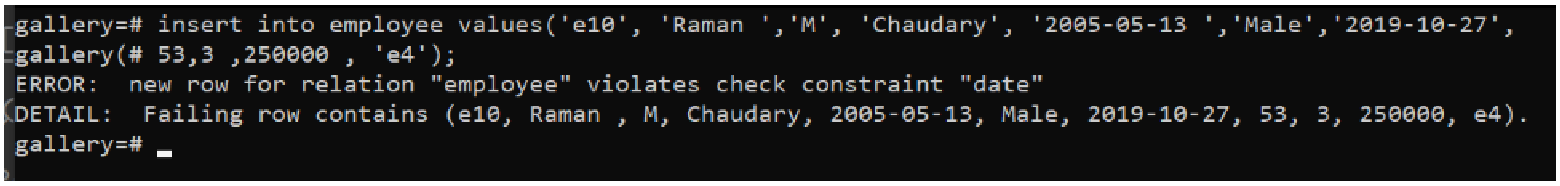
```
gallery=# ALTER TABLE painting
gallery-# ALTER COLUMN p_name TYPE varchar(50);
ALTER TABLE
gallery=# \d+ painting;
                                        Table "public.painting"
 Column  |         Type          | Collation | Nullable | Default | Storage  | Stats target |
Description
---------+-----------------------+-----------+----------+---------+----------+--------------+-
------------
 p_id    | character varying(30) |           | not null |         | extended |              |
 p_name  | character varying(50) |           | not null |         | extended |              |
 p_price | double precision      |           | not null | 0       | plain    |              |
 a_id    | character varying(30) |           | not null |         | extended |              |
 c_id    | character varying(30) |           |          |         | extended |              |
Indexes:
    "painting_pkey" PRIMARY KEY, btree (p_id)
    "painting_p_name_key" UNIQUE CONSTRAINT, btree (p_name)
Foreign-key constraints:
    "painting_a_id_fkey" FOREIGN KEY (a_id) REFERENCES artist(a_id)
    "painting_c_id_fkey" FOREIGN KEY (c_id) REFERENCES customer(c_id)
Referenced by:
    TABLE "auction" CONSTRAINT "auction_p_id_fkey" FOREIGN KEY (p_id) REFERENCES painting(p_id
)
    TABLE "instalments" CONSTRAINT "instalments_p_id_fkey" FOREIGN KEY (p_id) REFERENCES paint
ing(p_id)
Access method: heap
```

2. Setting a constraint on the e_dob of employee relation. The dob cannot contain a year greater than 2004.



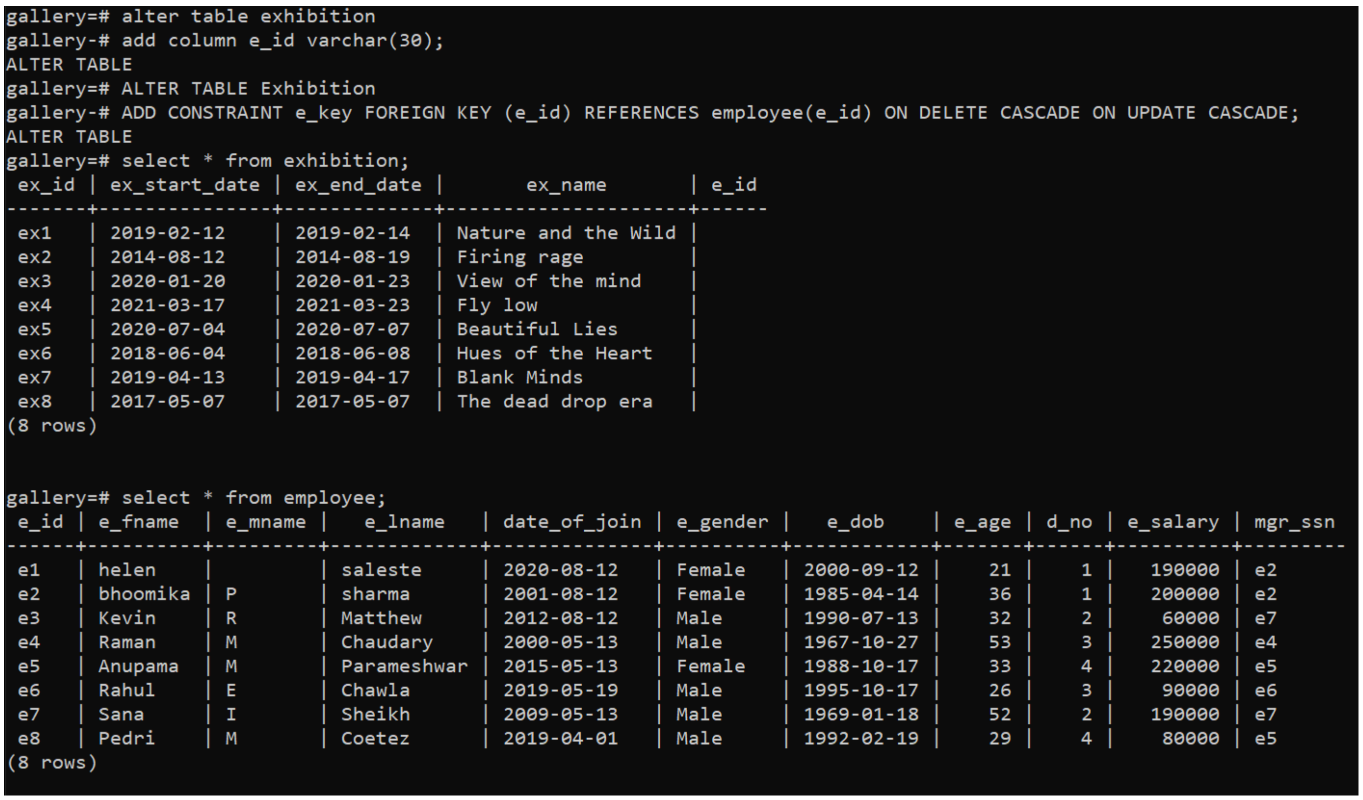
```
gallery=# alter table employee add constraint date check(e_dob <= '01-01-2004');
ALTER TABLE
gallery=# \d+ employee;
                                              Table "public.employee"
    Column    |         Type          | Collation | Nullable |          Default           | Storage  | Stats target | Description
--------------+-----------------------+-----------+----------+----------------------------+----------+--------------+-------------
 e_id         | character varying(30) |           | not null |                            | extended |              |
 e_fname      | character varying(30) |           | not null | 'None'::character varying  | extended |              |
 e_mname      | character varying(30) |           |          |                            | extended |              |
 e_lname      | character varying(30) |           | not null | 'None'::character varying  | extended |              |
 date_of_join | date                  |           | not null | CURRENT_DATE               | plain    |              |
 e_gender     | character varying(6)  |           |          | 'None'::character varying  | extended |              |
 e_dob        | date                  |           | not null | CURRENT_DATE               | plain    |              |
 e_age        | integer               |           | not null | 0                          | plain    |              |
 d_no         | integer               |           | not null |                            | plain    |              |
 e_salary     | double precision      |           | not null | 0.0                        | plain    |              |
 mgr_ssn      | character varying(30) |           | not null | 'unk'::character varying   | extended |              |
Indexes:
    "employee_pkey" PRIMARY KEY, btree (e_id)
Check constraints:
    "date" CHECK (e_dob <= '2004-01-01'::date)
    "int" CHECK (e_age >= 18)
Foreign-key constraints:
    "employee_d_no_fkey" FOREIGN KEY (d_no) REFERENCES department(d_no)
Referenced by:
    TABLE "exhibition" CONSTRAINT "e_key" FOREIGN KEY (e_id) REFERENCES employee(e_id) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "employee_address" CONSTRAINT "employee_address_e_id_fkey" FOREIGN KEY (e_id) REFERENCES employee(e_id)
Triggers:
    emp_sal BEFORE INSERT ON employee FOR EACH ROW EXECUTE FUNCTION empsalary()
Access method: heap
```

Inserting an invalid value throws an exception.



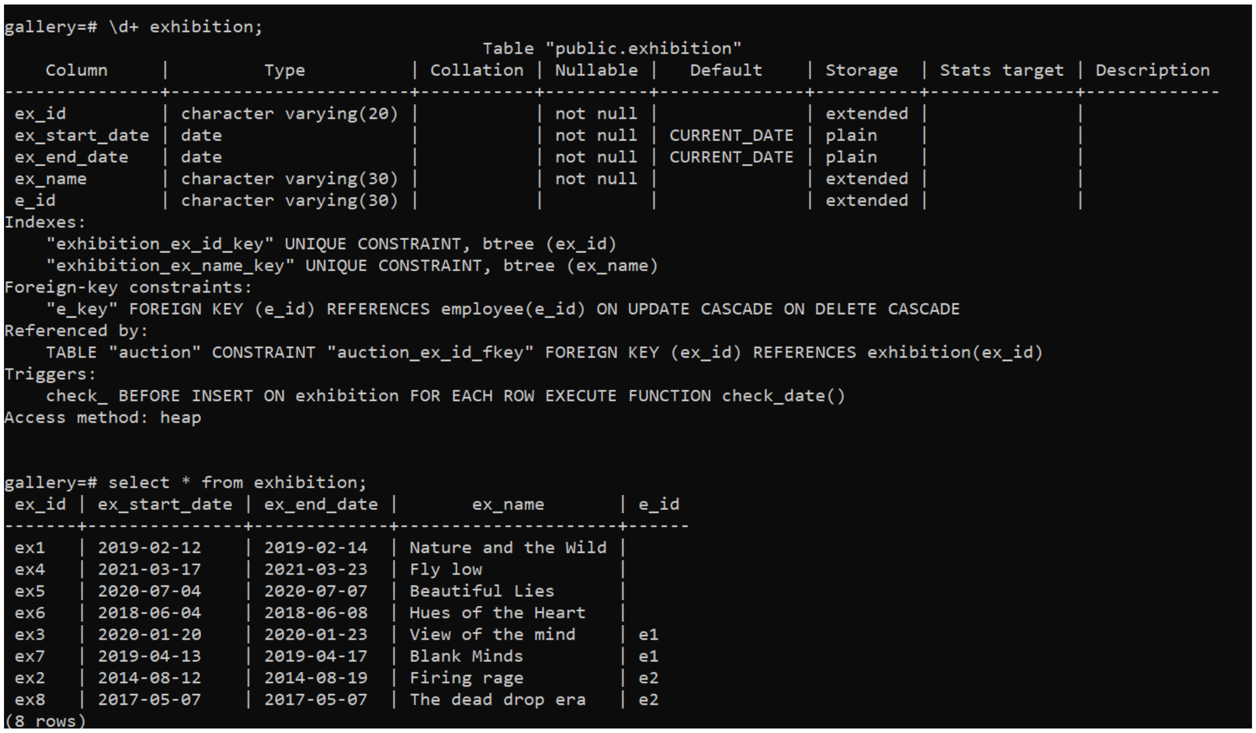
```
gallery=# insert into employee values('e10', 'Raman ','M', 'Chaudary', '2005-05-13 ','Male','2019-10-27',
gallery(# 53,3 ,250000 , 'e4');
ERROR:  new row for relation "employee" violates check constraint "date"
DETAIL:  Failing row contains (e10, Raman , M, Chaudary, 2005-05-13, Male, 2019-10-27, 53, 3, 250000, e4).
gallery=# _
```

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.



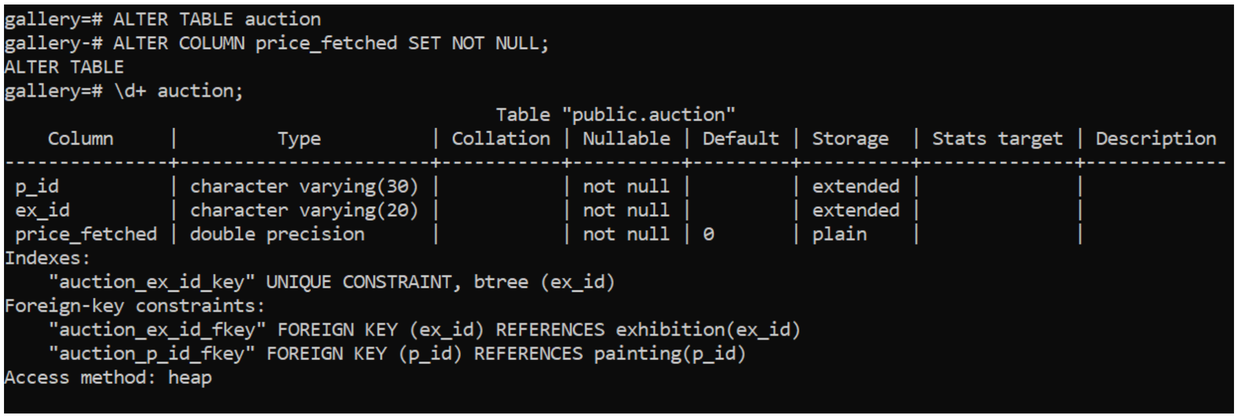
```
gallery=# alter table exhibition
gallery-# add column e_id varchar(30);
ALTER TABLE
gallery=# ALTER TABLE Exhibition
gallery-# ADD CONSTRAINT e_key FOREIGN KEY (e_id) REFERENCES employee(e_id) ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE
gallery=# select * from exhibition;
 ex_id | ex_start_date | ex_end_date |       ex_name       | e_id
-------+---------------+-------------+---------------------+------
 ex1   | 2019-02-12    | 2019-02-14  | Nature and the Wild |
 ex2   | 2014-08-12    | 2014-08-19  | Firing rage         |
 ex3   | 2020-01-20    | 2020-01-23  | View of the mind    |
 ex4   | 2021-03-17    | 2021-03-23  | Fly low             |
 ex5   | 2020-07-04    | 2020-07-07  | Beautiful Lies      |
 ex6   | 2018-06-04    | 2018-06-08  | Hues of the Heart   |
 ex7   | 2019-04-13    | 2019-04-17  | Blank Minds         |
 ex8   | 2017-05-07    | 2017-05-07  | The dead drop era   |
(8 rows)


gallery=# select * from employee;
 e_id | e_fname  | e_mname |   e_lname   | date_of_join | e_gender |   e_dob    | e_age | d_no | e_salary | mgr_ssn
------+----------+---------+-------------+--------------+----------+------------+-------+------+----------+---------
 e1   | helen    |         | saleste     | 2020-08-12   | Female   | 2000-09-12 |    21 |    1 |   190000 | e2
 e2   | bhoomika | P       | sharma      | 2001-08-12   | Female   | 1985-04-14 |    36 |    1 |   200000 | e2
 e3   | Kevin    | R       | Matthew     | 2012-08-12   | Male     | 1990-07-13 |    32 |    2 |    60000 | e7
 e4   | Raman    | M       | Chaudary    | 2000-05-13   | Male     | 1967-10-27 |    53 |    3 |   250000 | e4
 e5   | Anupama  | M       | Parameshwar | 2015-05-13   | Female   | 1988-10-17 |    33 |    4 |   220000 | e5
 e6   | Rahul    | E       | Chawla      | 2019-05-19   | Male     | 1995-10-17 |    26 |    3 |    90000 | e6
 e7   | Sana     | I       | Sheikh      | 2009-05-13   | Male     | 1969-01-18 |    52 |    2 |   190000 | e7
 e8   | Pedri    | M       | Coetez      | 2019-04-01   | Male     | 1992-02-19 |    29 |    4 |    80000 | e5
(8 rows)
```

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



```
gallery=# \d+ exhibition;
                                              Table "public.exhibition"
    Column     |         Type          | Collation | Nullable |   Default    | Storage  | Stats target | Description
---------------+-----------------------+-----------+----------+--------------+----------+--------------+-------------
 ex_id         | character varying(20) |           | not null |              | extended |              |
 ex_start_date | date                  |           | not null | CURRENT_DATE | plain    |              |
 ex_end_date   | date                  |           | not null | CURRENT_DATE | plain    |              |
 ex_name       | character varying(30) |           | not null |              | extended |              |
 e_id          | character varying(30) |           |          |              | extended |              |
Indexes:
    "exhibition_ex_id_key" UNIQUE CONSTRAINT, btree (ex_id)
    "exhibition_ex_name_key" UNIQUE CONSTRAINT, btree (ex_name)
Foreign-key constraints:
    "e_key" FOREIGN KEY (e_id) REFERENCES employee(e_id) ON UPDATE CASCADE ON DELETE CASCADE
Referenced by:
    TABLE "auction" CONSTRAINT "auction_ex_id_fkey" FOREIGN KEY (ex_id) REFERENCES exhibition(ex_id)
Triggers:
    check_ BEFORE INSERT ON exhibition FOR EACH ROW EXECUTE FUNCTION check_date()
Access method: heap


gallery=# select * from exhibition;
 ex_id | ex_start_date | ex_end_date |       ex_name       | e_id
-------+---------------+-------------+---------------------+------
 ex1   | 2019-02-12    | 2019-02-14  | Nature and the Wild |
 ex4   | 2021-03-17    | 2021-03-23  | Fly low             |
 ex5   | 2020-07-04    | 2020-07-07  | Beautiful Lies      |
 ex6   | 2018-06-04    | 2018-06-08  | Hues of the Heart   |
 ex3   | 2020-01-20    | 2020-01-23  | View of the mind    | e1
 ex7   | 2019-04-13    | 2019-04-17  | Blank Minds         | e1
 ex2   | 2014-08-12    | 2014-08-19  | Firing rage         | e2
 ex8   | 2017-05-07    | 2017-05-07  | The dead drop era   | e2
(8 rows)
```

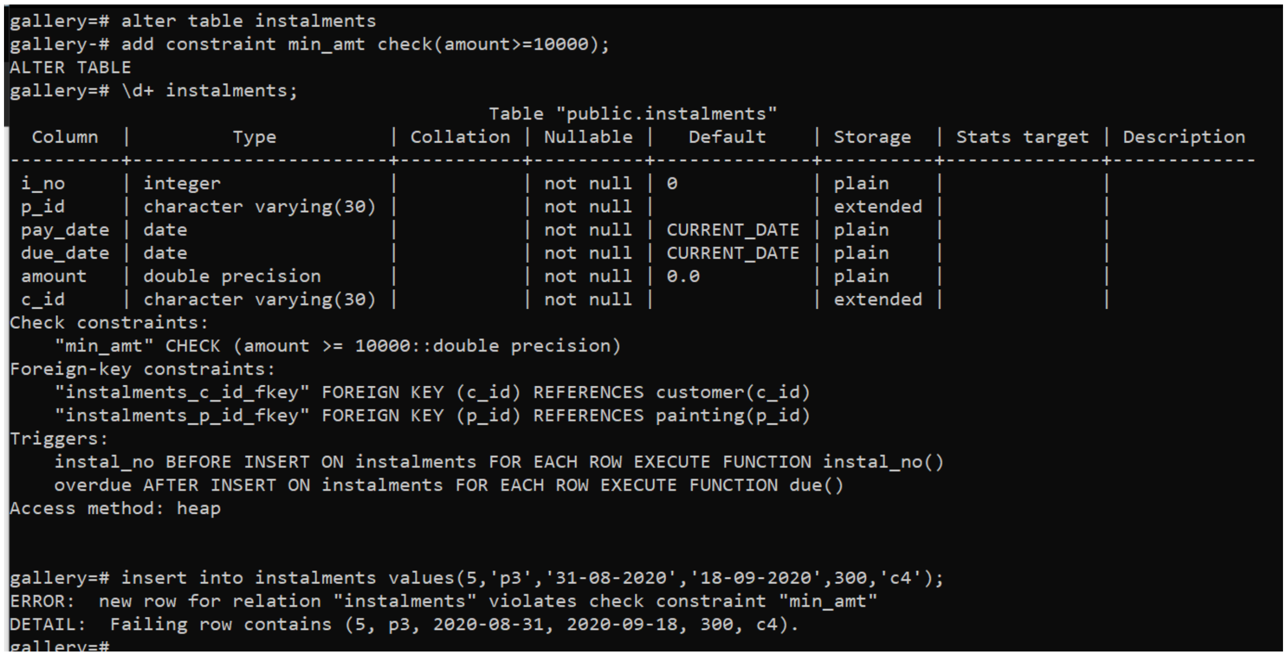
4. Setting a not null constraint and default value of 0 to the price_fetched attribute of auction relation. Two additional constraints are added to the auction relation.



```
gallery=# ALTER TABLE auction
gallery-# ALTER COLUMN price_fetched SET NOT NULL;
ALTER TABLE
gallery=# \d+ auction;
                                            Table "public.auction"
    Column     |         Type          | Collation | Nullable | Default | Storage  | Stats target | Description
---------------+-----------------------+-----------+----------+---------+----------+--------------+-------------
 p_id          | character varying(30) |           | not null |         | extended |              |
 ex_id         | character varying(20) |           | not null |         | extended |              |
 price_fetched | double precision      |           | not null | 0       | plain    |              |
Indexes:
    "auction_ex_id_key" UNIQUE CONSTRAINT, btree (ex_id)
Foreign-key constraints:
    "auction_ex_id_fkey" FOREIGN KEY (ex_id) REFERENCES exhibition(ex_id)
    "auction_p_id_fkey" FOREIGN KEY (p_id) REFERENCES painting(p_id)
Access method: heap
```

5. 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.



```
gallery=# alter table instalments
gallery-# add constraint min_amt check(amount>=10000);
ALTER TABLE
gallery=# \d+ instalments;
                                           Table "public.instalments"
  Column   |         Type          | Collation | Nullable |    Default     | Storage  | Stats target | Description
-----------+-----------------------+-----------+----------+----------------+----------+--------------+-------------
 i_no      | integer               |           | not null | 0              | plain    |              |
 p_id      | character varying(30) |           | not null |                | extended |              |
 pay_date  | date                  |           | not null | CURRENT_DATE   | plain    |              |
 due_date  | date                  |           | not null | CURRENT_DATE   | plain    |              |
 amount    | double precision      |           | not null | 0.0            | plain    |              |
 c_id      | character varying(30) |           | not null |                | extended |              |
Check constraints:
    "min_amt" CHECK (amount >= 10000::double precision)
Foreign-key constraints:
    "instalments_c_id_fkey" FOREIGN KEY (c_id) REFERENCES customer(c_id)
    "instalments_p_id_fkey" FOREIGN KEY (p_id) REFERENCES painting(p_id)
Triggers:
    instal_no BEFORE INSERT ON instalments FOR EACH ROW EXECUTE FUNCTION instal_no()
    overdue AFTER INSERT ON instalments FOR EACH ROW EXECUTE FUNCTION due()
Access method: heap


gallery=# insert into instalments values(5,'p3','31-08-2020','18-09-2020',300,'c4');
ERROR:  new row for relation "instalments" violates check constraint "min_amt"
DETAIL:  Failing row contains (5, p3, 2020-08-31, 2020-09-18, 300, c4).
gallery=#
```

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.