CODING CHALLENGE -1

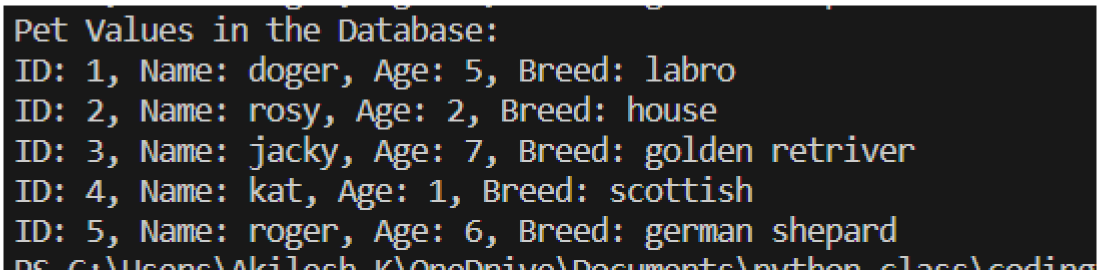
PETPALS

AKILESH K

TASK 1

1)

```
from connector import create_connection
from mysql.connector import Error
class Pet:
    def __init__(self, name, age, breed):
        self.name = name
        self.age = age
        self.breed = breed



    def display_pet_values():
        try:
            # Database connection
            connection = create_connection()
            if connection:
                try:
                    cursor = connection.cursor()

                    # Retrieve pet values from the database
                    cursor.execute("SELECT * FROM pets")
                    pets = cursor.fetchall()

                    # Display pet values
                    print("Pet Values in the Database:")
                    for pet in pets:
                        print(f"ID: {pet[0]}, Name: {pet[1]}, Age: {pet[2]}, Breed: {pet[3]}")
                except Error as e:
                    print(f"Error retrieving pet values from the database: {e}")

                finally:
                    connection.close()

        except Error as e:
            print(f"Error connecting to the database: {e}")
    display_pet_values()
```

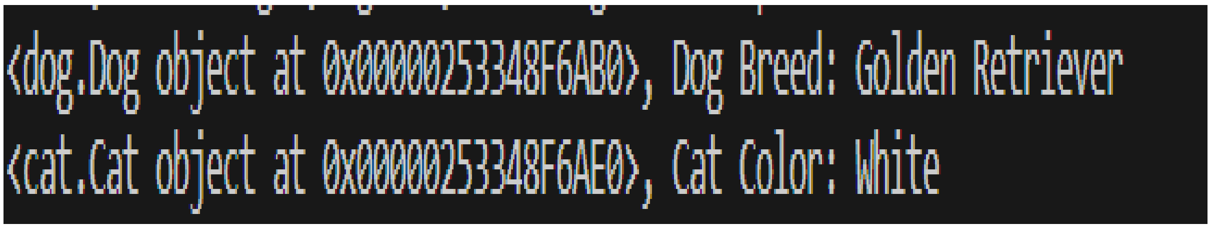


```
Pet Values in the Database:
ID: 1, Name: doger, Age: 5, Breed: labro
ID: 2, Name: rosy, Age: 2, Breed: house
ID: 3, Name: jacky, Age: 7, Breed: golden retriver
ID: 4, Name: kat, Age: 1, Breed: scottish
ID: 5, Name: roger, Age: 6, Breed: german shepard
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding
```

2)

```
from pet import Pet


class Dog(Pet):
    def __init__(self, name, age, breed, dog_breed):
        super().__init__(name, age, breed)
        self.dog_breed = dog_breed

    def __str__(self):
        return super().__str__() + f", Dog Breed: {self.dog_breed}"



from pet import Pet
```

```
class Cat(Pet):
    def __init__(self, name, age, breed, cat_color):
        super().__init__(name, age, breed)
        self.cat_color = cat_color

    def __str__(self):
        return super().__str__() + f", Cat Color: {self.cat_color}"
```



```
<dog.Dog object at 0x00000253348F6AB0>, Dog Breed: Golden Retriever
<cat.Cat object at 0x00000253348F6AE0>, Cat Color: White
```

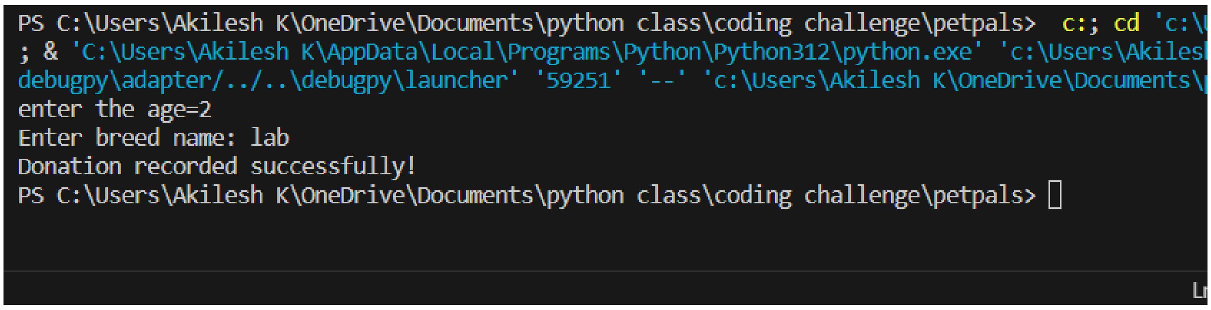
3)

```
from connector import create_connection
from mysql.connector import Error
from pet import Pet
pidc=103
class PetShelter:
    def __init__(self,available_pets):
        self.available_pets=available_pets







    def add_pet_to_database():
        connection = create_connection()
        if connection:
```

```
      try:
        name=("enter name:")
        age=input("enter the age=")
        breed = input("Enter breed name: ")
        pid=102

        cursor = connection.cursor()
        cursor.execute("INSERT INTO pets (petid, name, age, breed) VALUES (%s, %s, %s, %s)",
                (pid,name,age,breed,))
        connection.commit()

        print("Donation recorded successfully!")

      except (Error, ValueError) as e:
        print(f"Error recording donation: {e}")
      finally:
        connection.close()
```

display pets

```
def display_pet_values():
    try:
      # Database connection
      connection = create_connection()
      if connection:
        try:
          cursor = connection.cursor()

          # Retrieve pet values from the database
          cursor.execute("SELECT * FROM pets")
          pets = cursor.fetchall()
```



```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petpals>  c:; cd 'c:\
; & 'C:\Users\Akilesh K\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\Akiles
debugpy\adapter/../..\debugpy\launcher' '59251' '--' 'c:\Users\Akilesh K\OneDrive\Documents\
enter the age=2
Enter breed name: lab
Donation recorded successfully!
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petpals> 
```

```
            # Display pet values

            print("Pet Values in the Database:")

            for pet in pets:

                print(f"ID: {pet[0]}, Name: {pet[1]}, Age: {pet[2]}, Breed: {pet[3]}")


        except Error as e:

            print(f"Error retrieving pet values from the database: {e}")


        finally:

            connection.close()


    except Error as e:

        print(f"Error connecting to the database: {e}")
```

4)

```
# Donation class (Abstract)

from abc import ABC, abstractmethod


class Donation(ABC):

    def __init__(self, donor_name, amount):

        self.donor_name = donor_name

        self.amount = amount
    @abstractmethod
    def record_donation(self):
        pass
```

5)

```
# CashDonation class
class CashDonation(Donation):
    def __init__(self, donor_name, amount, donation_date):
        super().__init__(donor_name, amount)
        self.donation_date = donation_date

    def record_donation(self):
        print(f"Cash donation of ${self.amount} recorded on {self.donation_date}")

# ItemDonation class
class ItemDonation(Donation):
    def __init__(self, donor_name, amount, item_type):
        super().__init__(donor_name, amount)
        self.item_type = item_type

    def record_donation(self):
        print(f"Item donation of {self.item_type} recorded")
```

TASK 5

```
# IAdoptable interface/abstract class
class IAdoptable(ABC):
    @abstractmethod
    def adopt(self):
        pass

# AdoptionEvent class
```

```
class AdoptionEvent:
   def __init__(self):
      self.participants = []

   def host_event(self):
      print("Adoption event hosted!")

   def register_participant(self, participant):
      self.participants.append(participant)
```

TASK 6

EXCEPTIONS:

```
# Custom AdoptionException
class AdoptionException(Exception):
   pass

# Pet class with InvalidPetAgeException
class Pet:
   def __init__(self, name, age, breed):
      if not isinstance(age, int) or age <= 0:
         raise ValueError("Invalid pet age. Age must be a positive integer.")
      self.name = name
      self.age = age
      self.breed = breed

   def __str__(self):
      return f"{self.name} - Age: {self.age}, Breed: {self.breed}"
```

```
# PetShelter class with NullReferenceException
class PetShelter:
    def __init__(self):
        self.available_pets = []

    def add_pet(self, pet):
        if pet is None or any(prop is None for prop in [pet.name, pet.age, pet.breed]):
            raise NullPointerException("Pet information is missing.")
        self.available_pets.append(pet)

    def list_available_pets(self):
        for pet in self.available_pets:
            if any(prop is None for prop in [pet.name, pet.age, pet.breed]):
                raise NullPointerException("Pet information is missing.")
            print(pet)

# Donation class with InsufficientFundsException
class Donation:
    def __init__(self, donor_name, amount):
        if not isinstance(amount, (int, float)) or amount < 10:
            raise InsufficientFundsException("Insufficient donation amount. Minimum donation is $10.")
        self.donor_name = donor_name
        self.amount = amount

    def record_donation(self):
        print(f"Donation of ${self.amount} recorded.")

# File handling with FileHandlingException
class PetFileHandler:
    @staticmethod
    def read_pets_from_file(filename):
        try:
            with open(filename, 'r') as file:
                # Assuming each line in the file represents a pet's information
                pet_data = [line.strip().split(',') for line in file.readlines()]
                pets = [Pet(name, int(age), breed) for name, age, breed in pet_data]
                return pets
        except FileNotFoundError:
            raise FileHandlingException(f"File '{filename}' not found.")
        except Exception as e:
            raise FileHandlingException(f"Error reading file '{filename}': {str(e)}")
```

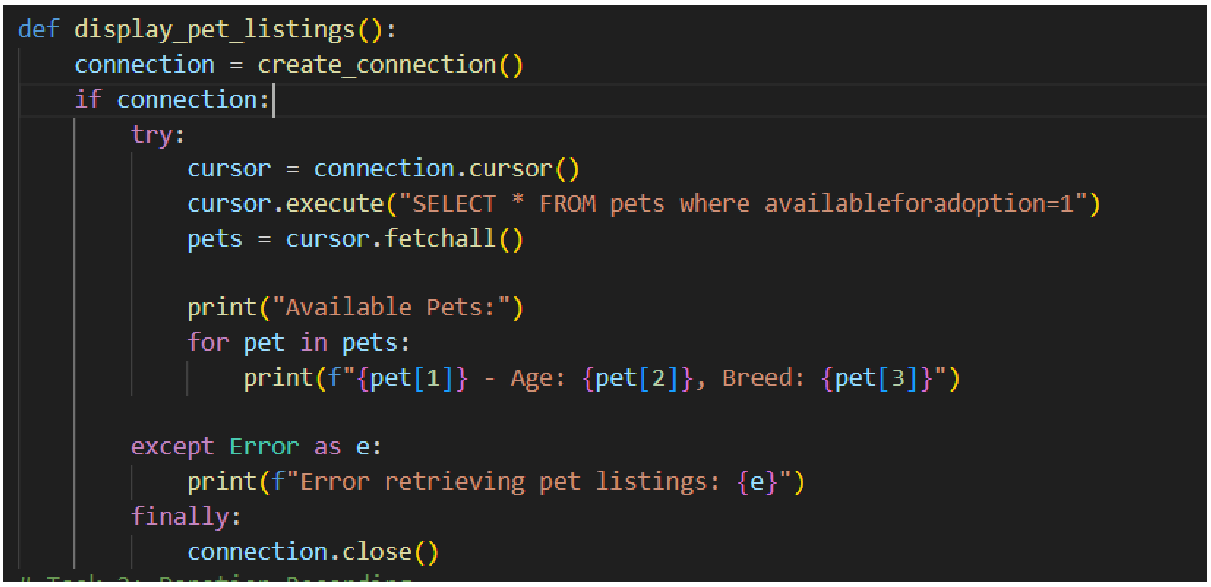
7)

Connecting to the MYSQL database

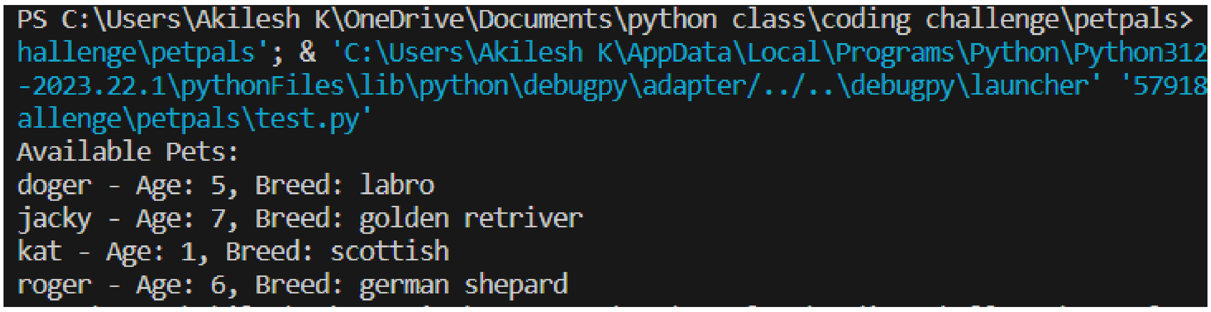
```
import mysql.connector
from mysql.connector import Error
from datetime import datetime


# Database connection
def create_connection():
    try:
        connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="root",
            port='3306',
            database="petpals"
        )
        return connection
    except Error as e:
        print(f"Error connecting to the database: {e}")
        return None
```

A) Displaying Pet Listings:

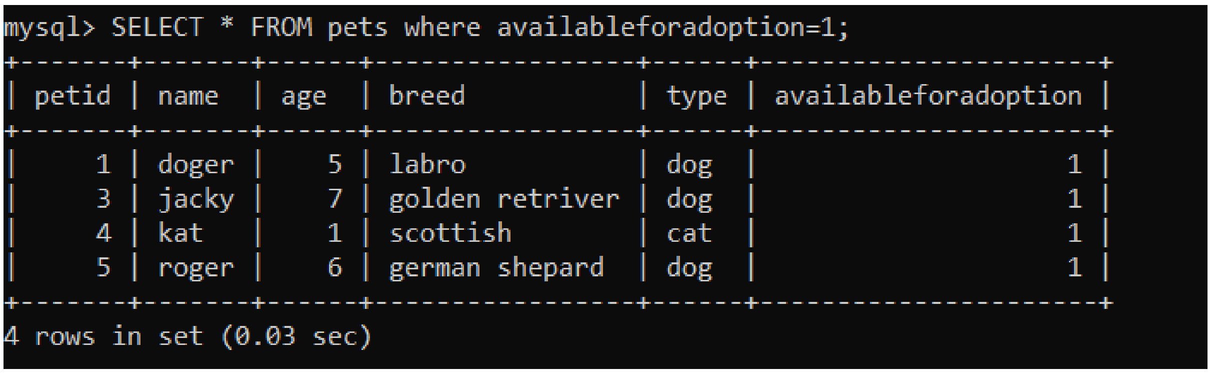
```
def display_pet_listings():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM pets where availableforadoption=1")
            pets = cursor.fetchall()

            print("Available Pets:")
            for pet in pets:
                print(f"{pet[1]} - Age: {pet[2]}, Breed: {pet[3]}")

        except Error as e:
            print(f"Error retrieving pet listings: {e}")
        finally:
            connection.close()
```



```python
def display_pet_listings():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM pets where availableforadoption=1")
            pets = cursor.fetchall()

            print("Available Pets:")
            for pet in pets:
                print(f"{pet[1]} - Age: {pet[2]}, Breed: {pet[3]}")

        except Error as e:
            print(f"Error retrieving pet listings: {e}")
        finally:
            connection.close()
```



```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petpals>
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Programs\Python\Python312
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '57918
allenge\petpals\test.py'
Available Pets:
doger - Age: 5, Breed: labro
jacky - Age: 7, Breed: golden retriver
kat - Age: 1, Breed: scottish
roger - Age: 6, Breed: german shepard
```



```
mysql> SELECT * FROM pets where availableforadoption=1;
+-------+-------+-----+-----------------+------+----------------------+
| petid | name  | age | breed           | type | availableforadoption |
+-------+-------+-----+-----------------+------+----------------------+
|     1 | doger |   5 | labro           | dog  |                    1 |
|     3 | jacky |   7 | golden retriver | dog  |                    1 |
|     4 | kat   |   1 | scottish        | cat  |                    1 |
|     5 | roger |   6 | german shepard  | dog  |                    1 |
+-------+-------+-----+-----------------+------+----------------------+
4 rows in set (0.03 sec)
```

B) Donation Recording

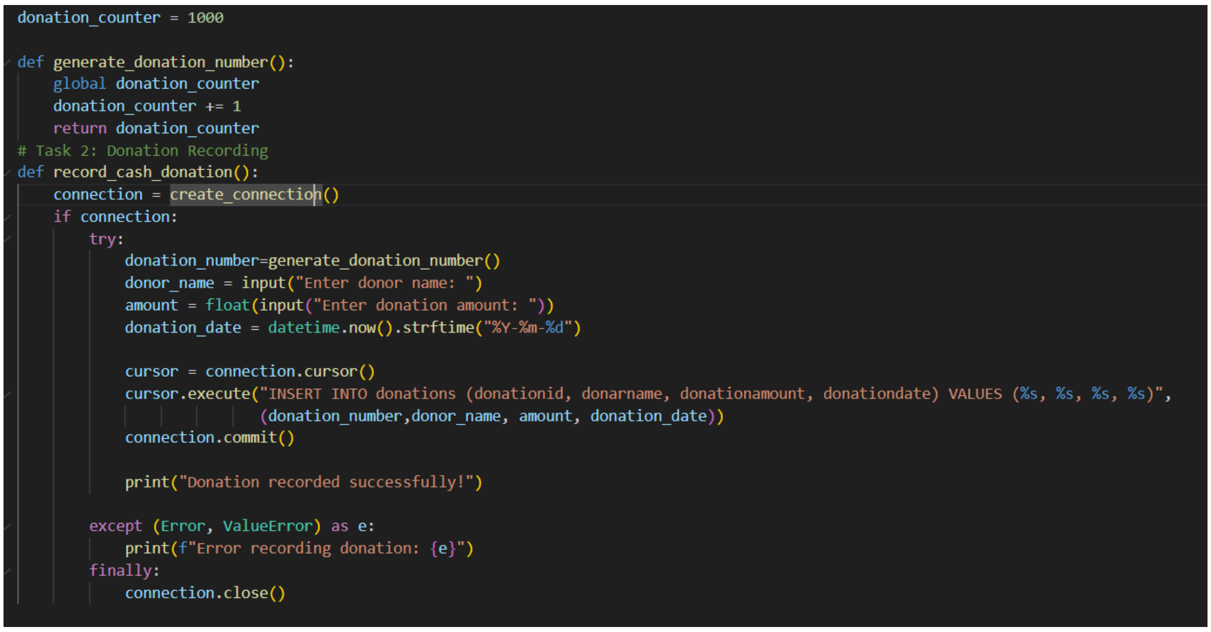
donation_counter = 1000

def generate_donation_number():

  global donation_counter

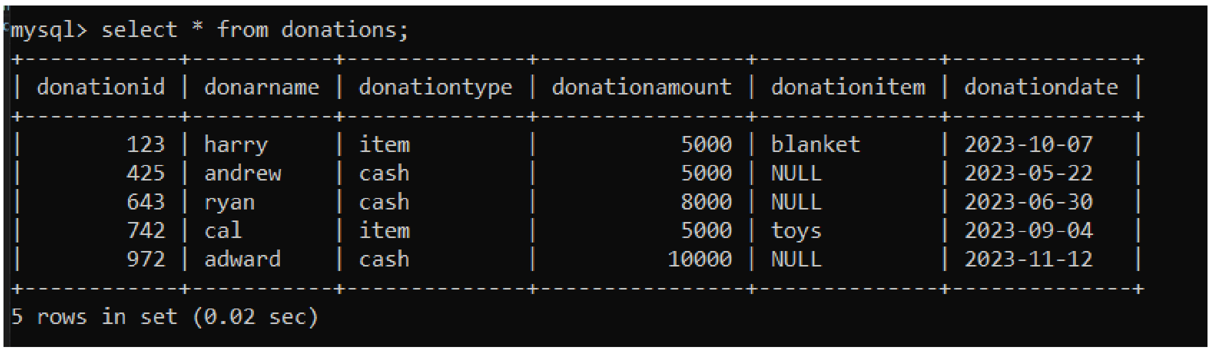
  donation_counter += 1

```
    return donation_counter


def record_cash_donation():
    connection = create_connection()
    if connection:
        try:
            donation_number=generate_donation_number()
            donor_name = input("Enter donor name: ")
            amount = float(input("Enter donation amount: "))
            donation_date = datetime.now().strftime("%Y-%m-%d")

            cursor = connection.cursor()
            cursor.execute("INSERT INTO donations (donationid, donarname, donationamount,
donationdate) VALUES (%s, %s, %s, %s)",
                           (donation_number,donor_name, amount, donation_date))
            connection.commit()

            print("Donation recorded successfully!")

        except (Error, ValueError) as e:
            print(f"Error recording donation: {e}")
        finally:
            connection.close()
```

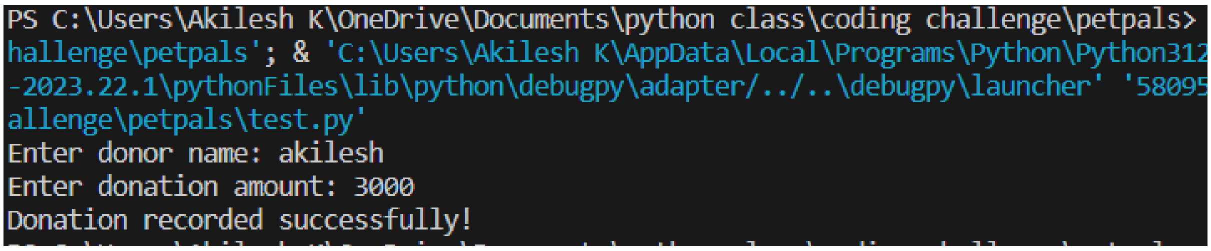


```python
donation_counter = 1000

def generate_donation_number():
    global donation_counter
    donation_counter += 1
    return donation_counter
# Task 2: Donation Recording
def record_cash_donation():
    connection = create_connection()
    if connection:
        try:
            donation_number=generate_donation_number()
            donor_name = input("Enter donor name: ")
            amount = float(input("Enter donation amount: "))
            donation_date = datetime.now().strftime("%Y-%m-%d")

            cursor = connection.cursor()
            cursor.execute("INSERT INTO donations (donationid, donarname, donationamount, donationdate) VALUES (%s, %s, %s, %s)",
                           (donation_number,donor_name, amount, donation_date))
            connection.commit()

            print("Donation recorded successfully!")

        except (Error, ValueError) as e:
            print(f"Error recording donation: {e}")
        finally:
            connection.close()
```

Before adding donor.

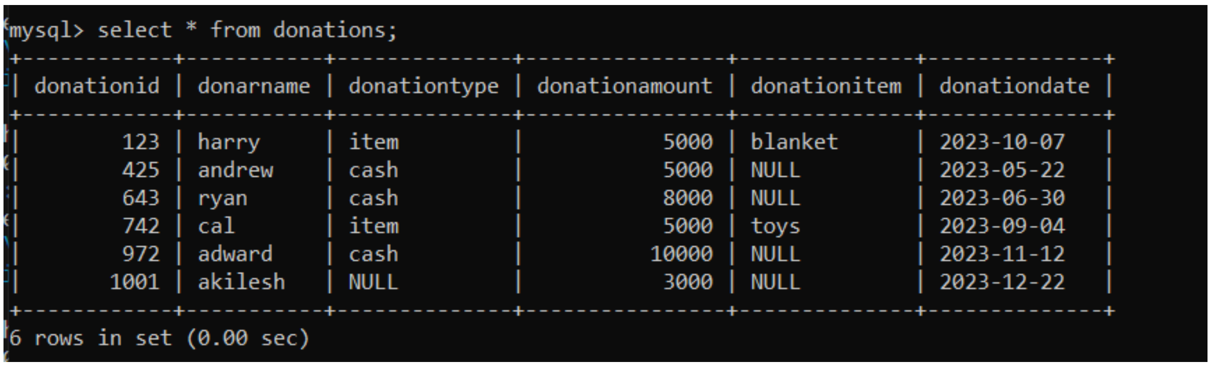


```
mysql> select * from donations;
+------------+-----------+--------------+----------------+--------------+--------------+
| donationid | donarname | donationtype | donationamount | donationitem | donationdate |
+------------+-----------+--------------+----------------+--------------+--------------+
|        123 | harry     | item         |           5000 | blanket      | 2023-10-07   |
|        425 | andrew    | cash         |           5000 | NULL         | 2023-05-22   |
|        643 | ryan      | cash         |           8000 | NULL         | 2023-06-30   |
|        742 | cal       | item         |           5000 | toys         | 2023-09-04   |
|        972 | adward    | cash         |          10000 | NULL         | 2023-11-12   |
+------------+-----------+--------------+----------------+--------------+--------------+
5 rows in set (0.02 sec)
```



```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petpals>
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Programs\Python\Python312
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '58095
allenge\petpals\test.py'
Enter donor name: akilesh
Enter donation amount: 3000
Donation recorded successfully!
```

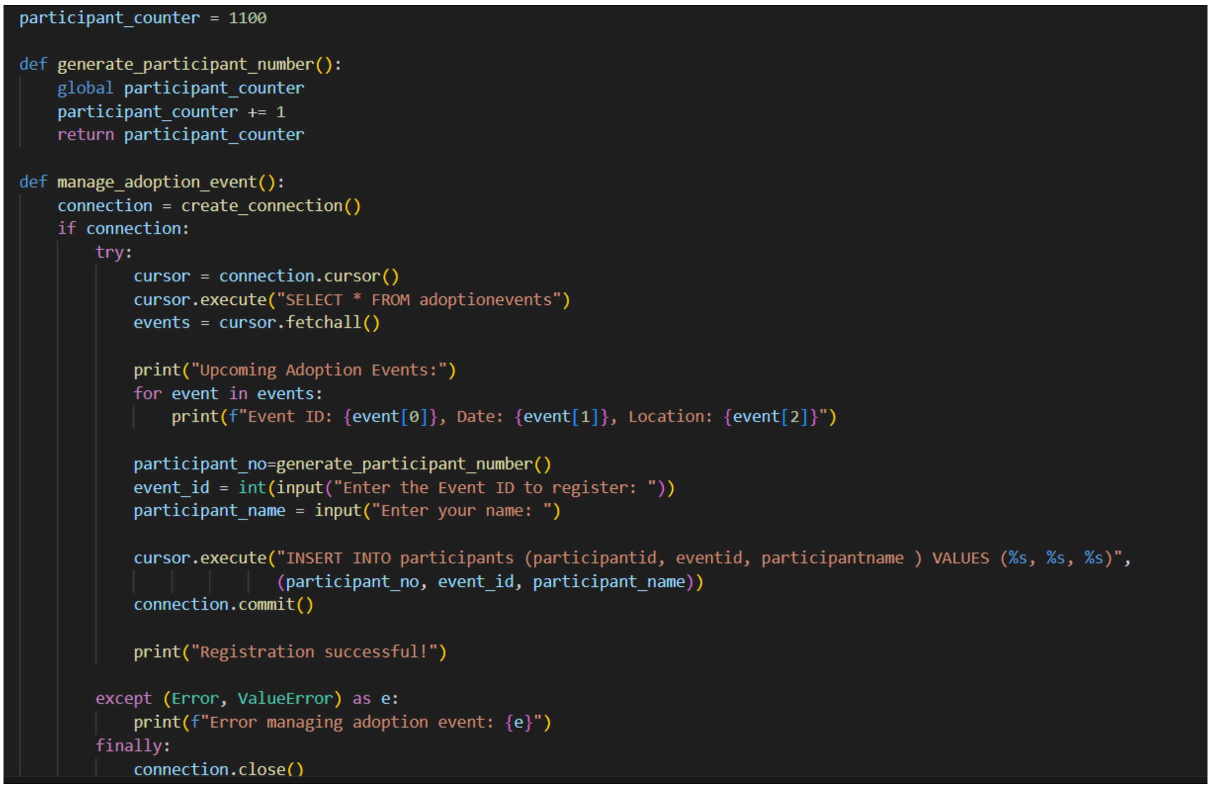
After adding donor details



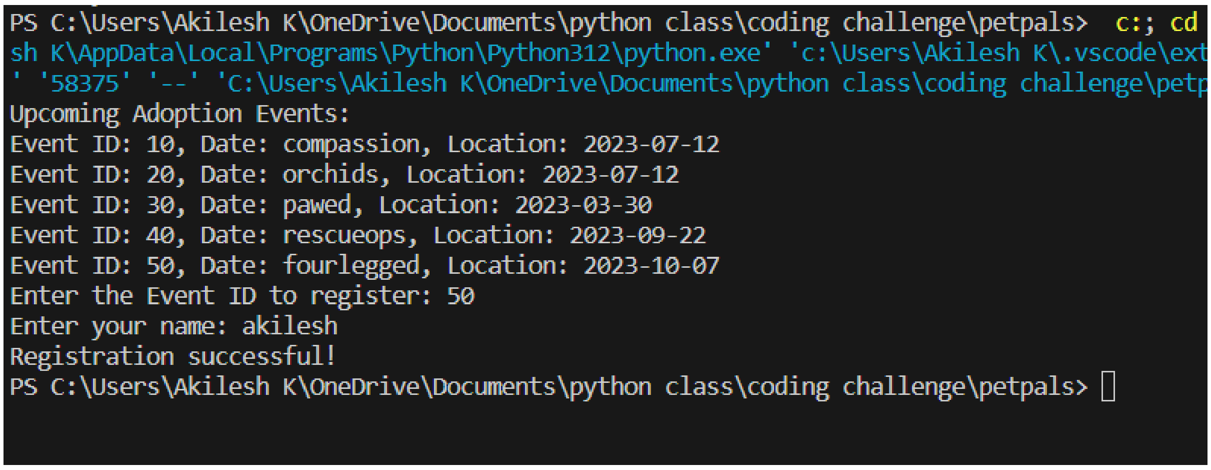
```
mysql> select * from donations;
+------------+-----------+--------------+----------------+--------------+--------------+
| donationid | donarname | donationtype | donationamount | donationitem | donationdate |
+------------+-----------+--------------+----------------+--------------+--------------+
|        123 | harry     | item         |           5000 | blanket      | 2023-10-07   |
|        425 | andrew    | cash         |           5000 | NULL         | 2023-05-22   |
|        643 | ryan      | cash         |           8000 | NULL         | 2023-06-30   |
|        742 | cal       | item         |           5000 | toys         | 2023-09-04   |
|        972 | adward    | cash         |          10000 | NULL         | 2023-11-12   |
|       1001 | akilesh   | NULL         |           3000 | NULL         | 2023-12-22   |
+------------+-----------+--------------+----------------+--------------+--------------+
6 rows in set (0.00 sec)
```

C) Adoption Event Management

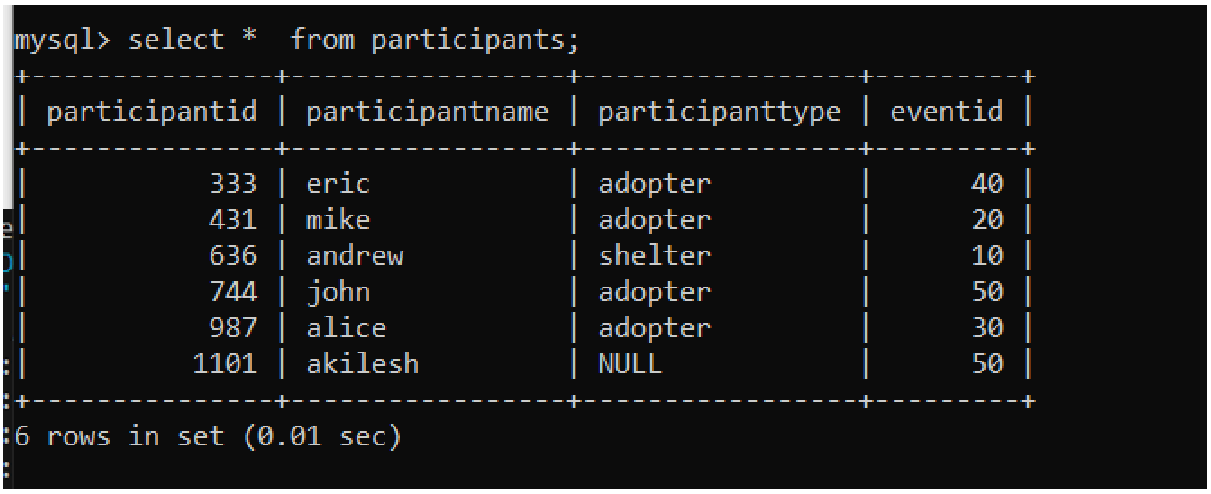
```
participant_counter = 1100


def generate_participant_number():
    global participant_counter
    participant_counter += 1
    return participant_counter


def manage_adoption_event():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM adoptionevents")
            events = cursor.fetchall()

            print("Upcoming Adoption Events:")
            for event in events:
                print(f"Event ID: {event[0]}, Date: {event[1]}, Location: {event[2]}")

            participant_no=generate_participant_number()
            event_id = int(input("Enter the Event ID to register: "))
            participant_name = input("Enter your name: ")


            cursor.execute("INSERT INTO participants (participantid, eventid, participantname ) VALUES
(%s, %s, %s)",
                            (participant_no, event_id, participant_name))
            connection.commit()


            print("Registration successful!")


        except (Error, ValueError) as e:
            print(f"Error managing adoption event: {e}")
        finally:
            connection.close()
```



```python
participant_counter = 1100

def generate_participant_number():
    global participant_counter
    participant_counter += 1
    return participant_counter

def manage_adoption_event():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM adoptionevents")
            events = cursor.fetchall()

            print("Upcoming Adoption Events:")
            for event in events:
                print(f"Event ID: {event[0]}, Date: {event[1]}, Location: {event[2]}")

            participant_no=generate_participant_number()
            event_id = int(input("Enter the Event ID to register: "))
            participant_name = input("Enter your name: ")

            cursor.execute("INSERT INTO participants (participantid, eventid, participantname ) VALUES (%s, %s, %s)",
                           (participant_no, event_id, participant_name))
            connection.commit()

            print("Registration successful!")

        except (Error, ValueError) as e:
            print(f"Error managing adoption event: {e}")
        finally:
            connection.close()
```



```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petpals>  c:; cd
sh K\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\Akilesh K\.vscode\ext
' '58375' '--' 'C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petp
Upcoming Adoption Events:
Event ID: 10, Date: compassion, Location: 2023-07-12
Event ID: 20, Date: orchids, Location: 2023-07-12
Event ID: 30, Date: pawed, Location: 2023-03-30
Event ID: 40, Date: rescueops, Location: 2023-09-22
Event ID: 50, Date: fourlegged, Location: 2023-10-07
Enter the Event ID to register: 50
Enter your name: akilesh
Registration successful!
PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petpals>
```



```
mysql> select *  from participants;
+---------------+-----------------+-----------------+---------+
| participantid | participantname | participanttype | eventid |
+---------------+-----------------+-----------------+---------+
|           333 | eric            | adopter         |      40 |
|           431 | mike            | adopter         |      20 |
|           636 | andrew          | shelter         |      10 |
|           744 | john            | adopter         |      50 |
|           987 | alice           | adopter         |      30 |
|          1101 | akilesh         | NULL            |      50 |
+---------------+-----------------+-----------------+---------+
6 rows in set (0.01 sec)
```