VIJAY R

Superset ID : 5371616

Saveetha Engineering College

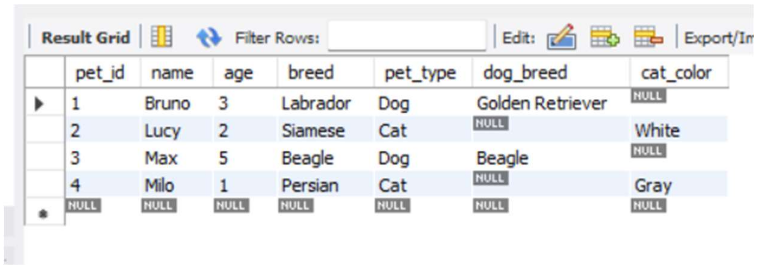
Coding Challenges: PetPals, The Pet Adoption Platform

Create SQL Schema from the pet and user class, use the class attributes for table column names.

Tables:

1.pets:

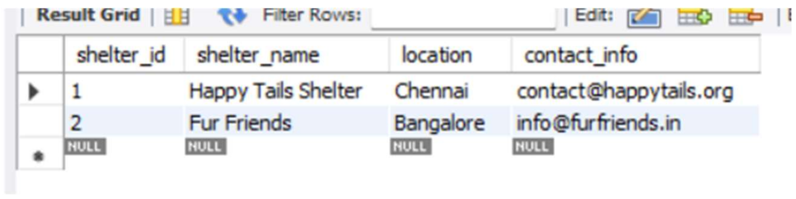
```
CREATE TABLE pets (
   pet_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   age INT NOT NULL,
   breed VARCHAR(100) NOT NULL,
   pet_type ENUM('Dog', 'Cat') NOT NULL,
   dog_breed VARCHAR(100),   -- Only for dogs
   cat_color VARCHAR(50)     -- Only for cats
);

INSERT INTO pets (name, age, breed, pet_type, dog_breed, cat_color) VALUES
('Bruno', 3, 'Labrador', 'Dog', 'Golden Retriever', NULL),
('Lucy', 2, 'Siamese', 'Cat', NULL, 'White'),
('Max', 5, 'Beagle', 'Dog', 'Beagle', NULL),
('Milo', 1, 'Persian', 'Cat', NULL, 'Gray');
```



	pet_id	name	age	breed	pet_type	dog_breed	cat_color
▶	1	Bruno	3	Labrador	Dog	Golden Retriever	NULL
	2	Lucy	2	Siamese	Cat	NULL	White
	3	Max	5	Beagle	Dog	Beagle	NULL
	4	Milo	1	Persian	Cat	NULL	Gray
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. pet_shelters :

```
CREATE TABLE pet_shelters (
    shelter_id INT AUTO_INCREMENT PRIMARY KEY,
    shelter_name VARCHAR(100) NOT NULL,
    location VARCHAR(100),
    contact_info VARCHAR(100)
);

INSERT INTO pet_shelters (shelter_name, location, contact_info) VALUES
('Happy Tails Shelter', 'Chennai', 'contact@happytails.org'),
('Fur Friends', 'Bangalore', 'info@furfriends.in');
```

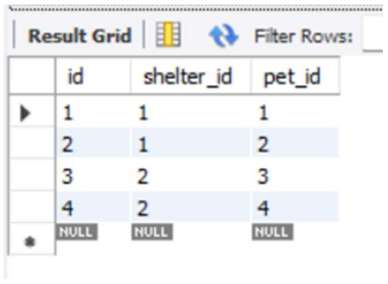


	shelter_id	shelter_name	location	contact_info
▶	1	Happy Tails Shelter	Chennai	contact@happytails.org
	2	Fur Friends	Bangalore	info@furfriends.in
*	NULL	NULL	NULL	NULL

3. shelter_pets:

```
CREATE TABLE shelter_pets (
    id INT AUTO_INCREMENT PRIMARY KEY,
    shelter_id INT,
    pet_id INT,
    FOREIGN KEY (shelter_id) REFERENCES pet_shelters(shelter_id),
    FOREIGN KEY (pet_id) REFERENCES pets(pet_id)
);
```

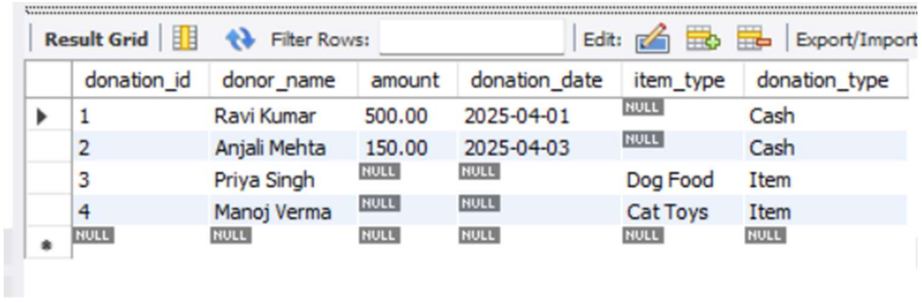
```
INSERT INTO shelter_pets (shelter_id, pet_id) VALUES
(1, 1), (1, 2), (2, 3), (2, 4);
```



id	shelter_id	pet_id
1	1	1
2	1	2
3	2	3
4	2	4
NULL	NULL	NULL

4.donations:

```
CREATE TABLE donations (
    donation_id INT AUTO_INCREMENT PRIMARY KEY,
    donor_name VARCHAR(100) NOT NULL,
    amount DECIMAL(10,2),         -- Nullable for item donation
    donation_date DATE,           -- Only for cash donation
    item_type VARCHAR(100),       -- Only for item donation
    donation_type ENUM('Cash', 'Item') NOT NULL
);

INSERT INTO donations (donor_name, amount, donation_date, donation_type) VALUES
('Ravi Kumar', 500.00, '2025-04-01', 'Cash'),
('Anjali Mehta', 150.00, '2025-04-03', 'Cash');

INSERT INTO donations (donor_name, item_type, donation_type) VALUES
('Priya Singh', 'Dog Food', 'Item'),
('Manoj Verma', 'Cat Toys', 'Item');
```

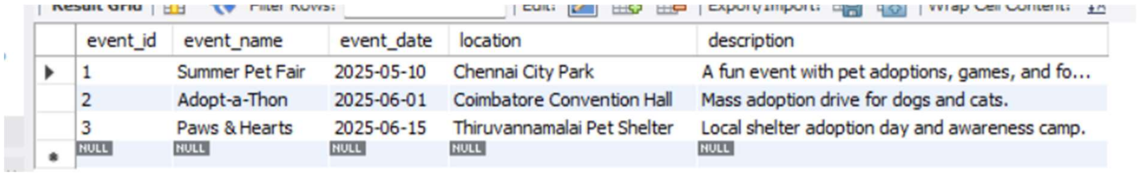


donation_id	donor_name	amount	donation_date	item_type	donation_type
1	Ravi Kumar	500.00	2025-04-01	NULL	Cash
2	Anjali Mehta	150.00	2025-04-03	NULL	Cash
3	Priya Singh	NULL	NULL	Dog Food	Item
4	Manoj Verma	NULL	NULL	Cat Toys	Item
NULL	NULL	NULL	NULL	NULL	NULL

5. adoption_events:

```
CREATE TABLE adoption_events (
    event_id INT AUTO_INCREMENT PRIMARY KEY,
    event_name VARCHAR(100),
    event_date DATE,
    location VARCHAR(100),
    description TEXT
);
```

```
INSERT INTO adoption_events (event_name, event_date, location, description) VALUES
('Summer Pet Fair', '2025-05-10', 'Chennai City Park', 'A fun event with pet adoptions, games, and food stalls.'),
('Adopt-a-Thon', '2025-06-01', 'Coimbatore Convention Hall', 'Mass adoption drive for dogs and cats.'),
('Paws & Hearts', '2025-06-15', 'Thiruvannamalai Pet Shelter', 'Local shelter adoption day and awareness camp.');
```

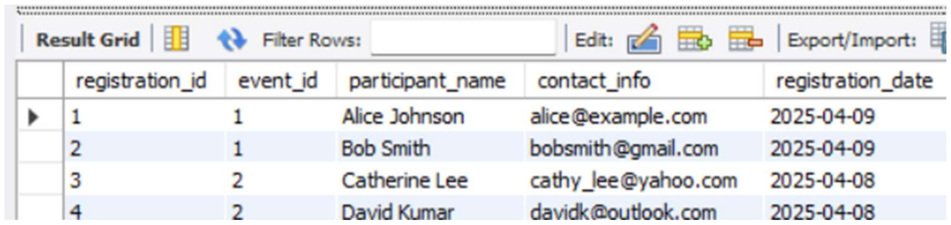


event_id	event_name	event_date	location	description
1	Summer Pet Fair	2025-05-10	Chennai City Park	A fun event with pet adoptions, games, and fo...
2	Adopt-a-Thon	2025-06-01	Coimbatore Convention Hall	Mass adoption drive for dogs and cats.
3	Paws & Hearts	2025-06-15	Thiruvannamalai Pet Shelter	Local shelter adoption day and awareness camp.
NULL	NULL	NULL	NULL	NULL

6. event_registrations:

```
CREATE TABLE event_registrations (
    registration_id INT AUTO_INCREMENT PRIMARY KEY,
    event_id INT,
    participant_name VARCHAR(100),
    contact_info VARCHAR(100),
    registration_date DATE,
    FOREIGN KEY (event_id) REFERENCES adoption_events(event_id)
);
```

```
INSERT INTO event_registrations (event_id, participant_name, contact_info,
registration_date)
VALUES
(1, 'Alice Johnson', 'alice@example.com', '2025-04-09'),
(1, 'Bob Smith', 'bobsmith@gmail.com', '2025-04-09'),
(2, 'Catherine Lee', 'cathy_lee@yahoo.com', '2025-04-08'),
(2, 'David Kumar', 'davidk@outlook.com', '2025-04-08');
```



registration_id	event_id	participant_name	contact_info	registration_date
1	1	Alice Johnson	alice@example.com	2025-04-09
2	1	Bob Smith	bobsmith@gmail.com	2025-04-09
3	2	Catherine Lee	cathy_lee@yahoo.com	2025-04-08
4	2	David Kumar	davidk@outlook.com	2025-04-08

1.Create and implement the mentioned class and the structure in your application.

Pet Class:

Attributes:

- Name (string): The name of the pet.
- Age (int): The age of the pet.
- Breed (string): The breed of the pet.

Methods:

- Constructor to initialize Name, Age, and Breed.
- Getters and setters for attributes.
- ToString() method to provide a string representation of the pet.

entity/pet.py:

```
class Pet:
    def __init__(self, name: str, age: int, breed: str):
        self.name = name
        self.age = age
        self.breed = breed

    def get_name(self): return self.name
    def set_name(self, name): self.name = name
    def get_age(self): return self.age
    def set_age(self, age): self.age = age

    def get_breed(self): return self.breed
    def set_breed(self, breed): self.breed = breed

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

Dog Class (Inherits from Pet):

Additional Attributes:

- DogBreed (string): The specific breed of the dog.

Additional Methods:

- Constructor to initialize DogBreed.
- Getters and setters for DogBreed.

Cat Class (Inherits from Pet):

Additional Attributes:

- CatColor (string): The color of the cat.

Additional Methods:

- Constructor to initialize CatColor.
- Getters and setters for CatColor.

entity/dog.py:

```
from entity.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 get_dog_breed(self): return self.dog_breed
    def set_dog_breed(self, breed): self.dog_breed = breed
```

entity/cat.py:

```
from entity.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 get_cat_color(self): return self.cat_color
    def set_cat_color(self, color): self.cat_color = color
```

3.PetShelter Class:

Attributes:

- availablePets (List of Pet): A list to store available pets for adoption.

Methods:

- AddPet(Pet pet): Adds a pet to the list of available pets.
- RemovePet(Pet pet): Removes a pet from the list of available pets.
- ListAvailablePets(): Lists all available pets in the shelter.

entity/petshelter.py:

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

    def add_pet(self, pet):
        self.available_pets.append(pet)

    def remove_pet(self, pet):
        self.available_pets.remove(pet)

    def list_available_pets(self):
        for pet in self.available_pets:
            print(pet)
```

4.Donation Class (Abstract):

Attributes:

- DonorName (string): The name of the donor.
- Amount (decimal): The donation amount.

Methods:

- Constructor to initialize DonorName and Amount.
- Abstract method RecordDonation() to record the donation (to be implemented in derived classes).

entity/donation.py:

```
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
```

CashDonation Class (Derived from Donation):

Additional Attributes:

- DonationDate (DateTime): The date of the cash donation.

Additional Methods:

- Constructor to initialize DonationDate.
- Implementation of RecordDonation() to record a cash donation.

entity/cash_donation.py:

```
from entity.donation import Donation
from datetime import datetime

class CashDonation(Donation):
    def __init__(self, donor_name, amount, donation_date=None):
        super().__init__(donor_name, amount)
        self.donation_date = donation_date or datetime.now()
    def record_donation(self):
        print(f"Recorded cash donation of ₹{self.amount} by {self.donor_name} on
{self.donation_date}")
```

ItemDonation Class (Derived from Donation):

Additional Attributes:

- ItemType (string): The type of item donated (e.g., food, toys).

Additional Methods:

- Constructor to initialize ItemType.
- Implementation of RecordDonation() to record an item donation.

entity/item_donation.py:

```
from entity.donation import Donation

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"Recorded item donation ({self.item_type}) worth ₹{self.amount} by
{self.donor_name}")
```

5.IAdoptable Interface/Abstract Class:

Methods:

- Adopt(): An abstract method to handle the adoption process.

AdoptionEvent Class:

Attributes:

- Participants (List of IAdoptable): A list of participants (shelters and adopters) in the adoption event.

Methods:

- HostEvent(): Hosts the adoption event.
- RegisterParticipant(IAdoptable participant): Registers a participant for the event.

entity/iadoptable.py:

```
from abc import ABC, abstractmethod

class IAdoptable(ABC):
    @abstractmethod
    def adopt(self):
        pass
```

entity/adoption_event.py:

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

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

    def host_event(self):
        print("Adoption Event Started!")
        for p in self.participants:
            p.adopt()
```

6.Exceptions handling:

Create and implement the following exceptions in your application.

• Invalid Pet Age Handling:

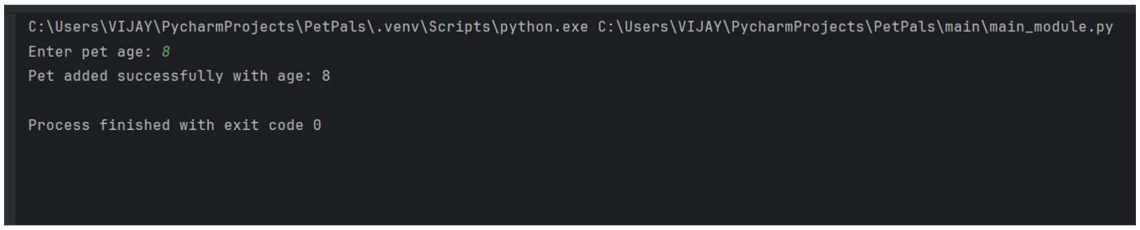
In the Pet Adoption Platform, when adding a new pet to a shelter, the age of the pet should be a positive integer. Write a program that prompts the user to input the age of a pet. Implement exception handling to ensure that the input is a positive integer. If the input is not valid, catch the exception and display an error message. If the input is valid, add the pet to the shelter.

exception/InvalidPetAgeException.py:

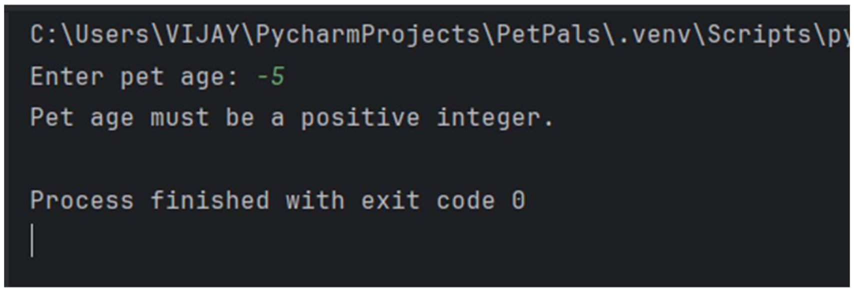
```
class InvalidPetAgeException(Exception):
    def __init__(self, message="Pet age must be a positive integer."):
        super().__init__(message)
```

Main.py:

```
from exception.InvalidPetAgeException import InvalidPetAgeException

try:
    age = int(input("Enter pet age: "))
    if age <= 0:
        raise InvalidPetAgeException()
    print("Pet added successfully with age:", age)
except ValueError:
    print("Please enter a valid integer.")
except InvalidPetAgeException as e:
    print(e)
```



```
C:\Users\VIJAY\PycharmProjects\PetPals\.venv\Scripts\python.exe C:\Users\VIJAY\PycharmProjects\PetPals\main\main_module.py
Enter pet age: 8
Pet added successfully with age: 8

Process finished with exit code 0
```



```
C:\Users\VIJAY\PycharmProjects\PetPals\.venv\Scripts\py
Enter pet age: -5
Pet age must be a positive integer.

Process finished with exit code 0
```

- **Null Reference Exception Handling:**

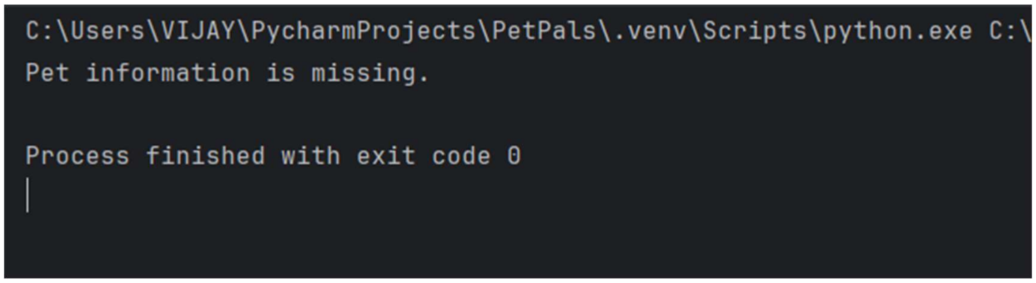
  In the Pet Adoption Platform, when displaying the list of available pets in a shelter, it's important to handle situations where a pet's properties (e.g., Name, Age) might be null. Implement exception handling to catch null reference exceptions when accessing properties of pets in the shelter and display a message indicating that the information is missing.

exception/NullReferenceException.py:

```
class NullReferenceException(Exception):
    def __init__(self, message="Pet information is missing."):
        super().__init__(message)
```

Main.py:

```
class Pet:
    def __init__(self, name=None, age=None):
        self.name = name
        self.age = age

from exception.NullReferenceException import NullReferenceException

try:
    pet = Pet(name=None, age=3)  # Name is missing
    if pet is None or pet.name is None or pet.age is None:
        raise NullReferenceException()
    print(f"{pet.name} - {pet.age}")
except NullReferenceException as e:
    print(e)
```



```
C:\Users\VIJAY\PycharmProjects\PetPals\.venv\Scripts\python.exe C:\
Pet information is missing.

Process finished with exit code 0
```

• **Insufficient Funds Exception:**

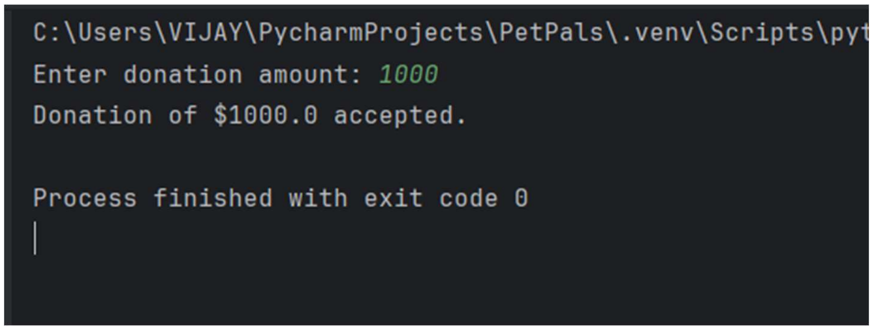
Suppose the Pet Adoption Platform allows users to make cash donations to shelters. Write a program that prompts the user to enter the donation amount. Implement exception handling to catch situations where the donation amount is less than a minimum allowed amount (e.g., $10). If the donation amount is insufficient, catch the exception and display an error message. Otherwise, process the donation.

exception/InsufficientFundsException.py:

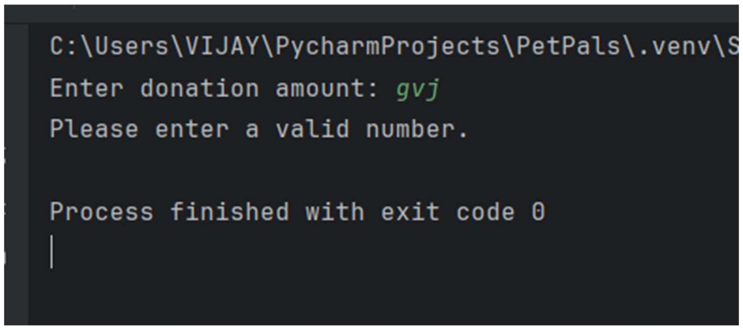
```
class InsufficientFundsException(Exception):
    def __init__(self, message="Donation must be at least $10."):
        super().__init__(message)
```

Main.py:

```
from exception.InsufficientFundsException import InsufficientFundsException

try:
    amount = float(input("Enter donation amount: "))
    if amount < 10:
        raise InsufficientFundsException()
    print(f"Donation of ${amount} accepted.")
except ValueError:
    print("Please enter a valid number.")
except InsufficientFundsException as e:
    print(e)
```



```
C:\Users\VIJAY\PycharmProjects\PetPals\.venv\Scripts\pyt
Enter donation amount: 1000
Donation of $1000.0 accepted.

Process finished with exit code 0
```



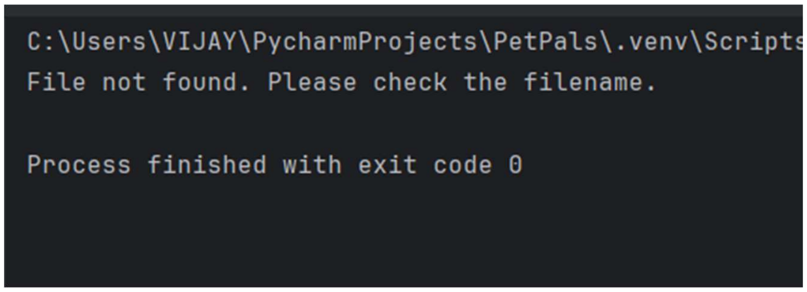
```
C:\Users\VIJAY\PycharmProjects\PetPals\.venv\S
Enter donation amount: gvj
Please enter a valid number.

Process finished with exit code 0
```

- **File Handling Exception:**

    In the Pet Adoption Platform, there might be scenarios where the program needs to read data from a file (e.g., a list of pets in a shelter). Write a program that attempts to read data from a file. Implement exception handling to catch any file-related exceptions (e.g., FileNotFoundException) and display an error message if the file is not found or cannot be read.

Main.py:

```
try:
    with open("pet.txt", "r") as file:
        data = file.read()
        print(data)
except FileNotFoundError:
    print("File not found. Please check the filename.")
except IOError:
    print("Error reading the file.")
```



```
C:\Users\VIJAY\PycharmProjects\PetPals\.venv\Scripts
File not found. Please check the filename.

Process finished with exit code 0
```

• Custom Exception for Adoption Errors:

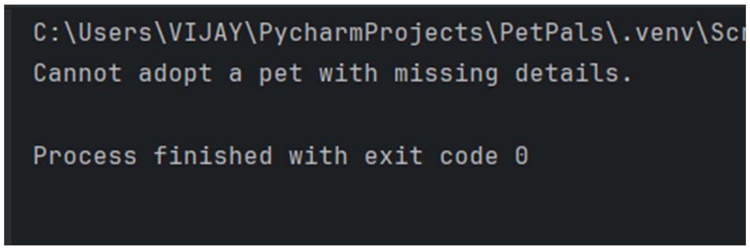
Design a custom exception class called AdoptionException that inherits from Exception. In the Pet Adoption Platform, use this custom exception to handle adoption-related errors, such as attempting to adopt a pet that is not available or adopting a pet with missing information. Create instances of AdoptionException with different error messages and catch them appropriately in your program.

exception/AdoptionException.py:

```
class AdoptionException(Exception):
    def __init__(self, message="Adoption failed due to invalid pet details or availability."):
        super().__init__(message)
```

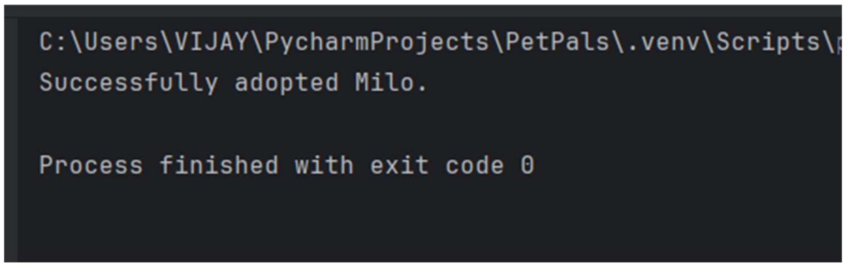
Main.py:

```
from exception.AdoptionException import AdoptionException


def adopt_pet(pet):

    if pet is None or pet.name is None:

        raise AdoptionException("Cannot adopt a pet with missing details.")

    print(f"Successfully adopted {pet.name}.")


try:

    pet = None  # simulate null or incomplete pet

    adopt_pet(pet)

except AdoptionException as e:

    print(e)
```


```
C:\Users\VIJAY\PycharmProjects\PetPals\.venv\Sc
Cannot adopt a pet with missing details.

Process finished with exit code 0
```

Main.py:

```
from exception.AdoptionException import AdoptionException
from entity.pet import Pet

def adopt_pet(pet):
    if pet is None or pet.get_name() is None:
        raise AdoptionException("Cannot adopt a pet with missing details.")
    print(f"Successfully adopted {pet.get_name()}.")

try:
    pet = Pet("Milo", 2, "Golden Retriever")  # Only 3 arguments
    adopt_pet(pet)
except AdoptionException as e:
    print(e)
```


```
C:\Users\VIJAY\PycharmProjects\PetPals\.venv\Scripts\
Successfully adopted Milo.

Process finished with exit code 0
```

7.Database Connectivity:

util/DBConnUtil.py:

```
import mysql.connector
from util.DBPropertyUtil import load_db_properties

def get_db_connection():
    config = load_db_properties()
    try:
        connection = mysql.connector.connect(
            host=config["host"],
            port=int(config["port"]),
            user=config["user"],
            password=config["password"],
            database=config["database"]
        )
        return connection
    except Exception as e:
        print("Error while establishing DB connection:", e)
        return None
```

util/DBPropertyUtil.py:

```
import os

def load_db_properties():
    props = {}
    try:
        path = os.path.join(os.path.dirname(os.path.dirname(__file__)), "resources",
"db.properties")
        print("Loading DB properties from:", path)
        with open(path, "r") as f:
            for line in f:
                line = line.strip()
                if line and not line.startswith("#") and "=" in line:
                    key, value = line.split("=", 1)
                    props[key.strip()] = value.strip()
    except FileNotFoundError:
        print("db.properties file not found at path.")
    except Exception as e:
        print(f"Error loading database properties: {e}")
    return props
```

Create and implement the following tasks in your application.

- **Displaying Pet Listings:**

  Develop a program that connects to the database and retrieves a list of available pets from the "pets" table. Display this list to the user. Ensure that the program handles database connectivity exceptions gracefully, including cases where the database is unreachable.

- **Donation Recording:**

  Create a program that records cash donations made by donors. Allow the user to input donor information and the donation amount and insert this data into the "donations" table in the database. Handle exceptions related to database operations, such as database errors or invalid inputs.

• **Adoption Event Management:**

Build a program that connects to the database and retrieves information about upcoming adoption events from the "adoption_events" table. Allow the user to register for an event by adding their details to the "participants" table. Ensure that the program handles database connectivity and insertion exceptions properly

main/main_module.py:

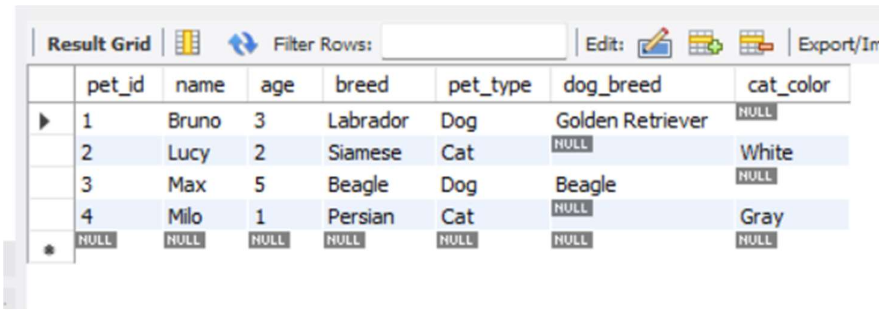
```
import sys
from util.DBConnUtil import get_db_connection
import mysql.connector


def list_available_pets():
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("SELECT pet_id, name, age, breed, pet_type, dog_breed, cat_color
FROM pets")
        pets = cursor.fetchall()
        print("\n--- Available Pets ---")
        for pet in pets:
            print(f"ID: {pet[0]}, Name: {pet[1]}, Age: {pet[2]}, Breed: {pet[3]}, Type: {pet[4]}",
end='')
            if pet[4] == 'Dog':
                print(f", Dog Breed: {pet[5]}")
            elif pet[4] == 'Cat':
                print(f", Cat Color: {pet[6]}")
            else:
                print()
    except Exception as e:
        print("Error while fetching pet data:", e)
    finally:
        if 'conn' in locals() and conn.is_connected():
            conn.close()


def record_cash_donation():
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        donor_name = input("Enter Donor Name: ")
        amount = float(input("Enter Donation Amount: "))
        donation_type = 'Cash'
        sql = "INSERT INTO donations (donor_name, amount, donation_date, donation_type)
```

```
VALUES (%s, %s, CURDATE(), %s)"
        values = (donor_name, amount, donation_type)
        cursor.execute(sql, values)
        conn.commit()
        print("Donation recorded successfully.")
    except Exception as e:
        print("Error while recording donation:", e)
    finally:
        if 'conn' in locals() and conn.is_connected():
            conn.close()


def view_adoption_events():
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("SELECT event_id, event_name, event_date, location FROM
adoption_events")
        events = cursor.fetchall()
        print("\n--- Adoption Events ---")
        for event in events:
            print(f"ID: {event[0]}, Name: {event[1]}, Date: {event[2]}, Location: {event[3]}")
    except Exception as e:
        print("Error while fetching events:", e)
    finally:
        if 'conn' in locals() and conn.is_connected():
            conn.close()


def register_for_event():
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        event_id = int(input("Enter Event ID to register for: "))
        participant_name = input("Enter Your Name: ")
        contact_info = input("Enter Contact Info: ")
        sql = "INSERT INTO event_registrations (event_id, participant_name, contact_info,
registration_date) VALUES (%s, %s, %s, CURDATE())"
        values = (event_id, participant_name, contact_info)
        cursor.execute(sql, values)
        conn.commit()
        print("Successfully registered for the event.")
    except Exception as e:
        print("Error during event registration:", e)
    finally:
        if 'conn' in locals() and conn.is_connected():
```

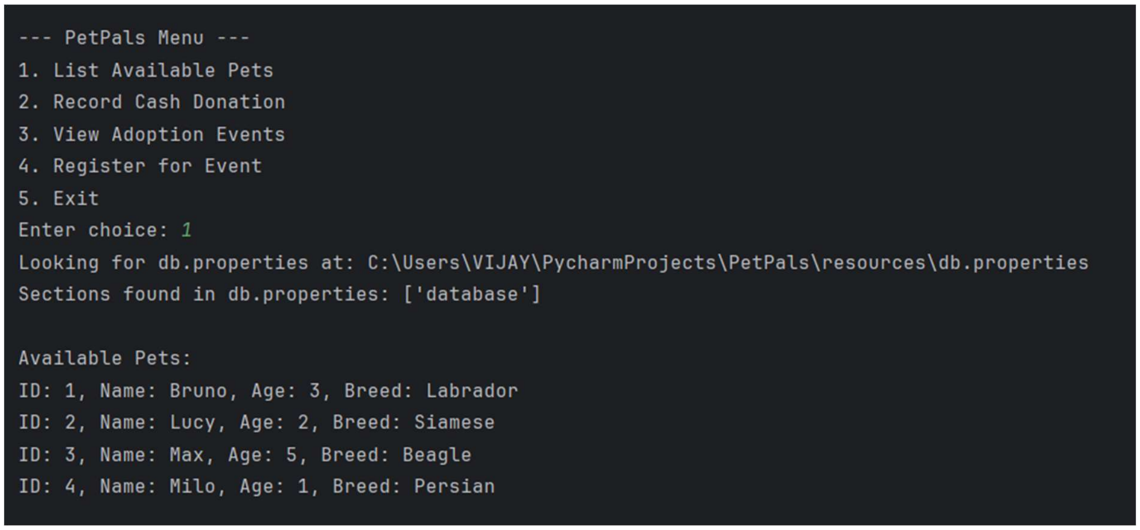
```
        conn.close()


def main():
    while True:
        print("\n--- PetPals Menu ---")
        print("1. List Available Pets")
        print("2. Record Cash Donation")
        print("3. View Adoption Events")
        print("4. Register for Event")
        print("5. Exit")

        choice = input("Enter choice: ")

        if choice == '1':
            list_available_pets()
        elif choice == '2':
            record_cash_donation()
        elif choice == '3':
            view_adoption_events()
        elif choice == '4':
            register_for_event()
        elif choice == '5':
            print("Exiting... Goodbye!")
            sys.exit()
        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    main()
```

OUTPUT:

1. List Available Pets:

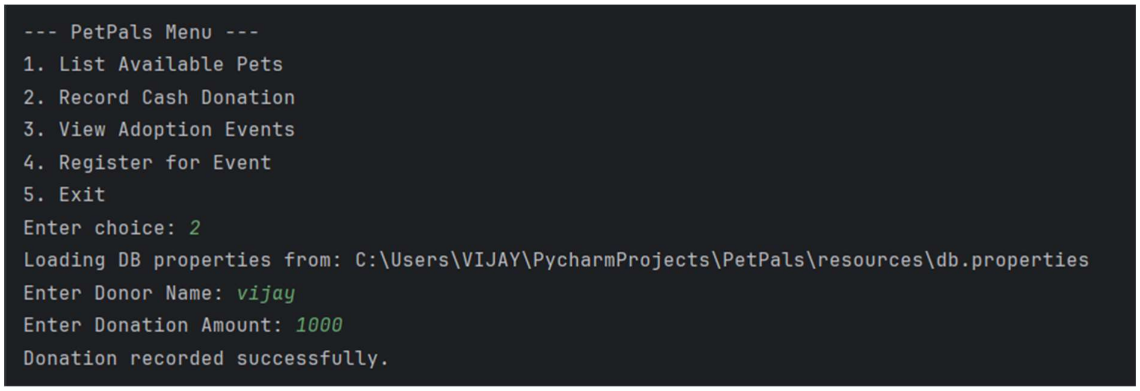


pet_id	name	age	breed	pet_type	dog_breed	cat_color
1	Bruno	3	Labrador	Dog	Golden Retriever	NULL
2	Lucy	2	Siamese	Cat	NULL	White
3	Max	5	Beagle	Dog	Beagle	NULL
4	Milo	1	Persian	Cat	NULL	Gray
NULL	NULL	NULL	NULL	NULL	NULL	NULL

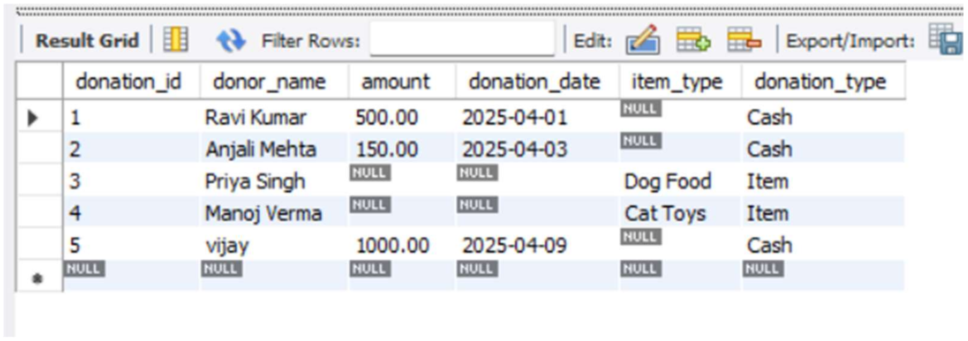


```
--- PetPals Menu ---
1. List Available Pets
2. Record Cash Donation
3. View Adoption Events
4. Register for Event
5. Exit
Enter choice: 1
Looking for db.properties at: C:\Users\VIJAY\PycharmProjects\PetPals\resources\db.properties
Sections found in db.properties: ['database']

Available Pets:
ID: 1, Name: Bruno, Age: 3, Breed: Labrador
ID: 2, Name: Lucy, Age: 2, Breed: Siamese
ID: 3, Name: Max, Age: 5, Breed: Beagle
ID: 4, Name: Milo, Age: 1, Breed: Persian
```

2. Record Cash Donation:

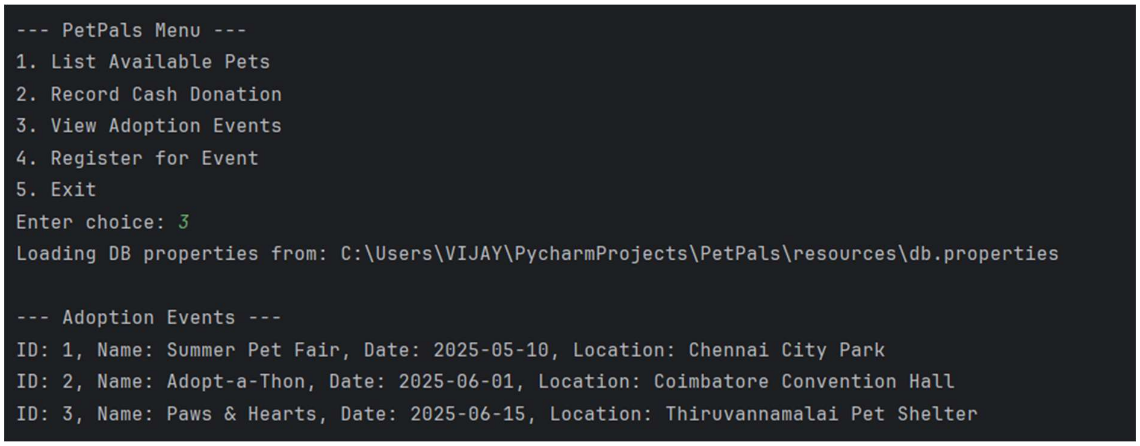


```
--- PetPals Menu ---
1. List Available Pets
2. Record Cash Donation
3. View Adoption Events
4. Register for Event
5. Exit
Enter choice: 2
Loading DB properties from: C:\Users\VIJAY\PycharmProjects\PetPals\resources\db.properties
Enter Donor Name: vijay
Enter Donation Amount: 1000
Donation recorded successfully.
```

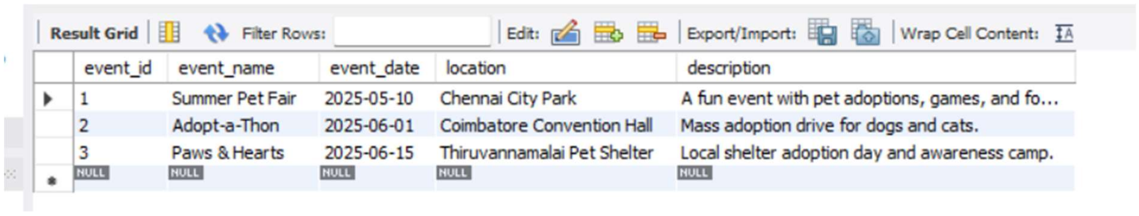


donation_id	donor_name	amount	donation_date	item_type	donation_type
1	Ravi Kumar	500.00	2025-04-01	NULL	Cash
2	Anjali Mehta	150.00	2025-04-03	NULL	Cash
3	Priya Singh	NULL	NULL	Dog Food	Item
4	Manoj Verma	NULL	NULL	Cat Toys	Item
5	vijay	1000.00	2025-04-09	NULL	Cash
NULL	NULL	NULL	NULL	NULL	NULL

3. View Adoption Events:

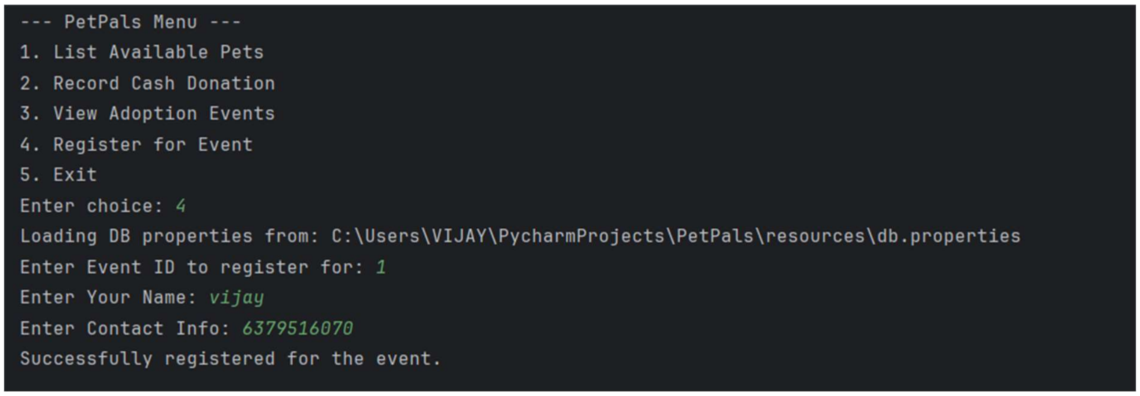


```
--- PetPals Menu ---
1. List Available Pets
2. Record Cash Donation
3. View Adoption Events
4. Register for Event
5. Exit
Enter choice: 3
Loading DB properties from: C:\Users\VIJAY\PycharmProjects\PetPals\resources\db.properties

--- Adoption Events ---
ID: 1, Name: Summer Pet Fair, Date: 2025-05-10, Location: Chennai City Park
ID: 2, Name: Adopt-a-Thon, Date: 2025-06-01, Location: Coimbatore Convention Hall
ID: 3, Name: Paws & Hearts, Date: 2025-06-15, Location: Thiruvannamalai Pet Shelter
```

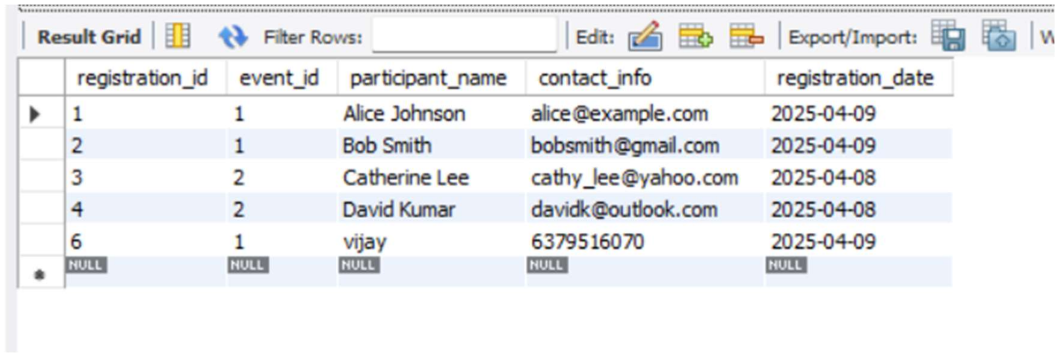


event_id	event_name	event_date	location	description
1	Summer Pet Fair	2025-05-10	Chennai City Park	A fun event with pet adoptions, games, and fo...
2	Adopt-a-Thon	2025-06-01	Coimbatore Convention Hall	Mass adoption drive for dogs and cats.
3	Paws & Hearts	2025-06-15	Thiruvannamalai Pet Shelter	Local shelter adoption day and awareness camp.
NULL	NULL	NULL	NULL	NULL

4. Register for Event:



```
--- PetPals Menu ---
1. List Available Pets
2. Record Cash Donation
3. View Adoption Events
4. Register for Event
5. Exit
Enter choice: 4
Loading DB properties from: C:\Users\VIJAY\PycharmProjects\PetPals\resources\db.properties
Enter Event ID to register for: 1
Enter Your Name: vijay
Enter Contact Info: 6379516070
Successfully registered for the event.
```



registration_id	event_id	participant_name	contact_info	registration_date
1	1	Alice Johnson	alice@example.com	2025-04-09
2	1	Bob Smith	bobsmith@gmail.com	2025-04-09
3	2	Catherine Lee	cathy_lee@yahoo.com	2025-04-08
4	2	David Kumar	davidk@outlook.com	2025-04-08
6	1	vijay	6379516070	2025-04-09
NULL	NULL	NULL	NULL	NULL