CODING CHALLENGE JAVA : PETPALS
SUBMITTED BY : SRINIDHI. V- BATCH 5

• The following Directory structure is to be followed in the application.

entity/model

▪ Create entity classes in this package. All entity class should not have any business logic.

dao

▪ Create Service Provider Interface/Abstract Class to showcase functionalities.

▪ Create the implementation class for the above Interface/Abstract Class with db interaction.

exception

▪ Create user defined exceptions in this package and handle exceptions whenever needed.

util

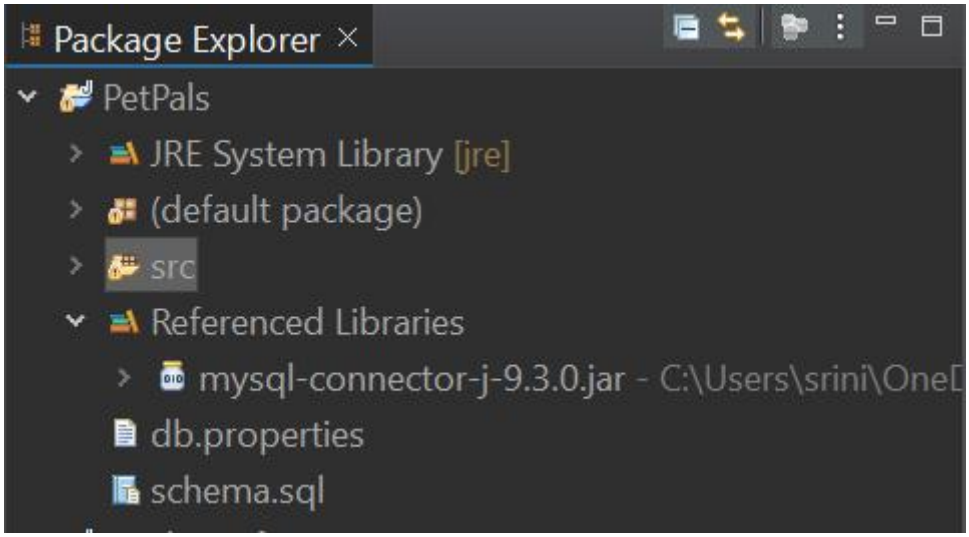
▪ Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.

▪ Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object (Use method defined in DBPropertyUtil class to get the connection String).

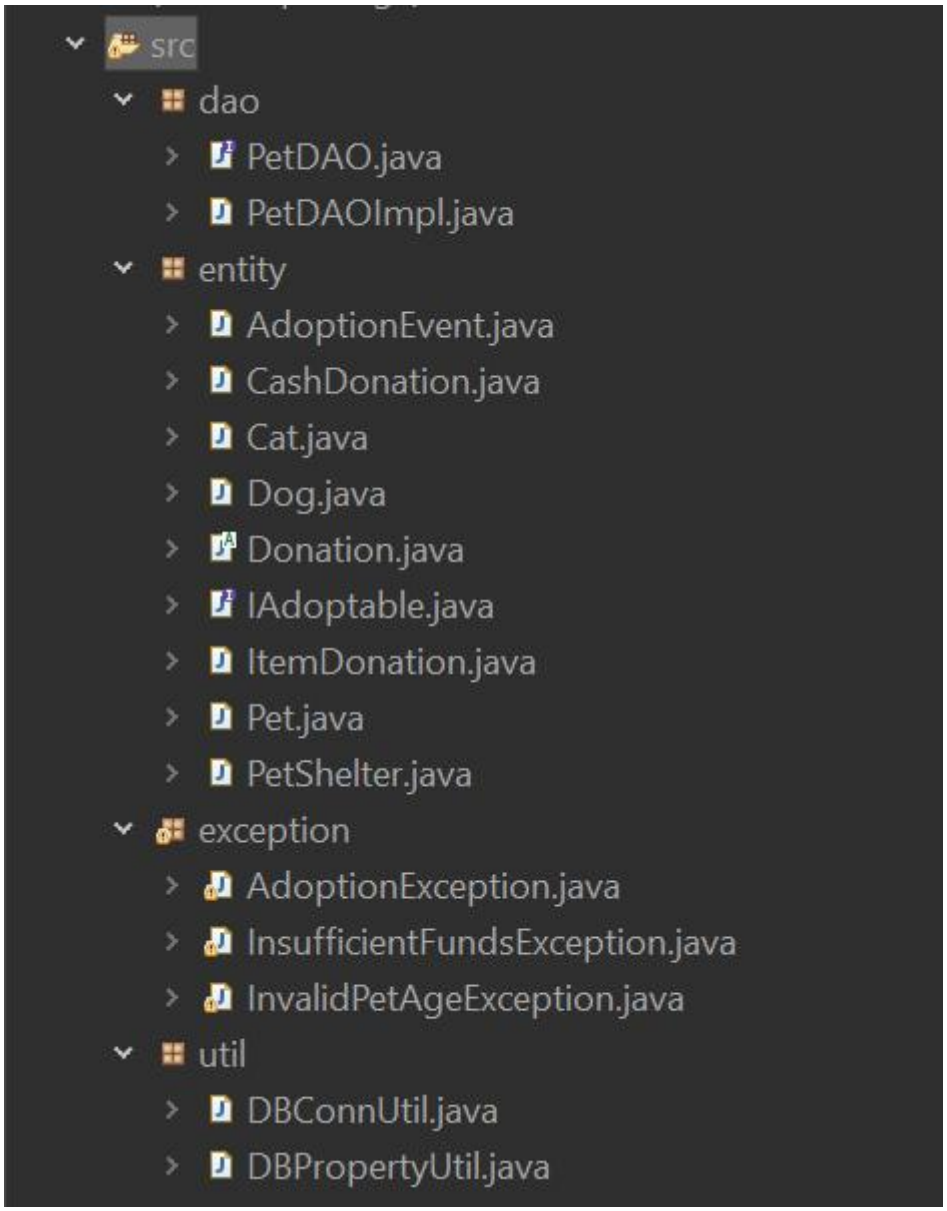
Main

▪ Create a class MainModule and demonstrate the functionalities in a menu driven application.

PROJECT FOLDERS IN ECLIPSE



PROJECT DIRECTORY STRUCTURE IN ECLIPSE



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.

Pet.java × CashDonatio... Donation.java PetShelter.java Cat.java Dog.java

```java
package entity;

public class Pet {
    private int id;
    private String name;
    private int age;
    private String breed;

    // Default constructor
    public Pet() {
    }

    // Constructor with parameters
    public Pet(String name, int age, String breed) {
        this.name = name;
        this.age = age;
        this.breed = breed;
    }

    // Constructor with ID
    public Pet(int id, String name, int age, String breed) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.breed = breed;
    }

    // Getters and setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
```

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.

ItemDonatio... CashDonatio... Donation.java PetShelter.java Cat.java Dog.java

```java
package entity;

public class Dog extends Pet {
    private String dogBreed;

    // Default constructor
    public Dog() {
        super();
    }

    // Constructor with parameters
    public Dog(String name, int age, String breed, String dogBreed) {
        super(name, age, breed);
        this.dogBreed = dogBreed;
    }

    // Constructor with ID
    public Dog(int id, String name, int age, String breed, String dogBreed) {
        super(id, name, age, breed);
        this.dogBreed = dogBreed;
    }

    // Getters and setters
    public String getDogBreed() {
        return dogBreed;
    }

    public void setDogBreed(String dogBreed) {
        this.dogBreed = dogBreed;
    }

    // ToString method
```

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.

IAdoptable.java | ItemDonatio... | CashDonatio... | Donation.java | PetShelter.java | Cat.java

```java
public class Cat extends Pet {
    private String catColor;

    // Default constructor
    public Cat() {
        super();
    }

    // Constructor with parameters
    public Cat(String name, int age, String breed, String catColor) {
        super(name, age, breed);
        this.catColor = catColor;
    }

    // Constructor with ID
    public Cat(int id, String name, int age, String breed, String catColor) {
        super(id, name, age, breed);
        this.catColor = catColor;
    }

    // Getters and setters
    public String getCatColor() {
        return catColor;
    }

    public void setCatColor(String catColor) {
        this.catColor = catColor;
    }

    // ToString method
    @Override
    public String toString() {
```

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.

package entity;

CODING

```
import java.util.ArrayList;
import java.util.List;

public class PetShelter {
    private int id;
    private String name;
    private String location;
    private List<Pet> availablePets;

    // Default constructor
    public PetShelter() {
        this.availablePets = new ArrayList<>();
    }

    // Constructor with parameters
    public PetShelter(String name, String location) {
        this.name = name;
        this.location = location;
        this.availablePets = new ArrayList<>();
    }

    // Constructor with ID
    public PetShelter(int id, String name, String location) {
        this.id = id;
        this.name = name;
        this.location = location;
        this.availablePets = new ArrayList<>();
    }

    // Getters and setters
    public int getId() {
        return id;
```

```
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public List<Pet> getAvailablePets() {
        return availablePets;
    }

    public void setAvailablePets(List<Pet> availablePets) {
        this.availablePets = availablePets;
    }

    // Methods
    public void addPet(Pet pet) {
        availablePets.add(pet);
    }

    public void removePet(Pet pet) {
        availablePets.remove(pet);
    }

    public List<Pet> listAvailablePets() {
        return availablePets;
```

```
    }

    // ToString method
    @Override
    public String toString() {
        return "PetShelter [id=" + id + ", name=" + name + ", location=" + location
+ "]";
    }
}
```

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).

schema.sql | Pet.java | IAdoptable.java | ItemDonatio... | CashDonatio... | Donation.java

```java
package entity;

import java.math.BigDecimal;

public abstract class Donation {
    private int id;
    private String donorName;
    private BigDecimal amount;
    private LocalDate donationDate;

    // Default constructor
    public Donation() {
    }

    // Constructor with parameters
    public Donation(String donorName, BigDecimal amount) {
        this.donorName = donorName;
        this.amount = amount;
        this.donationDate = LocalDate.now();
    }

    // Constructor with ID
    public Donation(int id, String donorName, BigDecimal amount, LocalDate donationDate) {
        this.id = id;
        this.donorName = donorName;
        this.amount = amount;
        this.donationDate = donationDate;
    }

    // Getters and setters
    public int getId() {
        return id;
```

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.

schema.sql | PetDAO.java | Pet.java | IAdoptable.java | ItemDonatio... | CashDonatio...

```java
package entity;

import java.math.BigDecimal;

public class CashDonation extends Donation {
    private String paymentMethod;

    // Default constructor
    public CashDonation() {
        super();
    }

    // Constructor with parameters
    public CashDonation(String donorName, BigDecimal amount, String paymentMethod) {
        super(donorName, amount);
        this.paymentMethod = paymentMethod;
    }

    // Constructor with ID
    public CashDonation(int id, String donorName, BigDecimal amount, LocalDate donationDat
        super(id, donorName, amount, donationDate);
        this.paymentMethod = paymentMethod;
    }

    // Getters and setters
    public String getPaymentMethod() {
        return paymentMethod;
    }

    public void setPaymentMethod(String paymentMethod) {
        this.paymentMethod = paymentMethod;
    }
```

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.

```java
package entity;

import java.math.BigDecimal;

public class ItemDonation extends Donation {
    private String itemType;
    private int quantity;

    // Default constructor
    public ItemDonation() {
        super();
    }

    // Constructor with parameters
    public ItemDonation(String donorName, BigDecimal estimatedValue, String itemType, int
        super(donorName, estimatedValue);
        this.itemType = itemType;
        this.quantity = quantity;
    }

    // Constructor with ID
    public ItemDonation(int id, String donorName, BigDecimal estimatedValue, LocalDate don
                        String itemType, int quantity) {
        super(id, donorName, estimatedValue, donationDate);
        this.itemType = itemType;
        this.quantity = quantity;
    }

    // Getters and setters
    public String getItemType() {
        return itemType;
    }
```

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.

```
package entity;

public interface IAdoptable {
    void adopt();
}
```

SQL CODE

```
-- Create the database if it doesn't exist
CREATE DATABASE IF NOT EXISTS Petpals;
USE Petpals;

-- Create pets table
CREATE TABLE IF NOT EXISTS pets (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    age INT NOT NULL,
    breed VARCHAR(100),
    pet_type VARCHAR(50) NOT NULL,
    dog_breed VARCHAR(100),
    cat_color VARCHAR(100)
);

-- Create donations table
CREATE TABLE IF NOT EXISTS donations (
    id INT AUTO_INCREMENT PRIMARY KEY,
    donor_name VARCHAR(100) NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    donation_date DATE NOT NULL,
    donation_type VARCHAR(50) NOT NULL,
    payment_method VARCHAR(50),
    item_type VARCHAR(100),
    quantity INT
);

-- Create adoption_events table
CREATE TABLE IF NOT EXISTS adoption_events (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    location VARCHAR(100) NOT NULL,
    event_date DATE NOT NULL
);
```

```sql
-- Create participants table
CREATE TABLE IF NOT EXISTS participants (
    id INT AUTO_INCREMENT PRIMARY KEY,
    event_id INT NOT NULL,
    participant_name VARCHAR(100) NOT NULL,
    participant_type VARCHAR(50) NOT NULL,
    FOREIGN KEY (event_id) REFERENCES adoption_events(id)
);
```

OUTPUT WHEN THE JAVA APPLICATION IS RUN

```
Connected to database successfully!
Database tables created successfully!

===== PetPals: Pet Adoption Platform =====
1. Manage Pets
2. Manage Donations
3. Manage Adoption Events
4. Exit
```

MANAGE PETS :VIEW ALL PETS

```
===== PetPals: Pet Adoption Platform =====
1. Manage Pets
2. Manage Donations
3. Manage Adoption Events
4. Exit
Enter your choice: 1

===== Pet Management =====
1. Add a Pet
2. View All Pets
3. View Pet Details
4. Update Pet Information
5. Remove a Pet
6. Back to Main Menu
Enter your choice: 2

===== All Pets =====
Pet [id=1, name=Cat, age=15, breed=Bull]
Pet [id=2, name=Dog, age=22, breed=Lab]
```

ADD A PET:DOG

```
===== Pet Management =====
1. Add a Pet
2. View All Pets
3. View Pet Details
4. Update Pet Information
5. Remove a Pet
6. Back to Main Menu
Enter your choice: 1

===== Add a Pet =====
1. Add a Dog
2. Add a Cat
3. Add a Generic Pet
Enter your choice: 1
Enter pet name: micks
Enter pet age: 34
Enter pet breed: Bull
Enter specific dog breed: BullDog
Dog added successfully with ID: 3
```

VIEW PET DETAILS

```
===== Pet Management =====
1. Add a Pet
2. View All Pets
3. View Pet Details
4. Update Pet Information
5. Remove a Pet
6. Back to Main Menu
Enter your choice: 3
Enter pet ID: 3

===== Pet Details =====
Dog [Pet [id=3, name=micks, age=34, breed=Bull], dogBreed=BullDog]
```

UPDATE PET INFORMATION

```
===== Pet Management =====
1. Add a Pet
2. View All Pets
3. View Pet Details
4. Update Pet Information
5. Remove a Pet
6. Back to Main Menu
Enter your choice: 4
Enter pet ID to update: 3

===== Update Pet =====
Current details: Dog [Pet [id=3, name=micks, age=34, breed=Bull], dogBreed=BullDog]
Enter new name (or press Enter to keep current): Micks
Enter new age (or press Enter to keep current): 35
Enter new breed (or press Enter to keep current): Bull
Enter new dog breed (or press Enter to keep current): BullDog
Pet updated successfully.
```

REMOVE A PET

```
===== Pet Management =====
1. Add a Pet
2. View All Pets
3. View Pet Details
4. Update Pet Information
5. Remove a Pet
6. Back to Main Menu
Enter your choice: 5
Enter pet ID to remove: 1
Are you sure you want to remove this pet?
Pet [id=1, name=Cat, age=15, breed=Bull]
Enter 'yes' to confirm: yes
Pet removed successfully.

===== Pet Management =====
1. Add a Pet
2. View All Pets
3. View Pet Details
4. Update Pet Information
5. Remove a Pet
6. Back to Main Menu
Enter your choice: 6
```

MANAGE DONATIONS: OUTPUT

```
===== PetPals: Pet Adoption Platform =====
1. Manage Pets
2. Manage Donations
3. Manage Adoption Events
4. Exit
Enter your choice: 2
```

MAKE A CASH DONATION

```
===== Donation Management =====
1. Make a Cash Donation
2. Make an Item Donation
3. View All Donations
4. View Donation Details
5. Back to Main Menu
Enter your choice: 1

===== Make a Cash Donation =====
Enter donor name: srinidhi
Enter donation amount: $500
Enter payment method (e.g., Credit Card, PayPal): PayPal
Cash donation recorded successfully with ID: 1
Cash donation of 500 from srinidhi recorded on 2025-04-19 via PayPal

===== Donation Management =====
```

MAKE AN ITEM DONATION

```
===== Donation Management =====
1. Make a Cash Donation
2. Make an Item Donation
3. View All Donations
4. View Donation Details
5. Back to Main Menu
Enter your choice: 2

===== Make an Item Donation =====
Enter donor name: sri
Enter item type (e.g., Food, Toys, Bedding): Food
Enter quantity: 4
Enter estimated value: $100
Item donation recorded successfully with ID: 2
Item donation of 4 Food(s) with estimated value 100 from sri recorded on 2025-04-19
```

VIEW ALL DONATIONS

```
===== Donation Management =====
1. Make a Cash Donation
2. Make an Item Donation
3. View All Donations
4. View Donation Details
5. Back to Main Menu
Enter your choice: 3

===== All Donations =====
CashDonation [Donation [id=1, donorName=srinidhi, amount=500.00, donationDate=2025-04-19], paymentMethod=PayPal]
ItemDonation [Donation [id=2, donorName=sri, amount=100.00, donationDate=2025-04-19], itemType=Food, quantity=4]
```

VIEW DONATION DETAILS

```
===== Donation Management =====
1. Make a Cash Donation
2. Make an Item Donation
3. View All Donations
4. View Donation Details
5. Back to Main Menu
Enter your choice: 4
Enter donation ID: 1

===== Donation Details =====
CashDonation [Donation [id=1, donorName=srinidhi, amount=500.00, donationDate=2025-04-19], paymentMethod=PayPal]
Cash donation of 500.00 from srinidhi recorded on 2025-04-19 via PayPal
```

ADOPTION EVENT OUTPUT

```
===== Adoption Event Management =====
1. Create an Adoption Event
2. View All Adoption Events
3. View Upcoming Adoption Events
4. Register for an Adoption Event
5. Back to Main Menu
Enter your choice: 1

===== Create an Adoption Event =====
Enter event name: Easyadopt
Enter event location: chennai
Enter event date (yyyy-MM-dd): 2025-05-02
Adoption event created successfully with ID: 1

===== Adoption Event Management =====
```

CREATE AN ADOPTION EVENT

```
===== Adoption Event Management =====
1. Create an Adoption Event
2. View All Adoption Events
3. View Upcoming Adoption Events
4. Register for an Adoption Event
5. Back to Main Menu
Enter your choice: 2

===== All Adoption Events =====
AdoptionEvent [id=1, name=Easyadopt, location=chennai, eventDate=2025-05-02, participants=0]
```

VIEW ALL ADOPTION EVENTS

```
===== Adoption Event Management =====
1. Create an Adoption Event
2. View All Adoption Events
3. View Upcoming Adoption Events
4. Register for an Adoption Event
5. Back to Main Menu
Enter your choice: 3

===== Upcoming Adoption Events =====
AdoptionEvent [id=1, name=Easyadopt, location=chennai, eventDate=2025-05-02, participants=0]
```

REGISTER AS AN ADOPTER FOR AN ADOPTION EVENT

```
===== Register for an Adoption Event =====
Available events:
AdoptionEvent [id=1, name=Easyadopt, location=chennai, eventDate=2025-05-02, participants=0]
Enter event ID to register: 1
Enter participant name: daisy
Select participant type:
1. Adopter
2. Shelter
3. Volunteer
Enter choice: 1
Successfully registered for the adoption event.
```

REGISTER AS AN SHELTER FOR AN ADOPTION EVENT

```
===== Adoption Event Management =====
1. Create an Adoption Event
2. View All Adoption Events
3. View Upcoming Adoption Events
4. Register for an Adoption Event
5. Back to Main Menu
Enter your choice: 4

===== Register for an Adoption Event =====
Available events:
AdoptionEvent [id=1, name=Easyadopt, location=chennai, eventDate=2025-05-02, participants=0]
Enter event ID to register: 1
Enter participant name: nidhi
Select participant type:
1. Adopter
2. Shelter
3. Volunteer
Enter choice: 2
Successfully registered for the adoption event.
```

REGISTER AS AN VOLUNTEER FOR AN ADOPTION EVENT

```
===== Register for an Adoption Event =====
Available events:
AdoptionEvent [id=1, name=Easyadopt, location=chennai, eventDate=2025-05-02, participants=0]
Enter event ID to register: 1
Enter participant name: raj
Select participant type:
1. Adopter
2. Shelter
3. Volunteer
Enter choice: 3
Successfully registered for the adoption event.
```

EXIT OUT OF APPLICATION

```
===== Adoption Event Management =====
1. Create an Adoption Event
2. View All Adoption Events
3. View Upcoming Adoption Events
4. Register for an Adoption Event
5. Back to Main Menu
Enter your choice: 5

===== PetPals: Pet Adoption Platform =====
1. Manage Pets
2. Manage Donations
3. Manage Adoption Events
4. Exit
Enter your choice: 4
Thank you for using PetPals! Goodbye!
```

6. INVALID PET AGE EXCEPTION

Pet.java | Donation.java | PetShelter.java | Cat.java | Dog.java | InvalidPetAg...

```java
package exception;

public class InvalidPetAgeException extends Exception {

    public InvalidPetAgeException() {
        super("Invalid pet age. Age must be a positive integer.");
    }

    public InvalidPetAgeException(String message) {
        super(message);
    }
}

```

DATAS STORED IN SQL DATABASE

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	donor_name	amount	donation_date	donation_type	payment_method	item_type	quantity
1	srinidhi	500.00	2025-04-19	Cash	PayPal	NULL	NULL
2	sri	100.00	2025-04-19	Item	NULL	Food	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	location	event_date
1	Easyadopt	chennai	2025-05-02
NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	event_id	participant_name	participant_type
1	1	nidhi	Shelter
2	1	1	Adopter
3	1	raj	Volunteer
4	1	daisy	Adopter
NULL	NULL	NULL	NULL