# Coding Challenges - PetPals, The Pet Adoption Platform

## Instructions

• Coding Challenge submissions should be done through the partcipants’ Github repository, and

the link should be shared with trainers and Hexavarsity.

Problem Statement:

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

SQL Schema:

Table: Pets

Attributes:

• PetID (Primary Key, int): Unique identifier for each pet.

• Name (string): The name of the pet.

• Age (int): The age of the pet.

• Breed (string): The breed of the pet.

• Type (string): Type of pet (e.g., "Dog," "Cat").

• AvailableForAdoption (bit): Indicates whether the pet is available for adoption (0 for not

available, 1 for available).

Table: Shelters

Attributes:

• ShelterID (Primary Key, int): Unique identifier for each shelter.

• Name (string): The name of the shelter.

• Location (string): The location or address of the shelter.

Table: Donations

Attributes:

• DonationID (Primary Key, int): Unique identifier for each donation.

• DonorName (string): The name of the donor.

• DonationType (string): Type of donation (e.g., "Cash," "Item").

• DonationAmount (decimal): The amount donated (for cash donations).

• DonationItem (string): The type of item donated (for item donations).

• DonationDate (datetime): Date and time of the donation.

Table: AdoptionEvents

Attributes:

• EventID (Primary Key, int): Unique identifier for each adoption event.

• EventName (string): The name or title of the event.

• EventDate (datetime): Date and time of the event.

• Location (string): The location or venue of the event.

Table: Participants

Attributes:

• ParticipantID (Primary Key, int): Unique identifier for each participant.

• ParticipantName (string): The name of the participant (shelter or adopter).

• ParticipantType (string): Type of participant (e.g., "Shelter," "Adopter").

• EventID (Foreign Key, int): References the EventID of the associated adoption event (if

applicable).

## Tasks:

1. Provide a SQL script that initializes the database for the Pet Adoption Platform”PetPals”.

CREATE DATABASE IF NOT EXISTS PetPals;

USE PetPals;



2. Create tables for pets, shelters, donations, adoption events, and participants.

CREATE TABLE Shelters (

ShelterID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

Location VARCHAR(255)

);

CREATE TABLE Pets (

PetID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100),

Age INT,

Breed VARCHAR(100),

Type VARCHAR(50),

AvailableForAdoption BIT DEFAULT 1,

OwnerID INT,

ShelterID INT,

FOREIGN KEY (ShelterID) REFERENCES Shelters(ShelterID)

);

CREATE TABLE Donations (

DonationID INT PRIMARY KEY AUTO\_INCREMENT,

DonorName VARCHAR(100),

DonationType VARCHAR(50),

DonationAmount DECIMAL(10, 2),

DonationItem VARCHAR(100),

DonationDate DATETIME,

ShelterID INT,

FOREIGN KEY (ShelterID) REFERENCES Shelters(ShelterID)

);

CREATE TABLE AdoptionEvents (

EventID INT PRIMARY KEY AUTO\_INCREMENT,

EventName VARCHAR(100),

EventDate DATETIME,

Location VARCHAR(255)

);

CREATE TABLE Participants (

ParticipantID INT PRIMARY KEY AUTO\_INCREMENT,

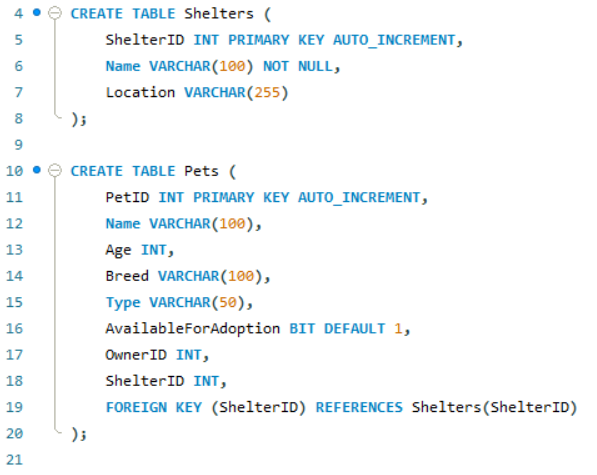
ParticipantName VARCHAR(100),

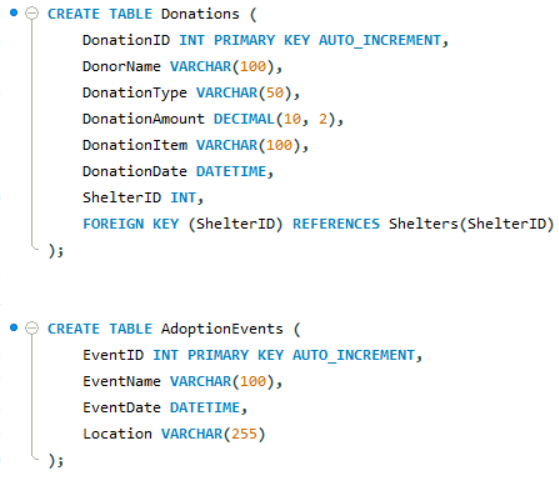
ParticipantType VARCHAR(50),

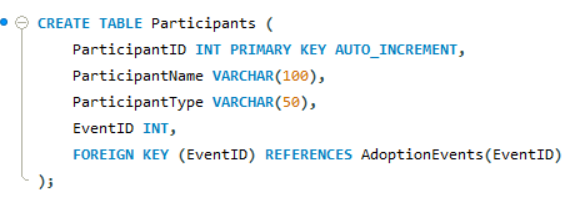
EventID INT,

FOREIGN KEY (EventID) REFERENCES AdoptionEvents(EventID)

);







3. Define appropriate primary keys, foreign keys, and constraints.

ShelterID INT PRIMARY KEY AUTO\_INCREMENT,

PetID INT PRIMARY KEY AUTO\_INCREMENT,

FOREIGN KEY (ShelterID) REFERENCES Shelters(ShelterID)

DonationID INT PRIMARY KEY AUTO\_INCREMENT,

FOREIGN KEY (ShelterID) REFERENCES Shelters(ShelterID)

EventID INT PRIMARY KEY AUTO\_INCREMENT,

ParticipantID INT PRIMARY KEY AUTO\_INCREMENT,

FOREIGN KEY (EventID) REFERENCES AdoptionEvents(EventID)

4. Ensure the script handles potential errors, such as if the database or tables already exist.

Using IF EXIST AND DELIMITERS

Dropping Table if the table already exists.

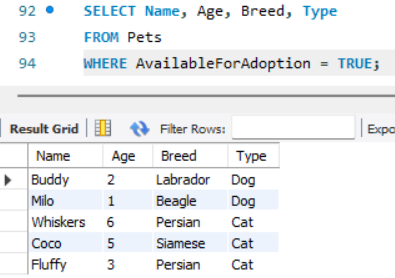
DROP TABLE IF EXISTS Participants, AdoptionEvents, Donations, Pets, Shelters, Users;

5. Write an SQL query that retrieves a list of available pets (those marked as available for adoption) from the "Pets" table. Include the pet's name, age, breed, and type in the result set. Ensure that the query filters out pets that are not available for adoption.

SELECT Name, Age, Breed, Type

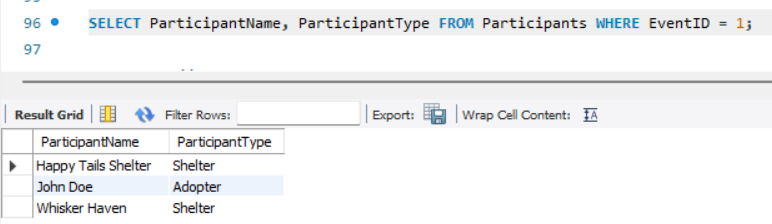
FROM Pets

WHERE AvailableForAdoption = TRUE;



6. Write an SQL query that retrieves the names of participants (shelters and adopters) registered for a specific adoption event. Use a parameter to specify the event ID. Ensure that the query joins the necessary tables to retrieve the participant names and types.

SELECT ParticipantName, ParticipantType FROM Participants WHERE EventID = 1;



7. Create a stored procedure in SQL that allows a shelter to update its information (name and location) in the "Shelters" table. Use parameters to pass the shelter ID and the new information. Ensure that the procedure performs the update and handles potential errors, such as an invalid shelter ID.

DELIMITER //

CREATE PROCEDURE UpdateShelterInfo(

IN p\_ShelterID INT,

IN p\_Name VARCHAR(100),

IN p\_Location VARCHAR(255)

)

BEGIN

IF EXISTS (SELECT 1 FROM Shelters WHERE ShelterID = p\_ShelterID) THEN

UPDATE Shelters

SET Name = p\_Name, Location = p\_Location

WHERE ShelterID = p\_ShelterID;

ELSE

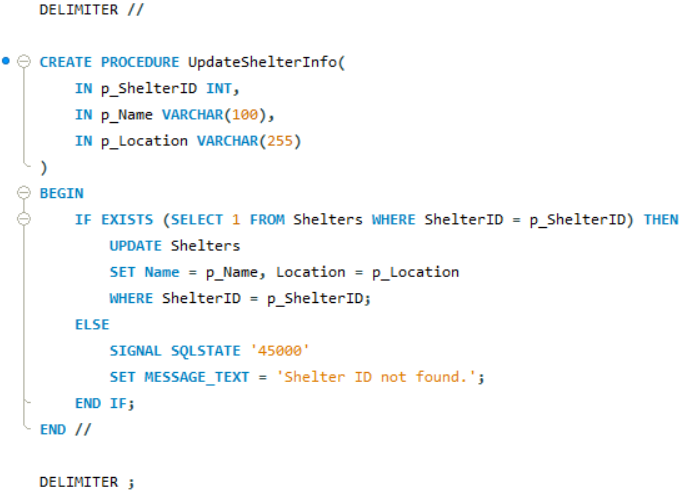
SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Shelter ID not found.';

END IF;

END //

DELIMITER ;



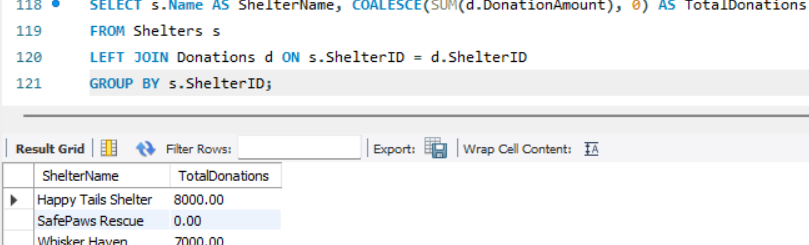
8. Write an SQL query that calculates and retrieves the total donation amount for each shelter (by shelter name) from the "Donations" table. The result should include the shelter name and the total donation amount. Ensure that the query handles cases where a shelter has received no donations.

SELECT s.Name AS ShelterName, COALESCE(SUM(d.DonationAmount), 0) AS TotalDonations

FROM Shelters s

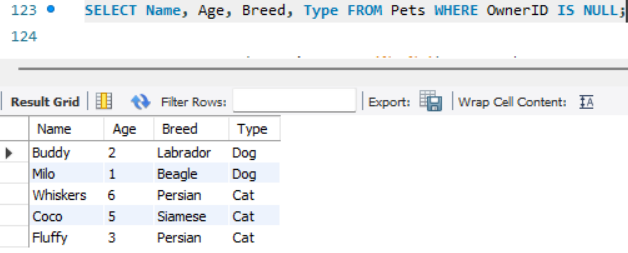
LEFT JOIN Donations d ON s.ShelterID = d.ShelterID

GROUP BY s.ShelterID;



9. Write an SQL query that retrieves the names of pets from the "Pets" table that do not have an owner (i.e., where "OwnerID" is null). Include the pet's name, age, breed, and type in the result set.

SELECT Name, Age, Breed, Type FROM Pets WHERE OwnerID IS NULL;



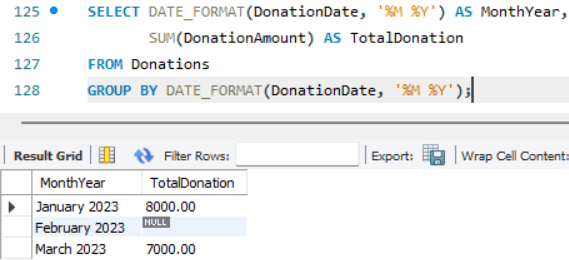
10. Write an SQL query that retrieves the total donation amount for each month and year (e.g. January 2023) from the "Donations" table. The result should include the month-year and the corresponding total donation amount. Ensure that the query handles cases where no donations were made in a specific month-year.

SELECT DATE\_FORMAT(DonationDate, '%M %Y') AS MonthYear,

SUM(DonationAmount) AS TotalDonation

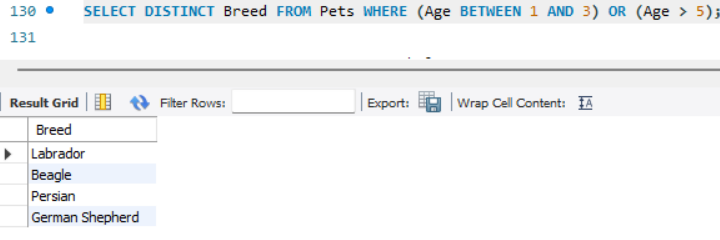
FROM Donations

GROUP BY DATE\_FORMAT(DonationDate, '%M %Y');



11. Retrieve a list of distinct breeds for all pets that are either aged between 1 and 3 years or older than 5 years.

SELECT DISTINCT Breed FROM Pets WHERE (Age BETWEEN 1 AND 3) OR (Age > 5);



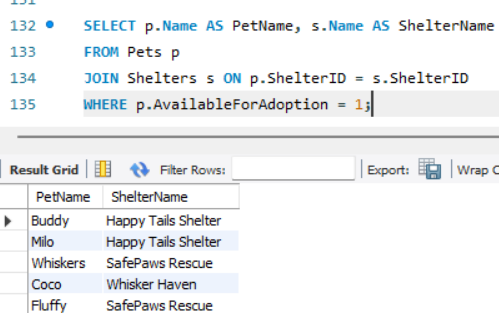
12. Retrieve a list of pets and their respective shelters where the pets are currently available for adoption.

SELECT p.Name AS PetName, s.Name AS ShelterName

FROM Pets p

JOIN Shelters s ON p.ShelterID = s.ShelterID

WHERE p.AvailableForAdoption = 1;



13. Find the total number of participants in events organized by shelters located in specific city. Example: City=Chennai

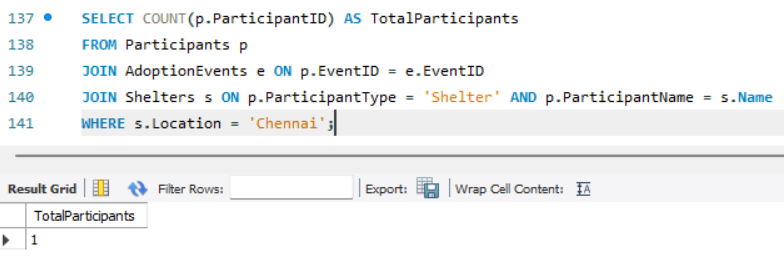
SELECT COUNT(p.ParticipantID) AS TotalParticipants

FROM Participants p

JOIN AdoptionEvents e ON p.EventID = e.EventID

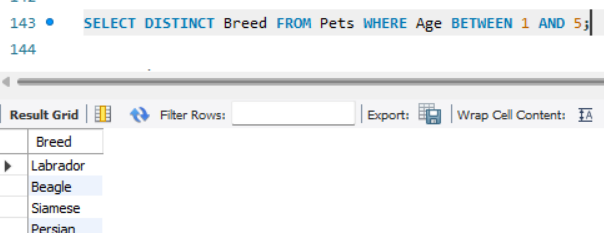
JOIN Shelters s ON p.ParticipantType = 'Shelter' AND p.ParticipantName = s.Name

WHERE s.Location = 'Chennai';



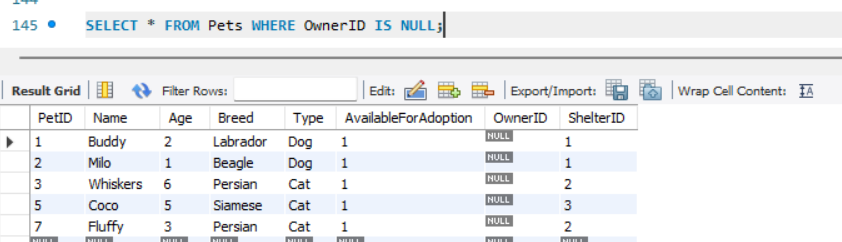
14. Retrieve a list of unique breeds for pets with ages between 1 and 5 years.

SELECT DISTINCT Breed FROM Pets WHERE Age BETWEEN 1 AND 5;



15. Find the pets that have not been adopted by selecting their information from the 'Pet' table.

SELECT \* FROM Pets WHERE OwnerID IS NULL;

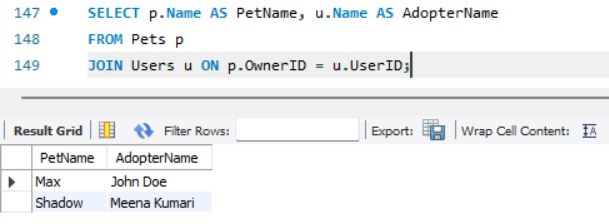


16. Retrieve the names of all adopted pets along with the adopter's name from the 'Adoption' and 'User' tables.

SELECT p.Name AS PetName, u.Name AS AdopterName

FROM Pets p

JOIN Users u ON p.OwnerID = u.UserID;



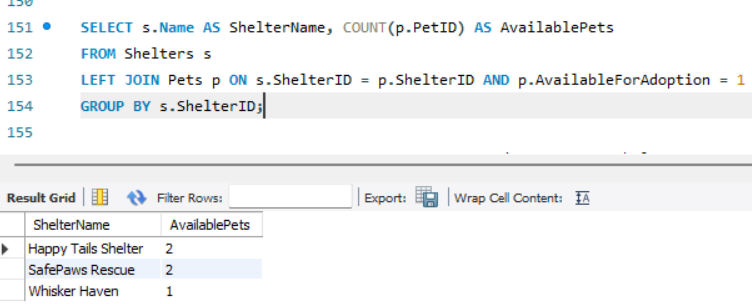
17. Retrieve a list of all shelters along with the count of pets currently available for adoption in each shelter.

SELECT s.Name AS ShelterName, COUNT(p.PetID) AS AvailablePets

FROM Shelters s

LEFT JOIN Pets p ON s.ShelterID = p.ShelterID AND p.AvailableForAdoption = 1

GROUP BY s.ShelterID;



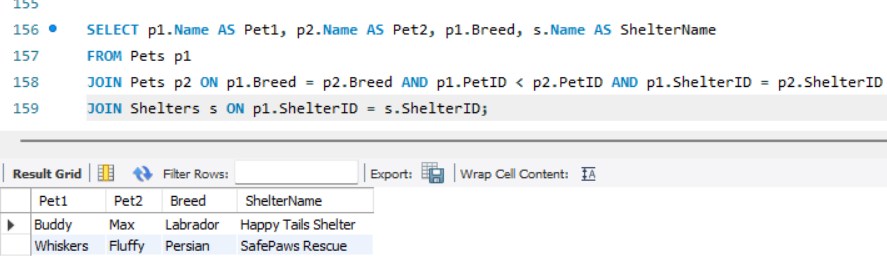
18. Find pairs of pets from the same shelter that have the same breed.

SELECT p1.Name AS Pet1, p2.Name AS Pet2, p1.Breed, s.Name AS ShelterName

FROM Pets p1

JOIN Pets p2 ON p1.Breed = p2.Breed AND p1.PetID < p2.PetID AND p1.ShelterID = p2.ShelterID

JOIN Shelters s ON p1.ShelterID = s.ShelterID;

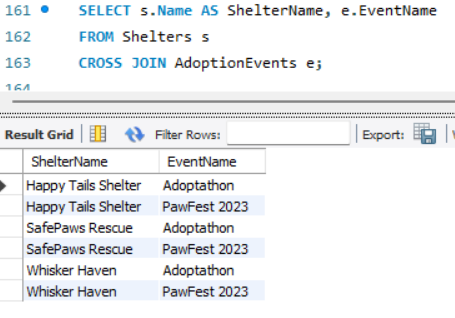


19. List all possible combinations of shelters and adoption events.

SELECT s.Name AS ShelterName, e.EventName

FROM Shelters s

CROSS JOIN AdoptionEvents e;



20. Determine the shelter that has the highest number of adopted pets.

SELECT s.Name, COUNT(p.PetID) AS AdoptedPets

FROM Pets p

JOIN Shelters s ON p.ShelterID = s.ShelterID

WHERE p.OwnerID IS NOT NULL

GROUP BY s.ShelterID

ORDER BY AdoptedPets DESC

LIMIT 1;

