**CPSC 304 Milestone 4**

**SQL Script**
This script is found as animal-hope.sql in the backend folder.

**GitHub Link**
We have pushed our project to the CPSC304 GitHub, however, we were working on a personal repository before this, found in this link, https://github.com/TracyGan/animal-hope.git .

**Description**
Our project provides functionalities to manage an animal shelter, covering areas such as animals, staff (like doctors, caretakers, and administrative staff), volunteers, adoptions, animal fostering, and other essential operations. The user interface allows users to manage shelter animals and their food supplies, control client relationships and donations, and organize fundamental activities like animal treatment, feeding, and walking schedules.

Specifically, the system provides the following capabilities:

- Authenticate staff through username and password.
- Delete animals from the system (DELETE).
- Record walks for animals (INSERT).
- Update the information on animals being fed (UPDATE).
- Search for name, breed, type, ID and age of animals (SELECT).
- Perform various calculations related to donations, described in sections 2.1.7 to 2.1.10.

Additionally, all user input from text fields is validated to prevent SQL injections and ensure security.

**Difference from Initial Schema**
Since data of type CHAR are fixed in length, attributes of datatype CHAR were updated to VARCHAR to remove whitespace characters. Additionally, data of type DATETIME were changed to TIMESTAMP, which Oracle supports. Also, we realized that the Name attribute can uniquely identify an animal's food. So, we changed the primary key of the Food relation from Brand and Name to Name.

**SQL Queries (2.1.1-2.1.6)**

| **SQL Query** | **File: Line _** |
|---|---|
| Insert | /backend/appService.js: Line 419-420 |
| Delete | /backend/appService.js: Line 316 |

| | |
|---|---|
| Update | /backend/appService.js: Line 246-249 |
| Selection | /backend/appService.js: Line 465-470 |
| Projection | /backend/appService.js: Line 443-445 |
| Join | /backend/appService.js: Line 205-219 |

**SQL Queries (2.1.7-2.1.10)**

**2.1.7** Aggregation with Group By
- /backend/appService.js: Line 432
- This query returns the number of walks that each volunteer has done/will do, as well as the related volunteer's name

```
SELECT COUNT(w.ID), v.Name
FROM Walks w
JOIN Volunteer v ON v.ID = w.Volunteer_ID
GROUP BY v.Name
```

**2.1.8** Aggregation with Having
- /backend/appService.js: Line 341-345
- This query returns the breeds who have an average age that is greater than the average age of all the animals.

```
SELECT A.Breed, AVG(A.Age)
FROM Animal A
GROUP BY A.Breed
HAVING AVG(A.age) >= ALL (SELECT AVG(A2.Age)
                          FROM Animal A2)
```

**2.1.9** Nested Aggregation with Group By
- /backend/appService.js: Line 232-236
- This query finds the clients that donated in total more than the average donations from all clients.

```
SELECT Client.ID, Client.Name, Client.EmailAddress,
SUM(D.Amount), Client.FosterPersonCertificationID,
Client.AdopterPersonCertificationID
     FROM Donation D
```

```
        JOIN Client ON Client.ID = D.Client_ID
        GROUP BY Client.ID, Client.Name, Client.EmailAddress,
        Client.FosterPersonCertificationID,
        Client.AdopterPersonCertificationID
        HAVING SUM(D.Amount) > (SELECT AVG(d1.Amount) FROM Donation
d1)
```

**2.1.10** Division

- /backend/appService.js: Line 326-332
- This query returns every volunteer that has walked every animal in our database.

```
SELECT Name
      FROM Volunteer V
      WHERE NOT EXISTS
        ((SELECT A.ID FROM Animal A) MINUS
             (SELECT W.Animal_ID
             FROM Walks W
             WHERE V.ID = W.Volunteer_ID))
```