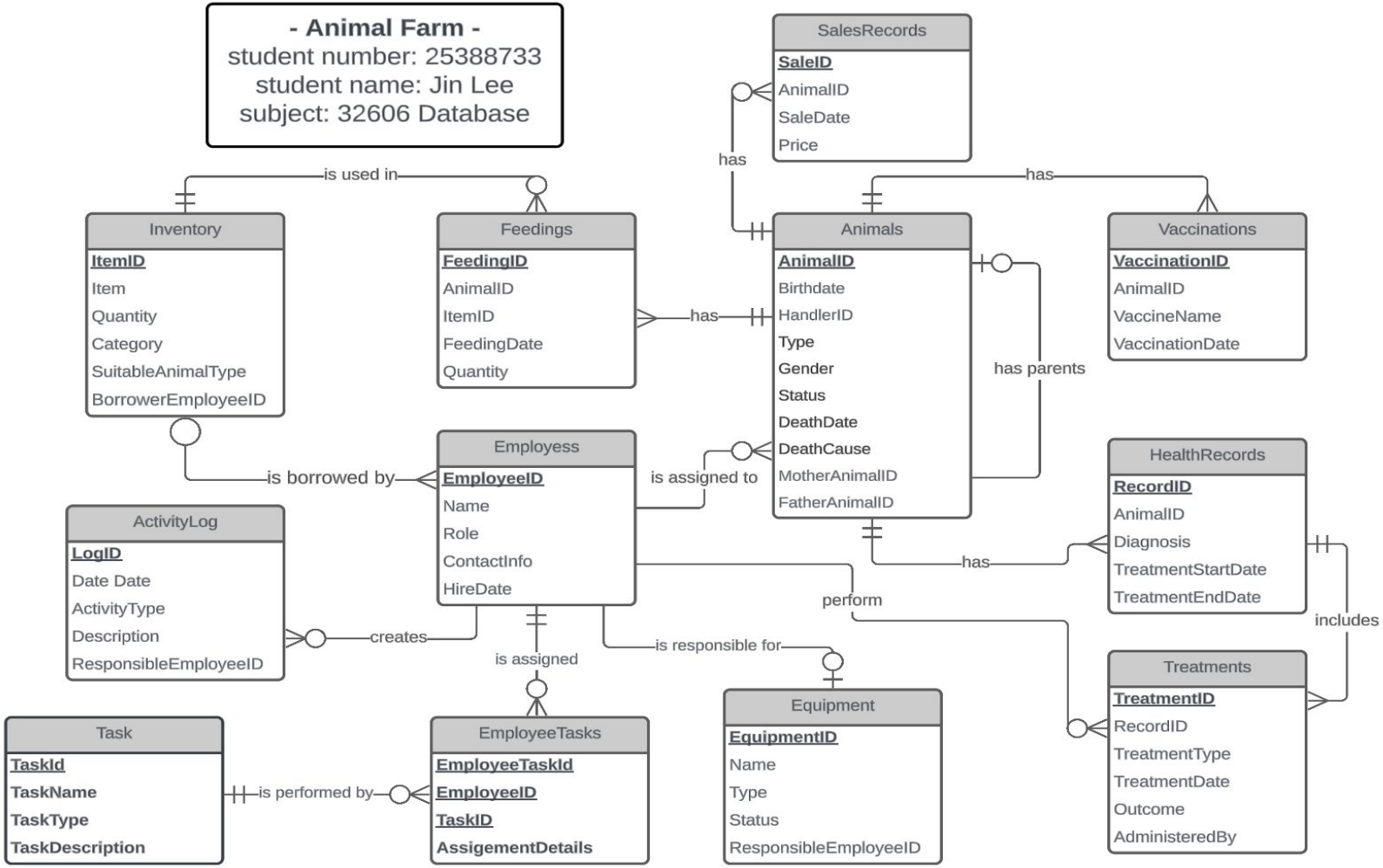

- Animal Farm -
student number: 25388733
student name: Jin Lee
subject: 32606 Database
SalesRecords
SaleID
AnimalID
SaleDate
Price
has
is used in
has
Inventory
ItemID
Item
Quantity
Category
SuitableAnimalType
BorrowerEmployeeID
Feedings
FeedingID
AnimalID
ItemID
FeedingDate
Quantity
has
Animals
AnimalID
Birthdate
HandlerID
Type
Gender
Status
DeathDate
DeathCause
MotherAnimalID
FatherAnimalID
Vaccinations
VaccinationID
AnimalID
VaccineName
VaccinationDate
has parents
is borrowed by
Employess
EmployeeID
Name
Role
ContactInfo
HireDate
is assigned to
HealthRecords
RecordID
AnimalID
Diagnosis
TreatmentStartDate
TreatmentEndDate
has
ActivityLog
LogID
Date Date
ActivityType
Description
ResponsibleEmployeeID
creates
is assigned
is responsible for
perform
includes
Treatments
TreatmentID
RecordID
TreatmentType
TreatmentDate
Outcome
AdministeredBy
Task
TaskId
TaskName
TaskType
TaskDescription
is performed by
EmployeeTasks
EmployeeTaskId
EmployeeID
TaskID
AssigementDetails
Equipment
EquipmentID
Name
Type
Status
ResponsibleEmployeeID

Animal Farm DB is Inspired by Kumdon Farm in korea

During my undergraduate studies Animal Science, I was inspired by my experiences visiting farms as part of extracurricular activities, which led me to design this database.

http://www.xn--2r1b60look51cbuw.com/main

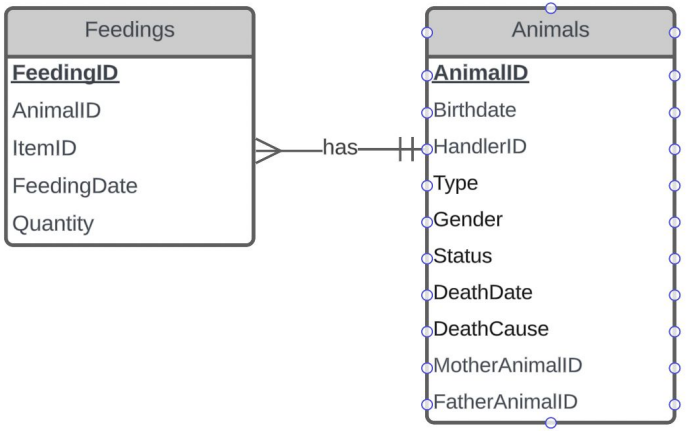
32606 Database - Autumn 2024

25388733 Jin Lee

A Single One-To-Many Relationship:

A Single Animal's feedings can be recorded through 1 to many Feedings.

```
FarmData=# SELECT Animals.AnimalID, Feedings.FeedingID
FarmData-# FROM Animals, Feedings
WHERE Animals.AnimalID = Feedings.AnimalID;
FROM Animals, Feedings
FarmData-# WHERE Animals.AnimalID = Feedings.AnimalID;
WHERE Animals.AnimalID = Feedings.AnimalID;
 animalid | feedingid
----------+-----------
 ANM1001  | F011
 ANM1002  | F012
 ANM1006  | F013
 ANM1007  | F014
 ANM1012  | F015
 ANM1011  | F016
 ANM1010  | F017
 ANM1009  | F018
 ANM1008  | F019
 ANM1014  | F020
(10 rows)
```

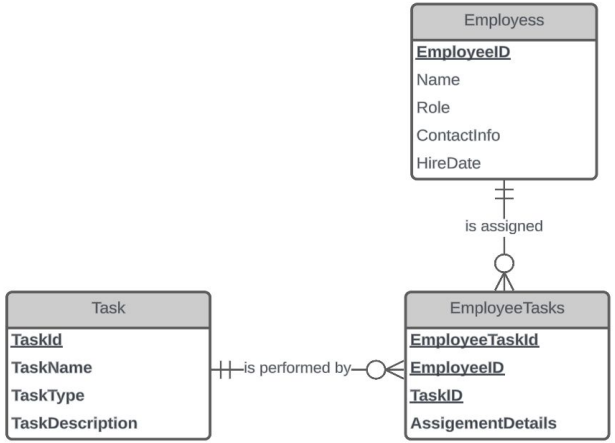


A Single Many-To-Many Relationship

A Single Employee can perform Many Tasks and One Task can be performed by Many Employees.

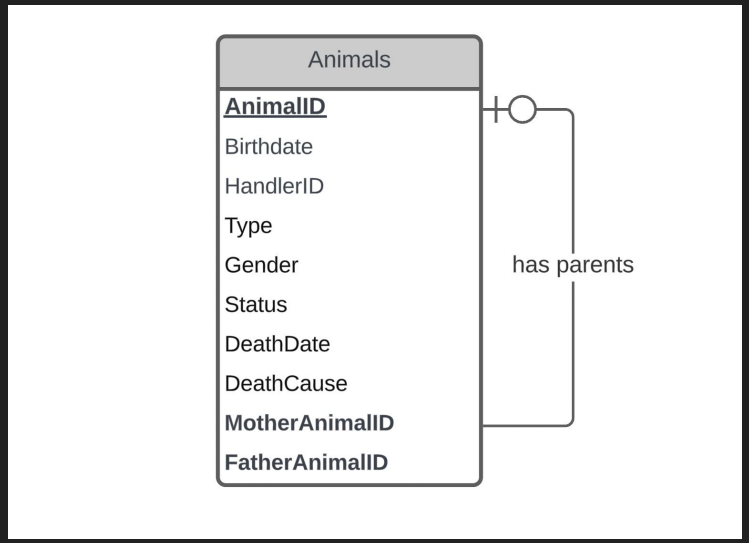
```
FarmData=# SELECT DISTINCT
    E.Name AS EmployeeName,
    E.EmployeeID,
    T.TaskName,
    T.TaskID,
    ET.AssignmentDetails
FROM
    employees E, tasks T, employeetasks ET
WHERE
    E.EmployeeID = ET.EmployeeID
    AND T.TaskID = ET.TaskID ORDER BY employeeid;
 employeename  | employeeid |      taskname       | taskid |                        assignmentdetails

---------------+------------+---------------------+--------+-----------------------------------------------------------------------
--
 Alice Johnson | EMP1001    | Feed Animals        | TSK01  | Daily feeding of all cows.
 Alice Johnson | EMP1001    | Train New Employees | TSK10  | Provide onboarding and training for new farm employees.
 Alice Johnson | EMP1001    | Update Inventory    | TSK09  | Ensure the farm inventory is updated and restocked as necessary.
 Bob Smith     | EMP1002    | Animal Health Check | TSK07  | Conduct regular health checks on all farm animals.
 Bob Smith     | EMP1002    | Harvest Crops       | TSK08  | Harvest crops according to the seasonal schedule.
 Bob Smith     | EMP1002    | Milk Cows           | TSK03  | Carry out het milking process.
 Bob Smith     | EMP1002    | Repair Fences       | TSK05  | Inspect and repair any damaged sections of farm fencing.
 Bob Smith     | EMP1002    | Tractor Maintenance | TSK06  | Perform routine maintenance checks on farm tractors and equipment
.
 Lisa Johnson  | EMP1007    | Repair Fences       | TSK05  | Inspect and repair any demage.
(9 rows)
```



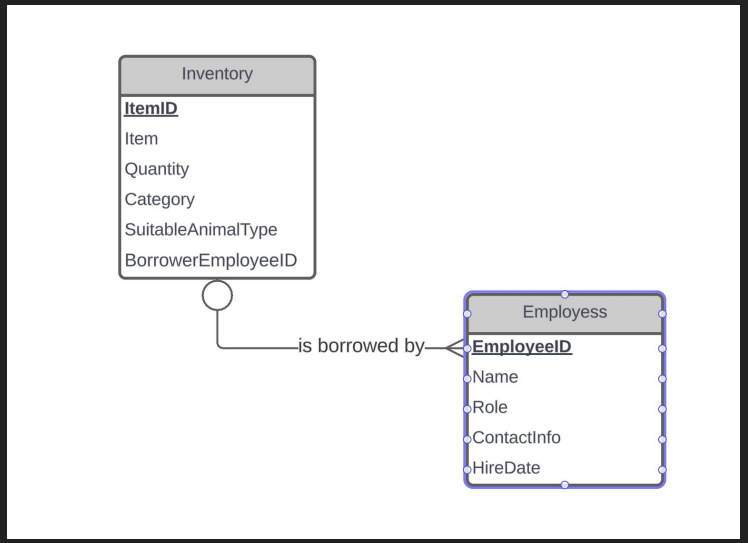
Self-Referencing:

‘Animals’ reference themselves with ‘MotherAnimalID’ and ‘FatherAnimalID’ to track lineage and genetics.



Zero-to-One:

An ‘Inventory’ item may or may not be borrowed by an ‘Employee’, making this relationship optional.



A simple query of a single table.

List all Animal's Id, Type, Gender, Birthdate for all pigs or cow from the Animals table.

```
FarmData=# SELECT AnimalID, Type, Gender, Birthdate FROM Animals WHERE Type ='Pig' OR TYPE = 'Cow' ORDER BY type;
 animalid | type | gender | birthdate
----------+------+--------+------------
 ANM1004  | Cow  | Male   | 2020-08-05
 ANM1005  | Cow  | Female | 2021-09-20
 ANM1002  | Cow  | Female | 2017-06-22
 ANM1016  | Cow  | Male   | 2019-06-01
 ANM1003  | Cow  | Female | 2019-07-15
 ANM1019  | Cow  | Female | 2019-09-30
 ANM1001  | Cow  | Male   | 2018-05-10
 ANM1020  | Pig  | Male   | 2020-10-12
 ANM1006  | Pig  | Female | 2018-03-11
 ANM1007  | Pig  | Male   | 2017-04-18
 ANM1010  | Pig  | Male   | 2021-12-15
 ANM1009  | Pig  | Female | 2020-11-30
 ANM1008  | Pig  | Male   | 2019-10-22
 ANM1017  | Pig  | Female | 2018-07-15
(14 rows)
```

A query which uses the words "natural join".

List employees hired after 2021 by their names, along with their employee IDs, task names, and task IDs, and order the results by the employee's name.

```
FarmData=# SELECT DISTINCT
    E.EmployeeID,
    E.hiredate,
    E.Name AS EmployeeName,
    T.TaskName,
    T.TaskID
FROM employees E NATURAL JOIN employeeTasks ET NATURAL JOIN tasks T
WHERE E.hiredate >= '2021-01-01'
ORDER BY E.Name;
 employeeid |  hiredate  | employeename |      taskname       | taskid
------------+------------+--------------+---------------------+--------
 EMP1002    | 2021-06-20 | Bob Smith    | Animal Health Check | TSK07
 EMP1002    | 2021-06-20 | Bob Smith    | Harvest Crops       | TSK08
 EMP1002    | 2021-06-20 | Bob Smith    | Milk Cows           | TSK03
 EMP1002    | 2021-06-20 | Bob Smith    | Repair Fences       | TSK05
 EMP1002    | 2021-06-20 | Bob Smith    | Tractor Maintenance | TSK06
 EMP1007    | 2021-05-21 | Lisa Johnson | Repair Fences       | TSK05
(6 rows)
```

The cross product equivalent to the "natural join" query above.

```
FarmData=# SELECT DISTINCT
    E.EmployeeID,
    E.Name AS EmployeeName,
    T.TaskName,
    T.TaskID
FROM employees E, employeeTasks ET, tasks T
WHERE E.EmployeeID = ET.EmployeeID AND ET.TaskID = T.TaskID AND E.HireDate >= '2021-01-01'
ORDER BY E.Name;
 employeeid | employeename |      taskname       | taskid
------------+--------------+---------------------+--------
 EMP1002    | Bob Smith    | Animal Health Check | TSK07
 EMP1002    | Bob Smith    | Harvest Crops       | TSK08
 EMP1002    | Bob Smith    | Milk Cows           | TSK03
 EMP1002    | Bob Smith    | Repair Fences       | TSK05
 EMP1002    | Bob Smith    | Tractor Maintenance | TSK06
 EMP1007    | Lisa Johnson | Repair Fences       | TSK05
(6 rows)
```

A query involving a "Group by", perhaps also with a "HAVING".

List animal types fed at least twice, with their feeding counts, to identify frequently fed animals and their care needs in the database.

```
FarmData=# SELECT
    A.Type,
    COUNT(*) AS NumberOfFeedings
FROM animals A JOIN feedings F ON A.AnimalID = F.AnimalID
GROUP BY A.Type
HAVING COUNT(*) >= 2
ORDER BY A.Type;
  type    | numberoffeedings
----------+------------------
 Chicken |                3
 Cow     |                2
 Pig     |                5
(3 rows)
```

A query which uses a sub query.

A query utilizing a subquery to list all animals that have received feedings. This query identifies animals with at least one recorded feeding in the Animal Farm database.

```
FarmData=# SELECT AnimalID, Type, Gender, Birthdate
FROM animals WHERE AnimalID = ANY (SELECT DISTINCT AnimalID FROM Feedings);
 animalid |  type   | gender | birthdate
----------+---------+--------+------------
 ANM1008  | Pig     | Male   | 2019-10-22
 ANM1007  | Pig     | Male   | 2017-04-18
 ANM1002  | Cow     | Female | 2017-06-22
 ANM1010  | Pig     | Male   | 2021-12-15
 ANM1001  | Cow     | Male   | 2018-05-10
 ANM1012  | Chicken | Male   | 2017-02-17
 ANM1006  | Pig     | Female | 2018-03-11
 ANM1011  | Chicken | Female | 2018-01-25
 ANM1009  | Pig     | Female | 2020-11-30
 ANM1014  | Chicken | Female | 2020-04-21
(10 rows)
```

Across product which cannot be implemented using the words "natural join" (e.g. self join)

List the Family Tree for Cows

```
FarmData=# SELECT
    A.AnimalID AS "Offspring ID",
    A.Type AS "Animal Type",
    A.Gender,
    M.AnimalID AS "Mother ID",
    F.AnimalID AS "Father ID"
FROM animals A
LEFT JOIN animals M ON A.MotherAnimalID = M.AnimalID
LEFT JOIN animals F ON A.FatherAnimalID = F.AnimalID
WHERE A.Type = 'Cow'
    AND A.MotherAnimalID IS NOT NULL
    AND A.FatherAnimalID IS NOT NULL;
 Offspring ID | Animal Type | gender | Mother ID | Father ID
--------------+-------------+--------+-----------+-----------
 ANM1003      | Cow         | Female | ANM1002   | ANM1001
 ANM1004      | Cow         | Male   | ANM1002   | ANM1001
 ANM1005      | Cow         | Female | ANM1002   | ANM1001
 ANM1016      | Cow         | Male   | ANM1002   | ANM1001
 ANM1019      | Cow         | Female | ANM1002   | ANM1001
(5 rows)
```

Check Statements:

```
FarmData=# CREATE TABLE Animals (
    AnimalID VARCHAR(255) PRIMARY KEY,
    Birthdate DATE,
    HandlerID VARCHAR(255),
    Type VARCHAR(255),
    Gender CHAR(1),
    Status VARCHAR(255),
    DeathDate DATE,
    DeathCause VARCHAR(255),
    MotherAnimalID VARCHAR(255),
    FatherAnimalID VARCHAR(255),
    FOREIGN KEY (HandlerID) REFERENCES Employees(EmployeeID),
    FOREIGN KEY (MotherAnimalID) REFERENCES Animals(AnimalID),
    FOREIGN KEY (FatherAnimalID) REFERENCES Animals(AnimalID),
    CONSTRAINT chk_Gender CHECK (Gender IN ('Male', 'Female')),
    CONSTRAINT chk_Status CHECK (Status IN ('Healthy', 'Sick', 'Deceased')),
    CONSTRAINT chk_Type CHECK (Type IN ('Cow', 'Pig', 'Sheep', 'Chicken')),
    CONSTRAINT chk_Birthdate CHECK (Birthdate <= CURRENT_DATE),
    CONSTRAINT chk_DeathDate CHECK (DeathDate IS NULL OR DeathDate <= CURRENT_DATE),
    CONSTRAINT chk_DeathCause CHECK (DeathDate IS NOT NULL AND DeathCause IS NOT NULL OR DeathDate IS NULL AND DeathCause IS N
ULL)
);
```

Check Statements:

```
FarmData=# CREATE TABLE Feedings (
    FeedingID VARCHAR(255) PRIMARY KEY,
    AnimalID VARCHAR(255) NOT NULL,
    ItemID VARCHAR(255) NOT NULL,
    FeedingDate DATE NOT NULL,
    Quantity INT NOT NULL,
    FOREIGN KEY (AnimalID) REFERENCES Animals(AnimalID),
    FOREIGN KEY (ItemID) REFERENCES Inventory(ItemID),
    CONSTRAINT chk_FeedingDate CHECK (FeedingDate <= CURRENT_DATE),
    CONSTRAINT chk_Quantity CHECK (Quantity > 0)
);
```

```
FarmData=# CREATE TABLE Employees (
    EmployeeID VARCHAR(255) PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Role VARCHAR(255),
    ContactInfo VARCHAR(255),
    HireDate DATE,
    CONSTRAINT chk_Role CHECK (Role IN ('Manager', 'Veterinarian', 'Worker', 'Admin')),
    CONSTRAINT chk_HireDate CHECK (HireDate <= CURRENT_DATE AND HireDate IS NOT NULL),
    CONSTRAINT chk_ContactInfo CHECK (ContactInfo IS NOT NULL AND (ContactInfo LIKE '%@%' OR ContactInfo LIKE '(___) ___-____'
))
);
```

Check Statements:

```
FarmData=# CREATE TABLE Equipment (
    EquipmentID VARCHAR(255) PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Type VARCHAR(255) NOT NULL,
    Status VARCHAR(255),
    ResponsibleEmployeeID VARCHAR(255),
    FOREIGN KEY (ResponsibleEmployeeID) REFERENCES Employees(EmployeeID) ON DELETE SET NULL,
    CONSTRAINT chk_Type CHECK (Type IN ('Tractor', 'Harvester', 'Irrigation', 'Other')),
    CONSTRAINT chk_Status CHECK (Status IN ('Available', 'In Use', 'Maintenance', 'Out of Service'))
);
```

1.ON DELETE RESTRICT

```
FarmData=# CREATE TABLE Tasks (
    TaskID VARCHAR(255) PRIMARY KEY,
    TaskName VARCHAR(255) NOT NULL,
    TaskType VARCHAR(255) CHECK (TaskType IN ('Maintenance', 'Feeding', 'Medical', 'Cleaning')),
    TaskDescription TEXT,
    AssignedEmployeeID VARCHAR(255),
    FOREIGN KEY (AssignedEmployeeID) REFERENCES Employees(EmployeeID) ON DELETE RESTRICT,
    CONSTRAINT chk_TaskName NOT NULL,
    CONSTRAINT chk_TaskDescription CHECK (LENGTH(TaskDescription) > 10)
);
```

2.ON DELETE CASCADE

```
FarmData=# CREATE TABLE Vaccinations (
    VaccinationID VARCHAR(255) PRIMARY KEY,
    AnimalID VARCHAR(255),
    VaccineName VARCHAR(255),
    VaccinationDate DATE,
    FOREIGN KEY (AnimalID) REFERENCES Animals(AnimalID) ON DELETE CASCADE
);
```

CREATE VIEW:

1. View of Animal Feeding Records:

This view displays the latest feeding records for each animal. It includes the Animal ID, the date fed, the name of the item fed, and the quantity given.

```
FarmData=# CREATE VIEW AnimalFeedingRecords AS
SELECT
    A.AnimalID,
    A.Type AS AnimalType,
    F.FeedingDate,
    I.Item AS FedItem,
    F.Quantity
FROM
    Animals A, Feedings F, Inventory I
WHERE
    A.AnimalID = F.AnimalID AND
    F.ItemID = I.ItemID;
```

2. View of Equipment Usage:

This view shows the usage status of each piece of equipment. It contains the Equipment ID, name, type, current status, and the ID of the responsible employee.

```
FarmData=# CREATE VIEW EquipmentUsage AS
SELECT
    E.EquipmentID,
    E.Name AS EquipmentName,
    E.Type AS EquipmentType,
    E.Status AS EquipmentStatus,
    E.ResponsibleEmployeeID
FROM
    Equipment E;
```