Nunzio, Fela, Dean
CSC 315 Group 9
Phase IV

Task 2:

Normalization:

1NF:

- Every table is in 1NF because each attribute contains its own atomic value and there are no repeating values, meaning that no attributes have multiple values or repeating groups

2NF:

- Table 1: Goat is already in 2NF because there is only 1 candidate key, Goat_ID and all attributes are functionally dependent on it.
- In Table 2: Goat_Stats, the composite primary keys are (Points, Goat_ID). Because every attribute is dependent on the primary keys the table is already in 2NF
- In Table 3: Child, there is only 1 attribute besides the primary key so there are no partial dependencies so its in 2NF

3NF:

- In Table 1: Goat there are no transitive dependencies in this table
- In table 2: Goat_Stats, there are no dependencies in this table
- In table 3: There is only one other attribute besides the primary key so there are no transitive dependencies

BCNF:

- In table 1: Goat there are no non-trivial functional dependencies in this table so it satisfied BCNF
- In table 2: Goat there are no non-trivial functional dependencies in this table so it satisfied BCNF
- In table 3: Goat there are no non-trivial functional dependencies in this table so it satisfied BCNF

Task 3:

```sql
CREATE TABLE Goats(
    Goat_ID INT PRIMARY KEY,
    Points_Total INT,
    Goat_Rating_Number INT,
    Birth_Day INT,
    Birth_Month INT,
    Birth_Year INT
);

CREATE TABLE Goat_Status(
    Points INT PRIMARY KEY.
    Goat_ID INT,
    Birth_Weight INT,
    Milk_Rating INT,
    Vigor_Health INT,
    Sore_Mouth INT,
    Weening_Number INT,
    Mothering_Rating INT,
    Avg_Daily_Wgt_Gain INT,
    Chlymdia_Vaccine BOOLEAN,
    FOREIGN KEY(Goat_ID) REFERENCES Goats(Goat_ID),
    FOREIGN KEY(Points) REFERENCES Goats(Points_Total)
);

CREATE TABLE Goat_Family(
    Goat_ID INT PRIMARY KEY,
    Child_ID INT,
    Parent_ID INT,
    FOREIGN KEY(Goat_ID) REFERENCES Goats(Goat_ID)
);

/*View that will have the Goats ranked from highest to lowest*/

CREATE VIEW Goat_Ranking AS
SELECT Goats.Goat_ID, Goats.Goats_Rating_Number, Goats.Points_Total,  Goat_Status.Chlymdia_Vaccine
FROM Goats, Goat_Status
ORDER BY Goat_Rating_Number;
/*
Data Requirements: Goat_ID, Goats_Rating_Number, Points_Total

Transaction Requirements: Atomic, Consistency, Isolation, Durability
*/
```

```sql
/*View that will show Goats and all their health status*/

CREATE VIEW Goat_Health AS
SELECT Goats.Goat_ID, Goat_Status.Birth_Weight, Goat_Status.Milk_Rating, Goat_Status.Vigor_Health,
 Goat_Status.Sore_Mouth, Goat_Status.Weening_Number, Goat_Status.Mothering_Rating,
 Goat_Status.Avg_Daily_Wgt_Gain, Goat_Status.Chlymdia_Vaccine
FROM Goats, Goat_Status;
/*
Data Requirements: Goats.Goat_ID, Goat_Status.Birth_Weight, Goat_Status.Milk_Rating, Goat_Status.Vigor_Health,
 Goat_Status.Sore_Mouth, Goat_Status.Weening_Number, Goat_Status.Mothering_Rating,
 Goat_Status.Avg_Daily_Wgt_Gain, Goat_Status.Chlymdia_Vaccine

Transaction Requirements: Atomic, Consistency, Isolation, Durability
*/

/*View that will show relations with other goats*/
CREATE VIEW Family_Tree AS
SELECT Goats.Goat_ID, Goat_Family.Child_ID, Goat_Familt.Parent_ID
FROM Goats, Goat_Family;
/*
Data Requirements: Goats.Goat_ID, Goat_Family.Child_ID, Goat_Familt.Parent_ID

Transaction Requirements: Atomic, Consistency, Isolation, Durability
*/
```

Task 4:

```sql

  SELECT *
  FROM Goat_Ranking;

  /*Query to Retrive the Top ranked goats
  Data Requirements: Goat_ID, Goats_Rating_Number, Points_Total

  Transaction Requirements: Atomic, Consistency, Isolation, Durability*/


  SELECT *
  FROM Goat_Health;

  /*Query to retrive the goat health status
  Data Requirements: Goats.Goat_ID, Goat_Status.Birth_Weight, Goat_Status.Milk_Rating, Goat_Status.Vigor_Health,
   Goat_Status.Sore_Mouth, Goat_Status.Weening_Number, Goat_Status.Mothering_Rating,
   Goat_Status.Avg_Daily_Wgt_Gain, Goat_Status.Chlymdia_Vaccine

  Transaction Requirements: Atomic, Consistency, Isolation, Durability*/

  SELECT *
  FROM Family_Tree;

  /*Query to retrive the relations between goats
  Data Requirements: Goats.Goat_ID, Goat_Family.Child_ID, Goat_Familt.Parent_ID

  Transaction Requirements: Atomic, Consistency, Isolation, Durability
  */
```

Task 5:

For our first use case:

Use Case: Individual Goat Ranking Lookup

1. User selects individual goat
2. When individual is selected the user will be prompted to input ID
3. User enters ID

   3.a. Invalid Id entered

   1. Systems tells user invalid id
   2. Use case starts at step 3 again
4. ID will display all health information and ranked points of the goat in a list with the correlated information.
5. Users will be able to see the total ranking points for the goat ID entered and the individual points in each category
6. User can chose to check another individual goat or to main page

The view in task 3 we would use here is Goat_Health and the query we would use from task 4 will be

```
SELECT *
FROM Goat_Health;
```

For our second use case :

Use Case Goat Group Ranking Lookup:

1. System prompts user for group lookup

---

2. Guest selects which group they want to see, Total, Vaccinated, Unvaccinated
3. When vaccinated is selected, users will be able to see the overall ranking of points for the vaccinated goats.
4. When unvaccinated is selected, users will be able to see overall ranking of points for the non vaccinated goats.
5. When total is selected, users will be able to see overall ranking of points for all the goats entered in the database.
6. User can chose to check another group or to main page

We would use the view Goat Ranking view from task 3 and from task 4 we would use the query:

```
SELECT *
FROM Goat_Ranking;
```