Normalization

Our database is normalized to Boyce-Codd Normal Form. To justify this, we will walk through each normal form step by step.

1NF

First Normal Form states that a primary key must be specified, which can consist of one attribute or group of attributes, and that there are no multivalued attributes within the database. As we can see with the Goat relation, there is one primary key EID that is a unique identifier for each goat on the table. All attributes are atomic, meaning none can contain multiple values. This also applies to our second relation WEIGHT, which has a primary key consisting of EID, which is a foreign key, and WeighDate which contains a date that will be used in combination with the EID to point at which weight is being referenced to for each goat. Each of these attributes are also atomic and cannot be broken down any further.

2NF

Second Normal Form states that there must be no partial dependencies. This is only possible in a relation with a composite primary key, where part of the key is used to determine another attribute or set of attributes instead of the entire primary key. Since the Goat relation's primary key is only made up of one attribute, partial dependencies do not apply to that relation. The Weight relation has a composite primary key, however the entire primary key is needed to determine each of the other attributes in the table. Therefore, both relations are in 2NF.

3NF

The Third Normal Form gives us the information that there must be no transitive dependencies within the table. As we can see, this is true for our database, as no attributes point to other non-primary keys that ultimately point to a primary key; instead, each attribute in both tables that is not part of the primary key is dependent on the primary key and nothing but the primary key.

BCNF

Finally, there is the Boyce Codd Normal Form, which contains the last specification that dependencies must be superkeys. Which, conclusively, is shown in our relations to be true, as the key EID is a unique identifier in the Goat relation, as well as the primary key in Weights which is a superkey as well. There are no other dependencies in the system that must be considered.

In conclusion, we have found through these results that this schema is in Boyce Codd Normal Form.

SQL Views and Tablesr

```
CREATE TABLE Goat (
        visTag varchar(16) PRIMARY KEY,
        EID int UNIQUE,
        DOB date,
        Gender varchar(20),
        Dam varchar(16) REFERENCES Goat,
        Sire varchar(16) REFERENCES Goat
);

CREATE TABLE Weight (
        visTag varchar(16) REFERENCES Goat,
        WeighDate timestamp,
        W_Val int,
        W_Type text,
        PRIMARY KEY(visTag, WeighDate)
);

CREATE VIEW fAvgWt AS(
        SELECT AVG(W_Val)
        FROM Weight NATURAL JOIN Goat
        WHERE Gender = 'Female' AND W_Type = 'Birthweight'
);

CREATE VIEW mAvgWt AS(
        SELECT AVG(W_Val)
        FROM Weight NATURAL JOIN Goat
        WHERE Gender = 'Male' AND W_Type = 'Birthweight'
);

CREATE VIEW numKids_sire(Sire, CS) AS(
        SELECT Sire, COUNT(EID)
        FROM Goat
        GROUP BY Sire
);

CREATE VIEW numKids_dam(Dam, CD) AS(
        SELECT Dam, COUNT(EID)
        FROM Goat
        GROUP BY Dam
);
```

Using these views, we will be able to determine the number of kids of any sire or dam along with whether or not a specific goat falls above or below the average birthweight of their gender. These determinations will be used with outside scripts or additional queries.

SQL Queries

```
SELECT * FROM Goat;

SELECT * FROM Weight;

INSERT INTO Goat
VALUES (...)
;
-- The WHERE clause demands that the goat record does not exist

INSERT INTO Weight
VALUES (...)
;
-- The WHERE clause demands that the weight record does not exist

UPDATE Goat
SET ...
WHERE ...
;
-- The WHERE clause demands that the goat record already exists

UPDATE Weight
SET ...
WHERE ...
;
-- The WHERE clause demands that the weight record already exists

DELETE FROM Goat
WHERE ...
;
-- The WHERE clause demands that the goat record already exists

DELETE FROM Weight
WHERE ...
;
-- The WHERE clause demands that the weight record already exists
```

Explanation

The first two queries will be used to display the information from both relations into tables that the users can see when viewing the database. The additional attributes Num_kids and Above_or_below_avg_wt will be implemented later on with the use of SQL conditional statements or outside scripts. Num_kids will need to use the information from the views numKids_sire and numKids_dam, while Above_or_below_avg_wt will need to use the information from the views fAvgwt and mAvgWt. Viewing the two tables will be allowed by any user, whether they are an authorized employee or an unauthorized employee. The rest of the queries, which are the CRUD operations, are only allowed to be used by authorized employees. This is because these modify the database, which cannot be allowed by anyone that does not have real-world permissions to do so.