Raya Young

IST659 Section M404

Project Milestone 2

09/14/2019

INDEX

Raya Young
09/14/2019
Project Milestone 2

Project Summary

The purpose of this database is to track a breeding program at a local aviary. Breeding pairs must be tracked to ensure the health and genetic diversity of offspring and future generations. The importance of this tracking is far-reaching, as it not only benefits the birds staying at the aviary, but also benefits partnering zoos who may purchase birds for cost, and some birds may be released into the wild to help struggling populations. Birds at this aviary may be from partnering zoos, born in-house, or captured from the wild. The information gained from this database benefits veterinary care (offering family history), and observational data used for research about local versus wild populations.

Stakeholder Description

Stakeholders include:

- Zookeepers, origin zoo (aviary)
- Veterinarians
- Zookeepers, partner zoo
- Researchers
- Other aviary and animal care staff (avoid inbreeding, do not house certain pairs together for enclosure cleaning, etc.)

Data Questions

1. Who are the top 3 most/least prolific breeding pairs?

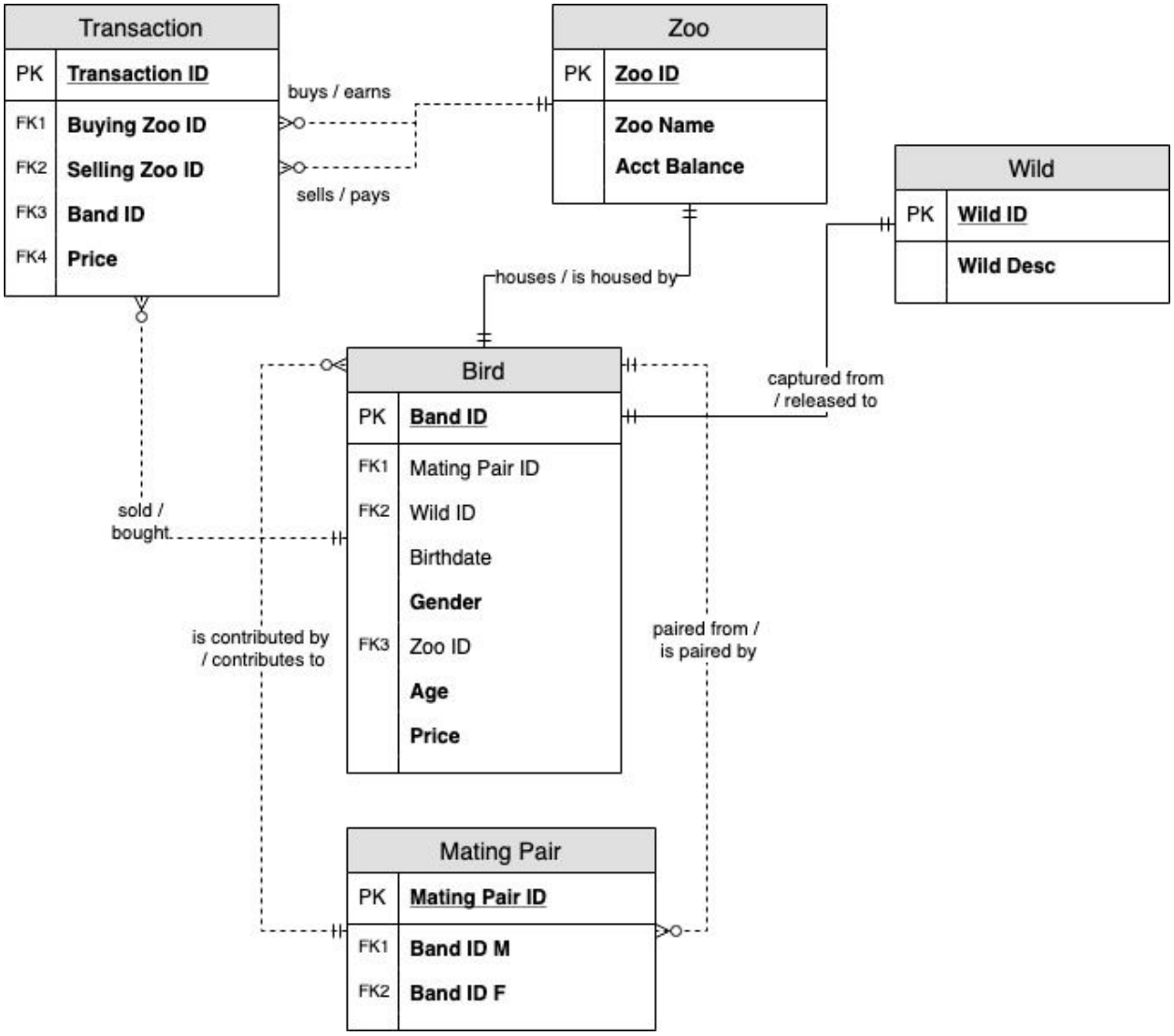
2. Who are the top 3 most/least prolific male/female of the breeding pairs?

3. What is the average number of offspring by zoo?

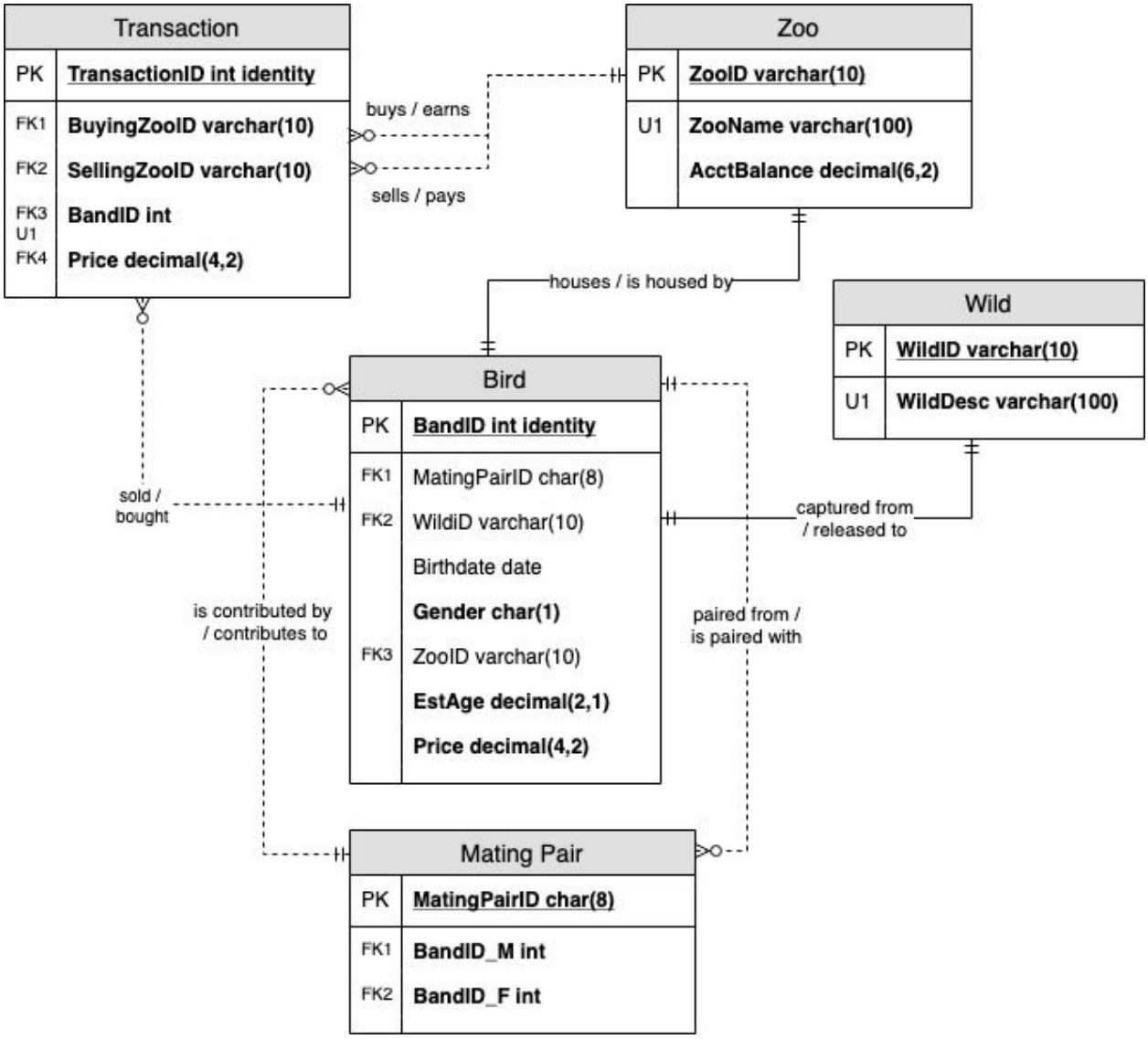
4. What is the average number of male/female offspring per zoo?

5. How much revenue has the producing facility earned (this year) through the breeding program?

Conceptual Model Diagram



Logical Model Diagram



Glossary

- **AcctBalance** Account balance of individual zoos.
- **BandID** Primary Key, identification attribute of Bird entity.
- **BandID_F** Female bird of mating pair.
- **BandID_M** Male bird of mating pair.
- **Bird** Entity containing all birds in the program.
- **Birthdate** Attribute of Bird entity, not required if bird was obtained from the wild.
- **BuyingZooID** Identification attribute of Transaction entity, zoo purchasing birds.
- **EstAge** Attribute of Bird entity. Estimated Age is important for breeding age. Birthdates are not available if bird was obtained from the wild, so estimated age is required for all birds.
- **Gender** Required attribute of Bird entity, male or female.
- **Mating_Pair** Entity, details about mating pairs.
- **MatingPairID** Composite Primary Key containing two band_ids (male and female), identification attribute of Mating Pair entity; Foreign key attribute of Bird entity. Some birds are obtained from the wild, so mating_pair_id is not required.
- **Price** Attribute of various entities, detailing cost or account balance.
- **SellingZooID** Identification attribute of Transaction entity, zoo selling birds.
- **Transaction** Entity, transaction details.
- **TransactionID** Primary Key, identification attribute of Transaction entity.
- **Wild** Entity, birds can be obtained from, or released to locations in the wild.
- **WildDesc** Text string name of a wild location.
- **WildID** Primary Key of Wild entity. Indicates current location of bird if filled.
- **Zoo** Entity, participates in transactions, houses birds.
- **ZooID** Primary Key, identification attribute of Zoo entity. Indicates current location of bird if filled.
- **ZooName** Text string name of a Zoo.

RAW DATA SAMPLES

What follows are the “origin” tables, ones that detail the participating locations, as well as the initial breeding stock. The output of the database, detailed in the reports section, show the interaction of these tables in the form of transactions between zoos, and successful mating pairs which produce offspring to insert into the Bird table.

WILD

Wild ID Code	Wild Facility Name
BAYOU	Bayou Sauvage National Wildlife Refuge
CWRS	Calgary Wildlife Rehabilitation Society
FOSSIL	Fossil Rim Wildlife Center
NYDEC	New York State Dept. of Environmental Conservation
WRCPA	Wild Resource Conservation Program, PA

ZOO

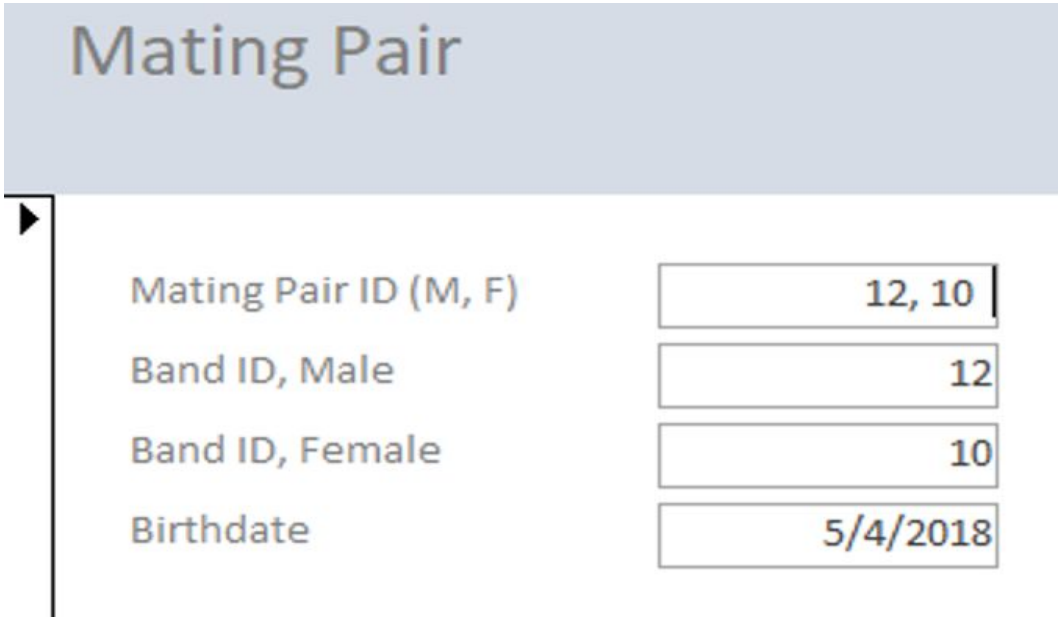
Zoo ID Code	Zoo Facility Name	Account Balance
ADBN	Audubon Zoo	$ 1,000.00
BUFF	Buffalo Zoo	$ 1,000.00
CALG	Calgary Zoo	$ 1,000.00
DALL	Dallas Zoo	$ 1,000.00
ERIE	Erie Zoo	$ 1,000.00

BIRD

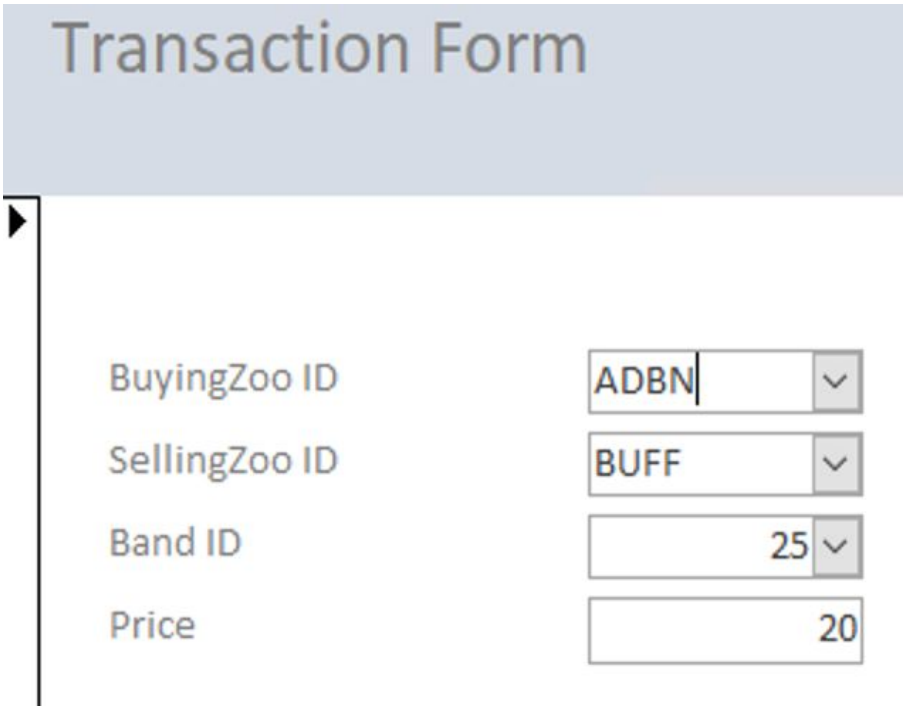
Band ID Number	Parents' Band ID Numbers (M,F) (if known)	Wild ID Code (if current location)	Birthdate (if known)	Gender	Zoo ID Code (if current location)	Estimated Age	Price
1	NULL	NULL	NULL	F	ADBN	2	$ 25.00
2	NULL	NULL	NULL	M	ADBN	2.5	$ 20.00
3	NULL	NULL	NULL	F	BUFF	3	$ 25.00
4	NULL	NULL	NULL	M	BUFF	3.5	$ 20.00
5	NULL	NULL	NULL	F	ADBN	4	$ 25.00

IMPLEMENTATION - Data entry/maintenance forms, reports for the 5 data questions

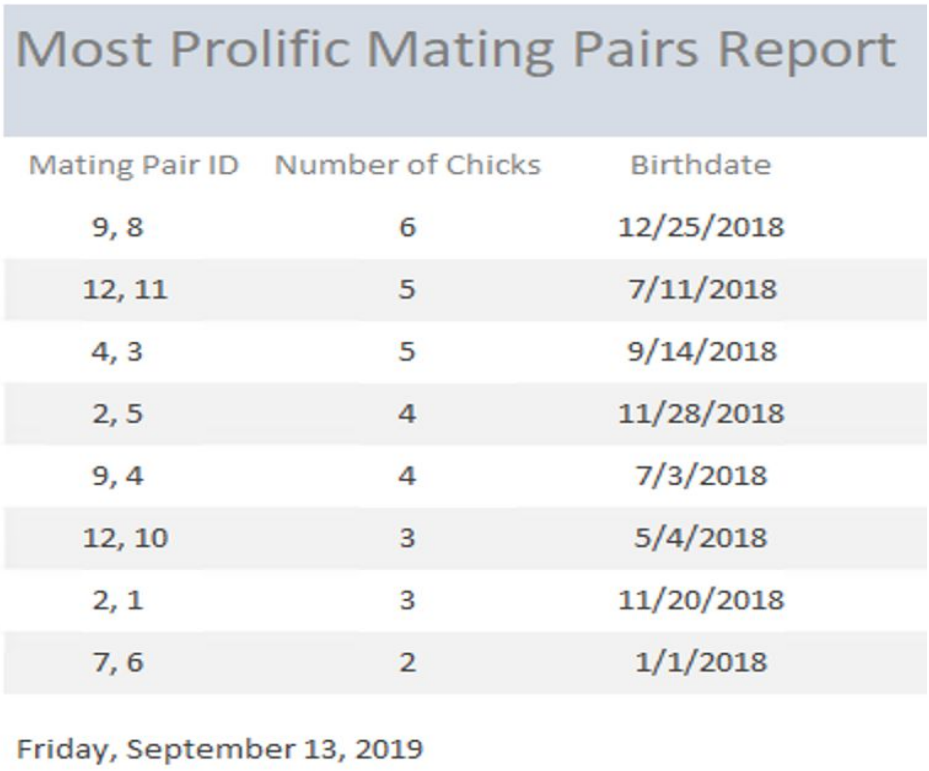
- **MATING PAIR FORM**
  For inputting Male, Female BandIDs as MatingPairIDs, and assigning this information to the profile of resulting chicks, including Birthdate (hatching)



- **TRANSACTION FORM**
  For inputting Buy/Sell interactions by Zoo ID. Includes input for each bird BandID, and their associated price.



1. **Who are the top 3 most/least prolific breeding pairs?**
   From the image below, we can see the top three breeding pairs are (9,8) First; (12,11) and (4,3) tied for Second; and (2,5) and (9,4) tied for Third.



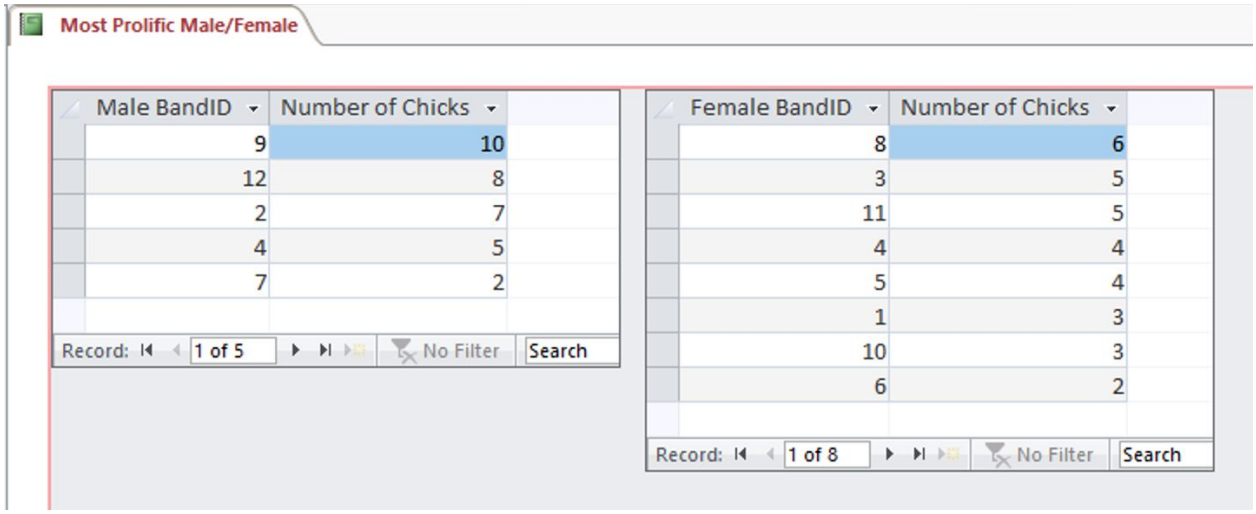
Most Prolific Mating Pairs Report

Mating Pair ID	Number of Chicks	Birthdate
9, 8	6	12/25/2018
12, 11	5	7/11/2018
4, 3	5	9/14/2018
2, 5	4	11/28/2018
9, 4	4	7/3/2018
12, 10	3	5/4/2018
2, 1	3	11/20/2018
7, 6	2	1/1/2018

Friday, September 13, 2019

```
--1. DATA QUESTION Who is the most prolific pair?
CREATE VIEW MostProlificPair AS
SELECT DISTINCT Bird.MatingPairID AS 'Mating Pair ID'
        , Bird.Birthdate AS 'Birthdate'
        , dbo.NumChicksPair(MatingPairID) AS NumberofChicks
FROM Bird WHERE MatingPairID IS NOT NULL
--ORDER BY NumberofChicks DESC
GO
```

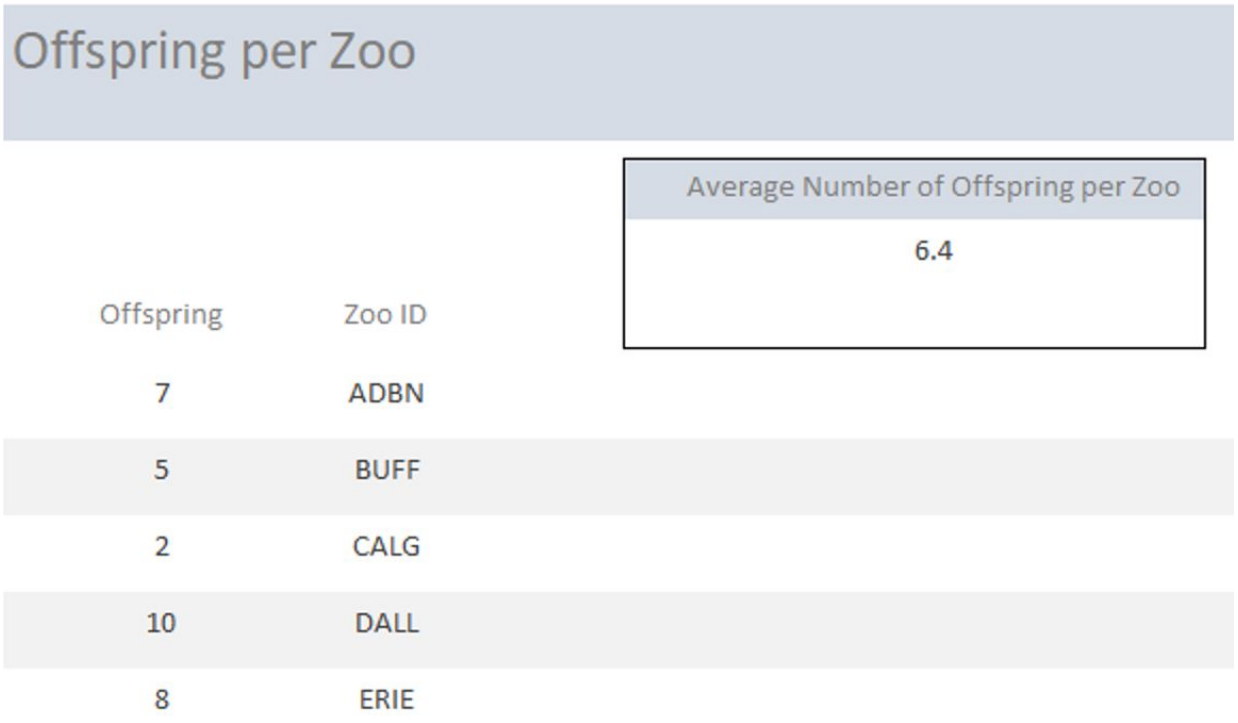
2. **Who are the top 3 most/least prolific male/female of the breeding pairs?**
   Top male performers are 10, 8, and 7, and top females are 8; 3 and 11 (tied); and 4 and 5 (tied). This output matches the Mating Pairs Report, because each female was only a member of one mating pair, while males participated in more (in this instance).



(SQL script on next page)

```
	--2. DATA QUESTION Most Prolific Male/Female

--NumChicksMale Function and SELECT DISTINCT
CREATE FUNCTION dbo.NumChicksMale(@bandID_M int)
RETURNS int AS -- COUNT() is an integer value, so return it as an int
BEGIN
	DECLARE @returnValue int -- matches the function's return type
	SELECT @returnValue = COUNT(*) FROM Mating_Pair
	INNER JOIN Bird ON Mating_Pair.MatingPairID = Bird.MatingPairID
	WHERE Mating_Pair.BandID_M = @bandID_M
	RETURN @returnValue
END
GO

SELECT DISTINCT Mating_Pair.BandID_M AS 'Male BandID'
	, dbo.NumChicksMale(BandID_M) AS 'Number of Chicks'
FROM Mating_Pair
	INNER JOIN Bird ON Mating_Pair.MatingPairID = Bird.MatingPairID
	WHERE BandID_M IS NOT NULL
	ORDER BY 'Number of Chicks' DESC

--Most Prolific Female FUNCTION and SELECT DISTINCT
GO
CREATE FUNCTION dbo.NumChicksFemale(@bandID_F int)
RETURNS int AS -- COUNT() is an integer value, so return it as an int
BEGIN
	DECLARE @returnValue int -- matches the function's return type
	SELECT @returnValue = COUNT(*) FROM Mating_Pair
	INNER JOIN Bird ON Mating_Pair.MatingPairID = Bird.MatingPairID
	WHERE Mating_Pair.BandID_F = @bandID_F
	RETURN @returnValue
END
GO

SELECT DISTINCT Mating_Pair.BandID_F AS 'Female BandID'
	, dbo.NumChicksFemale(BandID_F) AS 'Number of Chicks'
FROM Mating_Pair
	INNER JOIN Bird ON Mating_Pair.MatingPairID = Bird.MatingPairID
	WHERE BandID_F IS NOT NULL
	ORDER BY 'Number of Chicks' DESC
```

3. **What is the average number of offspring per zoo?**
   *Please note that this indicates current location, and not producing zoo.



Offspring per Zoo

Offspring	Zoo ID
7	ADBN
5	BUFF
2	CALG
10	DALL
8	ERIE

Average Number of Offspring per Zoo
6.4

Saturday, September 14, 2019

```
	--1. DATA QUESTION Who is the most prolific pair?
CREATE VIEW MostProlificPair AS
SELECT DISTINCT Bird.MatingPairID AS 'Mating Pair ID'
	, Bird.Birthdate AS 'Birthdate'
	, dbo.NumChicksPair(MatingPairID) AS NumberofChicks
FROM Bird WHERE MatingPairID IS NOT NULL
--ORDER BY NumberofChicks DESC
GO
```

4. **What is the average number of male/female offspring per zoo?**
   *Please note that this indicates current location, and not producing zoo.

Male/Female Offspring per Zoo

Male Offspring	Zoo ID
4	ADBN
4	BUFF
7	DALL
4	ERIE

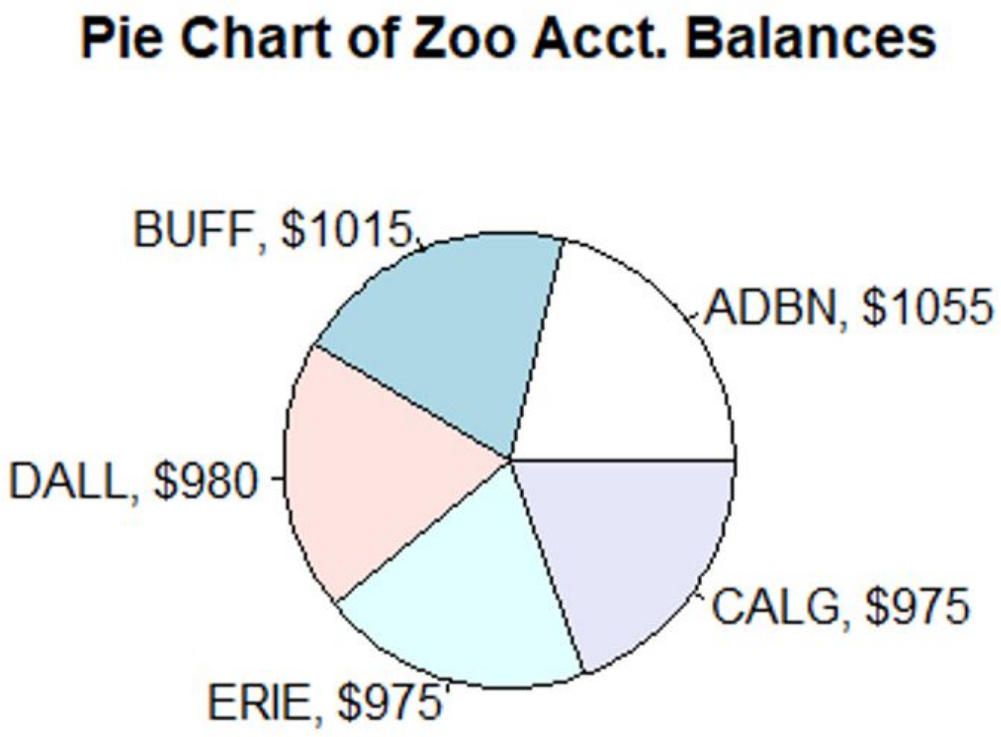
Female Offspring	Zoo ID
3	ADBN
1	BUFF
2	CALG
3	DALL
4	ERIE

Avg. # Male Offspring per Zoo
4.75

Avg. # Female Offspring per Zoo
2.6

```
--4. DATA QUESTION Average number of Male/Female offspring per zoo
GO
CREATE VIEW AvgFPerZoo AS
SELECT COUNT(Gender) AS FemaleOffspring, ZooID
FROM Bird
WHERE  Gender = 'F' AND MatingPairID IS NOT NULL
GROUP BY ZooID
GO

SELECT AVG(FemaleOffSpring) AS 'Avg. Female Offspring per Zoo'
FROM AvgFPerZoo

GO
CREATE VIEW AvgMPerZoo AS
SELECT COUNT(Gender) AS MaleOffspring, ZooID
FROM Bird
WHERE Gender = 'M' AND MatingPairID IS NOT NULL
GROUP BY ZooID
GO

SELECT AVG(MaleOffspring) AS 'Avg. Male Offspring per Zoo'
FROM AvgMPerZoo
```

5. **How much revenue has the producing facility earned (this year) through the breeding program?**
   Each zoo began the program with an account balance of $1000.00. Based on this information, we can see that Audubon Zoo (“ADBN”) is the highest earning zoo for this period.




```
--5. DATA QUESTION VIEW ZooRevenue
SELECT AcctBalance, ZooName
FROM Zoo
ORDER BY AcctBalance DESC
```

REFLECTION -

- **How did your assumptions from the start of the project change?**
- **What would you do differently?**
    - My assumptions at the start of the project were that this would be a fully functioning and interactive database, ready to deploy. Detailed further in the next section, I spent a lot of time trying to craft relationships for the input between tables, and even imagined a IF/THEN, to prevent inbreeding. I also found that in constructing the repeatable script, and later while creating the statements to answer the data questions - that some of the FKs and CONSTRAINTS need to be updated or omitted entirely, as some were found to be unnecessary. Truly, my learning evolved throughout this construction.

    - Based on what I have learned, some things I would do differently are:
        - Budget my time differently. As mentioned above, I spent the bulk of the time allotted trying to dream up some very complicated procedures, and learned later that it was okay to put in 'placeholder' data for the purposes of demonstration. I now see these ideas as opportunities for expansion, like creating a Breeding procedure, that generates a number of chicks and their attributes, placing them into the Bird table. I was able to do this successfully with the Transaction procedure. What I learned from this is that it may be best to put in the 'placeholder' data even if you will ultimately craft a procedure such as Breeding, so that you can pinpoint any errors in the interaction of these fields as you craft each statement. Later versions of this database would also eliminate the need for BandID_M and BandID_F as the Breeding procedure would check the intakes for the Gender attribute.

        - Other hiccups include the report for Data Question 2. I was able to generate an image using the 'Blank Report' template, but when trying to convert it to other views, it generated an endless loop of ODBC connections, sometimes crashing Access. My guess is that it was connecting for each instance of 'chick' for both male and female tables, which in this case would be 64 times. I used passthrough queries for each table, and perhaps there is another solution that could alleviate this issue.

        - Question 3 could have been answered more intuitively with a bit of database redesign, by including attribute 'Birthplace' or 'Origin' for each Bird.

- **Updates to Conceptual and Logical Models**
  - Updated Conceptual and Logical Models to reflect updates to decimals; updates to MatingPairID, removed some unnecessary UNIQUE CONSTRAINTS; as well updating columns as named in the SQL script.

- **Updates to Glossary**
  - Updated Glossary to match naming conventions used in the SQL script. Adding definitions for BandID_M, BandID_F.

- **Updates to Data Questions**
  - Changed Question 3 from ‘Average number of offspring per nest?’ to ‘Average number of offspring per zoo?’ because the original was too similar to Question 1. The revision is helpful information for the respective zoos.
  - Changed Question 4 from ‘Average number of male to female offspring per nest?’ to ‘Average number of male to female offspring per zoo?’ because the original was proving difficult to code with the existing database design. The revision provides helpful information for the respective zoos, similar to Revised Question 3.

SQL SCRIPT
PHYSICAL DATABASE DESIGN; DATA CREATION; DATA MANIPULATION

```
/*
        Author : Raya Young
        Course: IST659 M404
        Term    : July, 2019
*/

-- drop all constraints in reverse order of their dependencies.
IF OBJECT_ID('dbo.Transaction', 'U') IS NOT NULL
        ALTER TABLE "Transaction"
        DROP CONSTRAINT FK3_Transaction,
                CONSTRAINT FK2_Transaction,
                CONSTRAINT FK1_Transaction
GO

IF OBJECT_ID('dbo.Bird', 'U') IS NOT NULL
        ALTER TABLE Bird
        DROP CONSTRAINT FK2_Bird,
                CONSTRAINT FK1_Bird
GO

-- drop all tables in reverse order of their dependencies.
IF OBJECT_ID('dbo.Transaction', 'U') IS NOT NULL
        DROP TABLE "Transaction"
GO

IF OBJECT_ID('dbo.Mating_Pair', 'U') IS NOT NULL
        DROP TABLE Mating_Pair
GO

IF OBJECT_ID('dbo.Bird', 'U') IS NOT NULL
        DROP TABLE Bird
GO

IF OBJECT_ID('dbo.Wild', 'U') IS NOT NULL
        DROP TABLE Wild
GO

IF OBJECT_ID('dbo.Zoo', 'U') IS NOT NULL
        DROP TABLE Zoo
GO

-- drop all procedures, views, functions...
IF OBJECT_ID('BuySell', 'P') IS NOT NULL
        DROP PROCEDURE dbo.BuySell;
GO
```

```
IF OBJECT_ID('WildRelease', 'P') IS NOT NULL
	DROP PROCEDURE dbo.WildRelease;
GO

IF OBJECT_ID('ChicksPerPair', 'V') IS NOT NULL
	DROP VIEW dbo.ChicksPerPair;
GO

IF OBJECT_ID('MostProlificPair', 'V') IS NOT NULL
	DROP VIEW dbo.MostProlificPair;
GO

IF OBJECT_ID('AvgOPerZoo', 'V') IS NOT NULL
	DROP VIEW dbo.AvgOPerZoo;
GO

IF OBJECT_ID('AvgFPerZoo', 'V') IS NOT NULL
	DROP VIEW dbo.AvgFPerZoo;
GO

IF OBJECT_ID('AvgMPerZoo', 'V') IS NOT NULL
	DROP VIEW dbo.AvgMPerZoo;
GO

IF OBJECT_ID('NumChicksPair') IS NOT NULL
	DROP FUNCTION dbo.NumChicksPair;
GO

IF OBJECT_ID('NumChicksMale') IS NOT NULL
	DROP FUNCTION dbo.NumChicksMale;
GO

IF OBJECT_ID('NumChicksFemale') IS NOT NULL
	DROP FUNCTION dbo.NumChicksFemale;
GO

-- Creating the Zoo table
CREATE TABLE Zoo (
	-- Columns for the Zoo table
	ZooID varchar(10) NOT NULL,
	ZooName varchar(100) NOT NULL,
	AcctBalance decimal(6,2) NOT NULL,
	-- Constraints on the Zoo table
	CONSTRAINT PK_Zoo PRIMARY KEY (ZooID),
	CONSTRAINT U1_Zoo UNIQUE(ZooName)
)
-- End creating the Zoo table
```

```
-- Adding data to the Zoo table
INSERT INTO Zoo(ZooID, ZooName, AcctBalance)
	VALUES
		('ADBN', 'Audubon Zoo', 1000.00),
		('BUFF', 'Buffalo Zoo', 1000.00),
		('CALG', 'Calgary Zoo', 1000.00),
		('DALL', 'Dallas Zoo', 1000.00),
		('ERIE', 'Erie Zoo', 1000.00)

-- Creating the Wild table
CREATE TABLE Wild (
	-- Columns for the Wild table
	WildID varchar(10) NOT NULL,
	WildDescription varchar(100) NOT NULL,
	-- Constraints on the Wild table
	CONSTRAINT PK_Wild PRIMARY KEY (WildID),
	CONSTRAINT U1_Wild UNIQUE(WildDescription)
)
-- End creating the Wild table

-- Adding data to the Wild table
INSERT INTO Wild(WildID, WildDescription)
	VALUES
		('BAYOU', 'Bayou Sauvage National Wildlife Refuge'),
		('NYDEC', 'New York State Dept. of Environmental Conservation'),
		('CWRS', 'Calgary Wildlife Rehabilitation Society'),
		('FOSSIL', 'Fossil Rim Wildlife Center'),
		('WRCPA', 'Wild Resource Conservation Program, PA')

SELECT * FROM Wild

-- Creating the Bird table
CREATE TABLE Bird (
	-- Columns for the Bird table
	BandID int identity NOT NULL,
	MatingPairID char(8),
	WildID varchar(10),
	Birthdate date,
	Gender char(1) NOT NULL,
	ZooID varchar(10),
	EstAge decimal(2,1) NOT NULL,
	Price decimal(4,2) NOT NULL,
	-- Constraints on the Bird table
	CONSTRAINT PK_Bird PRIMARY KEY (BandID),
	CONSTRAINT FK1_Bird	FOREIGN KEY (ZooID) REFERENCES Zoo(ZooID),
	CONSTRAINT FK2_Bird FOREIGN KEY (WildID) REFERENCES Wild(WildID)
)
```

```
-- End creating the Bird table

-- Adding data to the Bird table
INSERT INTO Bird(Gender, ZooID, EstAge, Price)
	VALUES
		('F', 'ADBN', 2.0, 25.00),
		('M', 'ADBN', 2.5, 20.00),
		('F', 'BUFF', 3.0, 25.00),
		('M', 'BUFF', 3.5, 20.00),
		('F', 'ADBN', 4.0, 25.00),
		('F', 'CALG', 2.0, 25.00),
		('M', 'CALG', 2.5, 20.00),
		('F', 'DALL', 3.0, 25.00),
		('M', 'DALL', 3.5, 20.00),
		('F', 'ERIE', 4.0, 25.00),
		('F', 'ERIE', 3.0, 25.00),
		('M', 'ERIE', 3.5, 20.00)

SELECT Gender, ZooID, EstAge FROM Bird

-- Creating the Mating Pair table
CREATE TABLE Mating_Pair (
	-- Columns for the Mating Pair table
	MatingPairID char(8) NOT NULL,
	BandID_M int NOT NULL,
	BandID_F int NOT NULL,
	Birthdate date
	-- Constraints on the Mating Pair table
	CONSTRAINT PK_Mating_Pair PRIMARY KEY (MatingPairID)
)
-- Add Foreign Key Constraints from Bird table to Mating_Pair table
ALTER TABLE Mating_Pair
ADD CONSTRAINT FK1_Mating_Pair FOREIGN KEY (BandID_M) REFERENCES
Bird(BandID),
	CONSTRAINT FK2_Mating_Pair FOREIGN KEY (BandID_F) REFERENCES
Bird(BandID)
-- End creating the Mating Pair table

-- Adding data to Mating_Pair table
INSERT INTO Mating_Pair(MatingPairID, BandID_M, BandID_F, Birthdate)
	VALUES
		('2, 1', 2, 1, '2018-11-20'),
		('2, 5', 2, 5, '2018-11-28'),
		('4, 3', 4, 3, '2018-09-14'),
		('7, 6', 7, 6, '2018-01-01'),
		('9, 8', 9, 8, '2018-12-25'),
		('9, 4', 9, 4, '2018-07-03'),
		('12, 10', 12, 10, '2018-05-04'),
```

```
                ('12, 11', 12, 11, '2018-07-11')

SELECT * FROM Mating_Pair

-- Adding more data (offspring) to Bird table
INSERT INTO Bird(MatingPairID, Birthdate, Gender, ZooID, EstAge, Price)
        VALUES
                ('2, 1', '2018-11-20', 'F', 'ADBN', 0.5, 25.00),
                ('2, 1', '2018-11-20', 'F', 'ADBN', 0.5, 25.00),
                ('2, 1', '2018-11-20', 'M', 'ADBN', 0.5, 20.00),
                ('2, 5', '2018-11-28', 'F', 'ADBN', 0.5, 25.00),
                ('2, 5', '2018-11-28', 'M', 'ADBN', 0.5, 20.00),
                ('2, 5', '2018-11-28', 'M', 'ADBN', 0.5, 20.00),
                ('2, 5', '2018-11-28', 'M', 'ADBN', 0.5, 20.00),
                ('4, 3', '2018-09-14', 'M', 'BUFF', 1.0, 20.00),
                ('4, 3', '2018-09-14', 'F', 'BUFF', 1.0, 25.00),
                ('4, 3', '2018-09-14', 'M', 'BUFF', 1.0, 20.00),
                ('4, 3', '2018-09-14', 'M', 'BUFF', 1.0, 20.00),
                ('4, 3', '2018-09-14', 'M', 'BUFF', 1.0, 20.00),
                ('7, 6', '2018-01-01', 'F', 'CALG', 1.5, 25.00),
                ('7, 6', '2018-01-01', 'F', 'CALG', 1.5, 25.00),
                ('9, 4', '2018-07-03', 'F', 'DALL', 1.0, 25.00),
                ('9, 4', '2018-07-03', 'M', 'DALL', 1.0, 20.00),
                ('9, 4', '2018-07-03', 'M', 'DALL', 1.0, 20.00),
                ('9, 4', '2018-07-03', 'M', 'DALL', 1.0, 20.00),
                ('9, 8', '2018-12-25', 'M', 'DALL', 0.5, 20.00),
                ('9, 8', '2018-12-25', 'F', 'DALL', 0.5, 25.00),
                ('9, 8', '2018-12-25', 'M', 'DALL', 0.5, 20.00),
                ('9, 8', '2018-12-25', 'M', 'DALL', 0.5, 20.00),
                ('9, 8', '2018-12-25', 'M', 'DALL', 0.5, 20.00),
                ('9, 8', '2018-12-25', 'F', 'DALL', 0.5, 25.00),
                ('12, 10', '2018-05-04', 'F', 'ERIE', 1.5, 25.00),
                ('12, 10', '2018-05-04', 'F', 'ERIE', 1.5, 25.00),
                ('12, 10', '2018-05-04', 'M', 'ERIE', 1.5, 20.00),
                ('12, 11', '2018-07-11', 'F', 'ERIE', 1.0, 25.00),
                ('12, 11', '2018-07-11', 'M', 'ERIE', 1.0, 20.00),
                ('12, 11', '2018-07-11', 'M', 'ERIE', 1.0, 20.00),
                ('12, 11', '2018-07-11', 'M', 'ERIE', 1.0, 20.00),
                ('12, 11', '2018-07-11', 'F', 'ERIE', 1.0, 25.00)

-- Creating the Transaction table
CREATE TABLE "Transaction" (
        -- Columns for the Transaction table
        TransactionID int identity NOT NULL,
        BuyingZooID varchar(10) NOT NULL,
        SellingZooID varchar(10) NOT NULL,
        BandID int NOT NULL,
        Price decimal(4,2) NOT NUll,
```

```
	-- Constraints on the Transaction table
	CONSTRAINT PK_Transaction PRIMARY KEY (TransactionID),
	CONSTRAINT FK1_Transaction FOREIGN KEY (BuyingZooID) REFERENCES
Zoo(ZooID),
	CONSTRAINT FK2_Transaction FOREIGN KEY (SellingZooID) REFERENCES
Zoo(ZooID),
	CONSTRAINT FK3_Transaction FOREIGN KEY (BandID) REFERENCES Bird(BandID)
)
-- End creating the Transaction table

GO
-- Create Transaction ("BuySell") Procedure
CREATE PROCEDURE BuySell(@buyingZoo varchar(10), @sellingZoo varchar(10), @bandID
int) AS
BEGIN
	DECLARE @transID int
	SELECT @transID = TransactionID FROM "Transaction"
	WHERE	BuyingZooID = @buyingZoo
	DECLARE @price decimal(4,2)
	SELECT @price = Price FROM Bird
	WHERE	BandID = @bandID
	INSERT INTO "Transaction" (BuyingZooID, SellingZooID, BandID, Price)
	VALUES (@buyingZoo, @sellingZoo, @bandID, @price)
-- Update AcctBalance(s) in Zoo table
	UPDATE Zoo
	SET AcctBalance = AcctBalance - (SELECT Price FROM Bird WHERE BandID =
@bandID)
	WHERE ZooID = @buyingZoo;
	UPDATE Zoo
	SET AcctBalance = AcctBalance + (SELECT Price FROM Bird WHERE BandID =
@bandID)
	WHERE ZooID = @sellingZoo
-- Update Bird location (ZooID)
	UPDATE Bird
	SET ZooID = @buyingZoo
	WHERE BandID = @bandID
-- Return TransactionID
	RETURN SCOPE_IDENTITY()
END
GO

-- Creating data for the "Transaction" table
EXEC BuySell'ADBN', 'BUFF', 2
EXEC BuySell'DALL', 'BUFF', 4
EXEC BuySell'CALG', 'ADBN', 1
EXEC BuySell'BUFF', 'ADBN', 3
EXEC BuySell'ERIE', 'ADBN', 5
```

```
SELECT * FROM "Transaction"

SELECT * FROM Zoo

SELECT * FROM Bird

GO
 --Create WildRelease Procedure
CREATE PROCEDURE WildRelease(@bandID int, @wildID varchar(10)) AS
BEGIN
-- Update Bird location (WildID)
	UPDATE Bird
	SET WildID = @wildID
		, ZooID = NULL
	WHERE BandID = @bandID
END
GO

EXEC WildRelease 1, BAYOU

GO

CREATE FUNCTION dbo.NumChicksPair(@matingPairID char(8))
RETURNS int AS -- COUNT() is an integer value, so return it as an int
BEGIN
	DECLARE @returnValue int -- matches the function's return type
	SELECT @returnValue = COUNT(*) FROM Bird
	WHERE Bird.MatingPairID = @matingPairID
	RETURN @returnValue
END
GO

--DATA QUESTIONS

	--1. DATA QUESTION Who is the most prolific pair?

CREATE VIEW MostProlificPair AS
SELECT DISTINCT Bird.MatingPairID AS 'Mating Pair ID'
	, Bird.Birthdate AS 'Birthdate'
	, dbo.NumChicksPair(MatingPairID) AS NumberofChicks
FROM Bird WHERE MatingPairID IS NOT NULL
--ORDER BY NumberofChicks DESC
GO

	--2. DATA QUESTION Most Prolific Male/Female

--NumChicksMale Function and SELECT DISTINCT
CREATE FUNCTION dbo.NumChicksMale(@bandID_M int)
```

```
RETURNS int AS -- COUNT() is an integer value, so return it as an int
BEGIN
	DECLARE @returnValue int -- matches the function's return type
	SELECT @returnValue = COUNT(*) FROM Mating_Pair
	INNER JOIN Bird ON Mating_Pair.MatingPairID = Bird.MatingPairID
	WHERE Mating_Pair.BandID_M = @bandID_M
	RETURN @returnValue
END
GO

SELECT DISTINCT Mating_Pair.BandID_M AS 'Male BandID'
	, dbo.NumChicksMale(BandID_M) AS 'Number of Chicks'
FROM Mating_Pair
	INNER JOIN Bird ON Mating_Pair.MatingPairID = Bird.MatingPairID
	WHERE BandID_M IS NOT NULL
	ORDER BY 'Number of Chicks' DESC

--Most Prolific Female FUNCTION and SELECT DISTINCT
GO
CREATE FUNCTION dbo.NumChicksFemale(@bandID_F int)
RETURNS int AS -- COUNT() is an integer value, so return it as an int
BEGIN
	DECLARE @returnValue int -- matches the function's return type
	SELECT @returnValue = COUNT(*) FROM Mating_Pair
	INNER JOIN Bird ON Mating_Pair.MatingPairID = Bird.MatingPairID
	WHERE Mating_Pair.BandID_F = @bandID_F
	RETURN @returnValue
END
GO

SELECT DISTINCT Mating_Pair.BandID_F AS 'Female BandID'
	, dbo.NumChicksFemale(BandID_F) AS 'Number of Chicks'
FROM Mating_Pair
	INNER JOIN Bird ON Mating_Pair.MatingPairID = Bird.MatingPairID
	WHERE BandID_F IS NOT NULL
	ORDER BY 'Number of Chicks' DESC

	--3. DATA QUESTION Average number of offspring per zoo (current location)

GO
CREATE VIEW AvgOPerZoo AS
SELECT COUNT(BandID) AS Offspring, ZooID
FROM Bird
WHERE MatingPairID IS NOT NULL
GROUP BY ZooID
GO

SELECT AVG(Offspring) AS 'Avg. Offspring per Zoo'
```

```
FROM AvgOPerZoo

        --4. DATA QUESTION Average number of Male/Female offspring per zoo

GO
CREATE VIEW AvgFPerZoo AS
SELECT COUNT(Gender) AS FemaleOffspring, ZooID
FROM Bird
WHERE  Gender = 'F' AND MatingPairID IS NOT NULL
GROUP BY ZooID
GO

SELECT AVG(FemaleOffSpring) AS 'Avg. Female Offspring per Zoo'
FROM AvgFPerZoo

GO
CREATE VIEW AvgMPerZoo AS
SELECT COUNT(Gender) AS MaleOffspring, ZooID
FROM Bird
WHERE Gender = 'M' AND MatingPairID IS NOT NULL
GROUP BY ZooID
GO

SELECT AVG(MaleOffspring) AS 'Avg. Male Offspring per Zoo'
FROM AvgMPerZoo

        --5. DATA QUESTION VIEW ZooRevenue

SELECT AcctBalance, ZooName
FROM Zoo
ORDER BY AcctBalance DESC
```