Collecting and Analyzing data about Wildlife National Parks

Milestone: Use Case Study Report

Group 4
Aditya Bharadwaj S G
Suvroneil Ghosh

+1 (857) 832-0469
+1 (857) 540-6248

shivapuraguruprasa.a@northeastern.edu
ghosh.suv@northeastern.edu

Percentage of Effort Contributed by Aditya Bharadwaj: 50%

Percentage of Effort Contributed by Suvroneil Ghosh: 50%

Signature of Aditya Bharadwaj: Aditya Bharadwaj

Signature of Suvroneil Ghosh: Suvroneil Ghosh

Submission Date: 10th December, 2022

USE CASE STUDY REPORT

Group No.: Group 04

Student Names: Aditya Bharadwaj and Suvroneil Ghosh

Executive Summary:

The Primary Objective of this study was to design and setup database for the Massachusetts State Government along with few analytics systems for National Parks, which will help them in understanding the trends in the current wildlife population present in the National Parks. These insights will also help the government in making future decisions regarding conservation of wildlife population by setting up new Wildlife national parks correctly as part of their upcoming pilot project.

In order to design the database, a set of requirements was collected in accordance with the needs of the government. Initially, the EER and UML diagrams were modelled based on the requirements collected. After that the conceptual model was mapped to relational model with the Primary and Foreign Keys. In order to setup the database, for SQL based design, we chose MySQL and once this database was setup, we chose MongoDB as our choice of NoSQL based implementation.

After the databases were created, we used Python to connect to the MySQL database to analyze the data and the results were very helpful. Over the time, we added more data to the databases to have better understanding of trends. We were able to learn various trends about the different types of animals present in the parks, age of the current animals, the types of people that visited the national parks, revenue generated from ticket sales, etc.

I. Introduction

Over the last four decades, human activities have greatly pushed some animal species to near extinction with an estimated loss of about 10,000 species per year accounting for losing half of the world’s wildlife population. This year due to critical drought conditions in Massachusetts state, many animals have faced the consequences. In these alarming circumstances, the Massachusetts state government has decided to new National Parks as part of a pilot program. In order to understand the current trends in the National Parks, the government tends to analyze the data coming from national parks in Massachusetts in order to make future decisions with respect to conservation of wildlife.

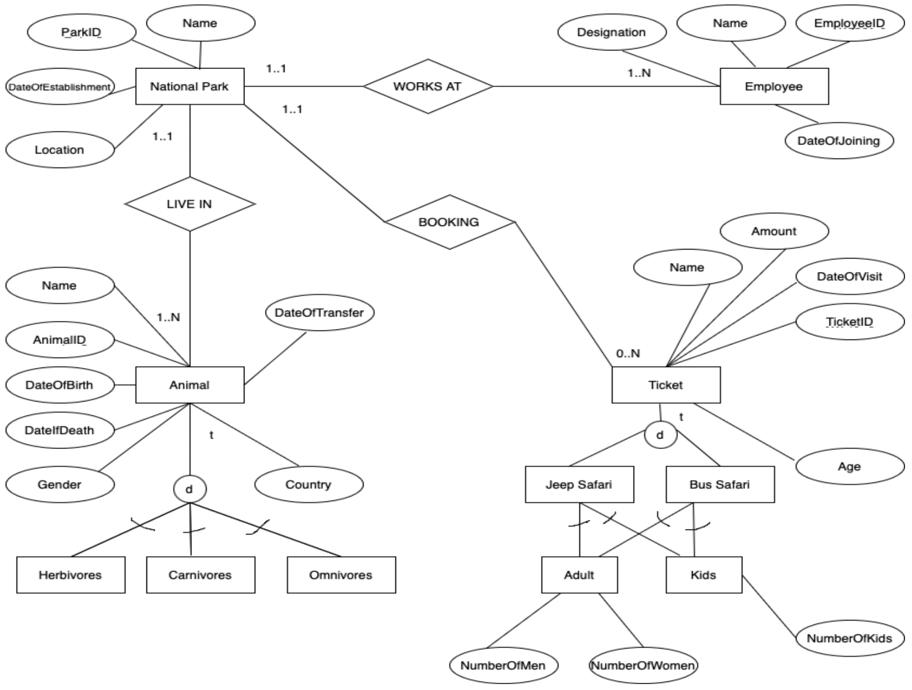
In our project, we aimed to help the government by coming up with a solution that helps them with analyzing different trends by keeping track of all the animals that are present in the national parks, keeping records of the different types of visitors, sales generated by the parks from ticket sales, etc. This was done by designing a database keeping in mind the requirements provided by the government. There are two parts to the project, the first deals with information about the national parks and the animals present in the parks. The second part deals with information about the ticket booking system.

The various requirements that we kept in mind while designing the database were:

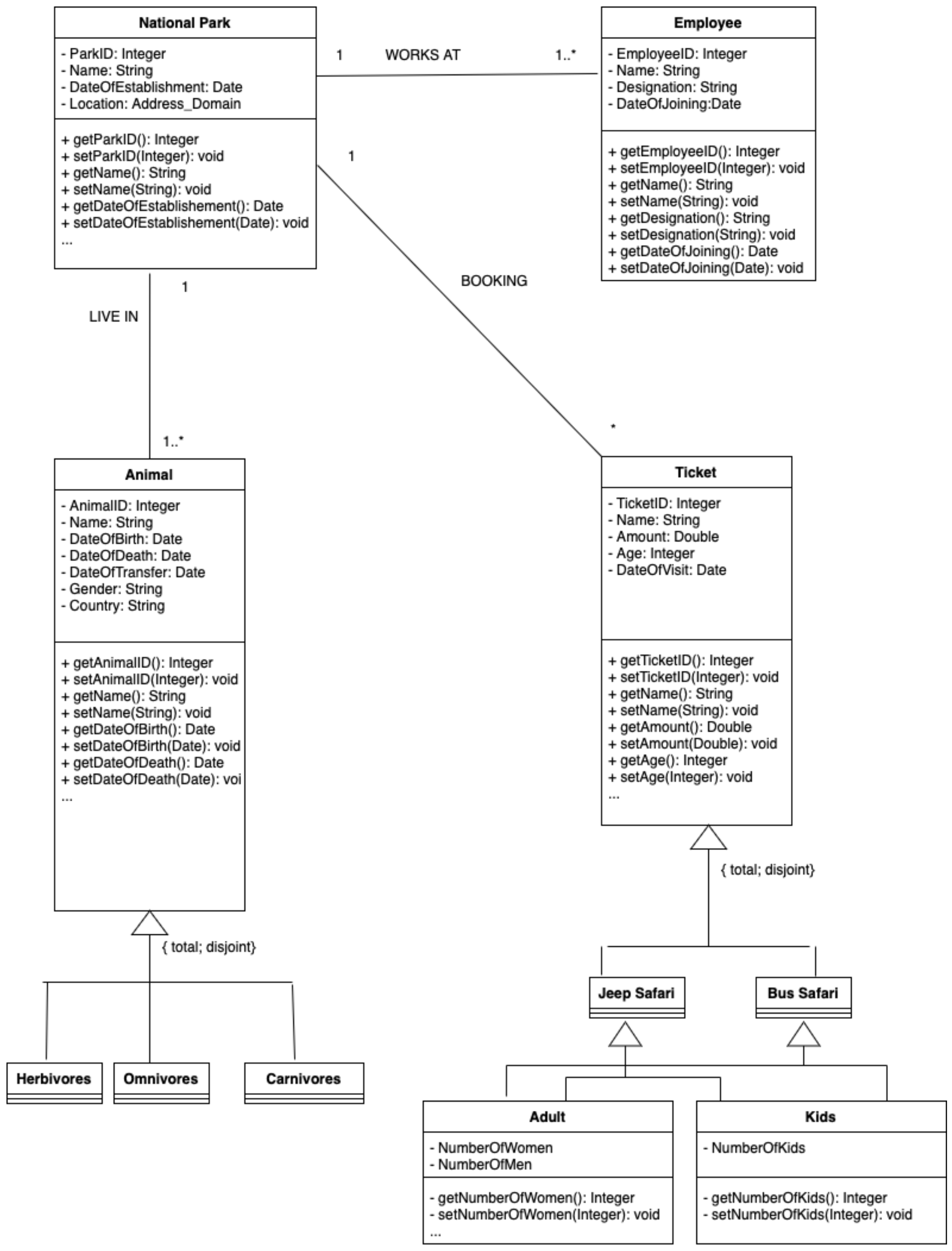
1) There can be one to N numbers of animals in a national park, but any animal can be part of only one national park
2) There will be a minimum of one employee and a maximum of N number of employees at the park who work in specific designations and one employee can work only at one national park at a time
3) There will be one or more caretakers assigned to the animals and one caretaker can be assigned to multiple animals of the same type
4) Migration of animals between national parks is possible and there can be zero to N animals that can be shifted.
5) There can be a sale of zero to an infinite number of tickets at the office on a particular day
6) Any visitor can buy at least one or many tickets on a particular date as per their requirement
7) A visitor can purchase tickets whose price varies for two main categories: Bus and Jeep safari. Both safaris can have a minimum of one visitor and a maximum of N number of visitors each

II. Conceptual Data Modeling

1. <u>EER Diagram:</u>



2. <u>**UML Diagram**</u>:



III. Mapping Conceptual Model to Relational Model

Primary Keys- **Bold**

Foreign Keys- *Italicized*

National Park (**ParkID**, Name, Date Of Establishment, Location)

Employee (**EmployeeID**, Name, Designation, Date Of Joining, *ParkID*)

- FOREIGN KEY ParkID refers to ParkID in National Park; NULL NOT ALLOWED

Animal (**AnimalID**, Name, Date Of Birth, Date Of Death, Gender, Date Of Transfer, Country, *ParkID*)

- FOREIGN KEY ParkID refers to ParkID in National Park; NULL NOT ALLOWED

HERBIVORES (*AnimalID*) AnimalID NOT NULL

CARNIVORES (*AnimalID*) AnimalID NOT NULL

OMNIVORES (*AnimalID*) AnimalID NOT NULL

TICKET (**TicketID**, Name, Amount, Date Of Visit, Age, *ParkID*)

- FOREIGN KEY ParkID refers to ParkID in National Park; NULL NOT ALLOWED

JEEP SAFARI (*TicketID*) TicketID NOT NULL

BUS SAFARI (*TicketID*) TicketID NOT NULL

ADULT (*TicketID*, Number Of Men, Number Of Women) TicketID NOT NULL

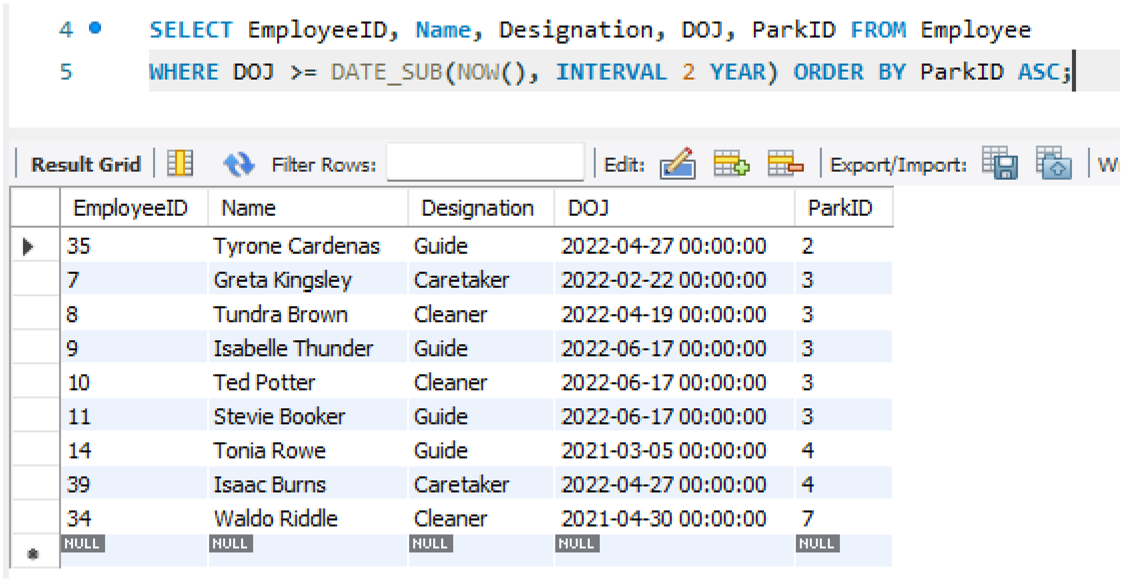
KIDS (TicketID, Number Of Kids) TicketID NOT NULL

IV. Implementation of Relation Model via MySQL and NoSQL

<u>MySQL implementation:</u>

The database was created in MySQL and the following queries were performed:

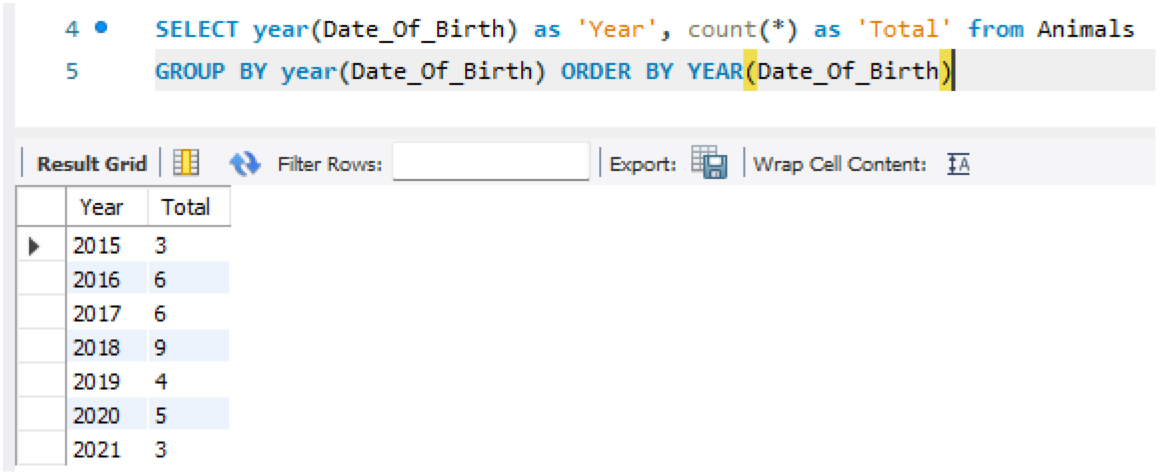
1. **Get the list of Employees who have joined within the last two year from current date:**



```
SELECT EmployeeID, Name, Designation, DOJ, ParkID FROM Employee
WHERE DOJ >= DATE_SUB(NOW(), INTERVAL 2 YEAR) ORDER BY ParkID ASC;
```

EmployeeID	Name	Designation	DOJ	ParkID
35	Tyrone Cardenas	Guide	2022-04-27 00:00:00	2
7	Greta Kingsley	Caretaker	2022-02-22 00:00:00	3
8	Tundra Brown	Cleaner	2022-04-19 00:00:00	3
9	Isabelle Thunder	Guide	2022-06-17 00:00:00	3
10	Ted Potter	Cleaner	2022-06-17 00:00:00	3
11	Stevie Booker	Guide	2022-06-17 00:00:00	3
14	Tonia Rowe	Guide	2021-03-05 00:00:00	4
39	Isaac Burns	Caretaker	2022-04-27 00:00:00	4
34	Waldo Riddle	Cleaner	2021-04-30 00:00:00	7
NULL	NULL	NULL	NULL	NULL

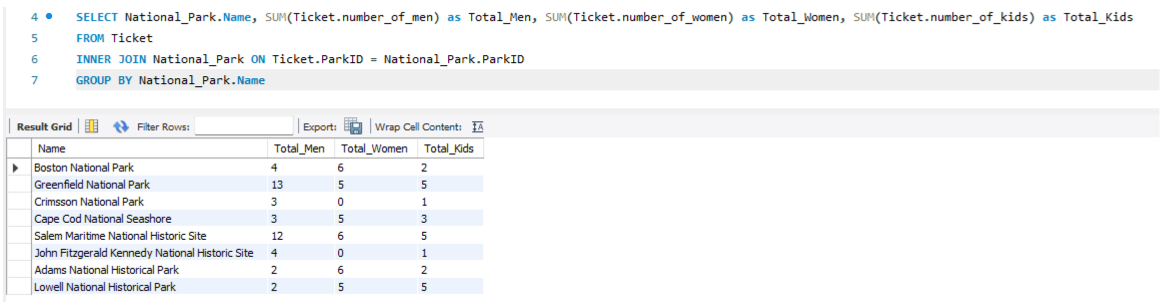
2. **Show the total number of Animal births per year:**



```
SELECT year(Date_Of_Birth) as 'Year', count(*) as 'Total' from Animals
GROUP BY year(Date_Of_Birth) ORDER BY YEAR(Date_Of_Birth)
```

Year	Total
2015	3
2016	6
2017	6
2018	9
2019	4
2020	5
2021	3

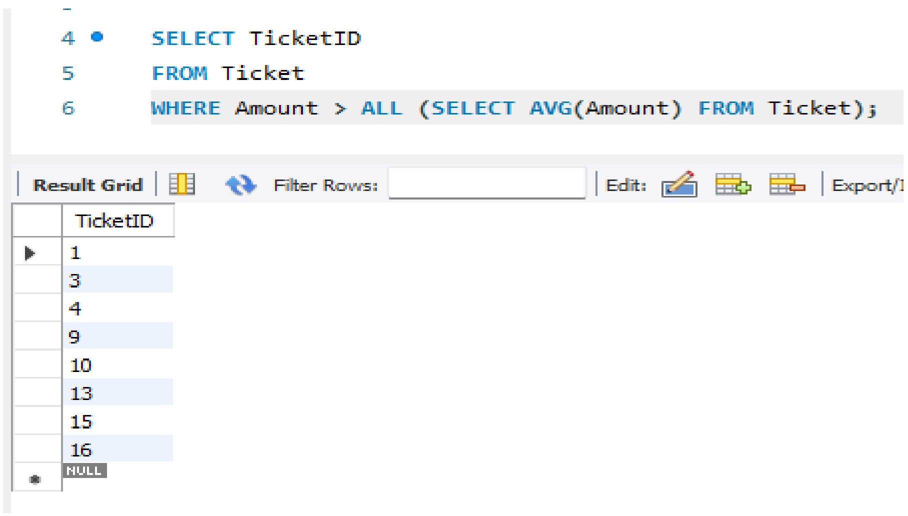
3. **Calculate the total number of men, women and kids who accompanied the main visitor for each park:**



```sql
SELECT National_Park.Name, SUM(Ticket.number_of_men) as Total_Men, SUM(Ticket.number_of_women) as Total_Women, SUM(Ticket.number_of_kids) as Total_Kids
FROM Ticket
INNER JOIN National_Park ON Ticket.ParkID = National_Park.ParkID
GROUP BY National_Park.Name
```

Name	Total_Men	Total_Women	Total_Kids
Boston National Park	4	6	2
Greenfield National Park	13	5	5
Crimsson National Park	3	0	1
Cape Cod National Seashore	3	5	3
Salem Maritime National Historic Site	12	6	5
John Fitzgerald Kennedy National Historic Site	4	0	1
Adams National Historical Park	2	6	2
Lowell National Historical Park	2	5	5

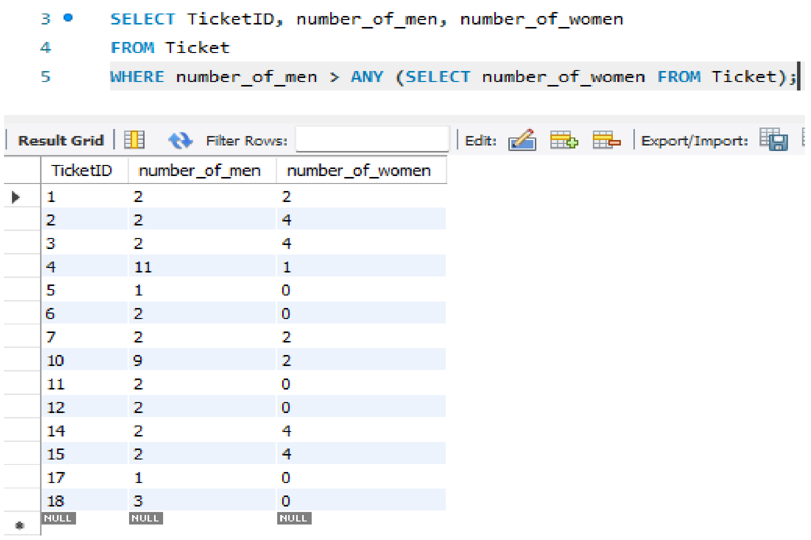
4. **Find all the Ticket IDs where the amount paid is greater than the average amount paid by all the visitors:**



```sql
SELECT TicketID
FROM Ticket
WHERE Amount > ALL (SELECT AVG(Amount) FROM Ticket);
```

TicketID
1
3
4
9
10
13
15
16
NULL

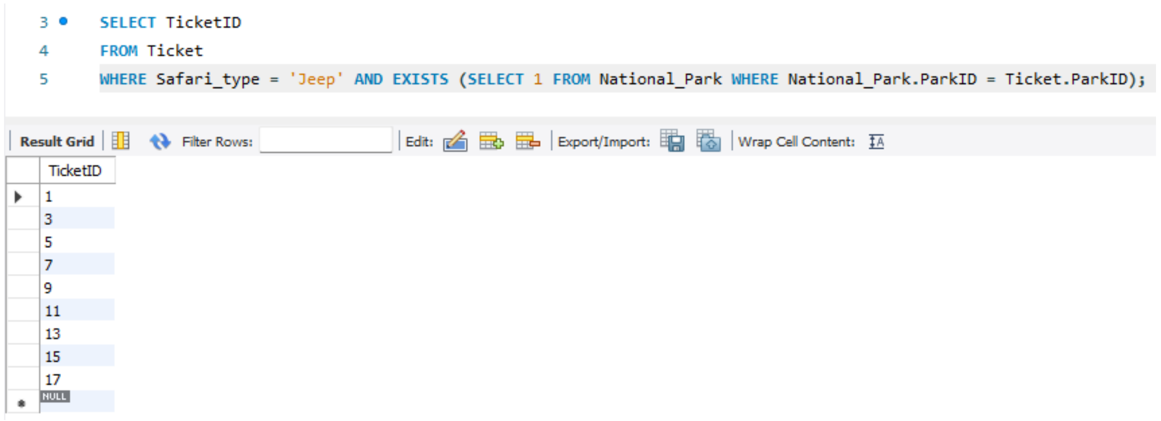
5. **Show all the Ticket IDs, number of men and number of women where the number of men who accompanied the main visitor is greater than number of women who accompanied any visitor:**



```sql
SELECT TicketID, number_of_men, number_of_women
FROM Ticket
WHERE number_of_men > ANY (SELECT number_of_women FROM Ticket);
```

TicketID	number_of_men	number_of_women
1	2	2
2	2	4
3	2	4
4	11	1
5	1	0
6	2	0
7	2	2
10	9	2
11	2	0
12	2	0
14	2	4
15	2	4
17	1	0
18	3	0
NULL	NULL	NULL

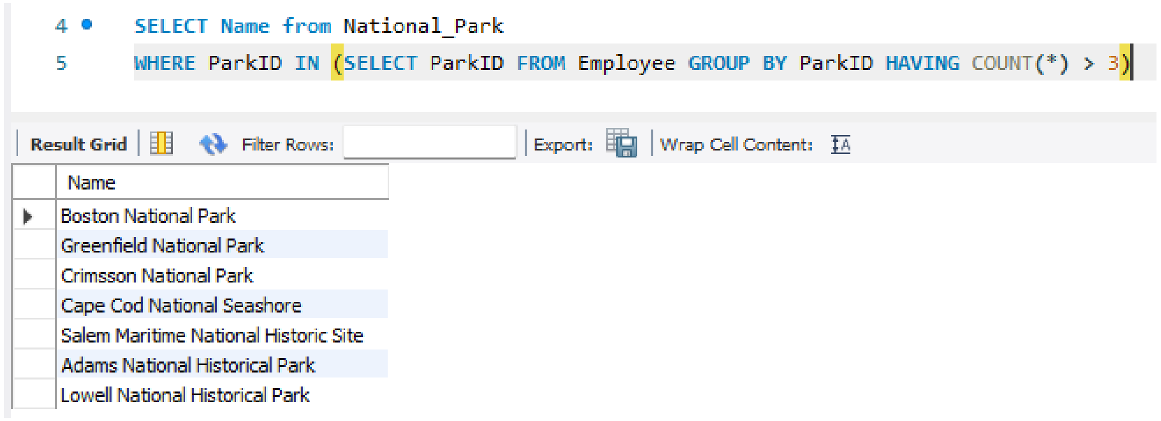
6. **Find all ticket IDs where the Safari Type is 'Jeep' and the National Park with the specified ParkID exists:**



```
SELECT TicketID
FROM Ticket
WHERE Safari_type = 'Jeep' AND EXISTS (SELECT 1 FROM National_Park WHERE National_Park.ParkID = Ticket.ParkID);
```

TicketID
1
3
5
7
9
11
13
15
17
NULL

7. **Show the list of National Parks having more than 3 employees:**



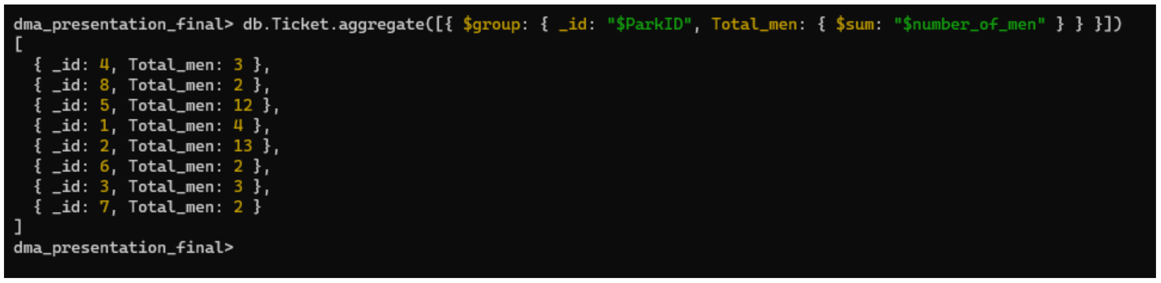
```
SELECT Name from National_Park
WHERE ParkID IN (SELECT ParkID FROM Employee GROUP BY ParkID HAVING COUNT(*) > 3)
```

Name
Boston National Park
Greenfield National Park
Crimsson National Park
Cape Cod National Seashore
Salem Maritime National Historic Site
Adams National Historical Park
Lowell National Historical Park

<u>NoSQL Implementation:</u>

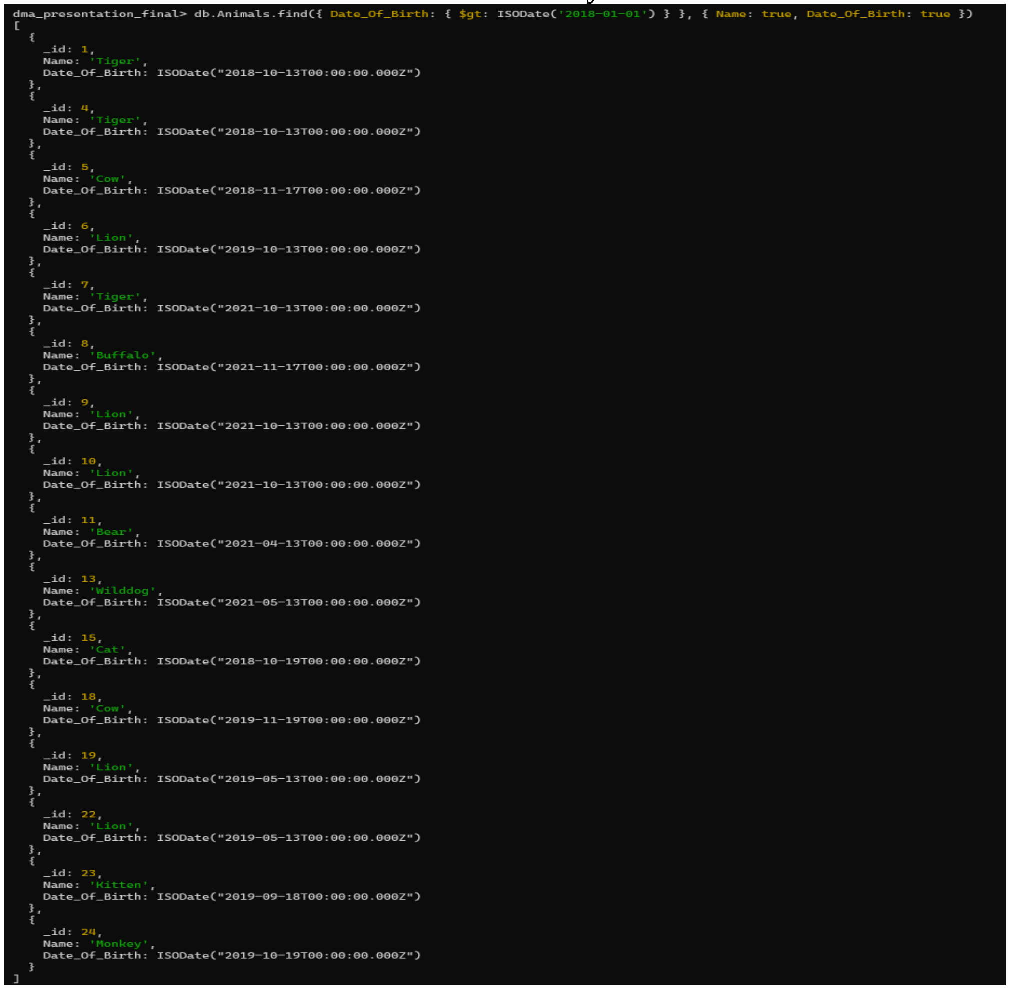
We used MongoDB for the NoSQL Implementation of the database. The following queries were performed:

1. **Total number of men visiting the National Parks:**



```
dma_presentation_final> db.Ticket.aggregate([{ $group: { _id: "$ParkID", Total_men: { $sum: "$number_of_men" } } }])
[
  { _id: 4, Total_men: 3 },
  { _id: 8, Total_men: 2 },
  { _id: 5, Total_men: 12 },
  { _id: 1, Total_men: 4 },
  { _id: 2, Total_men: 13 },
  { _id: 6, Total_men: 2 },
  { _id: 3, Total_men: 3 },
  { _id: 7, Total_men: 2 }
]
dma_presentation_final>
```

2. **Show all the animals born after 1st January 2018:**



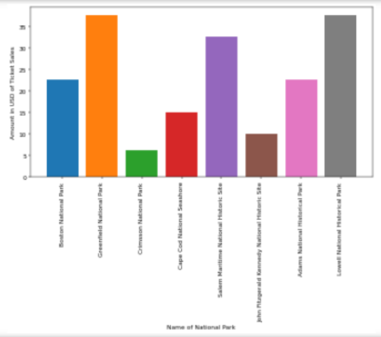
```
dma_presentation_final> db.Animals.find({ Date_Of_Birth: { $gt: ISODate('2018-01-01') } }, { Name: true, Date_Of_Birth: true })
[
  {
    _id: 1,
    Name: 'Tiger',
    Date_Of_Birth: ISODate("2018-10-13T00:00:00.000Z")
  },
  {
    _id: 4,
    Name: 'Tiger',
    Date_Of_Birth: ISODate("2018-10-13T00:00:00.000Z")
  },
  {
    _id: 5,
    Name: 'Cow',
    Date_Of_Birth: ISODate("2018-11-17T00:00:00.000Z")
  },
  {
    _id: 6,
    Name: 'Lion',
    Date_Of_Birth: ISODate("2019-10-13T00:00:00.000Z")
  },
  {
    _id: 7,
    Name: 'Tiger',
    Date_Of_Birth: ISODate("2021-10-13T00:00:00.000Z")
  },
  {
    _id: 8,
    Name: 'Buffalo',
    Date_Of_Birth: ISODate("2021-11-17T00:00:00.000Z")
  },
  {
    _id: 9,
    Name: 'Lion',
    Date_Of_Birth: ISODate("2021-10-13T00:00:00.000Z")
  },
  {
    _id: 10,
    Name: 'Lion',
    Date_Of_Birth: ISODate("2021-10-13T00:00:00.000Z")
  },
  {
    _id: 11,
    Name: 'Bear',
    Date_Of_Birth: ISODate("2021-04-13T00:00:00.000Z")
  },
  {
    _id: 13,
    Name: 'Wilddog',
    Date_Of_Birth: ISODate("2021-05-13T00:00:00.000Z")
  },
  {
    _id: 15,
    Name: 'Cat',
    Date_Of_Birth: ISODate("2018-10-19T00:00:00.000Z")
  },
  {
    _id: 18,
    Name: 'Cow',
    Date_Of_Birth: ISODate("2019-11-19T00:00:00.000Z")
  },
  {
    _id: 19,
    Name: 'Lion',
    Date_Of_Birth: ISODate("2019-05-13T00:00:00.000Z")
  },
  {
    _id: 22,
    Name: 'Lion',
    Date_Of_Birth: ISODate("2019-05-13T00:00:00.000Z")
  },
  {
    _id: 23,
    Name: 'Kitten',
    Date_Of_Birth: ISODate("2019-09-18T00:00:00.000Z")
  },
  {
    _id: 24,
    Name: 'Monkey',
    Date_Of_Birth: ISODate("2019-10-19T00:00:00.000Z")
  }
]
```

V. Database Access via R or Python

We connect to our MySQL Database using Python with the help of **mysql.connector** and **cursor.execute()** to execute the query and finally use **cursor.fetchall()** to fetch all the rows in the result set of the query. Below are visualizations of few queries that we performed to analyze the data:

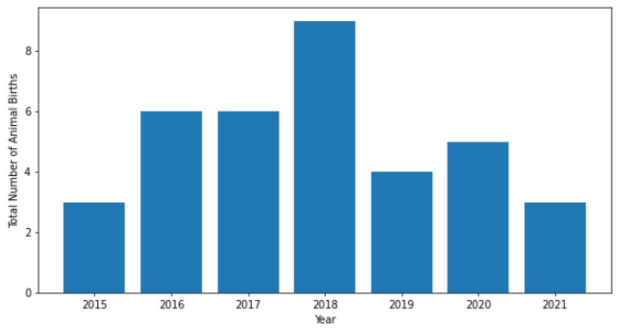
Plot-1: Graph for Amount of Ticket Sales for each park:

```
In [15]: sql_select_Query = "SELECT sum(t.amount), np.Name name FROM national_park np inner join ticket t on np.ParkID = t.ParkId group by
         cursor = connection.cursor()
         cursor.execute(sql_select_Query)
         records = cursor.fetchall()
         plt.figure(figsize=(10,5), dpi= 43, facecolor='w', edgecolor='k')
         for row in records:
             amount, Name = row
             plt.bar(Name, amount)
         plt.xlabel("Name of National Park")
         plt.ylabel("Amount in USD of Ticket Sales")
         plt.xticks(rotation='vertical')
```

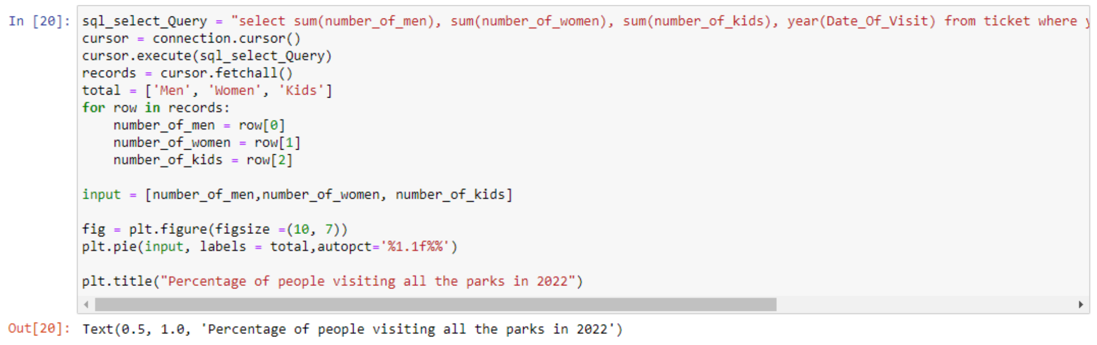


Plot-2: Graph for number of Animal Births per year in all the parks:

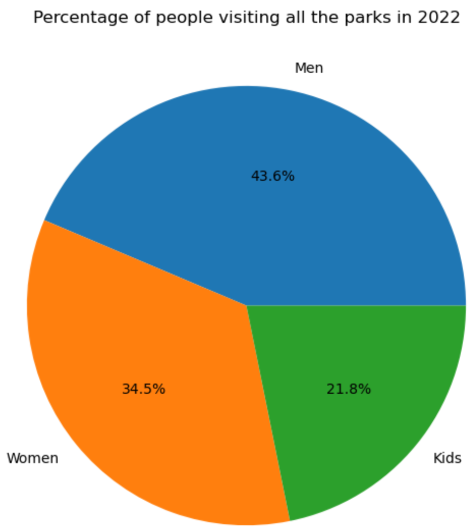
```
In [26]: sql_select_Query = "Select year(Date_Of_Birth), count(*) from Animals group by year(Date_Of_Birth) order by year(Date_Of_Birth)"
         cursor = connection.cursor()
         cursor.execute(sql_select_Query)
         records = cursor.fetchall()
         years = []
         count = []
         for row in records:
             years.append(row[0])
             count.append(row[1])

         plt.figure(figsize=(10,5), dpi= 70, facecolor='w', edgecolor='k')

         plt.bar(years, count)
         plt.xlabel("Year")
         plt.ylabel("Total Number of Animal Births")

Out[26]: Text(0, 0.5, 'Total Number of Animal Births')
```



Plot-3: Plotting number of men, women and children visiting parks in 2022:



```
In [20]: sql_select_Query = "select sum(number_of_men), sum(number_of_women), sum(number_of_kids), year(Date_Of_Visit) from ticket where y
         cursor = connection.cursor()
         cursor.execute(sql_select_Query)
         records = cursor.fetchall()
         total = ['Men', 'Women', 'Kids']
         for row in records:
             number_of_men = row[0]
             number_of_women = row[1]
             number_of_kids = row[2]

         input = [number_of_men,number_of_women, number_of_kids]

         fig = plt.figure(figsize =(10, 7))
         plt.pie(input, labels = total,autopct='%1.1f%%')

         plt.title("Percentage of people visiting all the parks in 2022")

Out[20]: Text(0.5, 1.0, 'Percentage of people visiting all the parks in 2022')
```



VII. Summary and Recommendation

The database that we described above, with the help of MySQL and Python will help the government make informed decisions about the infrastructure required to save the animals in the coming future. It will help them understand tourists interests as well and help in achieving the main goal of safeguarding wildlife. The analytics provided will also demonstrate the performance of each national park.

Improvement in this project would be to create a machine learning model to predict life span of the animals thereby taking necessary measures. It can also be extended to predict revenue that would be generated by the ticketing system. We can also extend our analysis by creating a time series analysis to understand the birth of each animal.
The shortcoming of this database design is that as the number of tables and amount of input data increases, the database querying will become slow. We plan on working on improving upon this shortcoming by introducing indexing to make the database more robust.