Single table SELECT Activity

For this activity, you will use the Zoo Database. If you haven't already, run the script to load the data into the database. Then, use the database to answer the questions below. The ERD diagram is at the bottom of the document.

Basic Select Statements

1. Let's see what tables are in the Database. Type in the following query to view all the tables. Which table do you think would list information about each animal?

   ```
   SHOW TABLES;
   ```

   The Animals table would show information about each animal.

2. Try out a basic **SELECT** query by keying in the following statement. What are the attributes (columns) of the table? How many rows are returned?

   ```
   SELECT * FROM Zookeepers;
   ```

   Columns are: ZookeeperID, FirstName, LastName, Email, AssignedHabitatID

   There were 8 rows returned.

3. Let's limit the number of columns returned by changing the * to the FirstName and LastName by entering the SQL statement below. Does it change the number of columns? Does it change the number of rows returned?

   ```
   SELECT FirstName, LastName FROM Zookeepers;
   ```

   It does not change the number of rows returned, still 8.

4. Modify the code so it only displays email. What does the SQL statement now look like?

   SELECT Email FROM Zookeepers;

5. Use the DISTINCT keyword in the SELECT field such as in the query below. What do you notice about the results?

   ```
   SELECT DISTINCT VetID FROM MedicalRecords;
   ```

   There are only two rows returned, and the IDs are 1 and 2. Each row is unique.

6. Try out the following query? What do you notice about how the following query orders the table?

   ```
   SELECT * FROM Animals ORDER BY Name;
   ```

   The results are in alphabetical order by the Name in the Animals table.

7. Add the keyword **DESC** after the query to get. What does that do to your results?

```
        SELECT * FROM Animals ORDER BY Name DESC;
```

Now the results are in reverse alphabetical order.

8. Modify the statement to order by the DateOfBirth. What does the select statement look like?

SELECT * FROM Animals ORDER BY DateOfBirth;

9. You can order by two values by placing a comma in between. What do you notice about the way the following query results are displayed?

```
        SELECT DateOfBirth, Name
        FROM Animals
        ORDER BY DateOfBirth, Name;
```

It sorts first by DateOfBirth, and for any animals that share a DateOfBirth, the results are then ordered by the Name.

10. Modify the statement to display the information by Species, then the Name of the animal in descending order. Copy your SQL statement here.

SELECT * FROM Animals ORDER BY Species, Name DESC;

11. Let's try another table. List all the columns of the Medical Records table in a SELECT statement. (You may use the SHOW TABLES command or just look at the diagram at the end of the page to find the exact spelling of the table) What would the SQL statement look like?

SELECT * FROM MedicalRecords;

12. Let's go a little more advanced. List the animalID and diagnosis of all the medical records in alphabetical order by diagnosis.

SELECT AnimalID, Diagnosis FROM MedicalRecords ORDER BY Diagnosis;

Where Clause

13. We will use the **WHERE** clause to limit the number of rows returned. Find out how many medical records have the animalID of 5. Run the following SQL query. How many rows are returned?
    ```
    SELECT * FROM MedicalRecords WHERE animalID = 5;
    ```

    2 rows returned.

14. An issue arises when the search item is text. Try out the following query, what error message do you receive?
    ```
    SELECT * FROM MedicalRecords WHERE Diagnosis = General check-up;
    ```

    Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Check-up' at line 1

15. Fix the problem by placing single quotes around '`General check-up`'. How many rows are returned?

    1 row returned.

16. What happens if you spell the name differently? Modify the text so it is all lowercase by changing it to ‘`general check-up`'. How many rows are returned? (Casing will matter in different database management systems.)

    1 row returned.

17. Sometimes a field may not have a value. It is said to be `NULL` or No Value. Find out how many animals do not have a habitat assigned yet by trying out <u>each</u> of these commands.
    ```
    SELECT * FROM Animals WHERE HabitatID = NULL;
    SELECT * FROM Animals WHERE HabitatID IS NULL;
    ```

    0 rows returned from first statement. 1 row returned from the second.

18. The **IS NULL** and **IS NOT NULL** keywords find when something is NULL or not NULL. The following lists all the animals that do not have a habitat assigned yet. Modify the query to show all the animals that do have a habitat assigned.
    ```
    SELECT * FROM Animals WHERE HabitatID IS NULL;
    ```

    SELECT * FROM Animals WHERE HabitatID IS NOT NULL;

19. The keywords **AND** and **OR** combine expressions in the WHERE clause. Run the two queries, what is the difference between the results? What is the difference between AND and OR?
    ```
    SELECT * FROM Animals WHERE healthStatus = 'healthy' AND habitatID = 3;
    SELECT * FROM Animals WHERE healthStatus = 'healthy' OR habitatID = 3;
    ```

The first statement only returned two rows where the animal was in Habitat 3 and it was healthy. The second statement returned 15 rows, any animal that was either in Habitat 3 or healthy was found.

20. The equal sign is used to find an exact match. You can also use > for greater than, < less than, >= greater than equal, <= less than equal, and != or not equal to. Write a query that shows all the Habitats with a square footage of below 4000.

    SELECT * FROM Habitats WHERE SizeSqFt < 4000;

21. There are other keywords when you do not have an exact match, BETWEEN and IN. **BETWEEN** finds values within a range, **IN** tests multiple conditions. Run the following SQL statements and determine the difference between the outputs.

    ```
    SELECT * FROM Habitats WHERE SizeSqFt BETWEEN 3000 AND 5000;
    SELECT * FROM Habitats WHERE SizeSqFt IN (3000, 5000);
    ```

    The first query returned 3 rows with SizeSqFt { 3000, 4000, 5000 }.

    The second query returned only 2 rows with SizeSqFt { 3000, 5000 }.

22. The **LIKE** keyword can be used when looking for a pattern instead of an exact item. The percent sign % means zero, one, or any amount of characters. The underscore _ means a single character. So if you wanted to find a person in the Student table that started with the letter A you could use the following query. Modify the query to return all the courses with a description that ends with the word Programming.

    ```
    SELECT * FROM Animals WHERE species LIKE 'T%';
    ```

    SELECT * FROM Courses WHERE Description LIKE '%Programming';

Simple Functions

23. We will use the **concat** function to combine two text fields into 1. How many columns are returned with the following query? What is the column name?

    ```
        SELECT concat(firstName, lastName) FROM zookeepers;
    ```

    8 rows returned. The column name is concat(firstName, lastName)

24. Modify the query to add a space. We will need to use quotation marks to specify that we are also concatenating a space? Is there now a space between?

    ```
        SELECT concat(firstName, ' ', lastName) FROM zookeepers;
    ```

    Yes there is now a space between.

25. Observe the name of the column, it is just the function call. Run the query below with the as Full_Name added. What does the as clause do to the column name?

```
SELECT concat(firstName, ' ', lastName) as Full_Name FROM zookeepers;
```

Instead of the column name being a 'concat' statement, the column name is FULL_Name.

26. Observe the use of the ucase, and lcase. How does each of the function affect the text?

```
SELECT UCASE(firstName), LCASE(firstName) FROM zookeepers;
```

UCASE makes the text all uppercase and LCASE makes the text all lowercase.

27. Notice the AM in the feedingSchedule column.

```
SELECT * FROM Diets;
```

What happens to the AM when you run this query? Why do you think that happens?

```
SELECT replace(feedingSchedule, 'AM', 'Morning') FROM Diets;
```

It replaces AM with Morning.

We can also nest the replace function. Observe what is happening here:

```
SELECT Replace(replace(feedingSchedule, 'AM', 'Morning'), 'PM', 'Evening') FROM Diets;
```

28. Now it is your turn to try out the REPLACE function on any table and column you would like and show your SELECT statement.
SELECT REPLACE(DietType, 'ivore', 'ologist') FROM Diets;
This resulted in:
'Herbologist'
'Carnologist'
'Omnologist'

Date Functions

29. Observe the following query. What does the MONTH function do to the start_date_time field?

```
SELECT DateOfBirth, MONTH(DateOfBirth) FROM animals;
```

It grabbed only the month number from the date that was returned.

30. What would you change it to to display the day?

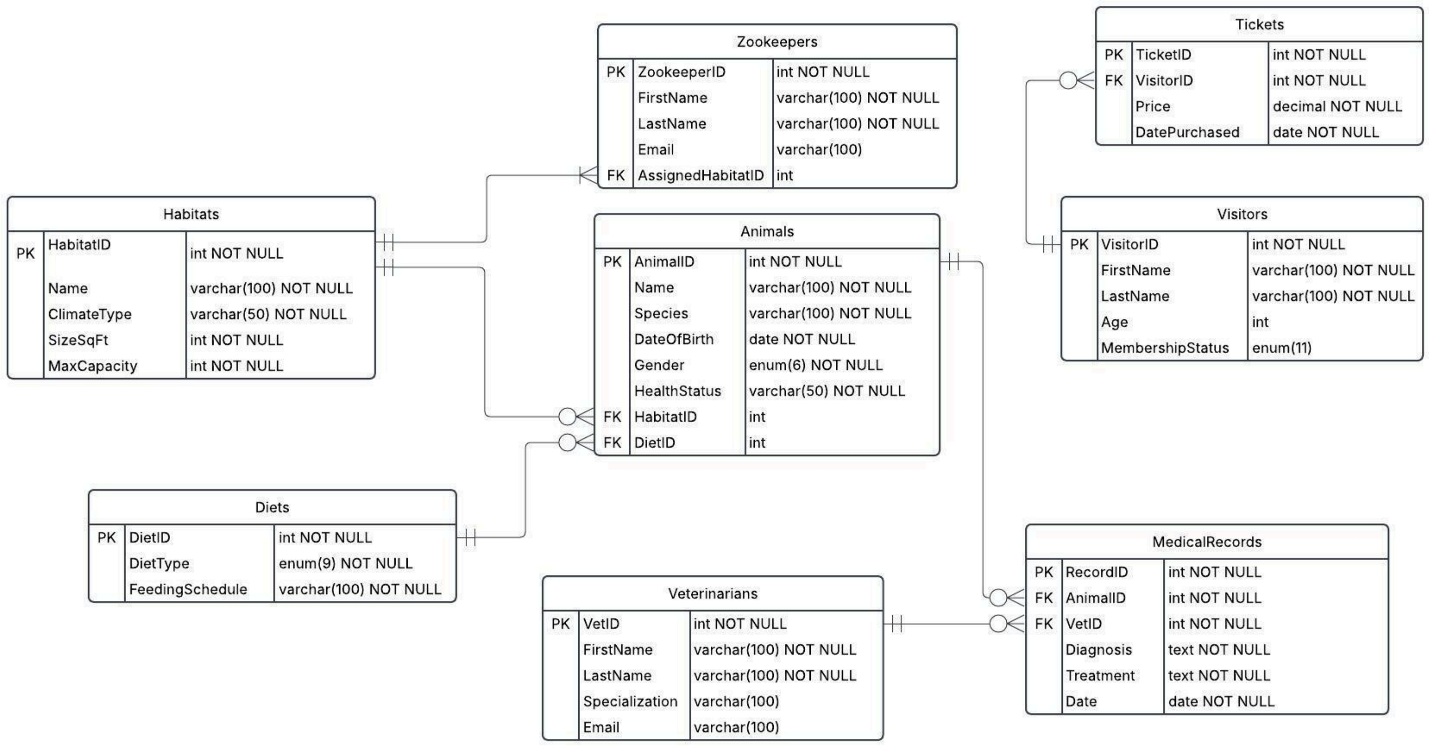
SELECT DateOfBirth, DAY(DateOfBirth) FROM Animals;

31. Modify the DateOfBirth and use the **concat** function to display in the format month-day. Order the values by the DateOfBirth.

SELECT CONCAT(MONTH(DateOfBirth), DAY(DateOfBirth)) FROM Animals ORDERBY DateOfBirth;

Zoo Database

The following is the ERD of the database used for this assignment.



Zookeepers		
PK	ZookeeperID	int NOT NULL
	FirstName	varchar(100) NOT NULL
	LastName	varchar(100) NOT NULL
	Email	varchar(100)
FK	AssignedHabitatID	int

Tickets		
PK	TicketID	int NOT NULL
FK	VisitorID	int NOT NULL
	Price	decimal NOT NULL
	DatePurchased	date NOT NULL

Habitats		
PK	HabitatID	int NOT NULL
	Name	varchar(100) NOT NULL
	ClimateType	varchar(50) NOT NULL
	SizeSqFt	int NOT NULL
	MaxCapacity	int NOT NULL

Animals		
PK	AnimalID	int NOT NULL
	Name	varchar(100) NOT NULL
	Species	varchar(100) NOT NULL
	DateOfBirth	date NOT NULL
	Gender	enum(6) NOT NULL
	HealthStatus	varchar(50) NOT NULL
FK	HabitatID	int
FK	DietID	int

Visitors		
PK	VisitorID	int NOT NULL
	FirstName	varchar(100) NOT NULL
	LastName	varchar(100) NOT NULL
	Age	int
	MembershipStatus	enum(11)

Diets		
PK	DietID	int NOT NULL
	DietType	enum(9) NOT NULL
	FeedingSchedule	varchar(100) NOT NULL

MedicalRecords		
PK	RecordID	int NOT NULL
FK	AnimalID	int NOT NULL
FK	VetID	int NOT NULL
	Diagnosis	text NOT NULL
	Treatment	text NOT NULL
	Date	date NOT NULL

Veterinarians		
PK	VetID	int NOT NULL
	FirstName	varchar(100) NOT NULL
	LastName	varchar(100) NOT NULL
	Specialization	varchar(100)
	Email	varchar(100)