NSS Joint Project:

Animal Management System for Dublin Zoo

Systems Analysis

Relational Database

Windows Programming

Object Orientated Programming

Student Name: Martin Fennell

Student Exam #:7959885G

Contents

|  |  |
| --- | --- |
| [Introduction](#_Introduction) | [Page 3](#_Introduction) |
| [Initial Phase](#_Initial_Phase) | [Page 4](#_Initial_Phase) |
| * [Appraisal of Existing System](#_Appraisal_of_Existing) | [Page 4](#_Appraisal_of_Existing) |
| * [Problems With Current System](#_Problems_with_Current) | [Page 4](#_Problems_with_Current) |
| * [Requirements For New System](#_Requirements_for_New) | [Page 5](#_Requirements_for_New) |
| [Project Development Phase](#_Project_Development_Phase) | [Page 6](#_Project_Development_Phase) |
| * [Project Development Plan](#_Project_Development_Plan) | [Page 6](#_Project_Development_Plan) |
| * [Gantt Chart](#_Gantt_Chart) | [Page 7](#_Gantt_Chart) |
| [System Requirements Phase](#_System_Requirements_Phase) | [Page 8](#_System_Requirements_Phase) |
| * [UML Use Case diagram](#_UML_Use_Case) | [Page 8](#_UML_Use_Case) |
| [Technical Solution Proposal](#_Technical_Solution_Proposal) | [Page 9](#_Technical_Solution_Proposal) |
| * [UML Class Diagram](#_UML_Class_Diagram) | [Page 9](#_UML_Class_Diagram) |
| * [UML Activity Diagram](#_UML_Activity_Diagram) | [Page 10](#_UML_Activity_Diagram) |
| [Design Document](#_Design_Document) | [Page 11](#_Design_Document) |
| * [Problem Partition](#_Problem_Partition) | [Page 11](#_Problem_Partition) |
| * [Design of control logic](#_Design_of_control) | [Page 12](#_Design_of_control) |
| * [User Interface](#_User_Interface) | [Page 13](#_User_Interface) |
| * [Database Tables](#_Database_Tables) | [Page 19](#_Database_Tables) |
| * [Queries](#_Queries) | [Page 22](#_Queries) |
| * [ER Diagram](#_ER_Diagram) | [Page 24](#_ER_Diagram) |
| * [Data Dictionary](#_Data_Dictionary) | [Page 25](#_Data_Dictionary) |
| * [Create Table Script](#_Create_Table_Script) | [Page 28](#_Create_Table_Script) |
| [Test Cases](#_Test_Cases) | [Page 29](#_Test_Cases) |
| * [Test Case 1](#_Test_Case_1) | [Page 29](#_Test_Case_1) |
| * [Test Case 2](#_Test_Case_2) | [Page 33](#_Test_Case_2) |
| * [Test Case 3](#_Test_Case_3) | [Page 37](#_Test_Case_3) |
| * [Test Case 4](#_Test_Case_4) | [Page 39](#_Test_Case_4) |
| * [Test Case 5](#_Test_Case_5) | [Page 42](#_Test_Case_5) |
| [Appendix](#_Appendix) | [Page 44](#_Appendix) |
| * [Application Code](#_Application_Code) | [Page 44](#_Application_Code) |

Introduction

The aim of this project is to design a menu-driven application which will allow Zoo management to manage Zoo animals and staff members by using a MySQL database and a Java program that are connected together. The application should to make it possible to enter new animals in the system as they arrive or are born at zoo. It should also make it possible to search for animals by a minimum of any 3 criteria.

The system should be able provide various types of reports, eg, All animals in a particular area, all animals of a certain type, etc. and also be user friendly with the minimum amount of input required.

My application allows 2 different admin users to access the system which managers and vets. The Managers can search, insert, update and delete records from both the animal and staff tables. Vets can search and update records from the animal table.

# Initial Phase of the system

Appraisal of Existing System

In current system for the zoo the capacity of animals and information has increased over the amount that the zoo can hold and over the last number of years and the current system has become obsolete. Various problems have arisen in the recent years with regards to the recording and managing of animals due to the current system. Also producing reports have been a major problem due to the fact that the information needed for these is stored in a Number of different locations and in different formats.

Another problem has been the managing of existing data. There have been numerous HR issues relating to who has the responsibility for recording and managing the data because the data has been store in a number of personal computers of the managers. Management has also concerns over the security of information. At the moment in theory at least all staff can access all data.

At present there is no specific policy with regards to the recording of animals and management of details. The Current system is mainly manual with records being variously kept in paper/hardcopy format or in electronic form, or in a mixture of both. The location of these records is, usually, only known to the particular area manager.

Problems with Current System

There are many problems with the current system these can be seen below:

* Current is system no longer functional
* Reports cannot be produced
* Data stored in multiple locations and formats
* No security or restrictions on different kinds of data
* No policy regarding to the recording of animals and management of details
* Current system is mainly manual
* Location of records only known to the particular area manager

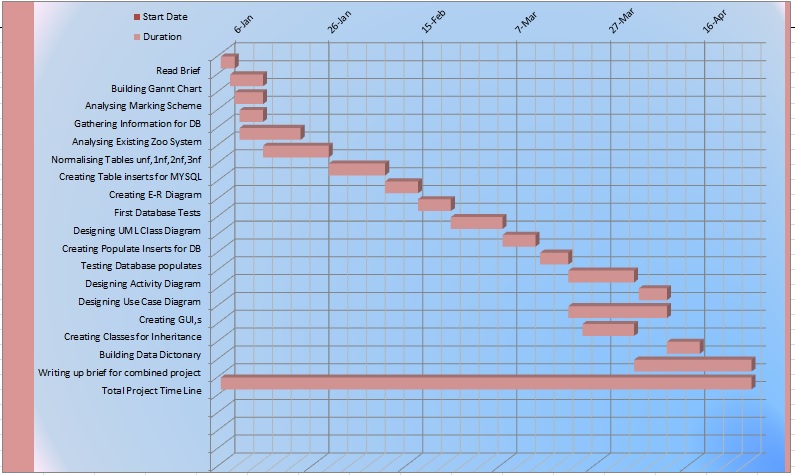
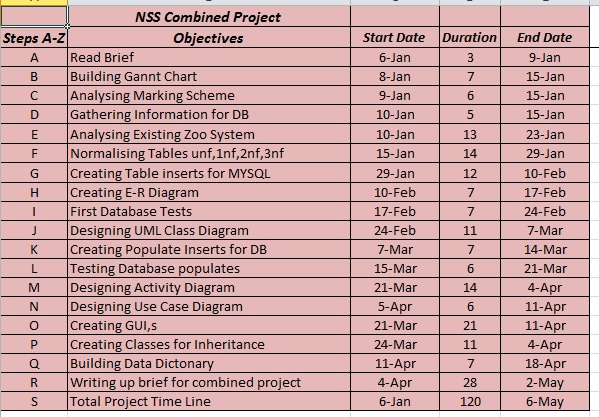
Requirements for New System

The new system requires a menu driven application which will allow zoo management to manage zoo animals and staff. It should be possible to enter new animals into the system as they arrive at or are born in the zoo, it should also be possible for the user to search, update and delete animal records from the system. It should be possible to generate various types of reports from the system.

The new system should also contain a record of all staff working in the zoo and all of their details and the system should be user friendly with the minimum amount of input required.

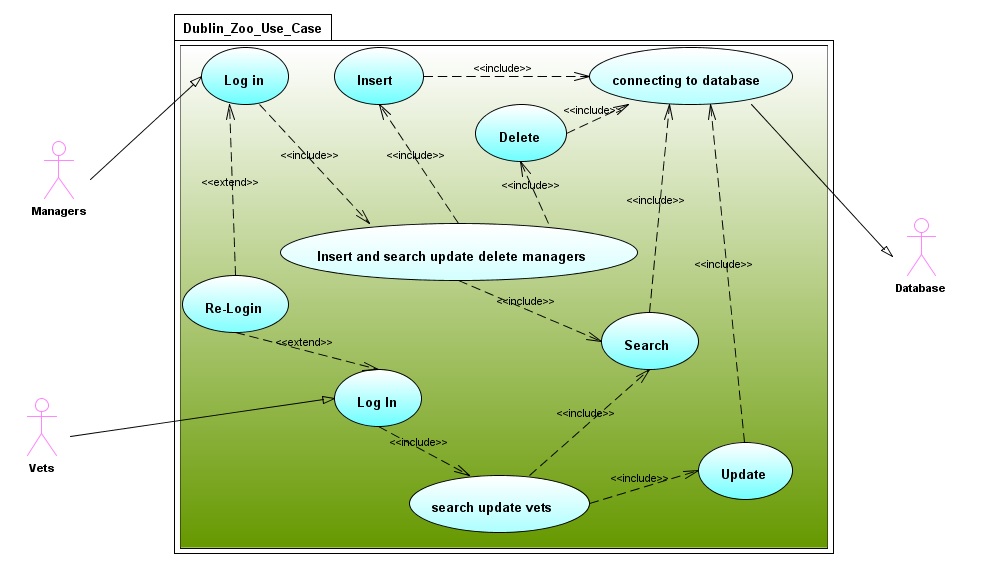
Project Development Phase

Project Development Plan

**Gantt Chart**

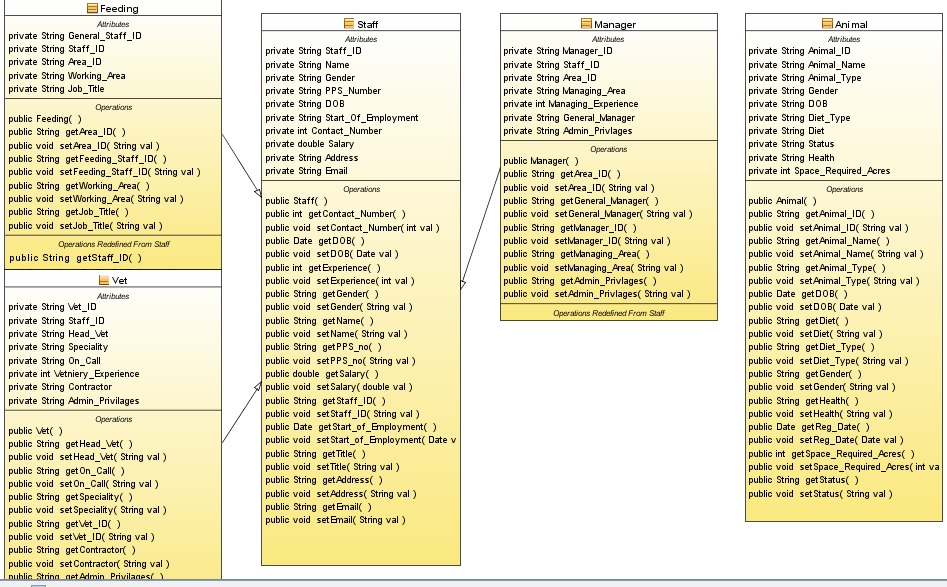
System Requirements Phase

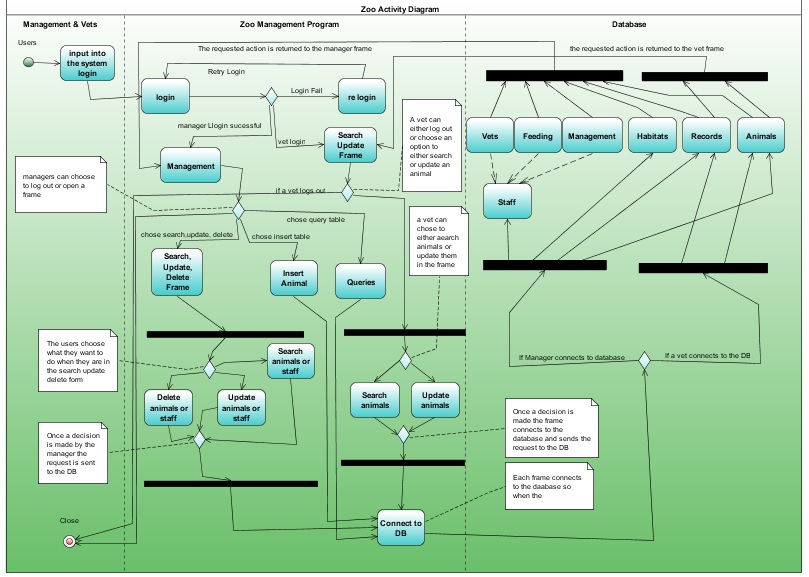
UML Use Case Diagram



Technical Solution Proposal

UML Class Diagram

****

UML Activity Diagram

Design Document

Problem Partition

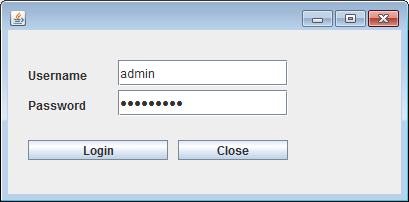
Design of control logic

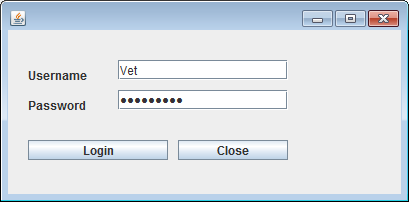
For this project I was asked to design and build a piece of software to manage data for Dublin zoo.

* First thing I did was spent some time analysing the requirements that the system would need from the brief.
* Then I designed a Gantt chart so I could plan out my time line of work.
* Then I took all the data I could get from the present zoo system and started normalising it from un-normalised form to third normal form.
* Then I created all the scripts with the data that I normalised for the database and designed it.
* Then I did my ER Diagram.
* Then I started to build my UML diagrams (Class, Use Case and Activity).
* Then I populated my database with information.
* Then I Designed and started building my GUI.
* Then I started coding the GUI.
* Finally I linked my GUI and database and began testing.
* I had some errors at first but I was able to successfully troubleshoot and debug them.

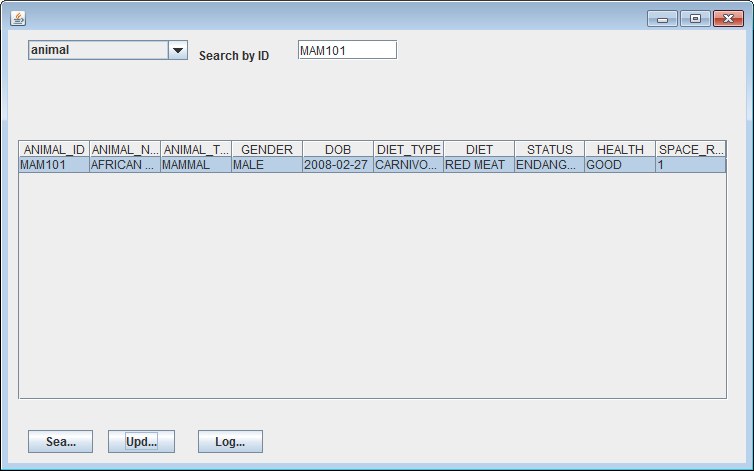
User Interface

The user interface for my project consists of many different frames; the first frame that opens when the program starts is the login frame from here either a vet or a manager can log into the system by using the usernames (Admin or Vet) and the password (dublinzoo).

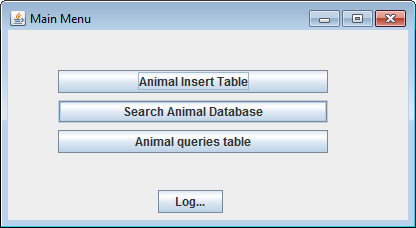




If the staff member is a Vet and the correct password is entered then the vet table is shown where the vet can search and update animals in the database.

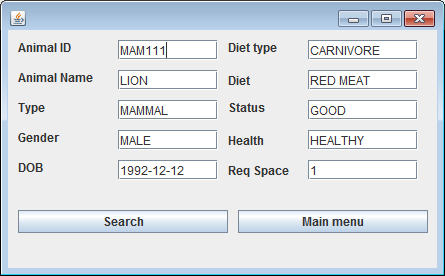


If the staff member is a manager and the correct password is entered then the manager screen is shown.

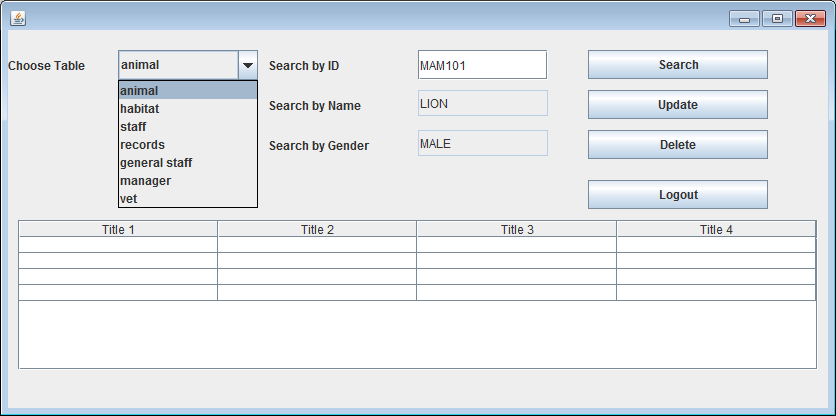


From the manager frame many different frames can be opened such as insert animals, search, update and delete animals and staff and the queries frame.

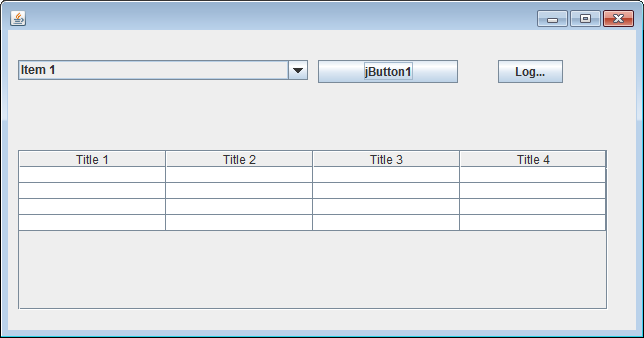
When animal insert table is chosen the frame related is opened.



When search animal database is chosen the frame related is opened where you can search animals and staff as well as update and delete animals.



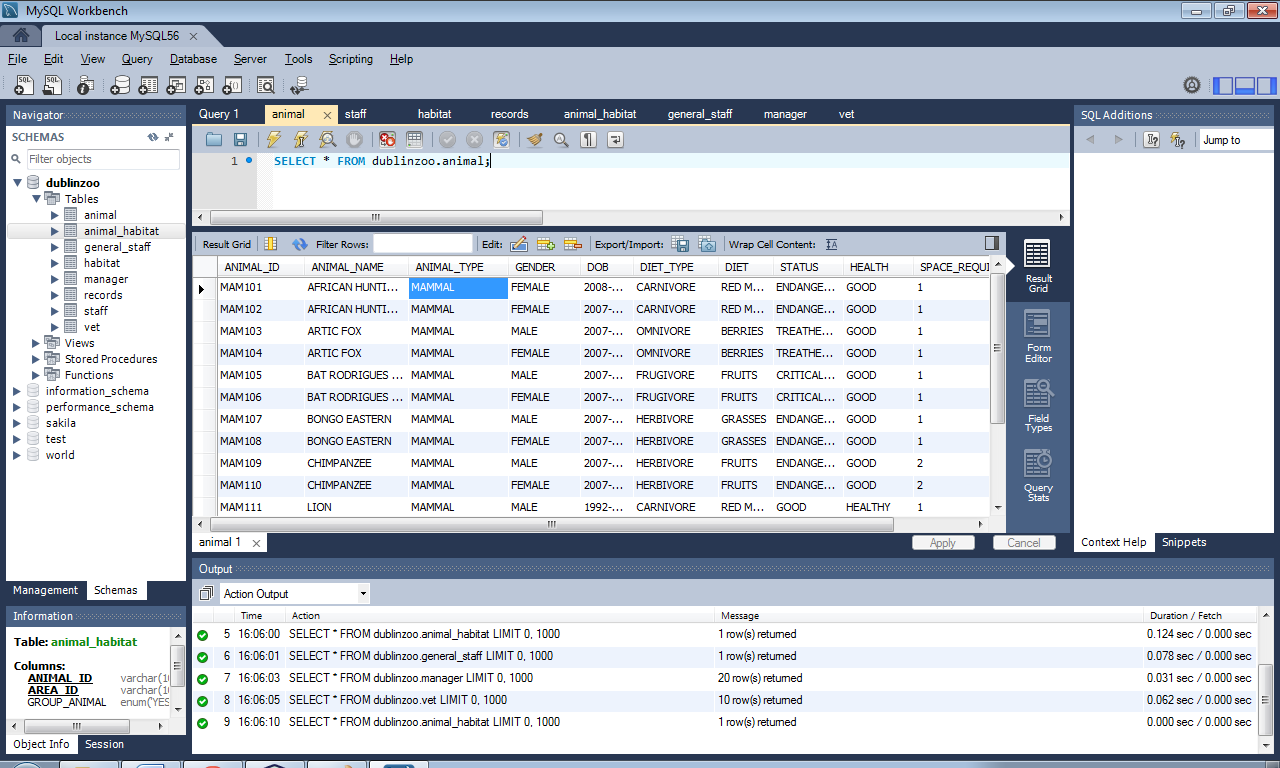
When Animal query is selected the related frame opens up.



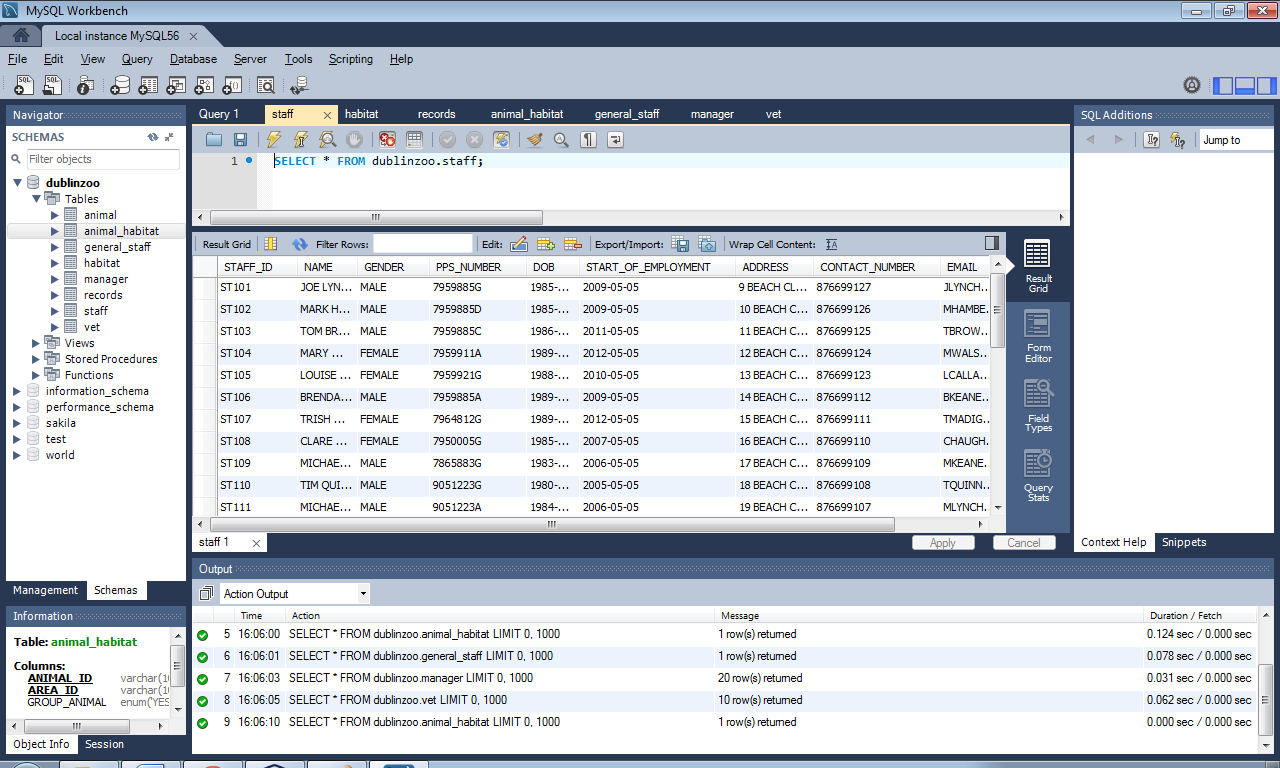
If the staff type is set to vet and the correct password is entered then the vet screen is shown.

Database Tables

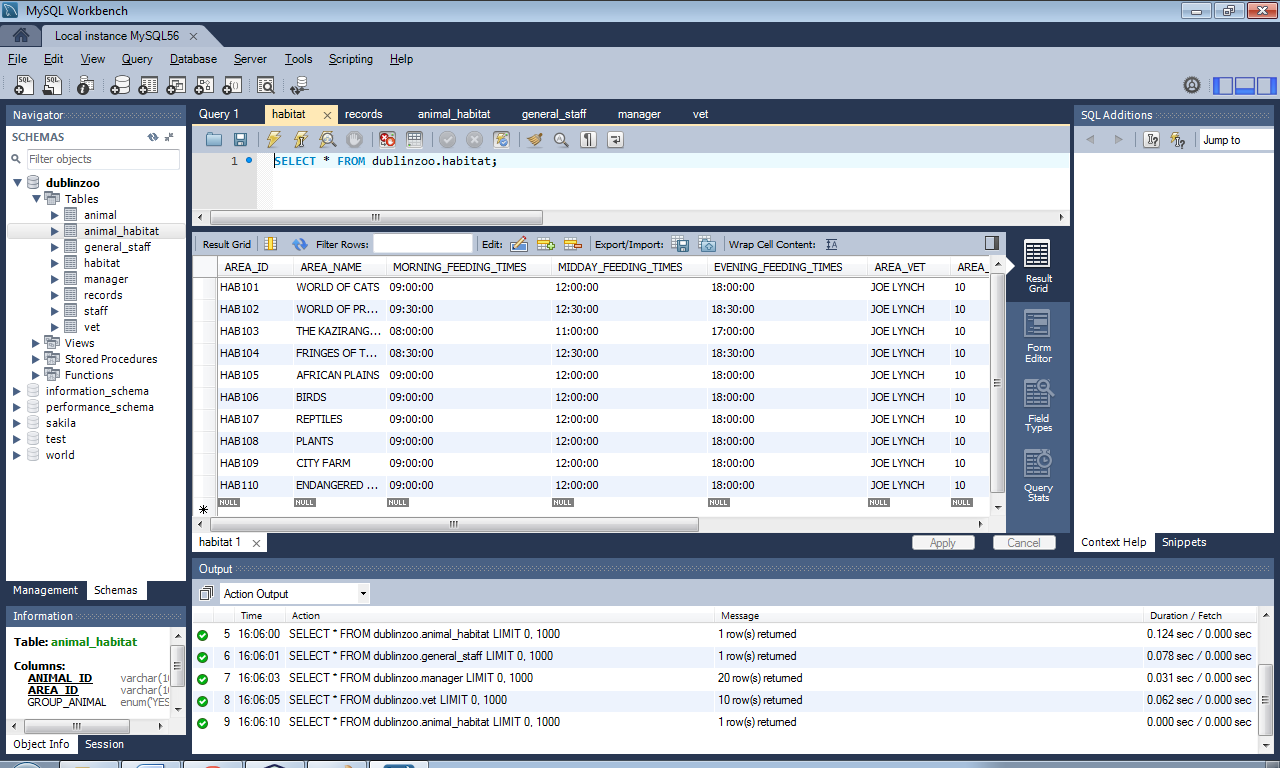
My database consists of 8 tables 4 main parent; animal, habitat, staff, records and it has 4 child tables which are; animal habitat, managers, vets and general staff.



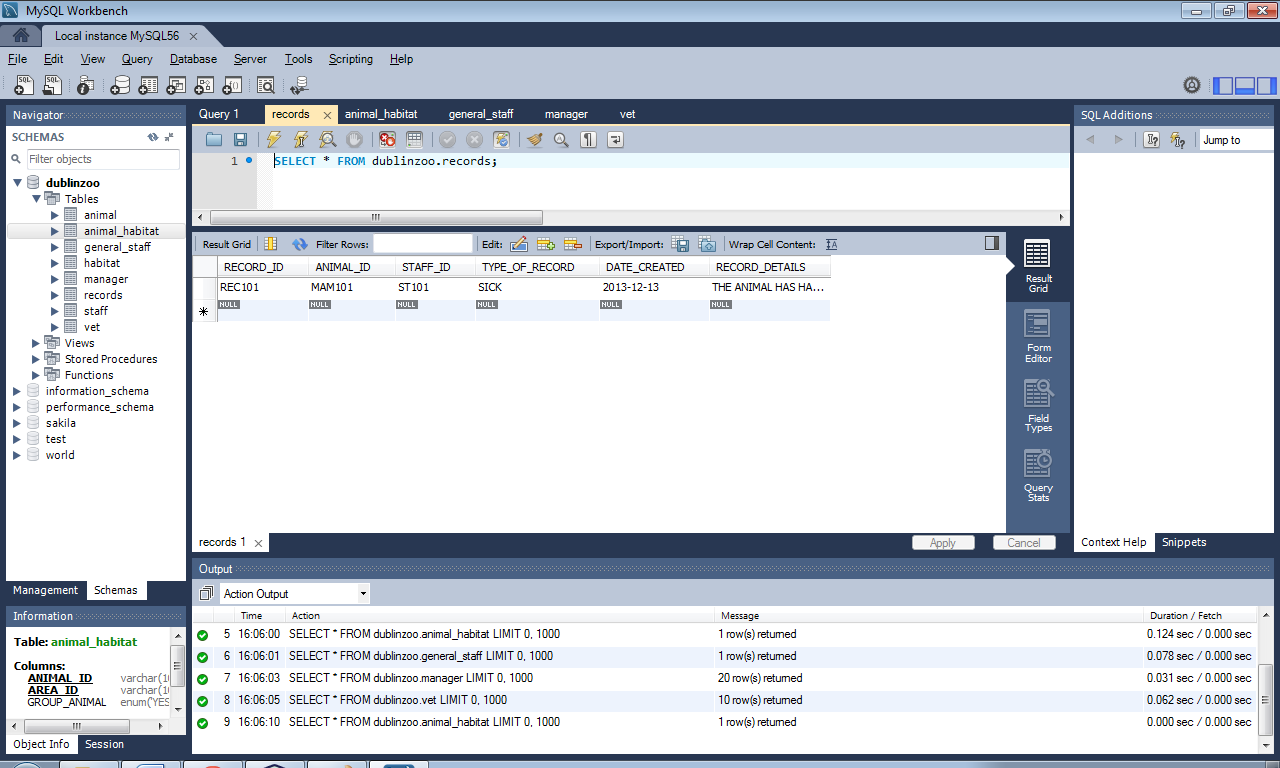
Above is the animal table.



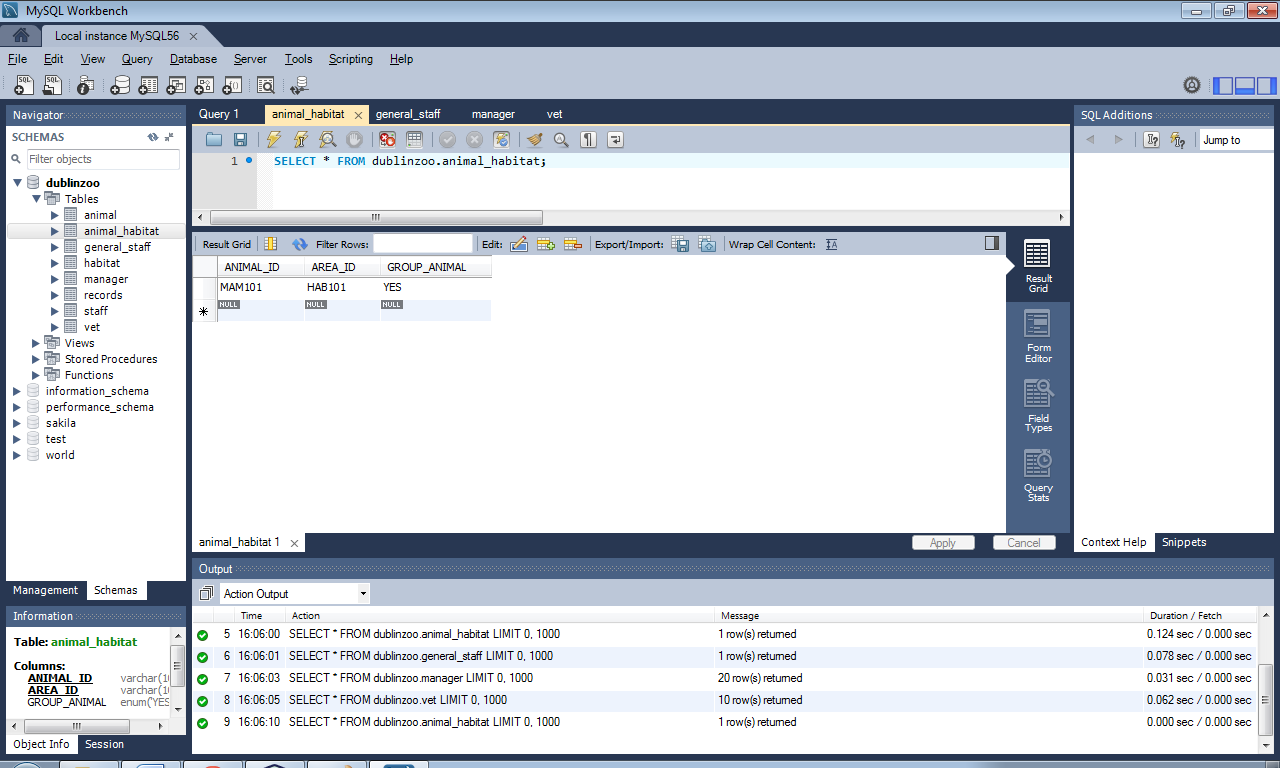
Above is the staff table.



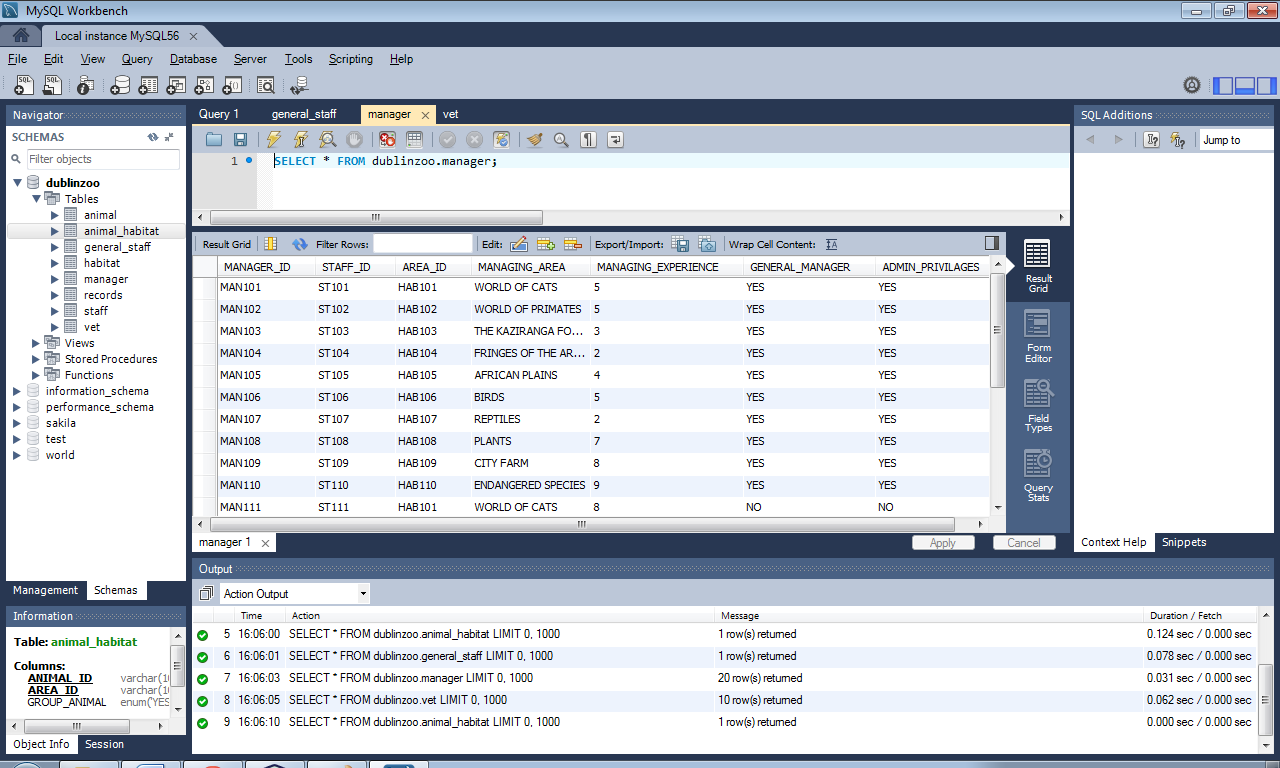
Above is the habitat table.



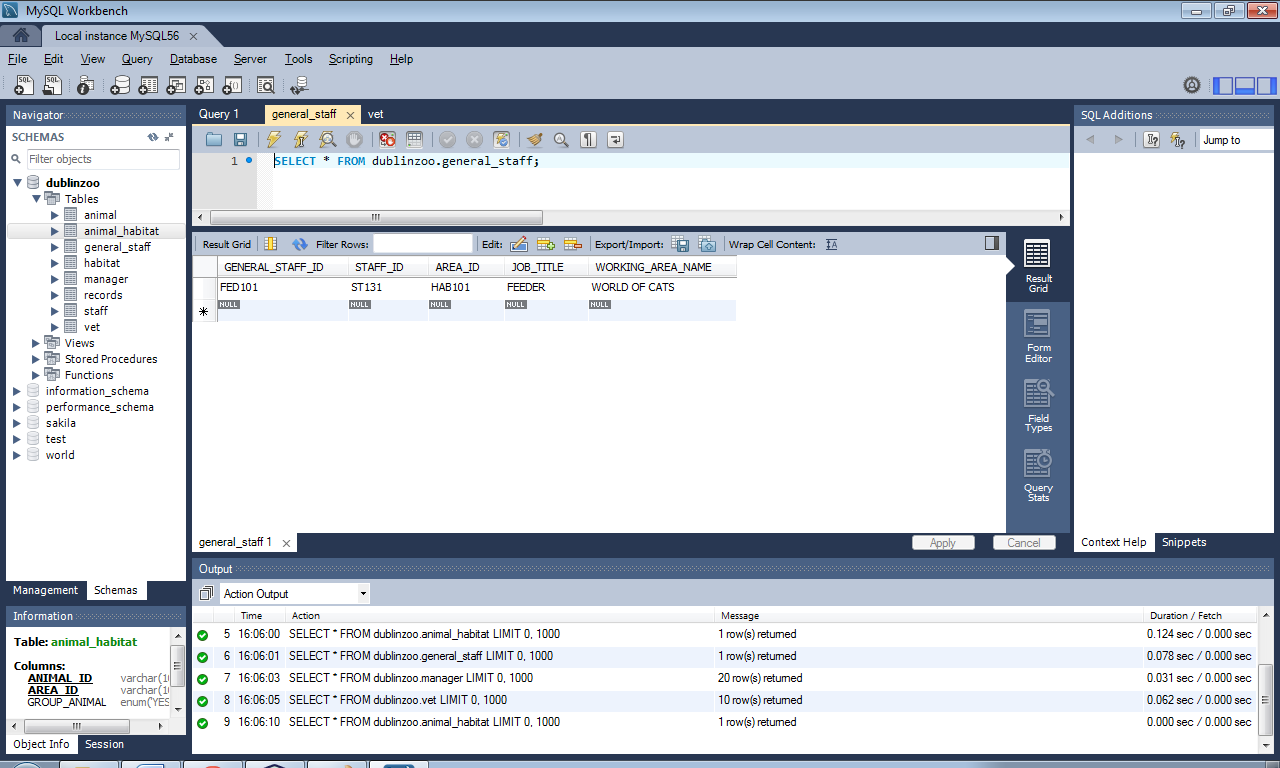
Above is the records table.



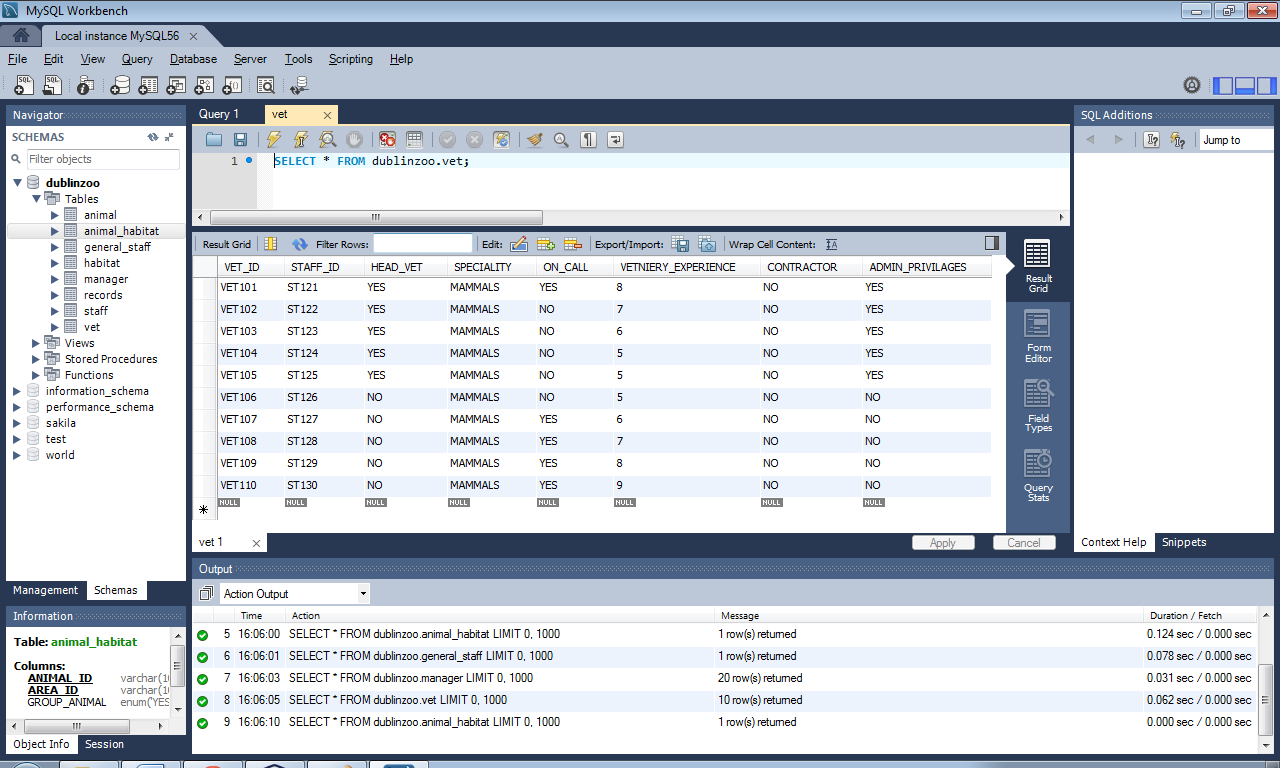
Above is the animal habitat table.



Above is the manager table.



Above is the general staff table.



Above is the vet table.

Queries

SELECT \* FROM animal

SELECT \* FROM animal

WHERE ANIMAL\_NAME="LION";

SELECT NAME,SALARY

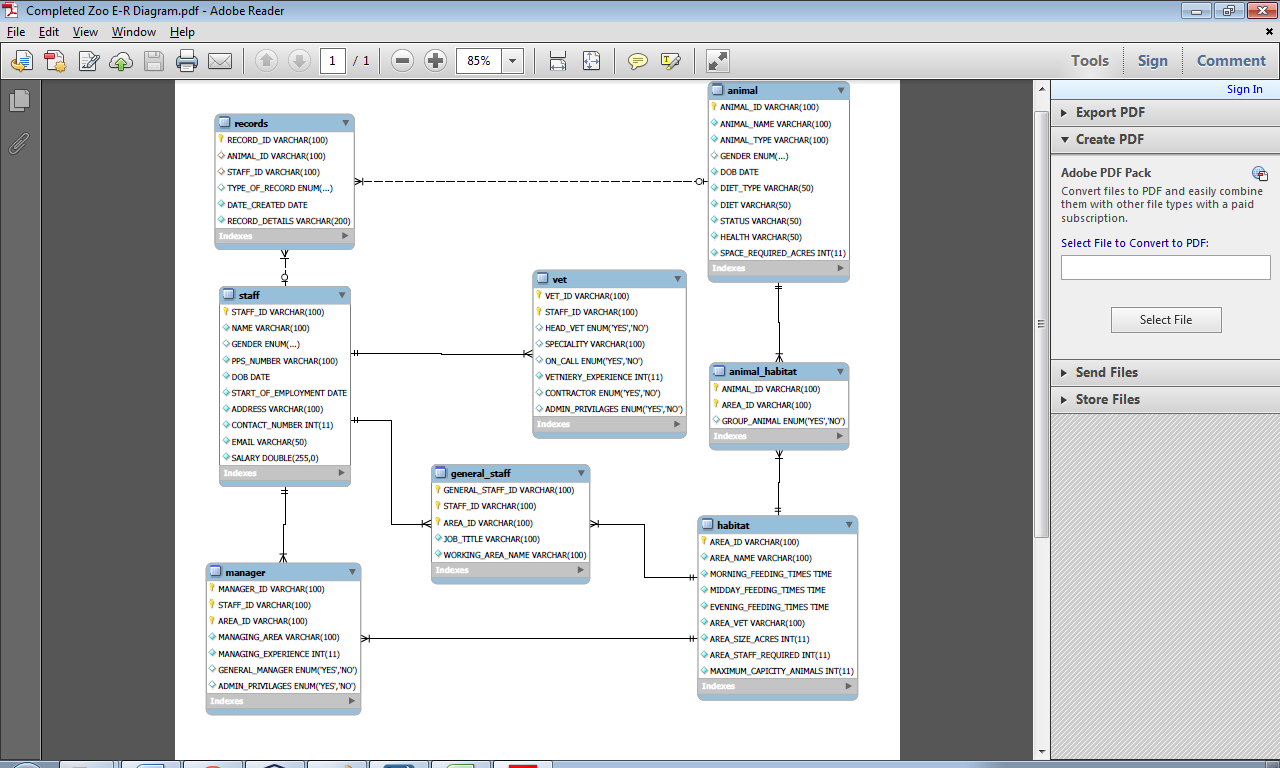
FROM staff

WHERE SALARY <=40000;

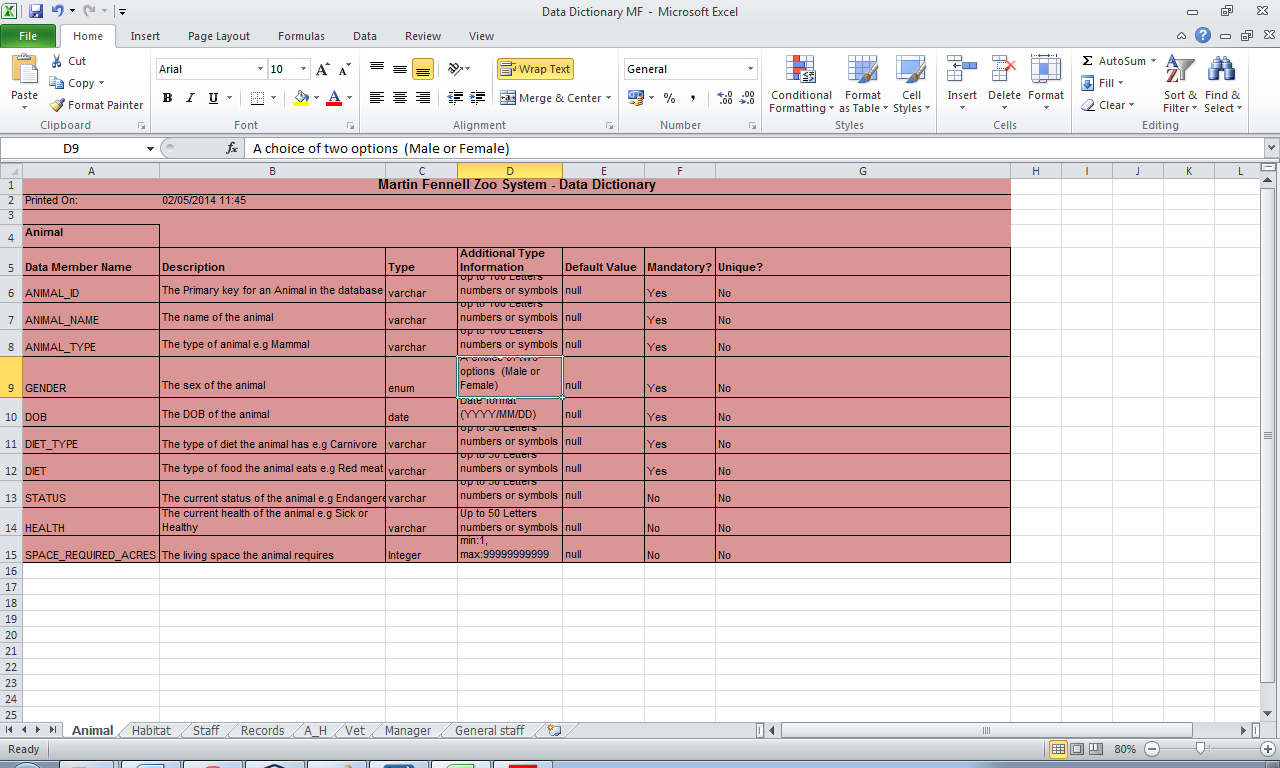
SELECT \* FROM staff where STAFF\_ID LIKE "%ST10%";

ER Diagram

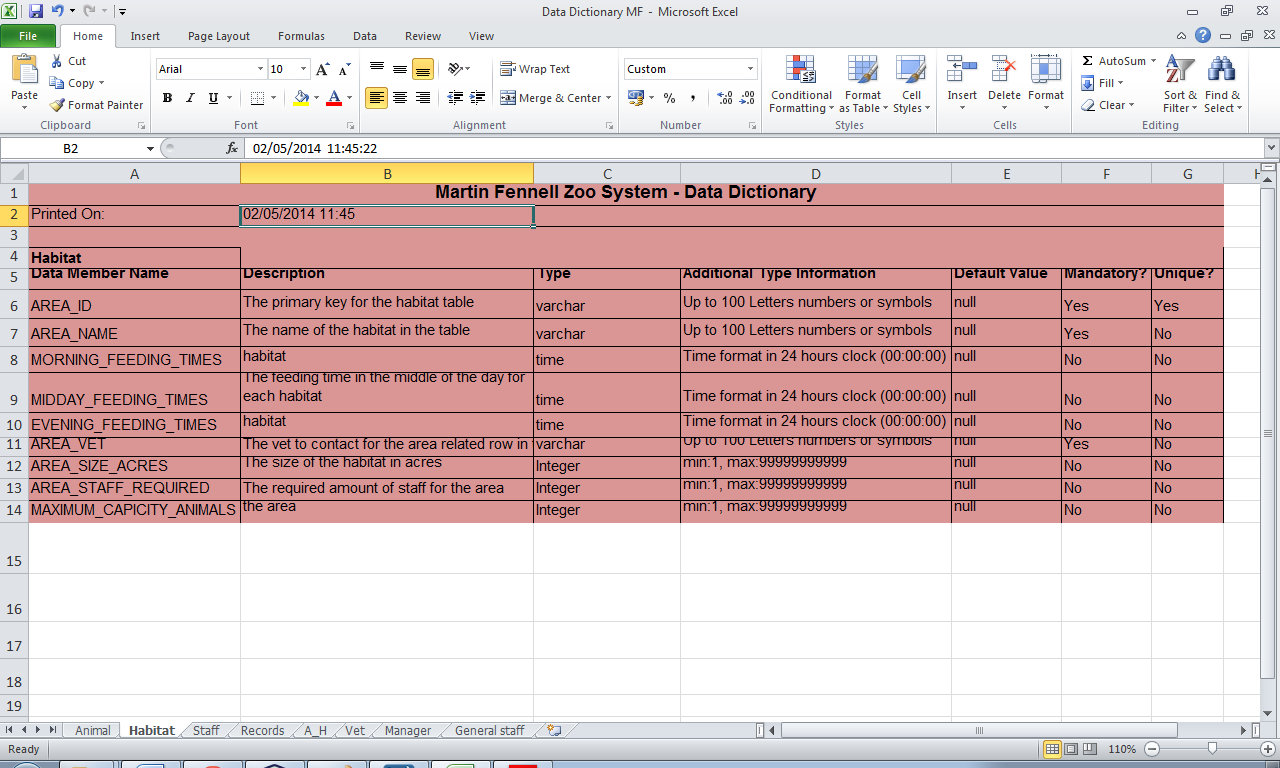
A screenshot of my ER Diagram can be seen below:

Data Dictionary

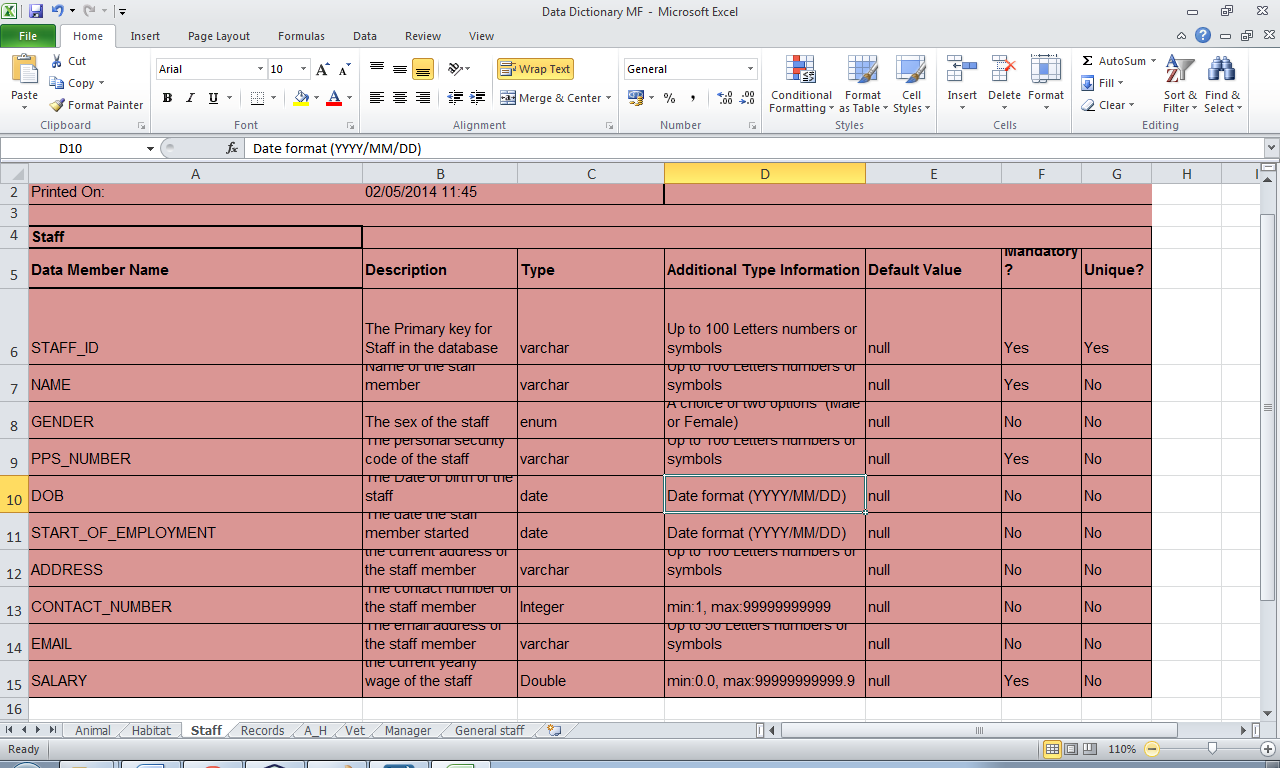
Animal Table:



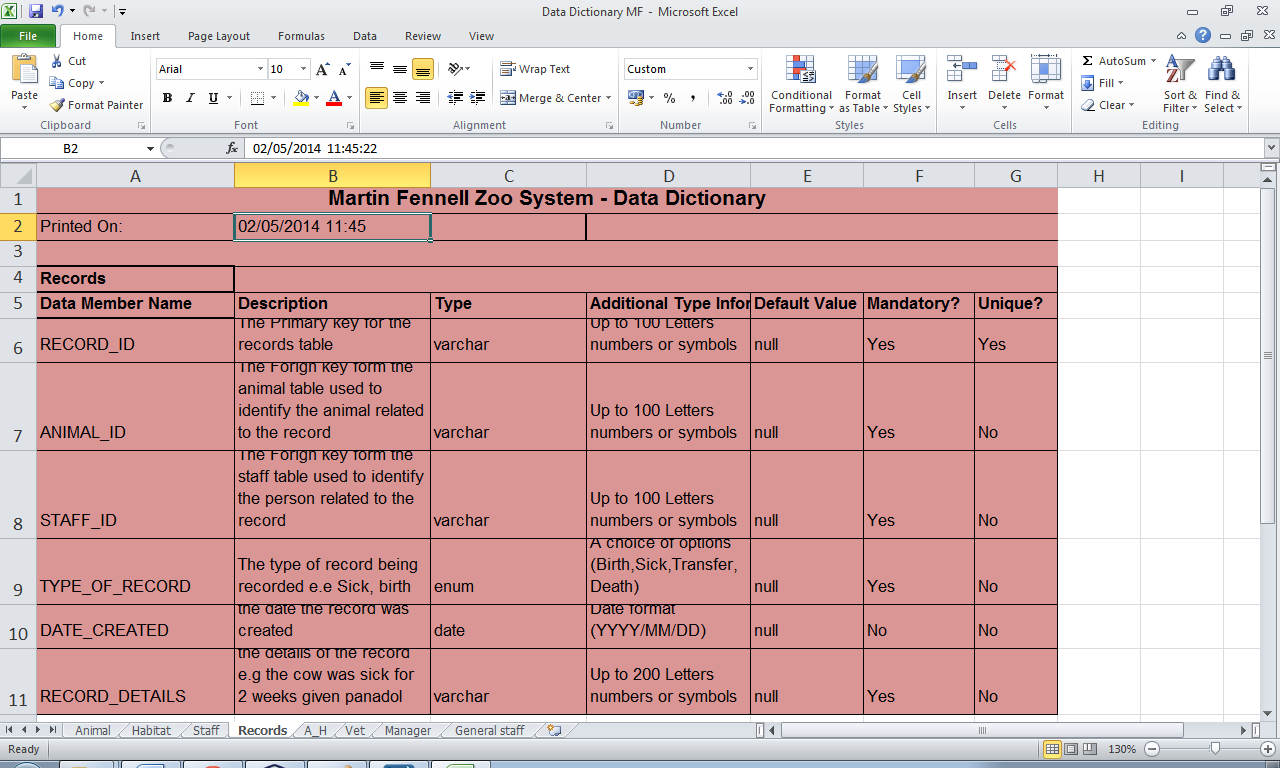
Habitat Table:



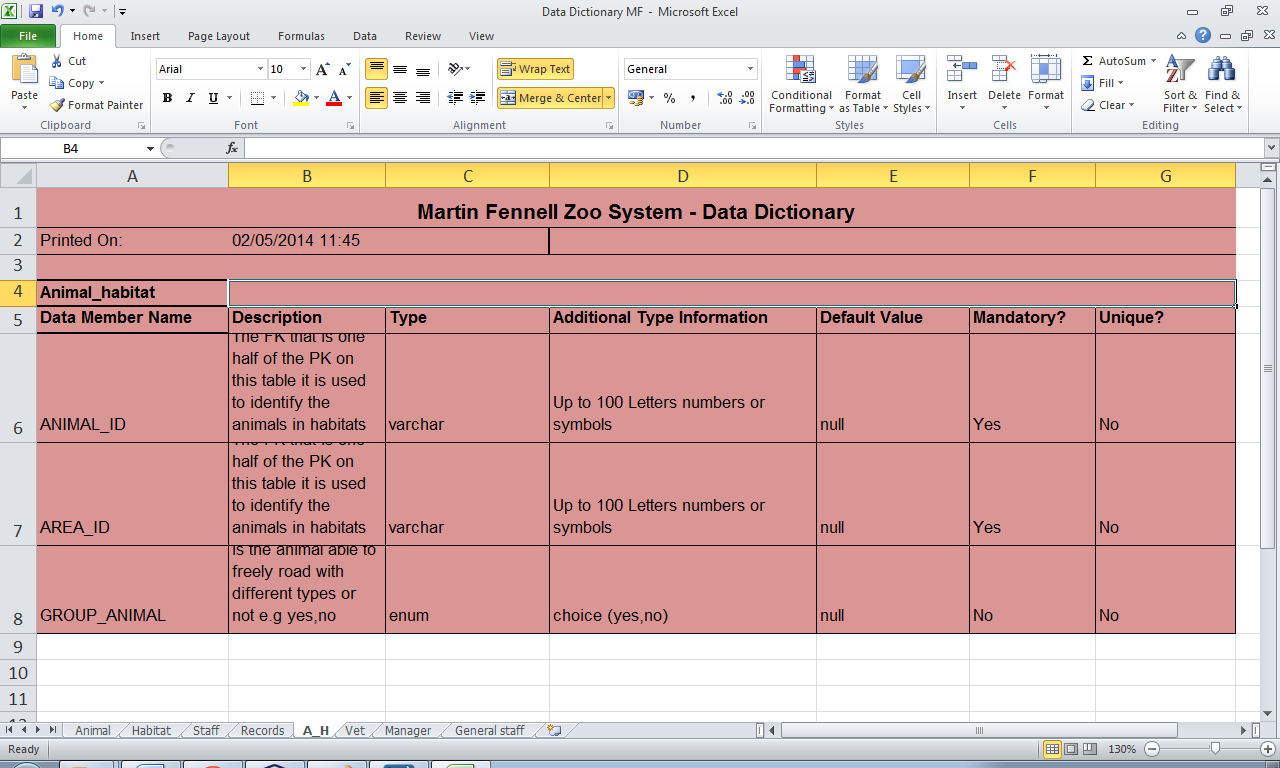
Staff Table:



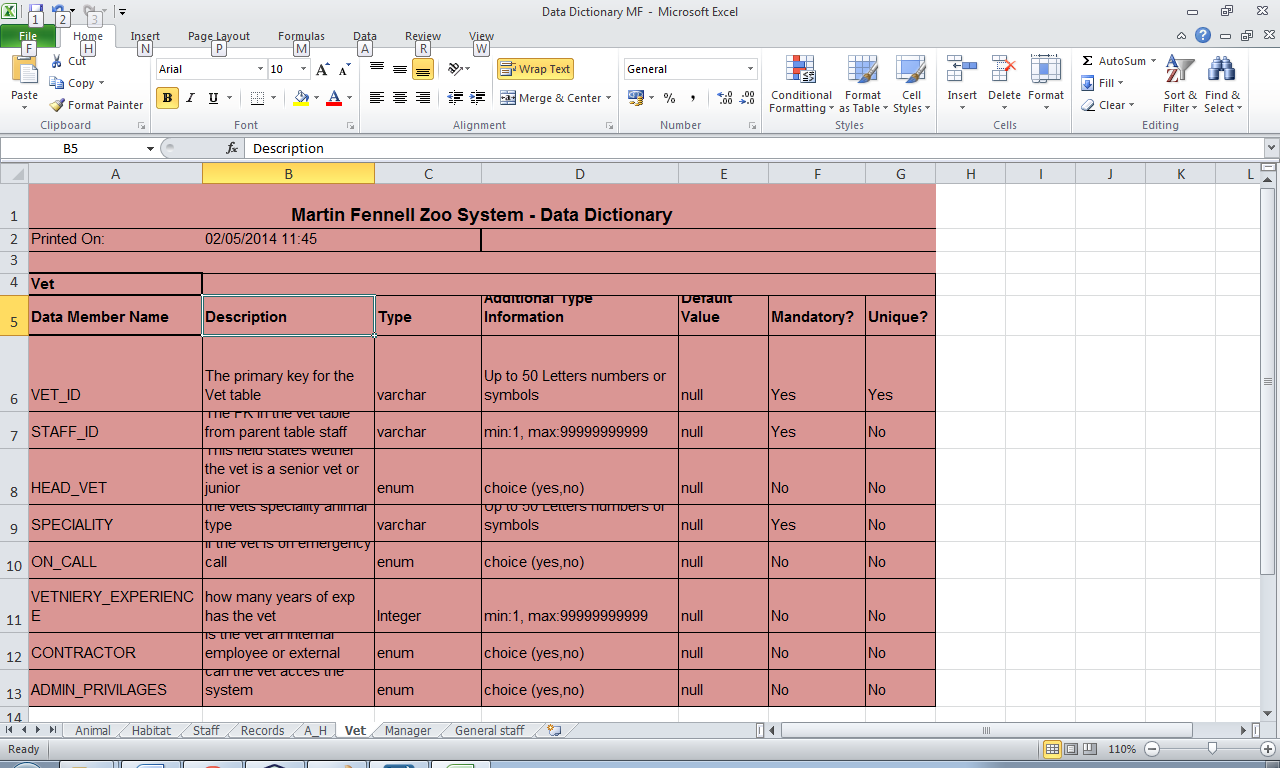
Records Table:



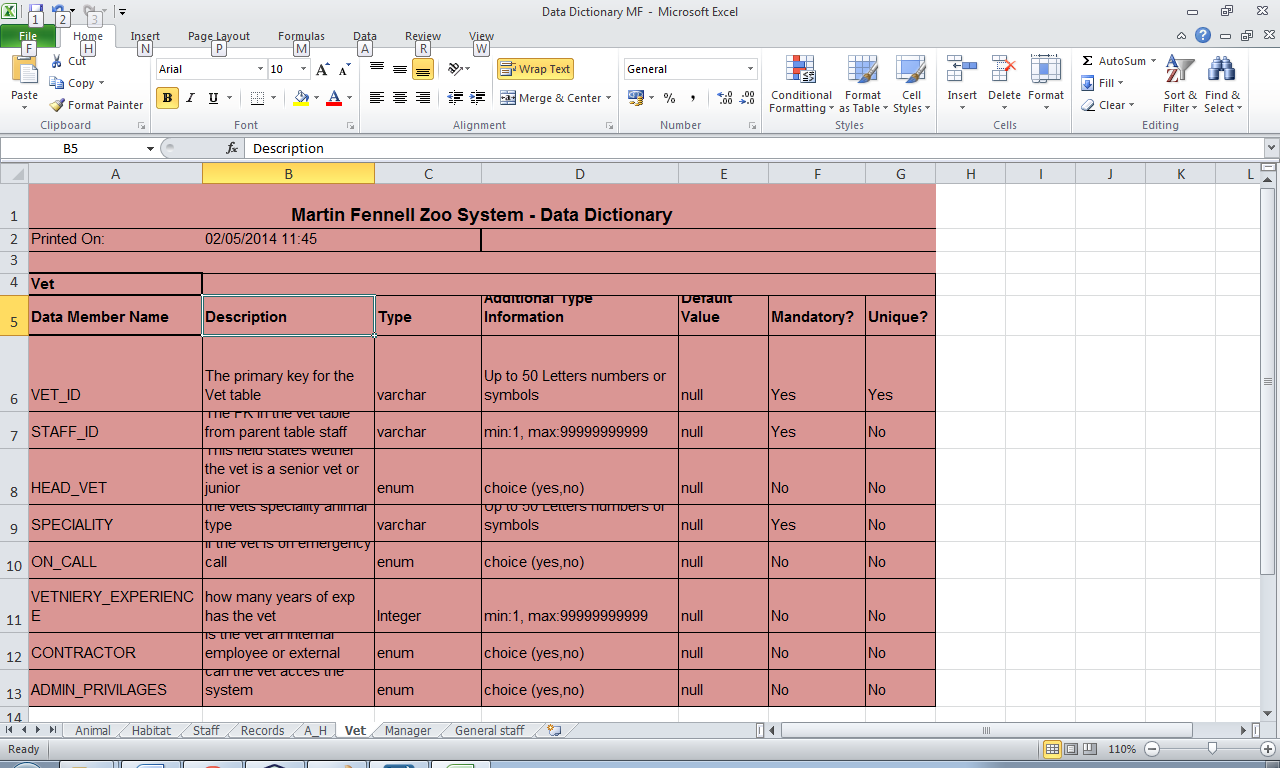
Animal habitat table:



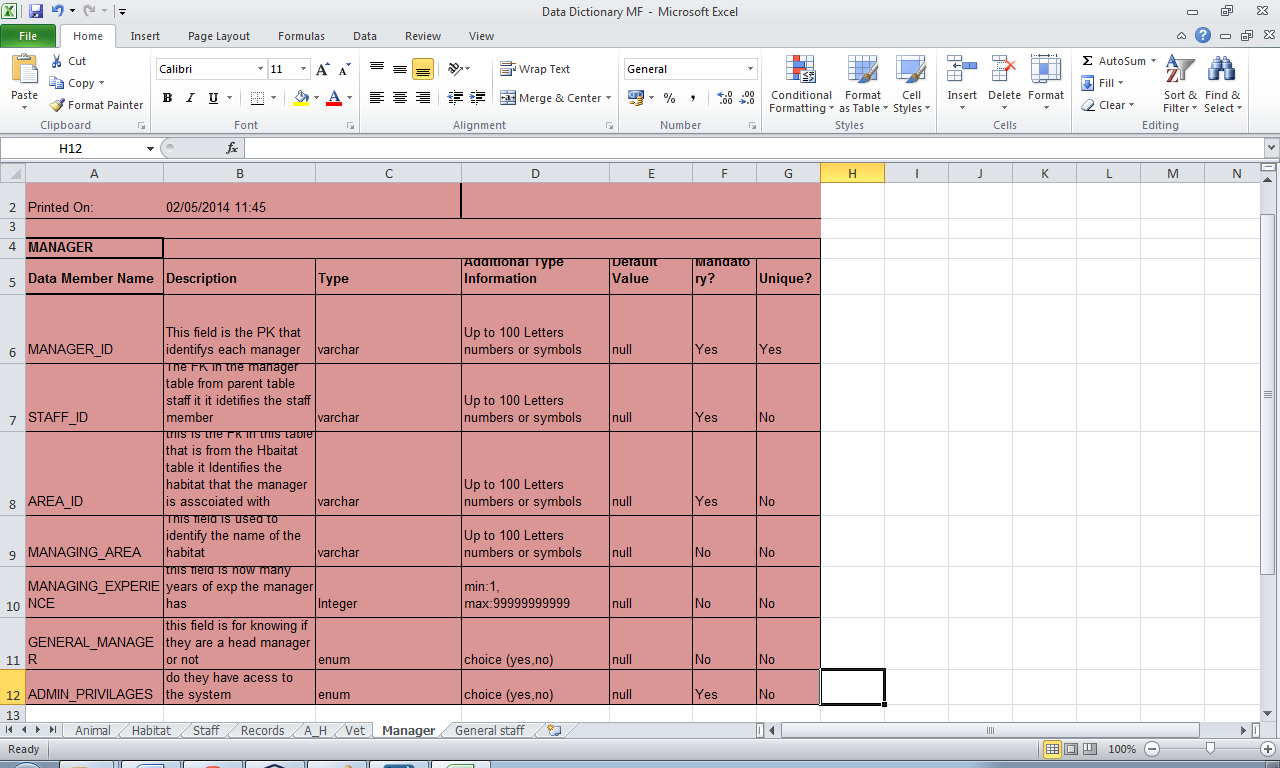
Manager Table:



Vet Table:



Animal General staff Table:



Create Table Script

Create Tables:

CREATE TABLE ANIMAL (

ANIMAL\_ID VARCHAR(100),

ANIMAL\_NAME VARCHAR(100) NOT NULL,

ANIMAL\_TYPE VARCHAR(100) NOT NULL,

GENDER ENUM('MALE','FEMALE'),

DOB DATE NOT NULL,

DIET\_TYPE VARCHAR(50) NOT NULL,

DIET VARCHAR(50) NOT NULL,

STATUS VARCHAR(50) NOT NULL,

HEALTH VARCHAR(50) NOT NULL,

SPACE\_REQUIRED\_ACRES INT NOT NULL,

PRIMARY KEY (ANIMAL\_ID) );

CREATE TABLE HABITAT (

AREA\_ID VARCHAR(100),

AREA\_NAME VARCHAR(100) NOT NULL,

MORNING\_FEEDING\_TIMES TIME NOT NULL,

MIDDAY\_FEEDING\_TIMES TIME NOT NULL,

EVENING\_FEEDING\_TIMES TIME NOT NULL,

AREA\_VET VARCHAR(100) NOT NULL,

AREA\_SIZE\_ACRES INT NOT NULL,

AREA\_STAFF\_REQUIRED INT NOT NULL,

MAXIMUM\_CAPICITY\_ANIMALS INT NOT NULL,

PRIMARY KEY (AREA\_ID) );

CREATE TABLE STAFF (

STAFF\_ID VARCHAR(100),

NAME VARCHAR(100) NOT NULL,

GENDER ENUM('MALE','FEMALE'),

PPS\_NUMBER VARCHAR(100) NOT NULL,

DOB DATE NOT NULL,

START\_OF\_EMPLOYMENT DATE NOT NULL,

ADDRESS VARCHAR(100) NOT NULL,

CONTACT\_NUMBER INT NOT NULL,

EMAIL VARCHAR(50) NOT NULL,

SALARY DOUBLE(255,00) NOT NULL,

PRIMARY KEY (STAFF\_ID) );

CREATE TABLE RECORDS (

RECORD\_ID VARCHAR(100),

ANIMAL\_ID VARCHAR(100),

STAFF\_ID VARCHAR(100),

TYPE\_OF\_RECORD ENUM('BIRTH','DEATH','TRANSFER','SICK'),

DATE\_CREATED DATE NOT NULL,

RECORD\_DETAILS VARCHAR(200) NOT NULL,

PRIMARY KEY (RECORD\_ID),

FOREIGN KEY (ANIMAL\_ID) REFERENCES ANIMAL(ANIMAL\_ID),

FOREIGN KEY (STAFF\_ID) REFERENCES STAFF(STAFF\_ID)

ON DELETE CASCADE

ON UPDATE CASCADE);

CREATE TABLE ANIMAL\_HABITAT (

ANIMAL\_ID VARCHAR(100),

AREA\_ID VARCHAR(100),

GROUP\_ANIMAL ENUM('YES','NO'),

PRIMARY KEY (ANIMAL\_ID, AREA\_ID),

FOREIGN KEY (ANIMAL\_ID) REFERENCES ANIMAL(ANIMAL\_ID),

FOREIGN KEY (AREA\_ID) REFERENCES HABITAT(AREA\_ID));

CREATE TABLE VET (

VET\_ID VARCHAR(100),

STAFF\_ID VARCHAR(100),

HEAD\_VET ENUM('YES','NO'),

SPECIALITY VARCHAR(100),

ON\_CALL ENUM('YES','NO'),

VETNIERY\_EXPERIENCE INT NOT NULL,

CONTRACTOR ENUM('YES','NO'),

ADMIN\_PRIVILAGES ENUM('YES','NO'),

PRIMARY KEY (VET\_ID,STAFF\_ID),

FOREIGN KEY (STAFF\_ID) REFERENCES STAFF(STAFF\_ID)

ON DELETE CASCADE

ON UPDATE CASCADE);

CREATE TABLE MANAGER (

MANAGER\_ID VARCHAR(100),

STAFF\_ID VARCHAR(100),

AREA\_ID VARCHAR(100),

MANAGING\_AREA VARCHAR(100)NOT NULL,

MANAGING\_EXPERIENCE INT NOT NULL,

GENERAL\_MANAGER ENUM('YES','NO'),

ADMIN\_PRIVILAGES ENUM('YES','NO'),

PRIMARY KEY (MANAGER\_ID,STAFF\_ID, AREA\_ID),

FOREIGN KEY (STAFF\_ID) REFERENCES STAFF(STAFF\_ID),

FOREIGN KEY (AREA\_ID) REFERENCES HABITAT(AREA\_ID)

ON DELETE CASCADE

ON UPDATE CASCADE);

CREATE TABLE GENERAL\_STAFF (

GENERAL\_STAFF\_ID VARCHAR(100),

STAFF\_ID VARCHAR(100),

AREA\_ID VARCHAR(100),

JOB\_TITLE VARCHAR(100) NOT NULL,

WORKING\_AREA\_NAME VARCHAR(100)NOT NULL,

PRIMARY KEY (GENERAL\_STAFF\_ID,STAFF\_ID, AREA\_ID),

FOREIGN KEY (STAFF\_ID) REFERENCES STAFF(STAFF\_ID),

FOREIGN KEY (AREA\_ID) REFERENCES HABITAT(AREA\_ID)

ON DELETE CASCADE

ON UPDATE CASCADE);

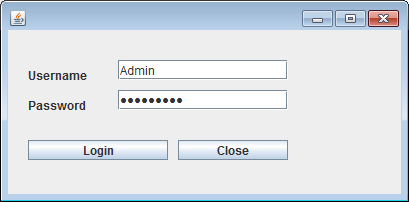
Test Cases

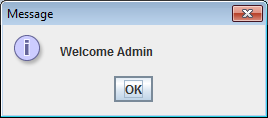
Test Case 1

Add animal to the database:

* Login to the system as a manager (Username:Admin) (Password: dublinzoo)
* Click animal insert button on the main frame
* Enter choose animals in the dropdown box.
* Press enter when animals is chosen.
* Enter an animal id.
* Press search

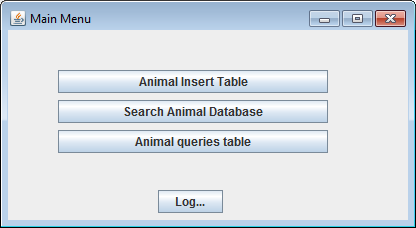
The login window is opened.



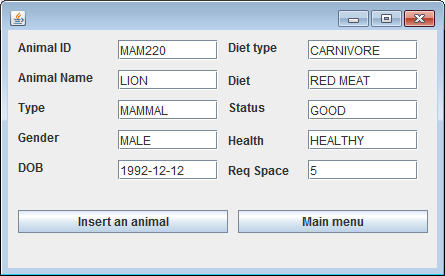


The manager window is opened.

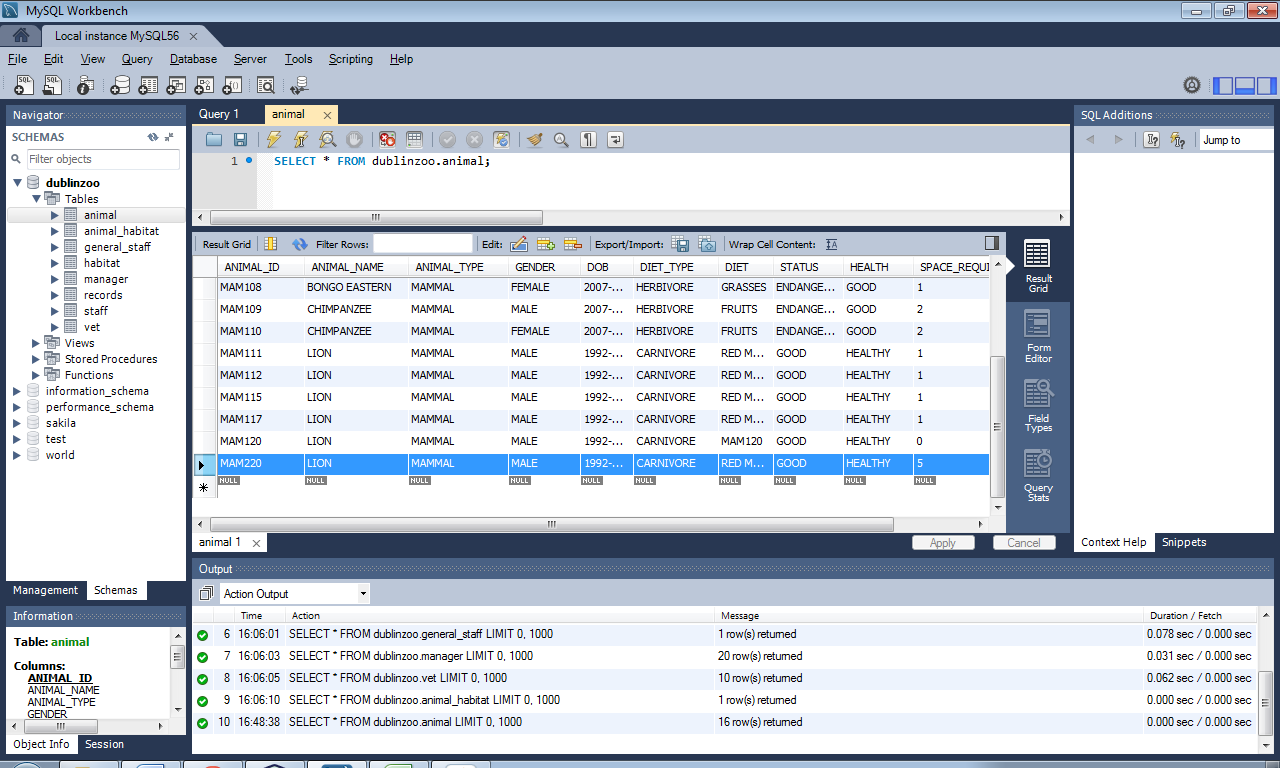
Select animal insert button from the options.



Select animal insert button from the options.



Input info into the txt fields and click the insert an animal button.

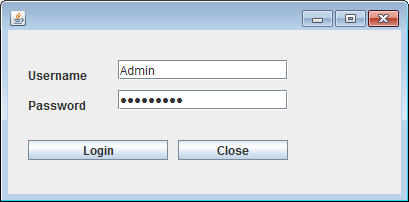


Above is the animal that was added to the database.

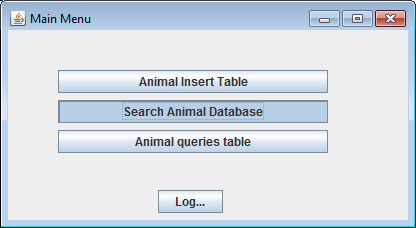
**Test Case 2**

Searching for an animal in the program with the vet and manager tables;

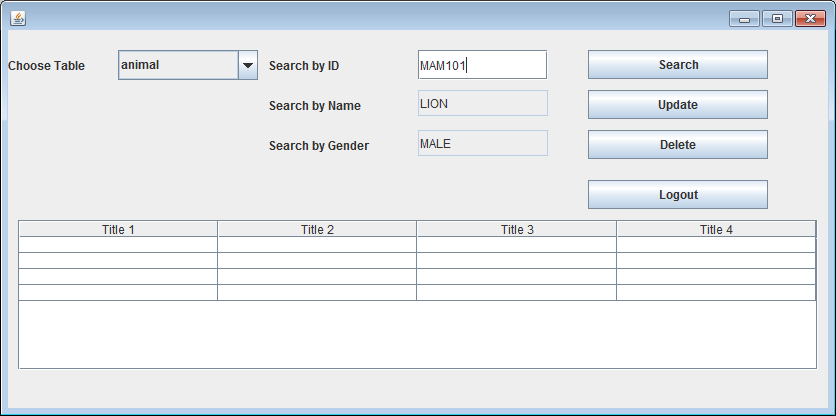
* Run the login form
* Enter the username and password for a manager
* Press search animal database.
* Choose animal then press enter to unlock search choices
* Type in an animal id that’s in the database e.g. MAM101.
* Press search.



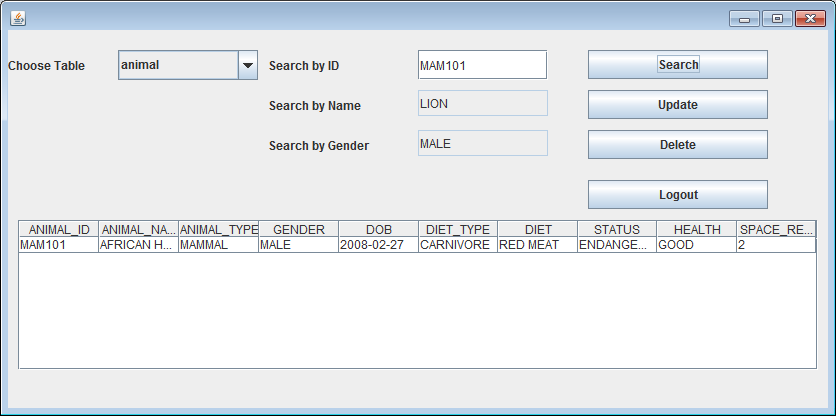
Above is the login frame with the manager details.



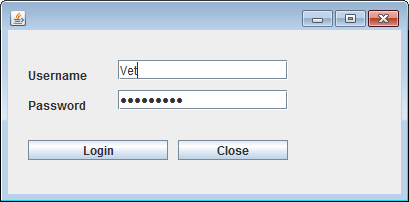
Above is the main frame for managers



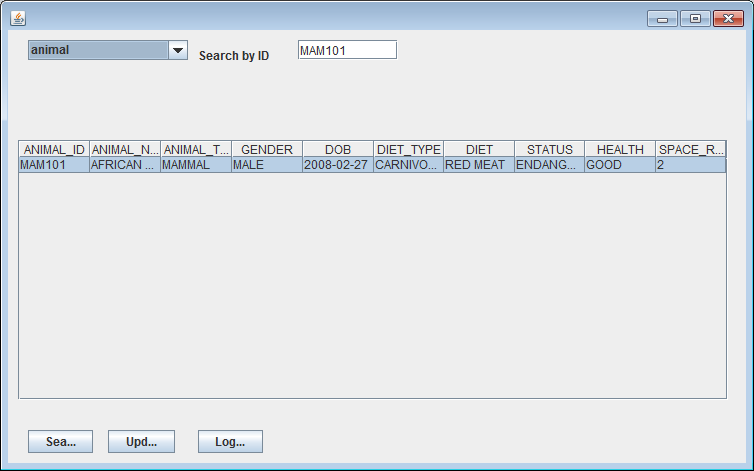
Above is the manager search update and delete frame.



Above is the result of a search for MAM101.



Above is the login frame for the program with the vet details.

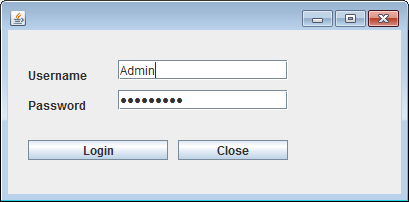


Above is the results after the vet searched MAM101.

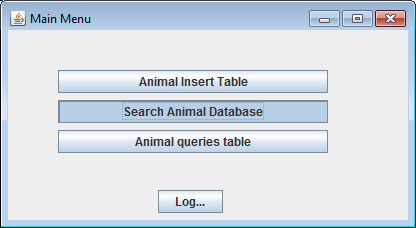
Test Case 3

The next test case is to update an animal from the database

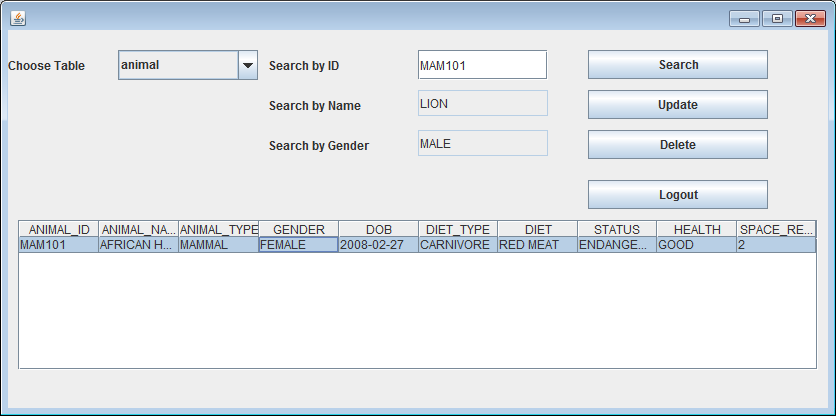
* Open the program and log in as a manager.
* Carry out a search for MAM101
* Edit a field when complete press enter.
* Press the update button.



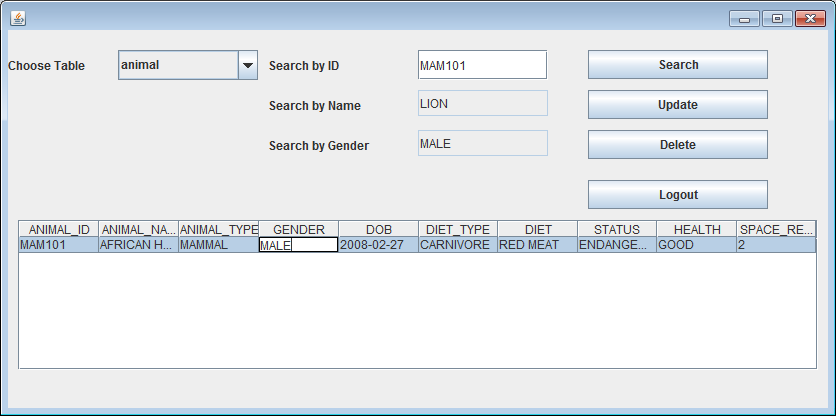
Above is the login form



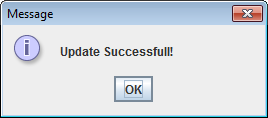
Above is the main manager frame



Above is the result for mam101 search.



Above is the column gender being changed.

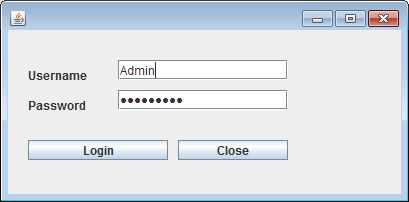


This is the message that appears after the update is complete.

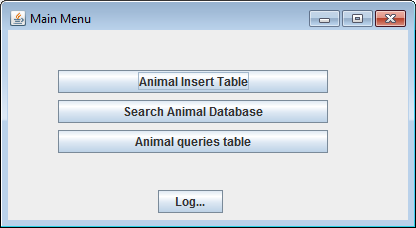
Test Case 4

The next test case is to delete an animal from the database

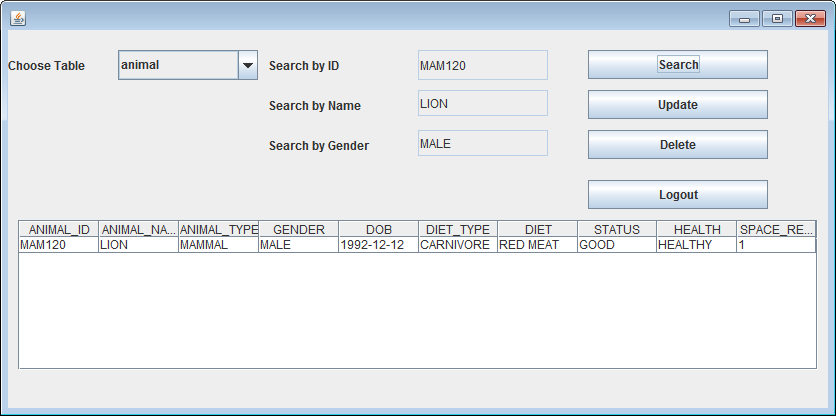
* Open the program and log in as a manager.
* Carry out a search for MAM120
* When the result comes back press the delete button to remove the animal from the database.



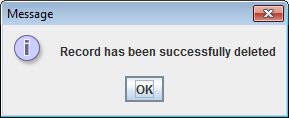
Above is the login frame.



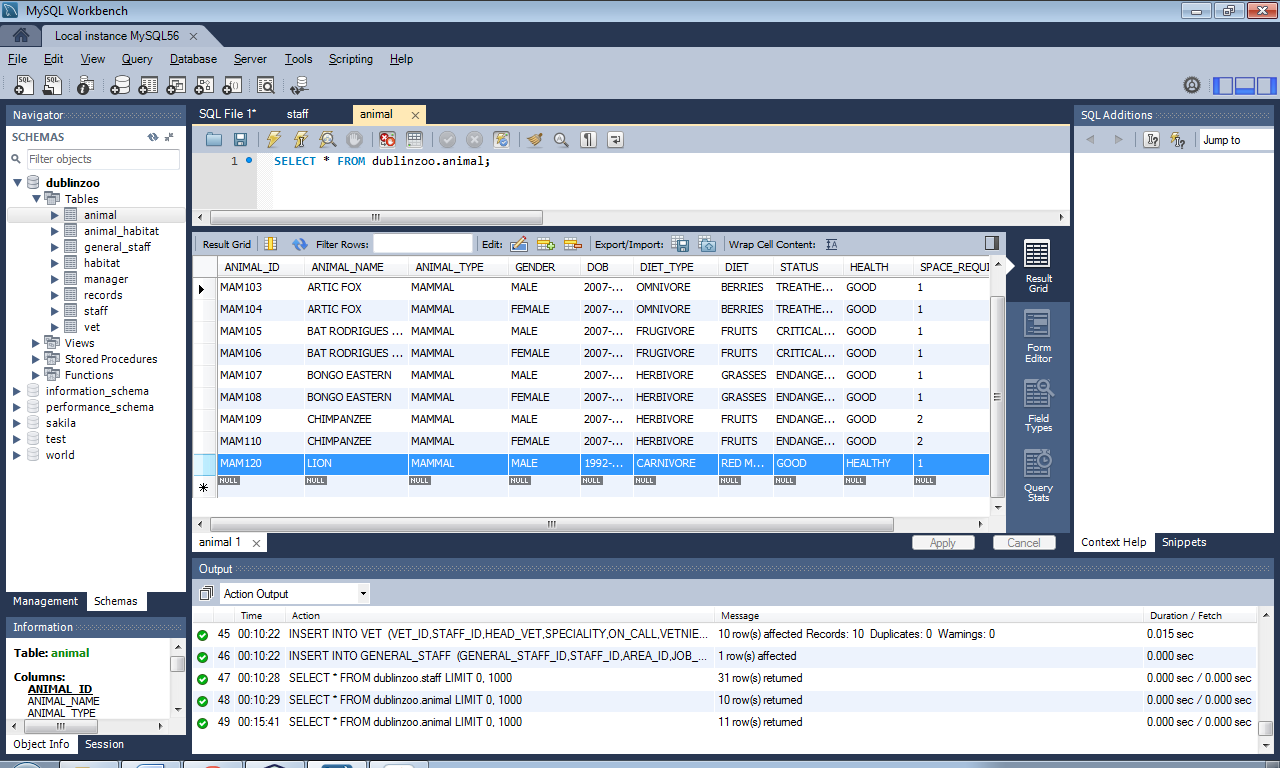
Above is the main frame for managers.



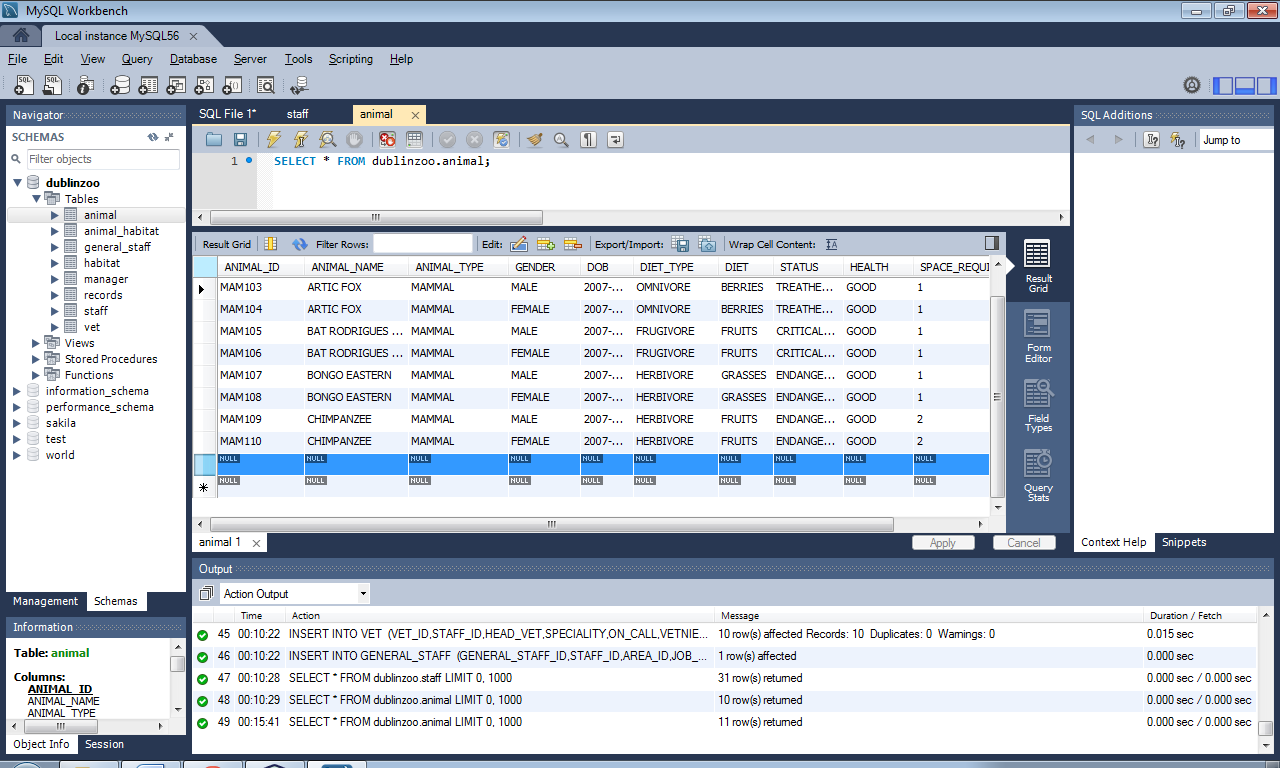
Above is the search update and delete table where the the search result for mam120 has come back.



Above is an image of the the message that popped up after the delete button was pressed.



Above is a screen shot of the animal MAM120 that was in the database before the delete button was pressed.



Above is a screenshot of the animal table where mam120 was before it got deleted from the database.