**Databases for Data Science**

**Term Project**

The first project consists of two parts. In the first part, you will create a database called “Portal Project Ecological Database.” This is a database from a scientific project for long-term monitoring and experimental manipulation of a Chihuahuan desert ecosystem near Portal, Arizona. Hence, they are real scientific data gathered over a long period of time. You will be installing this database in your SQLite database system instance that you have already created and will be using it for performing many data manipulation operations.

In the second part, you will create a database called “Northwind,” which consists of a schema for small businesses. This is a popular database used for learning about relational databases. You will use the SQLite system to install this database and perform data querying using SQL.

1. Portal Project: The data for the Portal Project can be found at the figshare site:

<https://figshare.com/articles/Portal_Project_Teaching_Database/1314459>

Create a folder called PortalProject, and download the Download All zip file (called 1314459.zip). Unzip it and you will see a set of .csv files in there as well as a .sqlite file.

I don’t want you to use the .sqlite file. Instead, create three tables (species, plots, surveys) using CREATE statements, and then upload the files in the .csv file into the database. Save the database in a file called YourName\_PortalProject.db, and upload that also as part of the project submission. You will also do some SQL queries and data manipulation as mentioned later.

1. Northwind: The data for Northwind is found in GitHub at the following location:

<https://github.com/jpwhite3/northwind-SQLite3>

Create a folder called Northwind, and then clone or download the zip file in there and unzip it. The website contains a READMe.md file, which shows the schema for the database. The file is also found in your directory as Northwind\_ERD. Unzip the Northwind\_large.sqlite.zip file. This contains the database that you want to upload into SQLite. In this case, you can open the .sqlite file as your database (no need for creating tables or uploading CSV files). You will do some SQL queries as mentioned next as part of the project submission.

**Queries to Be Answered**

Portal Project:

1. Find all genus and species of the rodent taxa.

SELECT genus, species FROM species s

WHERE taxa = 'Rodent';

1. Find all species (genus, species, and taxa) that were found by survey in 1987.

SELECT genus, species, taxa, year FROM species s

JOIN surveys sv ON sv.species\_id = s.species\_id

WHERE sv.year = 1987

1. Print hindfoot length and weight of all species found in “Long-term Krat Exclosure.”

SELECT hindfoot\_length, weight, plot\_type, species, genus, taxa FROM surveys s

INNER JOIN plots p ON p.plot\_id = s.plot\_id

INNER JOIN species sp on sp.species\_id = s.species\_id

WHERE p.plot\_type = 'Long-term Krat Exclosure';

Note: This is just a query for the requested parameters, and does not account for null values (improper logs) or differing taxa (e.g. “Bird’s” don’t have hind feet). If you were looking for the average parameters for each species present in the Long-term Krat Exclosure (excluding rows where data is missing), the query would be:

SELECT AVG(hindfoot\_length), AVG(weight), plot\_type, species, genus, taxa FROM surveys s

INNER JOIN plots p ON p.plot\_id = s.plot\_id

INNER JOIN species sp on sp.species\_id = s.species\_id

WHERE p.plot\_type = 'Long-term Krat Exclosure' AND s.hindfoot\_length IS NOT NULL AND s.weight IS NOT NULL

GROUP BY sp.species;

1. Find the dominant species in “Spectab exclosure.” What if you restrict to only 1998?

SELECT plot\_type AS Enclosure, species, COUNT(\*) AS [Number of Species] FROM surveys s

INNER JOIN plots p ON p.plot\_id = s.plot\_id

INNER JOIN species sp on sp.species\_id = s.species\_id

WHERE p.plot\_type = 'Spectab exclosure'

GROUP BY sp.species

ORDER BY [Number of Species] DESC;

SELECT plot\_type AS Enclosure, species, COUNT(\*) AS [Number of Species] FROM surveys s

INNER JOIN plots p ON p.plot\_id = s.plot\_id

INNER JOIN species sp on sp.species\_id = s.species\_id

WHERE p.plot\_type = 'Spectab exclosure' AND s.year = 1998

GROUP BY sp.species

ORDER BY [Number of Species] DESC;

Both are the species ‘merriami’, however 1998 is a smaller margin as compared to the overall of Spectab exclosure.

Northwind Project:

1. Find all suppliers who supply a beverage (category).

SELECT sup.CompanyName, sup.SupplierID, c.CategoryName FROM Suppliers sup

INNER JOIN Products p ON p.SupplierID = sup.SupplierID

INNER JOIN Categories c ON p.CategoryID = c.CategoryID

WHERE c.CategoryName = 'Beverages'

GROUP BY sup.SupplierID

1. Find suppliers (print supplier companyname and region) who supplied both condiments and confections (categories).

SELECT sup.SupplierID, sup.CompanyName, sup.Region FROM Suppliers sup

INNER JOIN Products p ON p.SupplierID = sup.SupplierID

INNER JOIN Categories c ON p.CategoryID = c.CategoryID

WHERE c.CategoryName = 'Condiments' OR c.CategoryName = 'Confections'

GROUP BY sup.SupplierID

HAVING COUNT(DISTINCT c.CategoryID) > 1;

We COUNT the unique values for CategoryID (*either 2 or 3*) and if we get more than one unique ID, we know that both Condiments and Confections appear for that SupplierID. Proud of myself for thinking of this solution.

1. Find all orders (print only orderId, orderdate, Product Name) by customer “Piccolo und mehr” through the employee named “Michael Suyama”.

SELECT ord.OrderID, ord.OrderDate, p.ProductName FROM Orders ord

INNER JOIN "Order Details" od ON ord.OrderID = od.OrderID

INNER JOIN Customers c ON c.CustomerID = ord.CustomerID

INNER JOIN Products p ON p.ProductID = od.ProductID

INNER JOIN Employees e on e.EmployeeID = ord.EmployeeID

WHERE c.**CompanyName** = "Piccolo und mehr" AND e.FirstName ='Michael' AND e.LastName = "Suyama";

Note: Customer “Piccolo und mehr” is actually the **company name**.

1. Count the number of shipments handled by each shipping company. What if you group by year?

SELECT COUNT(OrderID) as NumOrders, o.ShipVia, s.CompanyName FROM Orders o

INNER JOIN Shippers s on s.ShipperID = o.ShipVia

GROUP BY o.ShipVia;

SELECT COUNT(OrderID) as NumOrders, o.ShipVia, s.CompanyName, strftime('%Y', o.OrderDate) AS year FROM Orders o

INNER JOIN Shippers s on s.ShipperID = o.ShipVia

GROUP BY o.ShipVia, year

ORDER BY year, o.ShipVia;

I am assuming we want to check the number of shipments handled by each company ***individually*** by each year.