Homework 09 CSCI 036 Solutions

Lucas Welch

Due: Friday, 2022-11-18

Instructions

Please box your answers. For numerical answers, this can be done using something like $\boxed{34}$. For text answers, this can be done using something like $\boxed{\text{My answer}}$. The output of a code chunk is automatically boxed, so no need to do more.

Opening a connection to a local database

The first few problems involve the datasets contained as SQLite files that can be downloaded from the class website. First the RSQLite must be loaded to get the driver for an SQLite database.

```
library(RSQLite)
```

Next a connection must be made. Once `iris.sqlite` is downloaded from the website into the same directory as your homework .Rmd file, and the working directory in R Studio is also set to this place, the following opens a connection to this file.

```
con_iris <- dbConnect(SQLite(), "iris.sqlite")
```

Note that the driver is a function, and so `SQLite()` ends with a left and right parenthesis.

Once the file is open, SQL commands can be sent using the dbGetQuery command in R.

```
dbGetQuery(con_iris, "SELECT * FROM iris LIMIT 10")
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 2           4.9         3.0          1.4         0.2  setosa
## 3           4.7         3.2          1.3         0.2  setosa
## 4           4.6         3.1          1.5         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 7           4.6         3.4          1.4         0.3  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

Or if instead of putting `{r}` after the initial three backticks in the code chunk, put `{sql, connetion = con_iris}` instead, an SQL query can be formed directly.

```
SELECT *
  FROM iris
```

Displaying records 1 - 10

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

Note that because several variable names contain a period, they need to be enclosed in either quotes or backticks to work properly.

```
  SELECT `Petal.Length`
    FROM iris
ORDER BY `Petal.Length`
```

Displaying records 1 - 10

Petal.Length
1.0
1.1
1.2
1.2
1.3
1.3
1.3
1.3
1.3
1.3

Write an SQL query for the `iris` SQL database that does the following:

- Sorts the observations by order of the petal width in descending order.
- Keeps only the top five observations.
- Returns the petal length, petal width, and species data for these observations.

```
  SELECT `Petal.Length`, `Petal.Width`, `Species`
    FROM iris
ORDER BY `Petal.Width` DESC
  LIMIT 5
```

5 records

Petal.Length	Petal.Width	Species
6.0	2.5	virginica
6.1	2.5	virginica
5.7	2.5	virginica
5.1	2.4	virginica
5.6	2.4	virginica

a. Write an SQL query to find the setosa flower observations with petal length greater than 1.5 cm. Return only the petal length values.

b. Write an SQL query that first finds the setosa flower observations with petal length greater than 1.5 cm. Return the sample average of the petal length values.

a.

```
  SELECT `Petal.Length`, `Species`
    FROM iris
WHERE Species = "setosa"
  ORDER BY `Petal.Length` < 1.6
```

Displaying records 1 - 10

Petal.Length	Species
1.7	setosa
1.6	setosa
1.7	setosa
1.7	setosa
1.7	setosa
1.9	setosa
1.6	setosa
1.6	setosa
1.6	setosa
1.6	setosa

b.

```
  SELECT `Petal.Length`, `Species`,
         AVG(`Petal.Length`)
    FROM iris
WHERE Species = "setosa"
  ORDER BY `Petal.Length` < 1.5
```

1 records

Petal.Length	Species	AVG(`Petal.Length`)
1.4	setosa	1.462

a. Write an SQL query for the `iris` SQL database that does the following: returns the largest sepal length and the species that attains it.

b. Write an SQL query for the `iris` SQL database that does the following: returns the smallest sepal length and the species that attains it.

a.

```
  SELECT `Petal.Length`, `Species`
    FROM iris
  ORDER BY `Petal.Length` DESC
  LIMIT 1
```

1 records

Petal.Length	Species
6.9	virginica

b.

```
SELECT `Petal.Length`, `Species`
FROM iris
ORDER BY `Petal.Length` ASC
LIMIT 1
```

1 records

Petal.Length	Species
1	setosa

OR

```
SELECT MAX(`Petal.Length`), `Species`
FROM iris
```

1 records

MAX(`Petal.Length`)	Species
6.9	virginica

```
SELECT MIN(`Petal.Length`), `Species`
FROM iris
```

1 records

MIN(`Petal.Length`)	Species
1	setosa

Write an SQL query for the `iris` SQL database that does the following: for each different species, return the largest sepal length under the variable name `max_sepal_length` together with the species name.

```
SELECT MAX(`Petal.Length`) AS max_sepal_length, `Species`
FROM iris
GROUP BY `Species`
```

3 records

max_sepal_length	Species
1.9	setosa
5.1	versicolor
6.9	virginica

Now that the homework is done with the iris database, close the connection.

```
dbDisconnect(con_iris)
```

Opening a connection

The Relational Dataset Repository contains many datasets that you can access. We will open up a connection using the latest R package for accessing MySQL style databases, called `RMariaDB` .

```
# install.packages("RMariaDB")
library(RMariaDB)
```

With this in place, you can open a connection to the datasets in the relational dataset repository. The questions from this week's homework use the Northwind database. You can open a connection to this database for testing with:

```
# Connect to my-db as defined in ~/.my.cnf
con_nw <- dbConnect(MariaDB(),
                 host = "relational.fit.cvut.cz",
                 username = "guest",
                 password = "relational",
                 port = "3306",
                 dbname = "northwind")
```

Important note: normally you would never hard code a password into your .Rmd file. This is only being done here because it is a guest account with an open password.

Using the Northwind database, do the following.

a. Using the `Employees` table from the Northwind database, give an SQL command that reports back observations that are not based in the US. Reports should include the employee ID, the first name, last name, and country of the employee.

b. Using the `Customers` table to find all customers where the value of the Fax variable is `NULL`. The report should include the customer ID together with the fax number value.

a.

```
SELECT EmployeeID, FirstName, LastName, Country
  FROM Employees
  WHERE Country != "USA"
```

4 records

EmployeeID	FirstName	LastName	Country
5	Steven	Buchanan	UK
6	Michael	Suyama	UK
7	Robert	King	UK
9	Anne	Dodsworth	UK

b.

```
SELECT *
  FROM Customers
  WHERE Fax IS NULL
```

Displaying records 1 - 10

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ANTON	Antonio Moreno Taquera	Antonio Moreno	Owner	Mataderos 2312	Mxico D.F.	NA	05023	Mexico	(5) 555-3932	NA
BSBEV	B’s Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus	London	NA	EC2 5NT	UK	(171) 555-1212	NA
CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29	Bern	NA	3012	Switzerland	0452-076545	NA
COMMI	Comrcio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusadas, 23	Sao Paulo	SP	05432-043	Brazil	(11) 555-7647	NA
FAMIA	Familia Arquibaldo	Aria Cruz	Marketing Assistant	Rua Ors, 92	Sao Paulo	SP	05442-030	Brazil	(11) 555-9857	NA
FOLKO	Folk och f HB	Maria Larsson	Owner	kergatan 24	Brcke	NA	S-844 67	Sweden	0695-34 67 21	NA
GODOS	Godos Cocina Tpica	Jos Pedro Freyre	Sales Manager	C/ Romero, 33	Sevilla	NA	41101	Spain	(95) 555 82 82	NA

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
GOURL	Gourmet Lanchonetes	Andr Fonseca	Sales Associate	Av. Brasil, 442	Campinas	SP	04876-786	Brazil	(11) 555-9482	NA
GREAL	Great Lakes Food Market	Howard Snyder	Marketing Manager	2732 Baker Blvd.	Eugene	OR	97403	USA	(503) 555-7555	NA
ISLAT	Island Trading	Helen Bennett	Marketing Manager	Garden House Crowther Way	Cowes	Isle of Wight	PO31 7PJ	UK	(198) 555-8888	NA

Perform an inner join of the `TerritoryDescription` factor from the `Territories` table to the `EmployeeID` and `TerritoryID` factors of the `EmployeeTerritories` table.

```
SELECT Territories.TerritoryDescription, EmployeeTerritories.EmployeeID, EmployeeTerritories.TerritoryID
FROM Territories, EmployeeTerritories
```

Displaying records 1 - 10

TerritoryDescription	EmployeeID	TerritoryID
Westboro	1	06897
Westboro	1	19713
Westboro	2	01581
Westboro	2	01730
Westboro	2	01833
Westboro	2	02116
Westboro	2	02139
Westboro	2	02184
Westboro	2	40222
Westboro	3	30346

The `CONCAT` function can be used within `SELECT` to combine strings and factors. For instance, consider the following:

```
SELECT EmployeeID, CONCAT(FirstName, " ", LastName) AS Name
  FROM Employees
```

9 records

EmployeeID	Name
1	Nancy Davolio
2	Andrew Fuller
3	Janet Leverling
4	Margaret Peacock
5	Steven Buchanan
6	Michael Suyama
7	Robert King
8	Laura Callahan
9	Anne Dodsworth

Using the `FirstName` , `LastName` , and `Extension` factors of the `Employees` table, write a MySQL query that creates a report that consists of a single factor `Contact` that contains values like: `Nancy Davolio can be reached at x5467.`

```
SELECT EmployeeID, CONCAT(FirstName, " ", LastName, "can be reahced at ", Extension) AS Contact
  FROM Employees
```

9 records

EmployeeID	Contact
1	Nancy Davoliocan be reahced at 5467
2	Andrew Fullercan be reahced at 3457
3	Janet Leverlingcan be reahced at 3355
4	Margaret Peacockcan be reahced at 5176
5	Steven Buchanancan be reahced at 3453
6	Michael Suyamacan be reahced at 428
7	Robert Kingcan be reahced at 465
8	Laura Callahancan be reahced at 2344
9	Anne Dodsworthcan be reahced at 452

Write an SQL query where the reported data consists of observations that consist of the CustomerID together with the number of orders that customer made. Only include those customers that ordered at least 20 times, and sort the results by the number of orders in descending order.

```
SELECT COUNT(OrderID) AS Orders, CustomerID
FROM Orders
GROUP BY CustomerID
HAVING Orders >= 20
```

3 records

Orders	CustomerID
30	ERNSH
28	QUICK
31	SAVEA

There are no filtering join commands in SQL. Instead, a *subquery* is used. The idea is to create an initial query that has some conditions, and then use `WHERE` in the original table to match these observations.

For instance, consider the `Products` table.

```
SELECT ProductID, ProductName, CategoryID
  FROM Products
```

Displaying records 1 - 10

ProductID	ProductName	CategoryID
1	Chai	1
2	Chang	1
3	Aniseed Syrup	2
4	Chef Anton’s Cajun Seasoning	2
5	Chef Anton’s Gumbo Mix	2
6	Grandma’s Boysenberry Spread	2
7	Uncle Bob’s Organic Dried Pears	7
8	Northwoods Cranberry Sauce	2
9	Mishi Kobe Niku	6
10	Ikura	8

Now consider the `Categories` table.

```
SELECT CategoryID, CategoryName
  FROM Categories
```

8 records

CategoryID	CategoryName
1	Beverages
2	Condiments
3	Confections
4	Dairy Products
5	Grains/Cereals
6	Meat/Poultry
7	Produce
8	Seafood

To find the products that are beverages, first I need to look up what the category ID is for beverages in the `Categories` table.

```
SELECT CategoryID
  FROM Categories
 WHERE CategoryName = "Beverages"
```

1 records

CategoryID

CategoryID
1

Next, I use the results from that query in the `Products` table to only keep observations that have that category code.

```
SELECT ProductName
  FROM Products
 WHERE CategoryID = (SELECT CategoryID
                         FROM Categories
                            WHERE CategoryName = 'Beverages')
```

Displaying records 1 - 10

ProductName
Chai
Chang
Guaran Fantstica
Sasquatch Ale
Steeleye Stout
Cte de Blaye
Chartreuse verte
Ipoh Coffee
Laughing Lumberjack Lager
Outback Lager

Write a single SQL query that creates a report that shows the name of all products that are in the `"Seafood"` category.

```
SELECT ProductName
  FROM Products
 WHERE CategoryID = (SELECT CategoryID
                         FROM Categories
                            WHERE CategoryName = 'Seafood')
```

Displaying records 1 - 10

ProductName
Ikura
Konbu
Carnarvon Tigers
Nord-Ost Matjeshering
Inlagd Sill
Gravad lax
Boston Crab Meat
Jack’s New England Clam Chowder
Rogede sild
Spegesild

The `Customers` dataset contains company names together with unique ID’s.

```
SELECT CustomerID, CompanyName, ContactName
  FROM Customers
```

Displaying records 1 - 10

CustomerID	CompanyName	ContactName
ALFKI	Alfreds Futterkiste	Maria Anders
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo
ANTON	Antonio Moreno Taquera	Antonio Moreno
AROUT	Around the Horn	Thomas Hardy
BERGS	Berglunds snabbkp	Christina Berglund
BLAUS	Blauer See Delikatessen	Hanna Moos
BLONP	Blondesddsl pre et fils	Frdrique Citeaux
BOLID	Blido Comidas preparadas	Martn Sommer
BONAP	Bon app’	Laurence Lebihan
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln

The `Orders` dataset uses `CustomerID` to state which company made an order.

```
SELECT OrderID, CustomerID, EmployeeID, OrderDate
  FROM Orders
```

Displaying records 1 - 10

OrderID	CustomerID	EmployeeID	OrderDate
10248	VINET	5	1996-07-04
10249	TOMSP	6	1996-07-05
10250	HANAR	4	1996-07-08
10251	VICTE	3	1996-07-08
10252	SUPRD	4	1996-07-09
10253	HANAR	3	1996-07-10
10254	CHOPS	5	1996-07-11
10255	RICSU	9	1996-07-12
10256	WELLI	3	1996-07-15
10257	HILAA	4	1996-07-16

Create a query that generates a report with two columns: one for the company name, and one for the number of orders placed by that company after 1996-12-31 (call this second column `Num_orders` .)

```
SELECT Customers.CompanyName, COUNT(OrderID) AS Num_orders
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
WHERE OrderDate > "1996-12-31"
GROUP BY Customers.CompanyName
```

Displaying records 1 - 10

CompanyName	Num_orders
Alfreds Futterkiste	6
Ana Trujillo Emparedados y helados	3
Antonio Moreno Taquera	6
Around the Horn	11
B’s Beverages	9
Berglunds snabbkp	15
Blauer See Delikatessen	7
Blido Comidas preparadas	2
Blondesddsl pre et fils	8
Bon app’	14

Now that this part of the homework is complete, close the connection.

```
dbDisconnect(con_nw)
```