Relational Databases

#### SQL Introduction: Operations on columns and rows

Author: W.P.G.Peterson

Assignment Contents

EXPECTED TIME: 3 HRS

Overview

This and the next assignment are desiged to give some familiarity w ith relational databases and SQL. As this course is introductory, for our assignments, w e w ill not take the time to set up a local relational database server or connect to a remote database. There are a plethora of options w hen it comes to that sort of set-up. The particulars are quite particular and w ill be best learned w hen/if you actually need to connect to and use a database.

Warning: Many of the set-up issues not discussed here are "the hardest" part of using a new database or environment. In the last assignment, w e used the analogy of a "combination lock." In many cases, environment set-up and connecting to servers etc. is one of those tougher combination locks to bypass. How ever, once you're through, you should be all set.
These lessons are about how to deal w ith and think about relational databases once you have access. Next w eek, the lectures w ill go through the process of connecting to a local database. While that experience is excellent, it w ill not be a prerequisite for assesment.

This lesson is broken dow n into tw o parts: Visualizing and understanding the relationships betw een tables in a database, and learning a few basic SQL commands.
There are many relational databases. How ever, the few different flavors of SQL are some of the most popular; thus its inclusion here.

Finally, these SQL lessons are designed to primarily highlight retreival of data from SQL databases. SQL provides plenty of functionality beyond simple retrieval. How ever, the aim here is to:

1. Provide enough SQL know ledge to enable data retreival.
2. Start to introduce and test some data organization / manipulation principles.

These w ill enable you as a data-scientist. Becoming expert in SQL w ill enable you in other w ays, but our conceren here is *how to retreive data*. Not how to manage a database or become an SQL w izard.

Other fundamental SQL commands, e.g. `INSERT` w ill be show n, but not dw elt upon.

NB:

One of the draw backs of not going through the process of setting up a connection w ith a SQL server is that your ability to create SQL queries w ill not be directly evaluated. It is highly suggested that you practice creating and executing SQL queries.

Activities in this Assignment

- Think through database schemas
- Introduction to basic SQL commands:
  - Select
  - Where
  - Like
  - Sorting
  - Count, Sum, Avg, Min, Max

A Quick Note on Python SQL Integration

A few options w ill be covered next w eek during lecture, but it is useful to know that the `pyodbc` and `psycopg2` packages are standards for integrating SQL w ith `Python` .

Second, Jupyter has functions called cell magic/s [sic] w hich integrate w ith SQL. Here is an excellent article on their use.

Visualizing Relationships Between Tables in a Relational Database

As its name implies, `Relational` databases rely upon relationships betw een various tables. The below gives the first few row s of the four tables in a theoretical database. (Assume each table contains many many many more row s.) While "schema" has many different *particular* uses depending database system in use, the below offers a kind of schema in that it exposes tables available, as w ell as primary keys (pk) for the tables.

Pilots

Pilot ID (pk)	Name	Airline
2367	B.Smith	Delta
6678	N.Roy	American
2121	P.Mitchell	JetBlue

Trips

Itinerary ID (pk)	Pass. ID	Pilot ID	Depart. Loc.	Arrival Loc
001	1818	6678	DFW	JFK
002	1818	2121	ORD	ATL
003	1738	2367	SEA	SFO

Passengers

Passenger ID (pk)	Name
1738	L.Jones
1945	Y.Landry
1818	V.Miller

Airports

Airport (pk)	City	State	Airline Hub
DFW	Dallas	TX	American
JFK	NYC	NY	JetBlue
ORD	Chicago	IL	United
ATL	Atlanta	GA	Delta
SFO	San Fran.	CA	United

Question 1

```
### GRADED
### In a given table the primary key must be:

### 'a') all non-repeated values
### 'b') integers
### 'c') used to connect to other tables
### 'd') a primary key is not needed for all tables
### Assign the string associated with your choice to ans1

### YOUR ANSWER BELOW

### A Primary key MAY be used to connect to other tables, but will not always

ans1 = 'a'
```

The next few questions will rely upon the diagram above Question 1.

Question 2

```
### GRADED
### If the "Passengers" Table were to be connected to the "Trips" table, which column in "Passengers" should
### be used as the "Foreign Key". E.g. which column in Passengers will allow it to be related to "Trips"

### 'a') Passenter ID
### 'b') Name
### Assign the string-letter associated with your choice to ans1

### YOUR ANSWER BELOW


ans1 = 'a'
```

Question 3

```
### GRADED
### True of False:
### A key exists which would allow the connection of "Airports" and "Trips" tables
### Assign your choice, as a boolean, to ans1

### YOUR ANSWER BELOW

### the "Airport" column could be connected to either the
### Dep location or Arr Location of "Trips"
ans1 = True
```

Question 4

```
### GRADED
### True or False:
### A key exists which would allow the direct connection of the "Pilots" and "Airports" tables
### Assign your choice, as a boolean, to ans1

### YOUR ANSWER BELOW
### While it looks as if "Airline" and "Airline Hub" might provide
### an opportunity to connect the tables, there will be multiple values
### associated with both.
ans1 = False
```

Question 5

```
### GRADED
### What is the name of the passenger responsible for itinerary 001?

### 'a') L.Jones
### 'b') Y.Landry
### 'c') V.Miller
### Assign the string associated with your choice to ans1

### YOUR ANSWER BELOW

ans1 = 'c'
```

Question 6

```
### GRADED
### True or False:
### According to our Data,
### The airline responsible for the trip from Chicago to Atlanta has hubs at one or both of those locations
### Assign your choice, as a boolean,  to ans1

### YOUR ANSWER BELOW

ans1 = False
```

Question 7

```
### GRADED
### Which pilot flew L.Jones?

### 'a') B.Smith
### 'b') N.Roy
### 'c') P.Mitchell
### Assign the string associated with your choice to ans1

### YOUR ANSWER BELOW

ans1 = 'a'
```

Question 8

```
### GRADED
### Which airline has a hub at L.Jones' Destination?

### 'a') American
### 'b') JetBlue
### 'c') United
### 'd') Delta
### Assign string associated with your choice to ans1

### YOUR ANSWER BELOW

ans1 = 'c'
```

The above exercises w ere designed to give the opportunity to think about the steps you need to undertake to find an answ er.

- What information am I starting w ith?
- What information do I desire?
- What connects those pieces of information?

Only having 14 row s of data to search through makes this task trivial. When 1,000's, 100,000's or 1,000,000's or row s become involved, computers can help.

Basic SQL Commands -> Rows and Columns

While it is possible to accomplish the above tasks all in one SQL command, before w orking on joining data from different tables, w e w ill cover how to select row s and columns from single tables in a database. The lectures next w eek w ill cover this content, so this is something of a preview.
Hint: The questions w ill provide many clues, but finding some answ ers may require a little research!

For our w ork w e w ill be using the test interface exposed by w 3 Schools to practice our commands.

SELECT [col] FROM [table]

Opening up the w 3 SQL interface, the right side features the title **Your Database:** and a list of tables under "Tablename". Click on one of those "tablename"s and you w ill see the code in the **SQL Statement:** box change to `SELECT * FROM [<Tablename>]`, and under **Result:** you should see an n x p table of those records.
NB: Square Brackets around Table name are not needed.

Question 9

```
### GRADED
### In instance of `SELECT * FROM [<Tablename>]`;
### The * tells SQL to:
```

```
### 'a') include all rows
### 'b') include all columns
### 'c') include all tables
### 'd') include all databases
### Assign string associated with your choice to ans1

### YOUR ANSWER BELOW

ans1 = 'b'
```

Question 10

```
### GRADED
### Creating a SQL query: "SELECT ____ FROM Customers"
### Which of the following would NOT be appropariate to fill in the blank:

### 'a') CustomerID
### 'b') Postal Code
### 'c') City
### 'd') Country
### Assign string associated with your choice to ans1

### YOUR ANSWER BELOW

### Note the extra space in "Postal Code"
ans1 = 'b'
```

Question 11

```
### GRADED
### Investigate what happens when the command "SELECT * FROM Categories, Shippers" is run.

### Suppose the database added a table called Returns with 22 records.
### How many records would be returned from the call "SELECT * FROM Employees, Returns"?
### Assign int corresponding to number of records returned to ans1

### YOUR ANSWER BELOW
### 22 * 10 = 220
ans1 = 220
```

Question 12

```
### GRADED
### True or False:
### SQL functions are case-sensitive and MUST be upper-case
### Assign boolean answer to ans1

### YOUR ANSWER BELOW

ans1 = False
```

Question 13

```
### GRADED
### When requesting multiple columns, what character separates the column names?
### Assign string of a single character to ans1 for answer

### YOUR ANSWER BELOW

ans1 = ','
```

With the previous questions answ ered, you should be comfortable making a SQL query that returns a table w ith some subset of its columns. Before moving onto record selection --

The `LIMIT` Command is useful for limiting your output to a certain number of row s. Particularly if a query could return 1,000s of records, `LIMIT` can help give you a preview to ensure your query is w orking correctly.

E.G.
`SELECT * FROM Customers LIMIT 20`
Will give you the first 20 row s from Customers.

Where

With our potentially 1,000[,000]s of records, w e need the capability to specify w hich records w e need. The **`WHERE`** command enables the use of conditionals.

Question 14

```
### GRADED
### If a conditional were created in Python, which data-type should it evaluate to?

### 'a') Float
### 'b') Dictionary
### 'c') Int
### 'd') Boolean
### Assign string-character associated with answer to ans1

### YOUR ANSWER BELOW

ans1 = 'd'
```

Question 15

```
### GRADED
### A Query starts `SELECT * FROM Categories ... `
### Which "WHERE" statement will return records where CategoryID is greater than 5
### Note: Question not meant to trick you, but meant to be tricky -- Remember, test things out!

### 'a') WHERE CID > 5
### 'b') Where categoryid >5 IN Categories
### 'c') where CategoryID >5
### 'd') WHERE catID >5
### Assign string associated with answer to ans1

### YOUR ANSWER BELOW

ans1 = 'c'
```

`WHERE` statements can be expanded by using `OR` and `AND`, w hich operate as logical 'and' and 'or' operators.

Question 16

```
### GRADED
### A query starts `SELECT * FROM Categories`
### Which "WHERE" statement will return records that have:
### ### CategoryID Greater than 6 or less than 3 (exclusive)

### 'a') WHERE CategoryID > 3 OR CategoryID < 6
### 'b') Where CategoryID < 3 OR >6
### 'c') WHERE CategoryID < 3 OR CategoryID > 6
### Assign string associated with answer to ans1

### YOUR ANSWER BELOW

### 'a' is incorrect because the comparators are
### facing the wrong directions.
ans1 = 'c'
```

Question 17

```
### GRADED
### A query starts `SELECT * FROM Categories`
### Which "WHERE" statement returns records where CategoryID is between 2 and 6 (exclusive)

### 'a') WHERE CategoryID >2 AND CategoryID <6
### 'b') WHERE 2< CategoryID <6
### 'c') WHERE CategoryID <=6 AND CategoryID >2
### Assign string associated with answer to ans1

### YOUR ANSWER BELOW

ans1 = 'a'
```

`WHERE` statements can also utilize the "logical not" by invoking `NOT`.

Question 18

```
### GRADED
### A query starts `SELECT * FROM Categories`
### Which "WHERE statement returns records where CategoryID is not equal to 4

### 'a') WHERE CategoryID Not = 4
### 'b') WHERE NOT CategoryID = 4
### 'c') NOT WHERE CategoryID = 4
### Assign string associated with your choice to ans1

### YOUR ANSWER BELOW

ans1 = 'b'
```

Question 19

```
### GRADED
### The above exercise could be accomplished by changing the logical operator between CategoryID and 4

### e.g. SELECT * FROM Categories WHERE CategoryID <alternative operator> 4

### Assign the alternative logical operator to ans1 as a string

### YOUR ANSWER BELOW

ans1 = '!='
```

Question 20

The next few questions are about logical and/or and not.

Only boolean answ ers w ill be accepted. Strings, e.g. `"True"` w ill NOT be counted as correct.

```
### GRADED

### True AND False evaluates to:
```

```
### Assign boolean answer to ans1
### YOUR ANSWER BELOW

ans1 = False
```

Question 21

```
### GRADED

### False OR True evaluates to:

### Assign boolean answer to ans1
### YOUR ANSWER HERE

ans1 = True
```

LIKE

While `WHERE` support strict conditionals, `LIKE` allow s `WHERE` to matches to be made non-explicitly using the `<%>` as a w ildcard.

Question 22

```
### GRADED
### True or False:

### "WHERE <column> = 4"
### returns something different than
### "WHERE <column> LIKE 4"

### Assign boolean answer to ans1
### YOUR ANSWER HERE

ans1 = False
```

Question 23

```
### GRADED
### suppose a column - named "History" - in a SQL database contains a short personal history.
### What would be the command to find records that contain the word "college" *anywhere* in that personal history?

### 'a') WHERE History = "college"
### 'b') WHERE History = "%college"
### 'c') WHERE History LIKE "%college"
### 'd') WHERE History CONTAINS "college"
### 'e') WHERE History LIKE "%college%"
### Assign string associated with selection to ans1

### YOUR ANSWER HERE

### The "%" On either side are needed to ensure that "college"
### can be found at any location within the string column
ans1 = 'e'
```

Sorting and Summarizing

Like using the `LIMIT` command, it may be useful to sort and summarize data to develop a sense of w hat information a table / database contains.
NB: More advanced techniques for sorting and summarizing data w ill be covered later in `Python` focused assignments.

ORDER BY [col] [ASC/DESC]

The follow ing questions w ill test the creation of a SQL query that uses "ORDER BY".

Eventually, the query must return the columns "OrderDetailID" and "ProductID" From the "OrderDetails" Table. The record w ith the largest number associated w ith "Quantity" Should be returned first, and the record w ith the smallest number associated w ith "Quantity" should be returned last. How ever, the "Quantity" column should NOT be returned

Question 24

```
### GRADED

### Starting our query:

### Fill in the blank: "SELECT ________ FROM"
### assign string to ans1

### YOUR ANSWER HERE

ans1 = "OrderDetailID, ProductID"
```

Question 25

```
### GRADED
### Continuing from above:

### Fill in the blank: "SELECT <ans1> FROM ______"

### assign string to ans2.
### Do not include square-brackets ###

### NB:
```

```
### After filling in the blank, "SELECT <ans1> FROM <ans2>"
### should return the desired columns from the desired table in no particular order.



### YOUR ANSWER HERE

ans2 = "OrderDetails"
```

Question 26

```
### GRADED
### Finishing the Query:

### Fill in the blank "SELECT <ans1> FROM <ans2> __________


### After filling in ans3, the query "SELECT <ans1> FROM <ans2> <ans3>"
### should be complete,and function as described before Q25

### Assign answer string to ans3

### YOUR ANSWER HERE

ans3 = "ORDER BY Quantity DESC"
```

Summary functions

`COUNT` , `SUM` , `AVG` , `MIN` , and `MAX` are SQL functions that allow for the return of summary statistics from (primarily numerical) columns.

Try: `SELECT COUNT(*) FROM OrderDetails WHERE Quantity >10`

and : `SELECT AVG(Quantity), MAX(Quantity), MIN(Quantity) FROM OrderDetails WHERE Quantity >101`

and: `SELECT COUNT(ContactName) FROM Customers`

As you can see, `COUNT` can be called on either numeric or string columns.

Question 27

```
### GRADED
### What is the maximum productID in OrderDetails?
### Assign int answer to ans1.

### YOUR ANSWER HERE
### "SELECT MAX(ProductID) FROM OrderDetails"
ans1 = 77
```

Question 28

```
### GRADED
### Are there more customers in the Customers table from Germany or Mexico?

### 'a') Germany
### 'b') Mexico
### 'c') Same from each
### Assign string associated with answer to ans1

### YOUR ANSWER HERE
### "SELECT COUNT(*) FROM CUSTOMERS WHERE Country = "Germany"
### "SELECT COUNT(*) FROM CUSTOMERS WHERE Country = "Mexico"

ans1 = 'a'
```

Question 29

```
### GRADED
### What is the mean quantity of products ordered where ProductID is less than 40?
### Use table "OrderDetails"

### Assign float to ans1. Round to one decimal
### YOUR ANSWER HERE

### "SELECT AVG(Quantity) FROM OrderDetails WHERE ProductID <40
ans1 = 25.1
```

Editing Databases with SQL

For reference, below are a few commands that can be used to edit and update databases using SQL queries. These will be covered in lecture next week, but they will not be assesed in assignments.

INSERT

`INSERT INTO <tablename> VALUES (<val1>,<val2>,...,<valn>)`

Of note: the value of the primary key MUST BE UNIQUE, other wise that insert command will throw an error.
If a value is not known and you want to leave it blank, substitute "NULL" for that value. Otherwise, the term in parentheses must contain the same number of values as the table has columns.

DELETE

`DELETE FROM <tablename> WHERE <conditional or "LIKE" statement>`

The conditional / like statement may reference any column. Primary key or otherw ise:

UPDATE

`UPDATE <tablename> SET <colname> = <newValue> WHERE <conditional or "LIKE" statement>`

Coming Up…

Next lesson w ill cover additional advanced SQL and data manipulation topics, such as grouping and joining.

Comfort w ith the syntax and proceedures here w ill enable use and understanding of those more advanced topics.