Week 2
SQL queries and summarizing data

Introduction to SQL Queries

A database has tables, columns (also called fields) and rows. Tables and columns have names, rows have data. The value of a database is that it allows you to find data, group data, link data, and summarize data. For small data sets summaries may be easier made by hand, but with large data sets that are constantly changing, this is easier done with queries. SQL (Structured Query Language) is a simple language that is used to query database, and it can be used in MySQL, Oracle, Access, and many others.

The syntax of SQL queries is designed to be similar to a command in English. Most queries have at least two parts: a verb clause, where the verb is usually “select”, and a prepositional clause that tells from which table (or tables) data should be selected. The modifying clause puts conditions on the selection or the results. Several modifying clauses can be used, or none at all. The query above would select all columns (or fields) from the table “pbdb” for those rows where the Genus column contains the word “Hesperocyon” (an extinct dog-like genus).

The Select Clause

An asterisk (*) is used to select all columns in a table. If you only want to see specific columns, list them by name:

```
SELECT Genus, species, Locality, Age FROM pbdb;
```

If you just want a list of the taxa in your table, you can use the DISTINCT modifier to return only a list of the unique items in a field:

```
SELECT DISTINCT Genus FROM pbdb;
```

You can combine words from different fields using the CONCAT() function, which is useful here for combining genus and species names:

```
SELECT CONCAT(Genus, ' ', species) FROM pbdb;
```

In cases like this it is helpful to rename the fields, which you can do with the AS clause:

```
SELECT CONCAT(Genus, ' ', species) AS Taxon FROM pbdb;
```
There are other functions that operate on the data you retrieve:

- `Max(field_name)` returns the maximum value of a field.
- `Min(field_name)` returns minimum value
- `Count(field_name)` returns the number of rows
- `Avg(field_name)` returns the mean
- `Std(field_name)` returns the population standard deviation
- `Variance(field_name)` returns the population variance

The Where Clause

Use the WHERE clause to select parts of data:

```
SELECT * FROM pbdb WHERE Genus='Canis';
```

You can combine items in the WHERE clause using all the logic statements that you thought would be useless back when you learned them:

```
SELECT * FROM pbdb WHERE Genus='Canis' AND Continent='North America';
```

```
SELECT * FROM pbdb WHERE Genus='Canis' OR Genus='Felis';
```

```
SELECT * FROM pbdb WHERE Age > 65 AND Age < 55;
```

Order By Clause

Want results in alphabetical order? Add an ORDER BY clause.

```
SELECT DISTINCT Genus FROM pbdb ORDER BY Genus;
```

The Group By Clause

Here’s where the usefulness of a database really becomes apparent. If you’re clever and your data are arranged right you can calculate an entire table for a publication with one query:

```
SELECT Genus, Count(ID) AS N FROM pbdb GROUP BY Genus;
```

The previous query groups the data by genus and counts the rows for each genus, which gives you the sample size for that genus, and renames the count as “N”.

The following query gives you a summary of the stratigraphic ranges of all the genera in your database, ordered from the first appearance in the fossil record.

```
SELECT Genus, Count(ID) AS N, Max(AgeMidpoint) AS FAD, Min(AgeMidpoint) AS LAD
FROM pbdb GROUP BY Genus ORDER BY FAD;
```

The resulting table is very similar to the “ranges” table produced by the PBDB. If you downloaded the right data in the collection database you can recreate the ranges table exactly using just one SQL query. (oh, and you can put queries on more than one line if they are getting long).
Exporting your results

If you want to save the results you generated in the Query Browser, you can copy and paste them into Excel, or you can use the File menu to export the results as a comma-delimited file, which you can then import into Excel (the latter works better for large data sets).

Assignment for next week

Summarize your data

Use SQL and phpMyAdmin to summarize your data. Compile the following information and summarize it neatly, perhaps putting some of the information in a nicely formatted table:

1. Name of your group
2. Total number of occurrences in your database table
3. Number of distinct species
4. Number of distinct genera
5. Number of distinct families
6. Continents represented in your database
7. Number of distinct localities (i.e., distinct latitude and longitude combinations)
8. Using one age variable (e.g., median age), the maximum and minimum age of the occurrences in your data set
9. Number of distinct ages

Reading (available in OnCourse)
