Using SQL

- Everyone set up to do assignment 8?
- Able to run sql?
- Anyone need access to SPH virtual computer lab?

Basic process

- This is a small database, so you should be able to answer the questions manually without writing the SQL queries.
  - Practice on excel
- You might want to think of this assignment as writing the correct SQL queries to get the same results as your manual answer, noting that you could have made a mistake in your manual answer.
- So if your SQL query result is not matching your manual answer, think about why and figure out which was wrong.
- Note that conversely, even if you get the exact same answer as your manual answer the SQL query might not be fully correct.
  - That is, given the particular DB, you got the correct answer, but for a different DB it might not be correct.
Try it: Using the handout excel sheet from last week

- Fill out the gforms
  - https://forms.gle/UGRxo4XZsGuBwtnW9
- How many patients visited for vomiting?
- Who (what are their names) visited for vomiting?
- Which tables did you use to get this information?
- How did you combine the tables? That is, what columns did you use to combine the tables?
### Hand out: visits table

<table>
<thead>
<tr>
<th>PatientID</th>
<th>Date</th>
<th>Time</th>
<th>Reason</th>
<th>Diagnosis</th>
<th>NextAppt</th>
</tr>
</thead>
<tbody>
<tr>
<td>16577623</td>
<td>12/4/2018</td>
<td>9:00 AM</td>
<td>Vomiting</td>
<td>Stomach Virus</td>
<td>12/15/2018</td>
</tr>
<tr>
<td>16577660</td>
<td>1/28/2019</td>
<td>9:30 AM</td>
<td>Irritated Eye</td>
<td>Pink-Eye</td>
<td>-</td>
</tr>
<tr>
<td>16577665</td>
<td>12/5/2018</td>
<td>8:00 AM</td>
<td>Large Zit</td>
<td>MRSA</td>
<td>1/2/2019</td>
</tr>
<tr>
<td>16577615</td>
<td>12/18/2018</td>
<td>10:00 AM</td>
<td>Chills</td>
<td>Flu</td>
<td>12/28/2018</td>
</tr>
<tr>
<td>16577620</td>
<td>12/12/2018</td>
<td>1:30 PM</td>
<td>Headache</td>
<td>Headache</td>
<td>-</td>
</tr>
<tr>
<td>16577655</td>
<td>1/8/2019</td>
<td>12:00 PM</td>
<td>Fever</td>
<td>Sinus Infection</td>
<td>1/15/2019</td>
</tr>
<tr>
<td>16577690</td>
<td>1/9/2019</td>
<td>3:00 PM</td>
<td>Rash</td>
<td>Poison-Ivy</td>
<td>-</td>
</tr>
<tr>
<td>16577685</td>
<td>1/2/2019</td>
<td>8:30 AM</td>
<td>Follow-up</td>
<td>MRSA</td>
<td>2/15/2019</td>
</tr>
<tr>
<td>16577635</td>
<td>1/20/2019</td>
<td>12:30 PM</td>
<td>Abdominal Pain</td>
<td>Constipation</td>
<td>-</td>
</tr>
<tr>
<td>16577661</td>
<td>1/3/2019</td>
<td>1:00 PM</td>
<td>Lower Abdominal Pain</td>
<td>Urinary Tract Infection</td>
<td>-</td>
</tr>
<tr>
<td>16577645</td>
<td>12/19/2018</td>
<td>11:15 PM</td>
<td>Sore Throat</td>
<td>Streptococcal Pharyngitis</td>
<td>-</td>
</tr>
</tbody>
</table>
Hand out: join patients and visits table to get name

<table>
<thead>
<tr>
<th>PatientName</th>
<th>PatientID</th>
<th>Date</th>
<th>Time</th>
<th>Reason</th>
<th>Diagnosis</th>
<th>NextAppt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allison Bellame</td>
<td>16577623</td>
<td>12/4/2018</td>
<td>9:00 AM</td>
<td>Vomiting</td>
<td>Stomach Virus</td>
<td>12/15/2018</td>
</tr>
</tbody>
</table>

- SELECT patients.PatientName, visits.*
- FROM patients, visits
- WHERE patients.PatientID = visits.PatientID and Reason = “Vomiting”

Using SQL software and loading kum db?
CAUTION

- If you changed the database by accident, stop and exit from everything. And start over so that you are using the correct database.
- Remember to delete kum.db and upload again.
- Open database Read Only...

Take Away I
SQL - Structured Query Language

- Every statement yields a table of values as output
  - Sometimes there's only one row in the table!

<table>
<thead>
<tr>
<th>Keyword</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>select</td>
<td>columns and/or expressions</td>
</tr>
<tr>
<td>from</td>
<td>Tables</td>
</tr>
<tr>
<td>where</td>
<td>conditions on the rows</td>
</tr>
<tr>
<td>group by</td>
<td>group rows together</td>
</tr>
<tr>
<td>order by</td>
<td>order the rows</td>
</tr>
<tr>
<td>;</td>
<td></td>
</tr>
</tbody>
</table>
Are you comfortable?

- Specifying what variables (columns)?
- Specifying what rows (conditionals)?
- Understand variables types? What is the difference below?
  - SELECT * FROM patients where race='H';
  - SELECT * FROM patients where race='H';
  - SELECT * FROM patients where race='h';

Today: Outline

- Sort: ORDER BY [DESC]
- Calculate: AS
- Aggregates: Group by
- Combining multiple tables: JOIN
- Saving queries: Views
- Do the assignment
Sorting

- Can sort output by contents of a column
  - sort in ascending or descending order
  - sort by more than one column (second one breaks ties)
- Sort patients by last name
  - SELECT * FROM patients
  - ORDER BY lname DESC;
- TRY: Who are our 5 youngest patients?
  - Look at the table and answer
  - Now, what is the sql?
- Who are our 5 oldest patients?

Compute Columns

- Find discount amount
  - SELECT patientID, billed, covered, (billed-covered) AS discount
  - FROM payments;
- Nice names for output columns
  - Name following computed column (e.g., discount) will be used to name output column
- Find total paid amount
  - Total = copay+pat_pd+insur_pd

SELECT Column1, Column2
FROM Table
ORDER BY Column1 [DESC];

SELECT calculate AS NewColumnName
FROM Table;

--EXAMPLE;
SELECT (billed-covered) AS discount
FROM Table;
Grouping and Aggregates (multiple rows)

- Can make calculations on groups of rows
  - sum, avg, max, min, count
- Each different value for the GROUP BY fields defines a new group
  - One row of output is produced for each group
  - Several rows of input table may belong to same group. They are aggregated using aggregation operator.

```
SELECT f(Column2) AS ColumnName
FROM Table
GROUP BY Column1;
```

---

Grouping and Aggregates

- Can make calculations on groups of rows
  - sum, avg, max, min, count
- How many charges by bill type
  - SELECT count(charge) as ncharges
  - FROM charges
  - GROUP BY billtype;
- TRY: What is total billed by patient?

```
SELECT f(Column2) AS ColumnName
FROM Table
GROUP BY Column1;
```

---
Table Operations

- **Aggregate columns**: col1 \( \text{op} \) col2 \( \text{AS} \) col3

  \[
  \begin{array}{ccc}
  \text{col1} & \text{col2} \\
  \text{a} & \text{d} \\
  \text{b} & \text{e} \\
  \text{c} & \text{f} \\
  \end{array}
  \rightarrow
  \begin{array}{ccc}
  \text{col1} & \text{col2} & \text{col3} \\
  \text{a} & \text{d} & \text{a}+\text{d} \\
  \text{b} & \text{e} & \text{b}+\text{e} \\
  \text{c} & \text{f} & \text{c}+\text{f} \\
  \end{array}
  \]

- **Aggregate rows**: **Group BY**

  \[
  \begin{array}{c}
  \text{A} \\
  \text{B} \\
  \text{C} \\
  \end{array}
  \rightarrow
  \begin{array}{c}
  \text{D} \\
  \end{array}
  \]
  Where \( D=\text{function}(A,B,C) \)
  Examples of function are
  \( \text{Sum}(A,B,C) \), \( \text{Avg}(A,B,C) \), \( \text{Max}(A,B,C) \), \( \text{Min}(A,B,C) \), \( \text{Count}(A,B,C) \)

Joins

- Combine rows from one table with rows from another
- Usually join on some common column
  - Don’t combine rows unless their value in the common column is the same
  - WHERE clause says the common column must be same in each table
- Produce a list of bills for all patients with their name
  - SELECT patients.fname, patients.lname, patients.patientID, bills.
    - FROM patients, payments
    - WHERE patients.patientID=payments.patientID;
Different Syntax: Joins

```sql
SELECT Table1.Column1, Table2.Column2
FROM Table1, Table2
WHERE Table1.Column=Table2.Column;

-- Does same thing;
SELECT Table1.Column1, Table2.Column2
FROM Table1
JOIN Table2
ON Table1.Column=Table2.Column;
```

Different SQL JOINs

- **INNER JOIN**: Returns all rows when there is at least one match in **BOTH tables**
- **LEFT JOIN**: Return **all rows from the left table**, and the matched rows from the right table
- **RIGHT JOIN**: Return **all rows from the right table**, and the matched rows from the left table
- **FULL JOIN**: Return all rows when there is a match in **ONE of the tables**
Views: Permanent Queries

- Looks and feels like a table
- Saved queries
- Virtual table: not a real table in the DB
- Can treat it like a real table, as if it exists

Create View

```sql
CREATE VIEW panel AS
SELECT
    providers.fname as dr_first, providers.lname as dr_last,
    patients.fname, patients.lname
FROM providers, patients
WHERE providers.providerID=patients.primary_dr;
```
Take Away I
SQL - Structured Query Language

- Every statement yields a table of values as output
  - Sometimes there's only one row in the table!

<table>
<thead>
<tr>
<th>Keyword</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>select</td>
<td>columns and/or expressions</td>
</tr>
<tr>
<td>from</td>
<td>Tables</td>
</tr>
<tr>
<td>where</td>
<td>conditions on the rows</td>
</tr>
<tr>
<td>group by</td>
<td>group rows together</td>
</tr>
<tr>
<td>order by</td>
<td>order the rows</td>
</tr>
<tr>
<td>;</td>
<td></td>
</tr>
</tbody>
</table>

Take Away II
Table Operations

- Aggregate columns: col1 op col2 AS col3

\[
\begin{array}{ccc}
\text{col1} & \text{col2} & \text{col3} \\
\text{a} & \text{d} & \text{a+d} \\
\text{b} & \text{e} & \text{b+e} \\
\text{c} & \text{f} & \text{c+f} \\
\end{array}
\]

- Aggregate rows: Group BY

Where D=function(A,B,C)
Examples of function are
\[
\begin{align*}
\text{Sum}(A,B,C) & \quad \text{Avg}(A,B,C) \\
\text{Max}(A,B,C) & \quad \text{Min}(A,B,C) \\
\text{Count}(A,B,C) & \\
\end{align*}
\]
Take Away III
SQL - Structured Query Language

- Join multiple tables
- Save as View, then reuse

Take Away II
Boolean Logic: Truth Tables (1=T; 0=F)

- WHERE gender='Male' AND age>18;
- WHERE race='Hispanic' OR race='Black';

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>NOT</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Take Away III
Variable Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Stored value</th>
<th>Interpreted value</th>
<th>Label Interpreted Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>1000001 (65)</td>
<td>65</td>
<td>65 or older</td>
</tr>
<tr>
<td>Char/string (ASCII)</td>
<td>1000001 (65)</td>
<td>A</td>
<td>Asian</td>
</tr>
<tr>
<td>date</td>
<td>1000001 (65)</td>
<td>1960/3/6 (SAS)</td>
<td></td>
</tr>
</tbody>
</table>

- 1 0 0 0 0 0 1 = 64 + 1 = 65
- 64 32 16 8 4 2 1

Understand variables types? What is the difference below?
- SELECT * FROM patients where race='H';
- SELECT * FROM patients where race='h';

Raise your hand when you are back
GROUP PRESENTATION
GROUP 6: PEER REVIEW

Next week
- FOCUS on learning SQL. Will be on Final.
- Good for resume and real job skills
- Assignment 8 due: We may not be available on the weekend for help
- Class wrap up

Office hours:
- Dr. Kum, Wednesday, 3 to 5 (zoom: pinformatics)
- Michelle, Thursday, 3:30 to 5:30 (zoom: 606-413-8933)