**DSBA 6160 – PROBLEM SET 1 (Due 9/12/2019 at the beginning of class)**

**Instructions:** Use this document to capture your queries for ***each*** of the steps below. There is no need to provide your results unless we explicitly ask for them. Submit your completed Word document in Canvas.

**Problem #1 – Integer Division and CAST (20 points)**

The Intro to SQL for Data Science in DataCamp has introduced the concept of what happens with integer division (4/3 = 1) which is highly problematic and a common bug – even in big banks. However the fix shown in DataCamp (adding decimal points to operands – 4.0/3.0) isn’t realistic in practice as values are stored in fields in a table. What do we do when we have IntegerField1 / IntegerField2? Worse yet, what if we were to sum this calculation up across an entire table?

For this, we need to introduce the concept of CAST which is a way to transform data in one data type to another. You’ll need to do some research to solve this problem in SQLite:

<https://www.sqlite.org/datatype3.html>

**Note:** SQLite’s data types are very general compared to other database management systems which should make this a *little* easier.

Load the **Chinook** dataset into DBBrowser for SQLite. You’ll find it under the Week 2 module in Canvas.

* **Step 1:** Select all fields for the top 10 rows from the tracks table, ordered by TrackId. Note that each track has a particular length (in milliseconds) and storage size (in Bytes). Provide your query below.

﻿

SELECT \*

FROM tracks

ORDER BY TrackId

LIMIT 10

* **Step 2:** Calculate Bytes/Millisecond for Track 1 using literal values in SQL (i.e., you should not use the Milliseconds and Bytes field names in your SQL statement here). Apply the fix so we aren’t performing integer division. What value is returned when you do run this query both before and adding a decimal place to the operands? Provide your query AND the value below.

﻿SELECT 11170334/343719

FROM tracks

WHERE TrackId =1

=32

﻿SELECT 11170334.00/343719.00

FROM tracks

WHERE TrackId =1

=32.4984478600252

* **Step 3:** Calculate Bytes/Millisecond for the first 10 tracks of the table and alias it BytesPerMillisecond, ordered by TrackId. Note the decimal portion of each calculation is truncated. Use the link above to SQLite.org and adjust your query to avoid this truncation. Provide your query below.

﻿SELECT (Bytes\*1.00)/Milliseconds

FROM tracks

WHERE TrackId <=10

order by TrackId

* **Step 4:** Provide a query that displays the maximum Bytes/Millisecond across all tracks in the table.

﻿SELECT MAX((Bytes\*1.00)/Milliseconds)

FROM tracks

* **Step 5:** Identify the track, album and artist that has the largest Bytes/Millisecond. You don’t need to write SQL to do this or provide a query. You’ll learn to do this over the next week. For now, just provide the track name, album name and artist name.

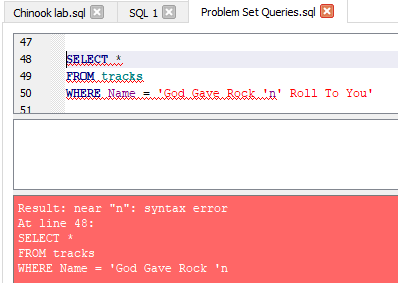
Better Halves, Heroes Season 1, Heroes

**Problem #2 – ESCAPE Sequences (20 points)**

Sometimes you need to find data which isn’t so easy to type into a filter condition. Take the following song written by the band Kiss which I really want to query from the tracks table:

God Gave Rock 'n' Roll To You

If I query the tracks table like this in DB Browser for SQLite, I get an error:



Can you figure out how to fix it? Load the **Chinook** dataset into DBBrowser for SQLite. You’ll find it under the Week 2 module in Canvas.

* **Step 1:** Query all fields for this track from the tracks table correctly based upon Name. Provide your final query below.

﻿SELECT \*

FROM tracks

WHERE name= "God Gave Rock 'n' Roll To You"

Hint: use the following link to learn at least one method for escaping single quotes. Note this is geared more towards Oracle but escaping rules are relatively consistent across database systems:

<https://www.databasestar.com/sql-escape-single-quote/>

* **Step 2:** Write a query to return ***all*** songs in the tracks table that contain the word ‘n’ in them. The song above is not the only one! Provide your query below.

**﻿**SELECT \*

FROM tracks

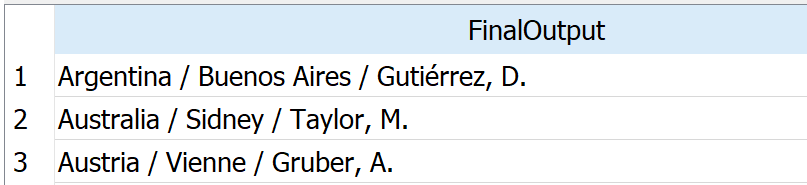
WHERE name LIKE "% 'n' %"

**Problem #3 – Playing with Strings (20 points)**

Marketing is asking for a list of customers, but they’re **very** particular in how they want the output formatted. They want all customers provided in the following format, ordered by Country, City, Last Name and First Name.

<Country> / <City> / <LastName>, <First letter of FirstName>.

Below are the first three rows of the output:



Use the following resource to provide a query that will produce the output above for all customers. Note, that this material has not been covered in class however this is a skill you’ll need to master – the ability to learn how to leverage functions you don’t know.

<https://www.sqlitetutorial.net/sqlite-string-functions/>

To be clear, your resultset should consist of multiple rows containing only one field named FinalOutput. This field should contain the full string for each row as shown above. Please provide your final query below.

﻿﻿SELECT Country || ' / ' || City || ' / '||LastName || ', ' || substr(FirstName,1,1) || '.' as FinalOutput

FROM customers

ORDER BY Country

**Problem #4 – Using OFFSET with LIMIT (20 points)**

Chinook, Inc, a music publisher, is celebrating their early success by awarding gift cards to the first 5 customers who ordered a track starting with their 100th invoice. The marketing department asks you to identify who these five customers are along with their names and addresses so they can send them the gift cards.

Load the **Chinook** dataset into DBBrowser for SQLite. You’ll find it under the Week 2 module in Canvas.

1. **Step 1:** Identify the 5 qualifying invoices and their associated customers. Use the link below to research the appropriate way to identify the five invoices. Provide your query below.

﻿SELECT \*

FROM invoices

LIMIT 5 OFFSET 99

Note: pay special attention to the discussion around LIMIT and ORDER BY

<https://www.sqlitetutorial.net/sqlite-limit/>

1. **Step 2:** Use the customers identified from Step 1 to pull the proper customer details for marketing. Note that Chinook’s customers are international so don’t forget to provide Country. Provide your final query.

﻿SELECT Country || ' / ' || City || ' / '||LastName || ', ' || substr(FirstName,1,1) || '.' as FinalOutput

FROM customers

WHERE CustomerId IN (5,9,15,24,38)

ORDER BY Country

**﻿**

**Problem #5 – Joining Data (20 points)**

Load the **IMDB data** dataset into DBBrowser for SQLite. You’ll find it under the Week 2 module in Canvas.

Single-table queries are useful for learning SQL and figuring things out one piece at a time but in the real-world queries typically require data from multiple tables to answer key business questions. You may not always have time to draft a pretty report for your business partners either – you may need to produce your results and report on them immediately using SQL.

Let’s say you’re working for a movie production company that’s looking to create the next big hit. They want to see which directors are the most profitable (and still alive!) in order to hire them for their next big concept film.

* **Step 1:** Select the most profitable films and display the film id, film title, and profit. Provide your query below.

﻿SELECT id, title, (gross-budget) as profit

FROM films

ORDER BY profit desc

LIMIT 25

**Hint:** Management is not going to care about the 90th most profitable movie. Limit your answers to the top 25 films by profitability.

* **Step 2:** Now that you have the 25 most profitable films, you’ll need to append the director of each film. Provide your query below.

﻿SELECT f.id, f.title, (f.gross-f.budget) as profit, p.name

FROM films as f

INNER JOIN roles as r

ON f.id=r.film\_id

INNER JOIN people as p

ON r.person\_id=p.id

WHERE r.role='director'

ORDER BY profit desc

LIMIT 25

**Hint:** You are going to need to join the films table with both the roles and people tables. SELECT \_\_\_\_ FROM films f JOIN roles R on…

* **Step 3:** Add to the above query to only show the directors that are still alive. Provide your query below.

﻿SELECT f.id, f.title, (f.gross-f.budget) as profit,

(SELECT name

FROM people

WHERE r.person\_id=people.id AND deathdate is NULL) as director

FROM films as f

INNER JOIN roles as r

ON f.id=r.film\_id

INNER JOIN people as p

ON r.person\_id=p.id

WHERE r.role='director'

ORDER BY profit desc

LIMIT 25