Northeastern University
CS5200 – Database Management Systems
Fall 2017, Derbinsky

HW2

This assignment has two parts. The first focuses on SQL Programming, whereas the second on using SQL to query the Chinook database. To receive credit, submit to Blackboard **a single ZIP file** that contains **only** the following files:

- `ChinookApp.java`: the single file in which you added code for part 1
- `p21.sql`: your SQL query for part 2, problem 1
- `p22.sql`: your SQL query for part 2, problem 2
- `p23.sql`: your SQL query for part 2, problem 3
- `p24.sql`: your SQL query for part 2, problem 4
- `p25.sql`: your SQL query for part 2, problem 5
- `p26.sql`: your SQL query for part 2, problem 6
- `p27.sql`: your SQL query for part 2, problem 7

1 SQL Programming (40 points)

1.1 Preliminaries

You will require some Java development tools for this part of the homework. There is an attempt to be platform agnostic, but you may have to do some searching to solve individual problems that arise.

1.1.1 Java

The Java Standard Edition (SE) Development Kit (or JDK for short) provides the ability to run programs written in Java (including applets in a browser), as well as compile new programs. By contrast, the Java Runtime Environment (JRE) can run programs, but not to make new ones.

We will use the latest version of the Oracle JDK, version 8 (you will see JDK8 as well as 1.8, which mean the same thing). Use the search term "jdk8" to find Oracles download page. Then, download and install the latest "update" (e.g. 8u144) for your platform (use 64-bit unless you have a particular reason not to).

Java is widely used, and thus it is common for vulnerabilities to be discovered/exploited in the JDK. *You should keep your version up-to-date in order to limit your exposure.*

Special Notes: Linux. OpenJDK is an open-source implementation of Java, and is what may be the default option for Java on common Linux distributions. It is possible to install Oracles JDK, but may require adding a non-free repository and/or a manual installation process. If you choose to use OpenJDK for this class, recognize that we will grade using the Oracle JDK, and thus you are taking a risk with respect to graded work.

1.1.2 Eclipse

The provided Java starter code is an Eclipse project. It is recommended, though not required, to use Eclipse to import, build, test, and run the application.

There are several good integrated-development environments (IDEs) for Java (e.g. NetBeans, IntelliJ). Eclipse has a highly extensible plug-in system, supports many languages/platforms (e.g. Android), and is widely used in industry.

First, navigate to the Eclipse Downloads page (search term "eclipse java download"). Next, download "Eclipse IDE for Java Developers" for your platform (match 32/64-bit to the JDK version above). Eclipse typically comes as a compressed folder you can extract the contents wherever is convenient and simply run the main eclipse executable (e.g. eclipse.exe on Windows, Eclipse.app on Mac OS).

1.2 Your Task

You are going to write a Java command-line application that supports five (5) queries on the Chinook database (via SQLite). To get a feel for the system, the following shows the "usage" statement that you would get if you run the completed program from the command line (with no arguments):

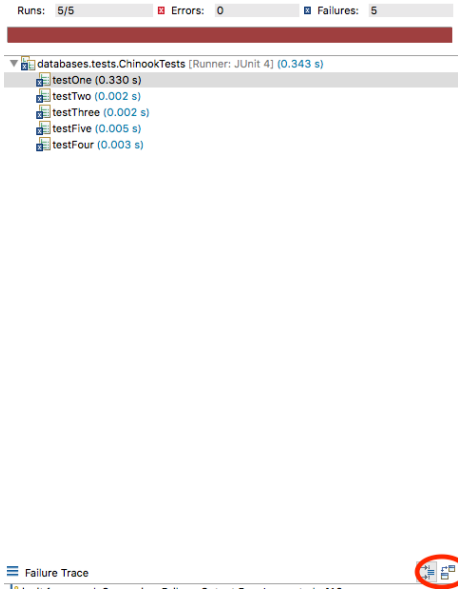
```
Usage: java databases.ChinookApp <path to Chinook database> <query #> [parameter value]

1) How many customers live in country [parameter value]?
2) List all employees (sort by employee id)
3) How many customers have been supported by employee id [parameter value]?
4) List all customers (sort by customer id)
5) List all invoices for customer #[parameter value] (sort by invoice id, each line by invoice line id)
```

So the first argument will be the path to the SQLite database, the second will be the query number (1 – 5, corresponding to the list above), and the third is a parameter value (only applicable to queries 1, 3, and 5).

You have been provided starter code (`starter.zip`) – unzip and import into Eclipse[1]. There are two important files in this project: `ChinookApp.java`, in which you will write your code and then **submit**, and `ChinookTests.java`, which has automated tests to help you make progress.

1.2.1 Testing with JUnit

To use the automated tests, right-click on `ChinookTests.java`, hover over "Run As", and choose "JUnit Test". A tab will pop-up that runs five tests (one per query) and shows your progress. To begin, you will fail all five tests; as you make progress, you will eventually pass all five. To see the code for a failed test, double click the test name – you will see something about the input and expected output in the editor. To see an input-output comparison, click the icon circled in red in the figure below.



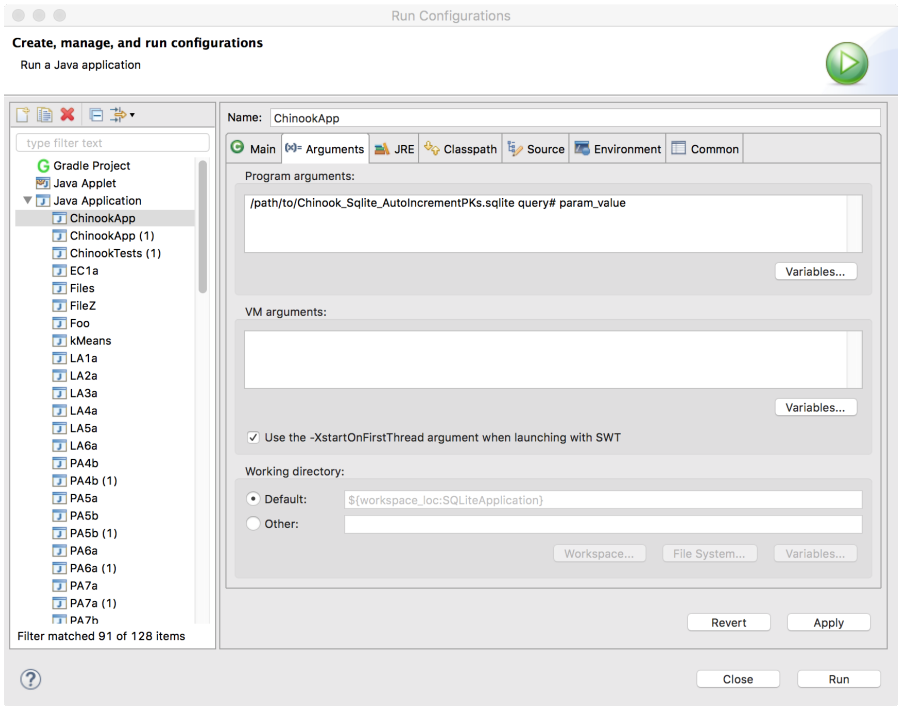
In the resulting "diff" window, pay close attention to spelling, whitespace, and missing/added lines (right-click anywhere and choose to show these items as necessary).

1.3 Arguments with Eclipse

To run your own code with command-line arguments, right-click `ChinookApp.java`, hover over "Run As", and choose "Java Application". You should then see the usage statement print to the console – this is expected, as you didn't supply any arguments.

To do so, click the "Run" menu, then "Run Configurations". Find `ChinookApp` in the list, and choose the "Arguments" tab. In the "Program Arguments" box, supply all command-line arguments in order (see usage above). Then click the "Run" button – change this as often as you wish.

[1] File, Import, "Existing Projects into Workspace" under "General"; "Browse" button, find the folder extracted, "Finish" button.



1.4 Submission

When you pass all of the JUnit tests, be sure to clean up and comment your code, including placing your name in the placeholders at the top of the file. Importantly, all of your code needs to be within the `ChinookApp.java` file – you are allowed to add methods if you wish (certainly not required), but can't change any of the provided methods (aside from where noted in the comments).

The JUnit tests will be used for grading, but other factors will be evaluated:

- Comments (including your name and section)
- Use of parameterized [i.e. safe] SQL that adheres to the query descriptions (including sorting)
- Results of additional tests (i.e. make sure your SQL addresses the prompts and you haven't modified the Chinook database)

These other factors aside, each of the five queries will be weighted equally.

2 Advanced Chinook Queries (70 points)

This part of the assignment has seven (7) problems, each weighted equally. For each problem, write an SQL query against the Chinook Database v1.4. Each query **must** run successfully using DB Browser for SQLite. A description of the correct result set for each problem is provided – your query must reproduce this result exactly (including attribute names/order and row order/contents).

To help, you have been provided an `SQLiteDiff` utility to compare the output of your query versus a supplied answer in CSV format. To use this program, create a text file that contains **only** your SQL query for a particular problem. Then run the program, supplying first the path to the supplied CSV file to compare against, then the path to your SQL file:

```
$ java -jar SQLiteDiff.jar p21.csv p21.sql
```

The program will either report success, or indicate the row/column where something differs.

Note that each question warns against using numeric ids (i.e. internal foreign key values). If your solution uses such identifiers, or tries to "game the system" (i.e. write a query that produces the correct output but does not adhere to the spirit/constraints of the question), you will receive **no** credit.

Each question also indicates a required sorting of the resulting rows. Because a database management system may produce rows in an arbitrary order, it is always good practice to explicitly indicate sorting in your SQL. Thus, if the result set has more than one row and your SQL does not fully specify row sorting order (according to the problem) you will lose 50% credit, even if the output happens to match the answer.

Problem 2.1 There are "lazy" artists in Chinook – artists that are not associated with any albums. Write a query to produce a list of these artists, sorted by artist name (alphabetically). The result set should have a single column titled `lazy_artist`. The figure below provides the first 10 rows of the result set; there are 71 in total. Your query must not hardcode any numeric ids (e.g. `ArtistId`, `AlbumId`).

lazy_artist
A Cor Do Som
Academy of St. Martin in the Fields, Sir Neville Marriner & William Bennett
Aerosmith & Sierra Leone's Refugee Allstars
Avril Lavigne
Azymuth
Baby Consuelo
Banda Black Rio
Barão Vermelho
Bebel Gilberto
Ben Harper

Problem 2.2 Aside from "Various Artists", "The Office", and "Lost", there are 9 artists that have more than 50 tracks. Write a query to generate a list of these artists, identifying their name (`artist_name`) and number of tracks (`num_tracks`). The list should be sorted by the number of tracks (greatest first), then by the name of the artist (alphabetically). Your query must not hardcode any numeric ids (e.g. `TrackId`, `AlbumId`, `ArtistId`).

artist_name	num_tracks
Iron Maiden	213
U2	135
Led Zeppelin	114
Metallica	112
Deep Purple	92
Pearl Jam	67
Lenny Kravitz	57
Faith No More	52
Van Halen	52

Problem 2.3 There is a single track with "odd" pricing: it is either an audio track whose price is not \$0.99 or it is a video track whose price is not \$1.99. Write a query to identify this track and retrieve information about the entire album; specifically, you are to retrieve for each album track the following fields in order: the title of the album (`album_title`), the album's artist's name (`artist_name`), the name of the track (`track_name`), the name of the media type of the track (`media_type`), and the unit price of the track with the \$ prepended (`unit_price`; e.g. \$0.99). The tracks should be sorted in order of the TrackId (smallest first). Your query must not hardcode any numeric ids (e.g. `TrackId`, `AlbumId`, `ArtistId`). Furthermore, your query must identify the track with "odd" pricing according to the conditions stated above (i.e. you cannot hardcode which of the two conditions will identify the track).

album_title	artist_name	track_name	media_type	unit_price
Revelations	Audioslave	Revelations	Protected AAC audio file	$0.99
Revelations	Audioslave	One and the Same	Protected AAC audio file	$0.99
Revelations	Audioslave	Sound of a Gun	Protected AAC audio file	$0.99
Revelations	Audioslave	Until We Fall	Protected AAC audio file	$0.99
Revelations	Audioslave	Original Fire	Protected AAC audio file	$0.99
Revelations	Audioslave	Broken City	Protected AAC audio file	$0.99
Revelations	Audioslave	Somedays	Protected AAC audio file	$0.99
Revelations	Audioslave	Shape of Things to Come	Protected AAC audio file	$0.99
Revelations	Audioslave	Jewel of the Summertime	Protected AAC audio file	$0.99
Revelations	Audioslave	Wide Awake	Protected AAC audio file	$0.99
Revelations	Audioslave	Nothing Left to Say But Goodbye	Protected AAC audio file	$0.99
Revelations	Audioslave	Moth	Protected AAC audio file	$0.99
Revelations	Audioslave	Show Me How to Live (Live at the Quart Festival)	Protected AAC audio file	$0.99
Revelations	Audioslave	Band Members Discuss Tracks from "Revelations"	Protected MPEG-4 video file	$0.99

Problem 2.4 Write a query to identify the single track that has no composer and is on both the "Grunge" and the "90's Music" playlists. Your query must not hardcode any numeric ids (e.g. `GenreID`, `TrackID`, `AlbumID`, `ArtistID`, `PlaylistID`). The result set (see figure below) should have a single row with the following columns (in order): track name (`track_name`), album title (`album_title`), album artist's name (`artist_name`), and genre name (`genre_name`).

track_name	album_title	artist_name	genre_name
Hunger Strike	Temple of the Dog	Temple of the Dog	Alternative

Problem 2.5 Write a query to generate a ranked list of employees based upon the amount of money brought in via customer invoices for which they were the support representative. The result set (see figure below) should have the following fields (in order) for all employees (even those that did not support any customers): ID (`e_id`), first name (`e_first_name`), last name (`e_last_name`), title (`e_title`), and invoice total (`total_invoices`). The rows should be sorted by the invoice total (greatest first), then by last name (alphabetically), then first name (alphabetically). The invoice total should be preceded by a dollar sign (`$`) and have two digits after the decimal point (rounded, as appropriate); in the case of employees without any invoices, you should output a `$0.00`, not `NULL`. You may find it useful to look at the `IFNULL`, `ROUND`, and `PRINTF` functions of SQLite[2].

e_id	e_first_name	e_last_name	e_title	total_invoices
3	Jane	Peacock	Sales Support Agent	$833.04
4	Margaret	Park	Sales Support Agent	$775.40
5	Steve	Johnson	Sales Support Agent	$720.16
1	Andrew	Adams	General Manager	$0.00
8	Laura	Callahan	IT Staff	$0.00
2	Nancy	Edwards	Sales Manager	$0.00
7	Robert	King	IT Staff	$0.00
6	Michael	Mitchell	IT Manager	$0.00

[2] `http://sqlite.org/lang_corefunc.html`

Problem 2.6 There are 17 artists that have **audio** tracks that are longer than 10 minutes. Write a query that identifies these artists (`long_artist`), as well as the name (`long_song`) and length, in full minutes (`full_minutes`), of their longest track. The results should be sorted by track length (longest first). Your query must not hardcode any numeric id's (e.g. `ArtistId`) and must discover the 17 artists via the criteria described.

long_artist	long_song	full_minutes
Led Zeppelin	Dazed And Confused	26
Deep Purple	Space Truckin'	19
Santana	We've Got To Get Together/Jingo	17
Miles Davis	My Funny Valentine (Live)	15
Terry Bozzio, Tony Levin & Steve Stevens	The Sun Road	14
Iron Maiden	Rime of the Ancient Mariner	13
The Doors	The End	11
Frank Zappa & Captain Beefheart	Advance Romance	11
Temple of the Dog	Reach Down	11
Metallica	Mercyful Fate	11
Rush	Xanadu	11
Creedence Clearwater Revival	I Heard It Through The Grapevine	11
Amy Winehouse	Amy Amy Amy (Outro)	11
Dennis Chambers	Outbreak	10
Black Sabbath	Sleeping Village	10
Jamiroquai	Revolution 1993	10
Guns N' Roses	Coma	10

Problem 2.7 There are only 2 artists that are distinguished by having at least one track that satisfies the following criteria...

a. the track is not of the "Classical" genre;

b. the track appears in more than three playlists; and

c. the track has appeared in more than one invoice line.

Write a query that identifies these artists (`distinguished_artist`), sorted by name (alphabetically). Your query must not hardcode any numeric id's (e.g. `GenreId`, `ArtistId`).

distinguished_artist
Djavan
Gilberto Gil