Homework 8

Sarah Cannon

Due: 11/5/2024

Question 1 (3 points)

> **As we get farther into this semester, rather than providing you with specific ethics resources, I'm going to start asking you to do some of your own research! This more closely mimics what you might need to do in the future, where you won't have a professor guiding your exploration and learning.**

> **Find (at least) three different sources that discuss *differential privacy*. List your three sources below. Using these three sources, explain in your own words in about one paragraph what differential privacy is, what its benefits are, and what some downsides are.**

Differential privacy is a method to keep individual data private while sharing useful insights from a dataset. It adds random noise to the data or the results of queries, so that the presence or absence of any single person's data doesn't noticeably change the outcome. This makes it hard for anyone to figure out personal details from the shared data. The main benefits are strong privacy protection and the ability to share data insights safely. However, it can be tricky to implement correctly, and the added noise might reduce the accuracy of the data. Balancing privacy and data usefulness is a key challenge.

Sources: - Wikipedia - Differential Privacy (https://en.wikipedia.org/wiki/Differential_privacy) - Towards Data Science - A Gentle Introduction to Differential Privacy (https://towardsdatascience.com/a-gentle-introduction-to-differential-privacy-330434437cdf) - Harvard Privacy Tools - What is Differential Privacy? (https://privacytools.seas.harvard.edu/differential-privacy)

Question 2 (2 points)

> **Explain the difference between read_csv and dbConnect(). When would you want to use read_csv(), and when would you want to use dbConnect()?**

`read_csv()` is a function used to read data from a CSV file into R as a data frame. It's better for smaller datasets that can be easily stored and manipulated in memory.

`dbConnect()` is used to establish a connection to a SQL database, allowing you to interact with large datasets stored externally. This is useful for working with big data that cannot be efficiently handled in memory.

Question 3 (8 points)

a. **Set up a connection to the database in the Chinook_Sqlite.sqlite file; the rest of this assignment will use this connection. This data is publicly available at https://github.com/lerocha/chinook-database (https://github.com/lerocha/chinook-database); this link also has come information about the tables included and the relationship between these tables.**

```
chinook_db <- dbConnect(SQLite(), dbname = "Chinook_Sqlite.sqlite")
```

b. **Use code to output a list of all the tables in the Chinook database.**

```
dbListTables(chinook_db)
```

```
##  [1] "Album"         "Artist"        "Customer"      "Employee"      "Genre"
##  [6] "Invoice"       "InvoiceLine"   "MediaType"     "Playlist"      "PlaylistTrack"
## [11] "Track"
```

c. **Set up a code chunk that can run SQL queries for this database. Use code to output all columns of the Album table.**

```
SELECT * FROM Album
```

Displaying records 1 - 10

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
2	Balls to the Wall	2
3	Restless and Wild	2
4	Let There Be Rock	1
5	Big Ones	3
6	Jagged Little Pill	4
7	Facelift	5

AlbumId	Title	ArtistId
8	Warner 25 Anos	6
9	Plays Metallica By Four Cellos	7
10	Audioslave	8

d. **Consider the table Track. How many rows does it have and how many columns does it have?**

```
SELECT COUNT(*) FROM Track
```

1 records

COUNT(*)
3503

Question 4 (8 points)

a. **Output just the AlbumId and Name columns from the Track data Table. The names of these two columns in your output should be “Album ID” (with a space between Album and ID) and “Name of Track”, respectively.**

```
SELECT AlbumId AS 'Album ID', Name AS 'Name of Track'
FROM Track
```

Displaying records 1 - 10

Album ID	Name of Track
1	For Those About To Rock (We Salute You)
2	Balls to the Wall
3	Fast As a Shark
3	Restless and Wild
3	Princess of the Dawn
1	Put The Finger On You
1	Let’s Get It Up
1	Inject The Venom

Album ID	Name of Track
1	Snowballed
1	Evil Walks

b. **How many different track names appear in the Track data table? Explain how you know.**

```
SELECT COUNT(DISTINCT Name) AS num_distinct_names
FROM Track
```

1 records

num_distinct_names
3257

c. **Arrange the Track table by UnitPrice. When there is a tie in UnitPrice, next arrange by Milliseconds. Be sure you display the UnitPrice, Milliseconds, and Name columns as the first three columns in your answer. Including more columns beyond these three is optional.**

```
SELECT UnitPrice, Milliseconds, Name
FROM Track
ORDER BY UnitPrice, Milliseconds
```

Displaying records 1 - 10

UnitPrice	Milliseconds	Name
0.99	1071	É Uma Partida De Futebol
0.99	4884	Now Sports
0.99	6373	A Statistic
0.99	6635	Oprah
0.99	7941	Commercial 1
0.99	11650	The Real Problem
0.99	21211	Commercial 2
0.99	29048	Bossa
0.99	32287	Casinha Feliz
0.99	33149	Mateus Enter

d. **Arrange the Track data table by Milliseconds, from highest to lowest. There is no need to include any tiebreakers. Be sure you display the Milliseconds and Name columns as the first two columns in your answer. Including more columns beyond these two is optional.**

```
SELECT Milliseconds, Name
FROM Track
ORDER BY Milliseconds DESC
```

Displaying records 1 - 10

Milliseconds	Name
5286953	Occupation / Precipice
5088838	Through a Looking Glass
2960293	Greetings from Earth, Pt. 1
2956998	The Man With Nine Lives
2956081	Battlestar Galactica, Pt. 2
2952702	Battlestar Galactica, Pt. 1
2935894	Murder On the Rising Star
2927802	Battlestar Galactica, Pt. 3
2927677	Take the Celestra
2926593	Fire In Space

Question 5 (3 points)

Make two new columns in the Track data set, one that displays the length of the track in Seconds and one that displays the length of the track in minutes; these new columns should be calls “Length in Seconds” and “Length in Minutes”, respectively. Note there are 1000.0 milliseconds in one second, and 60.0 seconds in one minute; be sure to use 1000.0 and 60.0 in any calculations you do rather than 1000 and 60. Be sure you display your two new columns, the Milliseconds column, and the Name column as the first four columns in your answer. Including more columns beyond these four is optional.

```
SELECT Milliseconds, Name, (Milliseconds / 1000.0) AS 'Length in Seconds', (Milliseconds
/ 60000.0) AS 'Length in Minutes'
FROM Track
```

Displaying records 1 - 10

Milliseconds	Name	Length in Seconds	Length in Minutes
343719	For Those About To Rock (We Salute You)	343.719	5.728650
342562	Balls to the Wall	342.562	5.709367
230619	Fast As a Shark	230.619	3.843650
252051	Restless and Wild	252.051	4.200850
375418	Princess of the Dawn	375.418	6.256967
205662	Put The Finger On You	205.662	3.427700
233926	Let's Get It Up	233.926	3.898767
210834	Inject The Venom	210.834	3.513900
203102	Snowballed	203.102	3.385033
263497	Evil Walks	263.497	4.391617

Question 6 (12 points)

In all parts below, make sure your answer displays the column(s) you are using for your filtering and the Name column; including additional columns is optional.

a. **Filter the Track data table to find all observations where the GenreID is not 2.**

```
SELECT *
FROM Track
WHERE GenreId != 2
```

Displaying records 1 - 10

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
1	For Those About To Rock (We Salute You)	1	1	1	Angus Young, Malcolm Young, Brian Johnson	343719	11170334	0.99
2	Balls to the Wall	2	2	1	NA	342562	5510424	0.99
3	Fast As a Shark	3	2	1	F. Baltes, S. Kaufman, U. Dirkscneider & W. Hoffman	230619	3990994	0.99
4	Restless and Wild	3	2	1	F. Baltes, R.A. Smith-Diesel, S. Kaufman, U. Dirkscneider & W. Hoffman	252051	4331779	0.99
5	Princess of the Dawn	3	2	1	Deaffy & R.A. Smith-Diesel	375418	6290521	0.99
6	Put The Finger On You	1	1	1	Angus Young, Malcolm Young, Brian Johnson	205662	6713451	0.99
7	Let’s Get It Up	1	1	1	Angus Young, Malcolm Young, Brian Johnson	233926	7636561	0.99
8	Inject The Venom	1	1	1	Angus Young, Malcolm Young, Brian Johnson	210834	6852860	0.99
9	Snowballed	1	1	1	Angus Young, Malcolm Young, Brian Johnson	203102	6599424	0.99
10	Evil Walks	1	1	1	Angus Young, Malcolm Young, Brian Johnson	263497	8611245	0.99

b. **Filter the Track data table to find all observations where the track Name contains the substring “Love”. It is fine if the string you’re looking for appears capitalized or lowercase, and it’s fine if it appears as a substring of a longer word.**

```
SELECT *
FROM Track
WHERE Name LIKE '%Love%'
```

Displaying records 1 - 10

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
24	Love In An Elevator	5	1	1	Steven Tyler, Joe Perry	321828	10552051	0.99
56	Love, Hate, Love	7	1	1	Jerry Cantrell, Layne Staley	387134	12575396	0.99
195	Let Me Love You Baby	20	1	6	Willie Dixon	175386	5716994	0.99
335	My Love	29	1	9	Jauperi/Zeu Góes	203493	6772813	0.99
341	The Girl I Love She Got Long Black Wavy Hair	30	1	1	Jimmy Page/John Bonham/John Estes/John Paul Jones/Robert Plant	183327	5995686	0.99
345	Whole Lotta Love	30	1	1	Jimmy Page/John Bonham/John Paul Jones/Robert Plant/Willie Dixon	373394	12258175	0.99
413	Loverman	35	1	3	Cave	472764	15446975	0.99
440	Love Gun	37	1	1	Paul Stanley	196257	6424915	0.99
444	Do You Love Me	37	1	1	Paul Stanley, B. Ezrin, K. Fowley	214987	6976194	0.99

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
449	Calling Dr. Love	37	1	1	Gene Simmons	225332	7395034	0.99

c. **Filter the Track data table to find all observations where the track Name contains “Love” or the track name contains “Hate”. It is fine if the strings you’re looking for appear capitalized or lowercase, and it’s fine if they appear as a substring of a longer word.**

```
SELECT *
FROM Track
WHERE Name LIKE '%Love%' OR Name LIKE '%Hate%'
```

Displaying records 1 - 10

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
24	Love In An Elevator	5	1	1	Steven Tyler, Joe Perry	321828	10552051	0.99
56	Love, Hate, Love	7	1	1	Jerry Cantrell, Layne Staley	387134	12575396	0.99
195	Let Me Love You Baby	20	1	6	Willie Dixon	175386	5716994	0.99
335	My Love	29	1	9	Jauperi/Zeu Góes	203493	6772813	0.99
341	The Girl I Love She Got Long Black Wavy Hair	30	1	1	Jimmy Page/John Bonham/John Estes/John Paul Jones/Robert Plant	183327	5995686	0.99
345	Whole Lotta Love	30	1	1	Jimmy Page/John Bonham/John Paul Jones/Robert Plant/Willie Dixon	373394	12258175	0.99
413	Loverman	35	1	3	Cave	472764	15446975	0.99
440	Love Gun	37	1	1	Paul Stanley	196257	6424915	0.99

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
444	Do You Love Me	37	1	1	Paul Stanley, B. Ezrin, K. Fowley	214987	6976194	0.99
449	Calling Dr. Love	37	1	1	Gene Simmons	225332	7395034	0.99

d. **Filter the Track data table to find all observations where the track Name contains “Love”, “Live”, or any other string that has an L, followed by any single character, followed by “ve”. It is fine if the strings you’re looking for appear capitalized or lowercase, and it’s fine if they appear as a substring of a longer word.**

```
SELECT *
FROM Track
WHERE Name LIKE '%L_ve%'
```

Displaying records 1 - 10

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
24	Love In An Elevator	5	1	1	Steven Tyler, Joe Perry	321828	10552051	0.99
56	Love, Hate, Love	7	1	1	Jerry Cantrell, Layne Staley	387134	12575396	0.99
86	Show Me How to Live	10	1	1	Audioslave/Chris Cornell	277890	6672176	0.99
95	Bring’em Back Alive	10	1	1	Audioslave/Chris Cornell	329534	7911634	0.99
195	Let Me Love You Baby	20	1	6	Willie Dixon	175386	5716994	0.99
335	My Love	29	1	9	Jauperi/Zeu Góes	203493	6772813	0.99

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
341	The Girl I Love She Got Long Black Wavy Hair	30	1	1	Jimmy Page/John Bonham/John Estes/John Paul Jones/Robert Plant	183327	5995686	0.99
345	Whole Lotta Love	30	1	1	Jimmy Page/John Bonham/John Paul Jones/Robert Plant/Willie Dixon	373394	12258175	0.99
388	À Vontade (Live Mix)	33	1	7	Bombom/Ed Motta	180636	5972430	0.99
395	Eu Vim Da Bahia - Live	34	1	7	Vários	157988	5115428	0.99

e. **Filter the Track data table to find all observations where there is no information on the composer of the track.**

```
SELECT *
FROM Track
WHERE Composer IS NULL
```

Displaying records 1 - 10

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
2	Balls to the Wall	2	2	1	NA	342562	5510424	0.99
63	Desafinado	8	1	2	NA	185338	5990473	0.99
64	Garota De Ipanema	8	1	2	NA	285048	9348428	0.99
65	Samba De Uma Nota Só (One Note Samba)	8	1	2	NA	137273	4535401	0.99
66	Por Causa De Você	8	1	2	NA	169900	5536496	0.99
67	Ligia	8	1	2	NA	251977	8226934	0.99
68	Fotografia	8	1	2	NA	129227	4198774	0.99

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
69	Dindi (Dindi)	8	1	2	NA	253178	8149148	0.99
70	Se Todos Fossem Iguais A Você (Instrumental)	8	1	2	NA	134948	4393377	0.99
71	Falando De Amor	8	1	2	NA	219663	7121735	0.99

f. **Filter the Track data table to display rows 1,001 to 1,500. Your answer should not use the TrackId column; your command should be one that would work on any table, even those without ID columns.**

```
SELECT *
FROM Track
LIMIT 500 OFFSET 1000
```

Displaying records 1 - 10

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
1001	Miracle	80	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris Shiflett/FOO FIGHTERS	209684	6877994	0.99
1002	Another Round	80	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris Shiflett/FOO FIGHTERS	265848	8752670	0.99
1003	Friend Of A Friend	80	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris Shiflett/FOO FIGHTERS	193280	6355088	0.99
1004	Over And Out	80	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris Shiflett/FOO FIGHTERS	316264	10428382	0.99

TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer	Milliseconds	Bytes	UnitPrice
1005	On The Mend	80	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris Shiflett/FOO FIGHTERS	271908	9071997	0.99
1006	Virginia Moon	80	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris Shiflett/FOO FIGHTERS	229198	7494639	0.99
1007	Cold Day In The Sun	80	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris Shiflett/FOO FIGHTERS	200724	6596617	0.99
1008	Razor	80	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris Shiflett/FOO FIGHTERS	293276	9721373	0.99
1009	All My Life	81	1	4	Foo Fighters	263653	8665545	0.99
1010	Low	81	1	4	Foo Fighters	268120	8847196	0.99

Question 7 (8 points)

a. **Calculate the total Bytes of all tracks in the Tracks data set.**

```
SELECT SUM(Bytes) AS total_bytes
FROM Track
```

1 records

total_bytes
117386255350

b. **Calculate the minimum, mean, and maximum milliseconds of the tracks listed in the Track data table.**

```
SELECT MIN(Milliseconds) AS min_milliseconds, AVG(Milliseconds) AS mean_milliseconds, MA
X(Milliseconds) AS max_milliseconds
FROM Track
```

1 records

min_milliseconds	mean_milliseconds	max_milliseconds
1071	393599.2	5286953

c. Do the same calculations as part (b), but only for tracks where composer information is available.

```
SELECT MIN(Milliseconds) AS min_milliseconds, AVG(Milliseconds) AS mean_milliseconds, MA
X(Milliseconds) AS max_milliseconds
FROM Track
WHERE Composer IS NOT NULL
```

1 records

min_milliseconds	mean_milliseconds	max_milliseconds
1071	270470.3	1612329

d. Do the same calculations as part (b), but separately for each GenreID.

```
SELECT GenreId, MIN(Milliseconds) AS min_milliseconds, AVG(Milliseconds) AS mean_millise
conds, MAX(Milliseconds) AS max_milliseconds
FROM Track
GROUP BY GenreId
```

Displaying records 1 - 10

GenreId	min_milliseconds	mean_milliseconds	max_milliseconds
1	1071	283910.0	1612329
2	126511	291755.4	907520
3	41900	309749.4	816509
4	4884	234353.8	558602
5	106266	134643.5	163265
6	135053	270359.8	589531
7	33149	232859.3	543007

GenreId	min_milliseconds	mean_milliseconds	max_milliseconds
8	173008	247177.8	366733
9	129666	229034.1	663426
10	32287	244370.9	383764

Question 8 (16 points)

a. (2 points) **What column(s) do the Track and Album tables have in common? What column(s) to the Album and Artist tables have in common? What column(s) to the Artist and Track table have in common? You should only include columns where the data in the two columns is the same, that is, when these are columns you might want to join by.**

```
# Check the column names of the Track table
dbGetQuery(chinook_db, "PRAGMA table_info(Track)")
```

```
##   cid         name          type notnull dflt_value pk
## 1   0      TrackId       INTEGER       1         NA  1
## 2   1         Name NVARCHAR(200)       1         NA  0
## 3   2      AlbumId       INTEGER       0         NA  0
## 4   3  MediaTypeId       INTEGER       1         NA  0
## 5   4      GenreId       INTEGER       0         NA  0
## 6   5     Composer NVARCHAR(220)       0         NA  0
## 7   6 Milliseconds       INTEGER       1         NA  0
## 8   7        Bytes       INTEGER       0         NA  0
## 9   8    UnitPrice NUMERIC(10,2)       1         NA  0
```

```
dbGetQuery(chinook_db, "PRAGMA table_info(Album)")
```

```
##   cid     name          type notnull dflt_value pk
## 1   0  AlbumId       INTEGER       1         NA  1
## 2   1    Title NVARCHAR(160)       1         NA  0
## 3   2 ArtistId       INTEGER       1         NA  0
```

```
dbGetQuery(chinook_db, "PRAGMA table_info(Artist)")
```

```
##   cid     name          type notnull dflt_value pk
## 1   0 ArtistId       INTEGER       1         NA  1
## 2   1     Name NVARCHAR(120)       0         NA  0
```

Answer: - Track and Album: AlbumId - Album and Artist: ArtistId - Artist and Track: ArtistId

b. (3 points) **Join the Track and Album tables on their common column(s) to create a new table with columns for TrackId, Name (of track), AlbumId, Title (of album), and ArtistId. Do this join *without* using the join keyword.**

```
SELECT Track.TrackId, Track.Name, Track.AlbumId, Album.Title, Album.ArtistId
FROM Track, Album
WHERE Track.AlbumId = Album.AlbumId
```

Displaying records 1 - 10

TrackId	Name	AlbumId	Title	ArtistId
1	For Those About To Rock (We Salute You)	1	For Those About To Rock We Salute You	1
2	Balls to the Wall	2	Balls to the Wall	2
3	Fast As a Shark	3	Restless and Wild	2
4	Restless and Wild	3	Restless and Wild	2
5	Princess of the Dawn	3	Restless and Wild	2
6	Put The Finger On You	1	For Those About To Rock We Salute You	1
7	Let's Get It Up	1	For Those About To Rock We Salute You	1
8	Inject The Venom	1	For Those About To Rock We Salute You	1
9	Snowballed	1	For Those About To Rock We Salute You	1
10	Evil Walks	1	For Those About To Rock We Salute You	1

c. (3 points) **Produce the same table as the previous part, but this time using the JOIN keyword and the USING keyword.**

```
SELECT Track.TrackId, Track.Name, Track.AlbumId, Album.Title, Album.ArtistId
FROM Track JOIN Album USING(AlbumId)
```

Displaying records 1 - 10

TrackId	Name	AlbumId	Title	ArtistId

TrackId	Name	AlbumId	Title	ArtistId
1	For Those About To Rock (We Salute You)	1	For Those About To Rock We Salute You	1
2	Balls to the Wall	2	Balls to the Wall	2
3	Fast As a Shark	3	Restless and Wild	2
4	Restless and Wild	3	Restless and Wild	2
5	Princess of the Dawn	3	Restless and Wild	2
6	Put The Finger On You	1	For Those About To Rock We Salute You	1
7	Let's Get It Up	1	For Those About To Rock We Salute You	1
8	Inject The Venom	1	For Those About To Rock We Salute You	1
9	Snowballed	1	For Those About To Rock We Salute You	1
10	Evil Walks	1	For Those About To Rock We Salute You	1

> d. (3 points) **Produce the same tibble as the previous two parts, but this time using the JOIN keyword and the ON keyword.**

```
SELECT Track.TrackId, Track.Name, Track.AlbumId, Album.Title, Album.ArtistId
FROM Track JOIN Album ON Track.AlbumId = Album.AlbumId
```

Displaying records 1 - 10

TrackId	Name	AlbumId	Title	ArtistId
1	For Those About To Rock (We Salute You)	1	For Those About To Rock We Salute You	1
2	Balls to the Wall	2	Balls to the Wall	2
3	Fast As a Shark	3	Restless and Wild	2
4	Restless and Wild	3	Restless and Wild	2
5	Princess of the Dawn	3	Restless and Wild	2
6	Put The Finger On You	1	For Those About To Rock We Salute You	1
7	Let's Get It Up	1	For Those About To Rock We Salute You	1

TrackId	Name	AlbumId	Title	ArtistId
8	Inject The Venom	1	For Those About To Rock We Salute You	1
9	Snowballed	1	For Those About To Rock We Salute You	1
10	Evil Walks	1	For Those About To Rock We Salute You	1

e. (2 points) **Explain when you can use the USING keyword, and when you can use the ON keyword.**

- USING: When the column names are the same in both tables.
- ON: When the column names are different in both tables.

f. (3 points) **Filter the tibble produced in parts (b)-(d) to only include Tracks where the Track does not mention “Walk” and the album title does not mention “Rock”.**

```
SELECT Track.TrackId, Track.Name, Track.AlbumId, Album.Title, Album.ArtistId
FROM Track JOIN Album ON Track.AlbumId = Album.AlbumId
WHERE Track.Name NOT LIKE '%Walk%' AND Album.Title NOT LIKE '%Rock%'
```

Displaying records 1 - 10

TrackId	Name	AlbumId	Title	ArtistId
2	Balls to the Wall	2	Balls to the Wall	2
3	Fast As a Shark	3	Restless and Wild	2
4	Restless and Wild	3	Restless and Wild	2
5	Princess of the Dawn	3	Restless and Wild	2
24	Love In An Elevator	5	Big Ones	3
25	Rag Doll	5	Big Ones	3
26	What It Takes	5	Big Ones	3
27	Dude (Looks Like A Lady)	5	Big Ones	3
28	Janie’s Got A Gun	5	Big Ones	3
29	Cryin’	5	Big Ones	3

Question 9 (13 points)

a. **(2 points) Make a table displaying all the different ArtistId's that appear in the Album table.If you've done this right, you should get a table with one column and 204 rows.**

```
SELECT DISTINCT ArtistId
FROM Album
```

Displaying records 1 - 10

ArtistId
1
2
3
4
5
6
7
8
9
10

b. **(2 points) Make a table displaying all the different ArtistId's that appear in the Artist table. If you've done this right, you should get a table with one column and 275 rows.**

```
SELECT DISTINCT ArtistId
FROM Artist
```

Displaying records 1 - 10

ArtistId
1
2
3

ArtistId
4
5
6
7
8
9
10

c. **(3 points) Using your answer to the previous two parts, write a code chunk that determines which ArtistIds appear in both the Album table and in the Artist table.**

```
SELECT Album.ArtistId
FROM Album JOIN Artist ON Album.ArtistId = Artist.ArtistId
```

Displaying records 1 - 10

ArtistId
1
1
2
2
3
4
5
6
6
7

d. **(3 points) Using your answers to parts (a) and (b), write a code chunk to determine which ArtistIds appear in the Artist table but not the Album table.**

```
SELECT Artist.ArtistId
FROM Artist LEFT JOIN Album ON Artist.ArtistId = Album.ArtistId
WHERE Album.ArtistId IS NULL
```

Displaying records 1 - 10

ArtistId
25
26
28
29
30
31
32
33
34
35

e. **(3 points) Use an appropriate join to add the Album information from the Album table to the Artist table, keeping the ArtistId, Artist Name, AlbumId, and Album Title column column. All artists that appear in the Artist table should appear in your output, whether or not they have an album in the Album table.**

```
SELECT Artist.ArtistId, Artist.Name, Album.AlbumId, Album.Title
FROM Artist LEFT JOIN Album ON Artist.ArtistId = Album.ArtistId
```

Displaying records 1 - 10

ArtistId	Name	AlbumId	Title
1	AC/DC	1	For Those About To Rock We Salute You
1	AC/DC	4	Let There Be Rock
2	Accept	2	Balls to the Wall
2	Accept	3	Restless and Wild
3	Aerosmith	5	Big Ones
4	Alanis Morissette	6	Jagged Little Pill

ArtistId	Name	AlbumId	Title
5	Alice In Chains	7	Facelift
6	Antônio Carlos Jobim	8	Warner 25 Anos
6	Antônio Carlos Jobim	34	Chill: Brazil (Disc 2)
7	Apocalyptica	9	Plays Metallica By Four Cellos

Question 10 (4 points)

a. **Explain, in your own words, the difference between WHERE and FILTER(WHERE), including where in an SQL query they would appear and when you'd want to use one vs. the other.**

WHERE is used to filter rows before any aggregation occurs, affecting all aggregation functions. FILTER(WHERE …) is used within an aggregation function to apply a filter only to that specific aggregation. WHERE appears before GROUP BY, while FILTER(WHERE …) is used within the SELECT clause.

b. **Explain, in your own words, the difference between WHERE and HAVING, including where in an SQL query they appear and when you'd want to use one vs. the other.**

WHERE is used to filter rows before aggregation, while HAVING is used to filter groups after aggregation. WHERE appears before GROUP BY, and HAVING appears after GROUP BY.

Question 11 (18 points)

a. **Create a table that has a column that contains each AlbumId that appears in Track, a column for how many times that AlbumId appears in Track, and a column for the average milliseconds of all tracks on that album. Give your columns appropriate names.**

```
SELECT AlbumId, COUNT(*) AS num_tracks, AVG(Milliseconds) AS avg_milliseconds
FROM Track
GROUP BY AlbumId
```

Displaying records 1 - 10

AlbumId	num_tracks	avg_milliseconds

AlbumId	num_tracks	avg_milliseconds
1	10	240041.5
2	1	342562.0
3	3	286029.3
4	8	306657.4
5	15	294113.9
6	13	265455.8
7	12	270780.4
8	14	207637.6
9	8	333925.9
10	14	280550.9

b. **Modify your answer to (a) so that the count and average only include the observations where the GenreId is 1 (Rock). That is, you should have a column for AlbumId, a column for the number of tracks on that album that have GenreId 1, and a column for the average length (in milliseconds) of the tracks on that album that have GenreId 1. Your answer should not have any rows for albums with no Rock tracks. Give your columns appropriate names.**

```
SELECT AlbumId, COUNT(*) AS RockTrackCount, AVG(Milliseconds) AS AvgMilliseconds
FROM Track
WHERE GenreId = 1
GROUP BY AlbumId
```

Displaying records 1 - 10

AlbumId	RockTrackCount	AvgMilliseconds
1	10	240041.5
2	1	342562.0
3	3	286029.3
4	8	306657.4
5	15	294113.9
6	13	265455.8
7	12	270780.4
10	14	280550.9

AlbumId	RockTrackCount	AvgMilliseconds
30	14	320708.6
31	9	273992.1

c. **Modify your answer to (b) so that you do have rows for the albums that have no tracks with genre 1; For example, there should be a row in your table for AlbumId 8, with 0 in the Number of Rock Tracks column and NA for the Average Length (in milliseconds) of the rock tracks on that album.**

```
  SELECT Album.AlbumId,
         COUNT(CASE WHEN Track.GenreId = 1 THEN Track.TrackId ELSE NULL END) AS RockTrac
kCount,
         AVG(CASE WHEN Track.GenreId = 1 THEN Track.Milliseconds ELSE NULL END) AS AvgMi
lliseconds
  FROM Album
  LEFT JOIN Track ON Album.AlbumId = Track.AlbumId
  GROUP BY Album.AlbumId
```

Displaying records 1 - 10

AlbumId	RockTrackCount	AvgMilliseconds
1	10	240041.5
2	1	342562.0
3	3	286029.3
4	8	306657.4
5	15	294113.9
6	13	265455.8
7	12	270780.4
8	0	NA
9	0	NA
10	14	280550.9

d. **Make a table that has three columns: Album Id, Total number of tracks on that album, and total number of rock tracks on that album.**

```
SELECT AlbumId,
       COUNT(*) AS num_tracks,
       SUM(CASE WHEN GenreId = 1 THEN 1 ELSE 0 END) AS num_rock_tracks
FROM Track
GROUP BY AlbumId
```

Displaying records 1 - 10

AlbumId	num_tracks	num_rock_tracks
1	10	10
2	1	1
3	3	3
4	8	8
5	15	15
6	13	13
7	12	12
8	14	0
9	8	0
10	14	14

e. **Modify your code from (d) to only keep those AlbumIds that have at least 10 tracks total.**

```
SELECT AlbumId,
       COUNT(*) AS num_tracks,
       SUM(CASE WHEN GenreId = 1 THEN 1 ELSE 0 END) AS num_rock_tracks
FROM Track
GROUP BY AlbumId
HAVING COUNT(*) >= 10
```

Displaying records 1 - 10

AlbumId	num_tracks	num_rock_tracks
1	10	10
5	15	15
6	13	13
7	12	12
8	14	0

AlbumId	num_tracks	num_rock_tracks
10	14	14
11	12	0
12	12	0
14	13	0
17	10	0

f. **Modify your code from (d) to also include a column for the Title of each album.**

```
SELECT Album.AlbumId,
       Album.Title,
       COUNT(Track.TrackId) AS num_tracks,
       SUM(CASE WHEN Track.GenreId = 1 THEN 1 ELSE 0 END) AS num_rock_tracks
FROM Album
LEFT JOIN Track ON Album.AlbumId = Track.AlbumId
GROUP BY Album.AlbumId
```

Displaying records 1 - 10

AlbumId	Title	num_tracks	num_rock_tracks
1	For Those About To Rock We Salute You	10	10
2	Balls to the Wall	1	1
3	Restless and Wild	3	3
4	Let There Be Rock	8	8
5	Big Ones	15	15
6	Jagged Little Pill	13	13
7	Facelift	12	12
8	Warner 25 Anos	14	0
9	Plays Metallica By Four Cellos	8	0
10	Audioslave	14	14

Question 12 (4 points)

Using any method you'd like, make a table with two columns: One for Genre (the full Genre name, not the GenreId), and one for how frequently the GenreId for that Genre appears in Tracks.

```
SELECT Genre.Name AS Genre,
       COUNT(Track.GenreId) AS Frequency
FROM Track
JOIN Genre ON Track.GenreId = Genre.GenreId
GROUP BY Genre.Name
```

Displaying records 1 - 10

Genre	Frequency
Alternative	40
Alternative & Punk	332
Blues	81
Bossa Nova	15
Classical	74
Comedy	17
Drama	64
Easy Listening	24
Electronica/Dance	30
Heavy Metal	28

Question 13 (1 points)

Close your connection to the database used throughout this assignment.

```
dbDisconnect(chinook_db)
```