Group 40
Janna Agustin - 260732684
Roshane Jochum - 260797793
Maanav - 260782837
Louise Oyarzabal - 260777303

Deliverable 2: Creating Your Database

COMP 421 - Database Systems

We used PostgreSQL for this assignment.

Q1. Modified Relational Schema

Entity Sets

- User(userID, name, username, country, email, date_of_birth, credit_info)
- Listener(userID) userID ref to User
- Artist(userID, stage_name, balance) userID ref to User
- Library(libid, number_of_songs)
- Playlist(name, libid, status, number_of_songs) libid ref to Library
- Article(articleID, title, release_date, price, genre)
- Album(articleID, type) articleID ref to Article
- Song(articleID, duration) articleID ref to Article
- ShoppingCart(orderID, total_amount, number_of_articles)

Relationships

- Releases(articleID, artistID) articleID ref to Article, artistID ref to Artist
- BelongsTo(songID, albumID) songID ref to Song (articleID), albumID ref to Album (articleID)
- ComprisesOf(libid,name,songID) libid ref to Library, name ref to Playlist, songID ref to Song (articleID)
- IsAddedTo(articleID, libid) Libid ref to Library, articleID ref to Article
- IsPartOf(name, libid) name ref to Playlist, libid ref to Library
- Contains(orderID, articleID) orderID ref to Shopping Cart, articleID ref to Article
- Creates(listenerID, libid, name) ListenerID ref to Listener (userID), libid ref to Library, name ref to Playlist
- Uses(listenerID, orderID) ListenerID ref to Listener (userID), orderID ref to Shopping Cart
- moneyEarned(artistID, orderID, amountEarned) artistID ref to Artist (userID), orderID ref to Shopping Cart
- Has(listenerID,libid) listenerID ref to Listener (userID), libid ref to Library

Modifications

- Deleted the HasGenre relation
- Added the Has relation (which appeared in our diagram but not in our relational statements)

Constraints not portrayed in the relational schema

All constraints mentioned in the previous deliverable and:

- No prices shall be negative and therefore no total amount for an order shall be negative.

Q2. SQL Database Schema

See the **create_statements.sql** file, as well as the **description_tables.txt** file for the descriptions of all created tables (results of *\d <table name>* statements).

Q3. 5 INSERT Commands

(See **insert_statements.sql** file)

Here are insert commands of 5 articles.

```
INSERT INTO articles(article_id,title,genre,release_date,price) values (25,'Born This Way','R&B','2007-06-14',0.55);
INSERT INTO articles(article_id,title,genre,release_date,price) values (68,'Part Of Me','R&B','2007-06-14',0.55);
INSERT INTO articles(article_id,title,genre,release_date,price) values (252,'Sucker','R&B','2007-06-14',0.55);
INSERT INTO articles(article_id,title,genre,release_date,price) values (21,'Patricia','R&B','2007-06-14',0.55);
INSERT INTO articles(article_id,title,genre,release_date,price) values (0,'Always Be My Baby','R&B','2007-06-14',0.55);
```

*Response of SELECT * FROM relationname*

```
cs421=> SELECT * FROM articles LIMIT 10;
 article_id |       title        | release_date | price | genre
------------+--------------------+--------------+-------+-------
         25 | Born This Way      | 2007-06-14   |  0.55 | R&B
         68 | Part Of Me         | 2007-06-14   |  0.55 | R&B
        252 | Sucker             | 2007-06-14   |  0.55 | R&B
         21 | Patricia           | 2007-06-14   |  0.55 | R&B
          0 | Always Be My Baby  | 2007-06-14   |  0.55 | R&B
```

Q4. Table Output

See **insert_statements.sql** file as well as the **tables_output.txt** file (contains the outputs of *SELECT * FROM ... LIMIT 10* for each table).

Q5. Queries

1- Display the listeners that have spent the most money buying songs and albums. This allows us to see how much our biggest customers are willing to spend on our platform. *Note: The total amount would be more significant once the platform reaches a bigger scale or is available for a longer amount of time.*

SQL Statement

```
SELECT uses.listener_id, total_amount
FROM uses, shopping_carts, listeners
WHERE shopping_carts.order_id = uses.order_id AND uses.listener_id = listeners.uid
ORDER BY total_amount DESC;
```

Script illustrating execution and output

```
cs421=> SELECT uses.listener_id, total_amount
cs421-> FROM uses, shopping_carts, listeners
cs421-> WHERE shopping_carts.order_id = uses.order_id AND uses.listener_id = listeners.uid
cs421-> ORDER BY total_amount DESC;
 listener_id | total_amount
-------------+--------------
         185 |          4.4
         179 |         4.35
         168 |         3.85
          69 |          3.8
          38 |          3.8
         107 |          3.3
         368 |         3.15
         320 |            3
         323 |          1.9
         310 |         1.65
         395 |         0.95
         164 |         0.95
         253 |         0.95
          34 |         0.95
         108 |         0.55
          12 |         0.55
         365 |            0
(17 rows)
```

2 - Select all listeners emails, orderid, total price for each. This would allow us to send each listener a receipt about their placed orders.

SQL Statement

```
SELECT l.email, u.order_id, s.total_amount
FROM Uses u, shopping_carts s, users l
WHERE u.order_id = s.order_id AND u.listener_id = l.uid;
```

Script illustrating execution and output

```
cs421=> SELECT l.email, u.order_id, s.total_amount
cs421-> FROM Uses u, shopping_carts s, users l
cs421-> WHERE u.order_id = s.order_id AND u.listener_id = l.uid;
             email               | order_id | total_amount
---------------------------------+----------+--------------
 Theo.Angelo@yahoo.com           |        0 |          3.3
 Zhang.Wang@yahoo.com            |        1 |          3.8
 Olive.Leone@gmail.com           |        2 |         0.95
 Kian.Kourilsky@gmail.com        |        3 |          3.8
 Tara.Rodriguez@yahoo.com        |        4 |          1.9
 Elio.Jones@mail.mcgill.ca       |        5 |         3.15
 Misgana.Rodriguez@mail.mcgill.ca |        6 |         0.55
 Tara.Markoski@yahoo.com         |        7 |         0.95
 Theo.Wilson@mail.mcgill.ca      |        8 |            0
 Magnus.Angelo@mail.mcgill.ca    |        9 |         4.35
 Sam.Kourilsky@mail.mcgill.ca    |       10 |            3
 Janna.Jones@gmail.com           |       11 |         0.95
 Sam.Nguyen@yahoo.com            |       12 |         0.95
 Karim.Porta@gmail.com           |       13 |         0.55
 Olive.Wilson@gmail.com          |       14 |         1.65
 Misgana.Porta@gmail.com         |       15 |         3.85
 Max.Williams@gmail.com          |       16 |          4.4
(17 rows)
```

3 - Display all the artists that released R&B music. If we see that the demand for the R&B music is increasing, this helps to know how many artists are releasing music of this genre on our platform (it could be helpful if need to contact them for a partnership).

SQL Statement

```
SELECT stage_name
FROM artists
WHERE uid = ANY (
    SELECT DISTINCT R.artist_id
    FROM articles A, releases R
    WHERE A.article_id = R.article_id AND A.genre = 'R&B');
```

Script illustrating execution and output

```
cs421=> SELECT stage_name
cs421-> FROM artists
cs421-> WHERE uid = ANY (
cs421(> SELECT DISTINCT R.artist_id
cs421(> FROM articles A, releases R
cs421(> WHERE A.article_id = R.article_id AND A.genre = 'R&B');
 stage_name
------------
 Zaryah
 The Fireman
 JP Edwards
(3 rows)
```

4 - Display the most frequently bought articles on the platform. This would give us information on the most popular articles on our platform (which could, for example, be helpful to establish future partnerships with the artists to do promotion, or to know which type of music is in demand).

SQL Statement

```
SELECT articles.title, COUNT(contains.article_id) AS times_bought, SUM(price)
FROM articles, contains, shopping_carts
WHERE articles.article_id = contains.article_id AND shopping_carts.order_id = contains.order_id
GROUP BY articles.title
ORDER BY COUNT(articles.article_id) DESC;
```

Script illustrating execution and output

```
cs421=> SELECT articles.title, COUNT(contains.article_id) AS times_bought, SUM(price)
cs421-> FROM articles, contains, shopping_carts
ns.order_idRE articles.article_id = contains.article_id AND shopping_carts.order_id = contai
cs421-> GROUP BY articles.title
cs421-> ORDER BY COUNT(articles.article_id) DESC LIMIT 20;
        title         | times_bought | sum
----------------------+--------------+------
 Back To December     |            4 |  3.8
 Speed Of Sound       |            3 | 1.65
 Right Above It       |            3 | 2.85
 Deja Vu              |            3 |  2.2
 Don t Cry (remix)    |            3 |    0
 Barbie Girl (remix)  |            3 |    0
 Psycho               |            3 | 2.85
 Begin Again          |            2 |    0
 Change The World     |            2 |  1.1
 Paris                |            2 |  1.1
 Please Me (remix)    |            2 |    0
 Say You ll Be There  |            2 |  1.1
 Don t Stop Believing |            2 | 2.75
 Strange Clouds       |            2 |    0
 Part Of Me           |            2 |  3.3
 Speak Now            |            2 |  4.4
 Runaway              |            2 | 4.75
 She Will             |            2 |  1.9
 All By Myself        |            2 |  1.1
 Circles              |            2 | 3.85
(20 rows)
```

(here we limited the display up to 20 articles)

5 - List all artists in ascending order by stage name with their respective genre. A regular tool that could be useful for us and for the listeners when searching for a particular artist.

SQL Statement

```
SELECT DISTINCT a.stage_name, m.genre
FROM artists a, releases r, articles m
WHERE r.artist_id = a.uid AND r.article_id = m.article_id
ORDER BY a.stage_name ASC;
```

Script illustrating execution and output

```
cs421=> SELECT DISTINCT a.stage_name, m.genre
cs421-> FROM artists a, releases r, articles m
cs421-> WHERE r.artist_id = a.uid AND r.article_id = m.article_id
cs421-> ORDER BY a.stage_name ASC;
      stage_name       |    genre
-----------------------+-------------
 22 Seconds to Boston | Rock
 Alec Wood            | Blues
 Alexandre Millet     | Soundtrack
 Alice In The Rain    | Independent
 Aya Nova             | Pop
 Ayden Page           | Independent
 Chitty Bang          | Hip-Hop/Rap
 Chitty Bang          | Rap
 David Olsen          | Electro Pop
 Dutch Dabber         | Electronic
 Emma Della           | Pop
 Francois Delacour    | Rock
 Greg McMiller        | Country
 How About Now        | Punk Rock
 Isiah Deville        | Pop
 John Williams        | Rock
 JP Edwards           | R&B
 Justin B             | Pop
 Kicking Dolls        | Punk Rock
 KM Kid               | Rap
 Lara O Mara          | Jazz
 LEIMA                | Indie Pop
 Lil Diggity          | Rap
 Michael James        | Independent
 NMN                  | Electronic
 No Box               | Rock
 Purpleplum           | Electronic
 The Fireman          | R&B
 The Kicking Eyes     | Soul
 The Lilies           | Independent
 The Sunny Boys       | Pop
 The Wheelies         | Country
 Zaryah               | R&B
 Zephiro              | Folk
(34 rows)
```

Q6. Data Modification Commands

1 - This query gives a reward of 10$ to all artists that submitted more than 5 free songs. It encourages new artists to release music and to have affordable options for listeners who are looking for free music.

SQL Statement

```sql
UPDATE artists
SET balance = balance + 10
WHERE uid IN (SELECT r.artist_id
    FROM Articles a, releases r
    WHERE a.article_id = r.article_id AND a.price = 0
    GROUP BY r.artist_id
    HAVING COUNT(r.article_id)>5) ;
```

Script illustrating execution

```
cs421=> UPDATE artists
cs421-> SET balance = balance + 10
cs421-> WHERE uid IN (SELECT r.artist_id
cs421(> FROM Articles a, releases r
cs421(> WHERE a.article_id = r.article_id AND a.price = 0
cs421(> GROUP BY r.artist_id
cs421(> HAVING COUNT(r.article_id)>5) ;
UPDATE 3
```

Output before and after

```
cs421=> SELECT * FROM artists
cs421-> ;
 uid |      stage_name       | balance
-----+-----------------------+---------
  89 | Zaryah                |    2.75
  13 | Emma Della            |    2.85
 334 | NMN                   |    2.85
 329 | Lil Diggity           |    4.75
 246 | Alec Wood             |    2.75
  27 | Justin B              |       0
 216 | Dutch Dabber          |       0
 315 | Aya Nova              |   11.15
 145 | Purpleplum            |       0
 270 | Francois Delacour     |     2.2
 113 | How About Now         |     7.7
 238 | Isiah Deville         |       0
 387 | LEIMA                 |       0
 236 | The Lilies            |       0
 146 | KM Kid                |       0
 358 | The Fireman           |       0
 124 | Michael James         |       0
 353 | Chitty Bang           |       0
 143 | Lara O Mara           |       0
 335 | JP Edwards            |       0
 144 | Alexandre Millet      |       0
 173 | The Sunny Boys        |       0
 163 | 22 Seconds to Boston  |       0
 171 | The Wheelies          |       0
 354 | No Box                |       0
 200 | The Kicking Eyes      |       0
 231 | John Williams         |       0
 281 | Ayden Page            |       0
 359 | Kicking Dolls         |       0
 118 | Alice In The Rain     |       0
 177 | Greg McMiller         |       0
 373 | David Olsen           |       0
  71 | Zephiro               |       0
(33 rows)
```

```
cs421=> SELECT * FROM artists;
 uid |      stage_name       | balance
-----+-----------------------+---------
  89 | Zaryah                |    2.75
  13 | Emma Della            |    2.85
 334 | NMN                   |    2.85
 329 | Lil Diggity           |    4.75
 246 | Alec Wood             |    2.75
  27 | Justin B              |       0
 216 | Dutch Dabber          |       0
 315 | Aya Nova              |   11.15
 270 | Francois Delacour     |     2.2
 113 | How About Now         |     7.7
 238 | Isiah Deville         |       0
 387 | LEIMA                 |       0
 236 | The Lilies            |       0
 146 | KM Kid                |       0
 358 | The Fireman           |       0
 124 | Michael James         |       0
 353 | Chitty Bang           |       0
 335 | JP Edwards            |       0
 144 | Alexandre Millet      |       0
 173 | The Sunny Boys        |       0
 163 | 22 Seconds to Boston  |       0
 354 | No Box                |       0
 200 | The Kicking Eyes      |       0
 231 | John Williams         |       0
 281 | Ayden Page            |       0
 359 | Kicking Dolls         |       0
 118 | Alice In The Rain     |       0
 177 | Greg McMiller         |       0
 373 | David Olsen           |       0
  71 | Zephiro               |       0
 145 | Purpleplum            |      10
 143 | Lara O Mara           |      10
 171 | The Wheelies          |      10
(33 rows)
```

2 - This query gives a 10% discount to all orders that are more than 3.50$. Flash sale: to encourage listeners to make a new order in the future.

SQL Statement

```
UPDATE shopping_carts
SET total_amount = total_amount*0.9
WHERE total_amount > 3.50;
```

Script illustrating execution

```
cs421=> UPDATE shopping_carts
cs421-> SET total_amount = total_amount*0.9
cs421-> WHERE total_amount > 3.50;
UPDATE 5
```

Output before and after

```
cs421=> SELECT * FROM shopping_carts
cs421-> ;
 order_id | total_amount | num_articles
----------+--------------+--------------
        0 |          3.3 |            7
        1 |          3.8 |            5
        2 |         0.95 |            3
        3 |          3.8 |            5
        4 |          1.9 |            3
        5 |         3.15 |            6
        6 |         0.55 |            3
        7 |         0.95 |            3
        8 |            0 |            3
        9 |         4.35 |            6
       10 |            3 |            5
       11 |         0.95 |            3
       12 |         0.95 |            3
       13 |         0.55 |            3
       14 |         1.65 |            4
       15 |         3.85 |            8
       16 |          4.4 |            9
(17 rows)
```

```
cs421=> SELECT * FROM shopping_carts
;
 order_id | total_amount | num_articles
----------+--------------+--------------
        0 |          3.3 |            7
        2 |         0.95 |            3
        4 |          1.9 |            3
        5 |         3.15 |            6
        6 |         0.55 |            3
        7 |         0.95 |            3
        8 |            0 |            3
       10 |            3 |            5
       11 |         0.95 |            3
       12 |         0.95 |            3
       13 |         0.55 |            3
       14 |         1.65 |            4
        1 |         3.42 |            5
        3 |         3.42 |            5
        9 |        3.915 |            6
       15 |        3.465 |            8
       16 |         3.96 |            9
(17 rows)
```

3 - Give a $3 gift to all artists younger than 25 who haven't made any revenue yet. This shows our support for young artists who are new to our platform or may not have gotten enough recognition yet. Making them feel welcome can build up their customer loyalty as they become bigger on our platform.

SQL Statement

```
UPDATE artists
SET balance = 3
WHERE balance = 0 AND
    uid IN (SELECT a.uid
        FROM users u, artists a
        WHERE u.dob> '1995-02-28' AND u.uid = a.uid);
```

Script illustrating execution

```
cs421=> UPDATE artists
cs421-> SET balance = 3
cs421-> WHERE balance = 0 AND
cs421-> uid IN (SELECT a.uid
cs421(> FROM users u, artists a
cs421(> WHERE u.dob> '1995-02-28' AND u.uid = a.uid);
UPDATE 6
```

Output before and after :

```
cs421=> SELECT * FROM artists;
 uid |      stage_name      | balance
-----+----------------------+---------
  89 | Zaryah               |    2.75
  13 | Emma Della           |    2.85
 334 | NMN                  |    2.85
 329 | Lil Diggity          |    4.75
 246 | Alec Wood            |    2.75
  27 | Justin B             |       0
 216 | Dutch Dabber         |       0
 315 | Aya Nova             |   11.15
 270 | Francois Delacour    |     2.2
 113 | How About Now        |     7.7
 238 | Isiah Deville        |       0
 387 | LEIMA                |       0
 236 | The Lilies           |       0
 146 | KM Kid               |       0
 358 | The Fireman          |       0
 124 | Michael James        |       0
 353 | Chitty Bang          |       0
 335 | JP Edwards           |       0
 144 | Alexandre Millet     |       0
 173 | The Sunny Boys       |       0
 163 | 22 Seconds to Boston |       0
 354 | No Box               |       0
 200 | The Kicking Eyes     |       0
 231 | John Williams        |       0
 281 | Ayden Page           |       0
 359 | Kicking Dolls        |       0
 118 | Alice In The Rain    |       0
 177 | Greg McMiller        |       0
 373 | David Olsen          |       0
  71 | Zephiro              |       0
 145 | Purpleplum           |      10
 143 | Lara O Mara          |      10
 171 | The Wheelies         |      10
(33 rows)
```

```
 uid |      stage_name      | balance
-----+----------------------+---------
  89 | Zaryah               |    2.75
  13 | Emma Della           |    2.85
 334 | NMN                  |    2.85
 329 | Lil Diggity          |    4.75
 246 | Alec Wood            |    2.75
  27 | Justin B             |       0
 216 | Dutch Dabber         |       0
 315 | Aya Nova             |   11.15
 270 | Francois Delacour    |     2.2
 113 | How About Now        |     7.7
 236 | The Lilies           |       0
 146 | KM Kid               |       0
 124 | Michael James        |       0
 335 | JP Edwards           |       0
 144 | Alexandre Millet     |       0
 173 | The Sunny Boys       |       0
 163 | 22 Seconds to Boston |       0
 200 | The Kicking Eyes     |       0
 231 | John Williams        |       0
 281 | Ayden Page           |       0
 118 | Alice In The Rain    |       0
 177 | Greg McMiller        |       0
 373 | David Olsen          |       0
  71 | Zephiro              |       0
 145 | Purpleplum           |      10
 143 | Lara O Mara          |      10
 171 | The Wheelies         |      10
 238 | Isiah Deville        |       3
 387 | LEIMA                |       3
 358 | The Fireman          |       3
 353 | Chitty Bang          |       3
 354 | No Box               |       3
 359 | Kicking Dolls        |       3
(33 rows)
```

4 - This query adds a * in every username of listeners who don't have any songs yet in their libraries. They are considered as “new” users. This is for future reference (reduction for first order, promos for new users...).

SQL Statement

```sql
UPDATE users
SET username = CONCAT('*', username)
FROM has h, libraries l
WHERE users.uid = h.listener_id AND h.lib_id = l.lib_id AND l.num_songs = 0;
```

Script illustrating execution

```
cs421=> UPDATE users
cs421-> SET username = CONCAT('*', username)
cs421-> FROM has h, libraries l
cs421-> WHERE users.uid = h.listener_id AND h.lib_id = l.lib_id AND l.num_songs = 0;
UPDATE 50
cs421=> SELECT * FROM users LIMIT 55;
```

Output before

```
cs421=>  SELECT * FROM users LIMIT 20 OFFSET 40;
 uid |      name       | username |    country    |             email              |    dob     |   credit_info
-----+-----------------+----------+---------------+--------------------------------+------------+------------------
 164 | Tara Markoski   | Tar227   | Russia        | Tara.Markoski@yahoo.com        | 1989-08-28 | 3101055029413001
 365 | Theo Wilson     | The376   | Bolivia       | Theo.Wilson@mail.mcgill.ca     | 1962-06-04 | 4889664053371458
 179 | Magnus Angelo   | Mag130   | Egypt         | Magnus.Angelo@mail.mcgill.ca   | 1993-01-10 | 6260165498455396
 320 | Sam Kourilsky   | Sam93    | China         | Sam.Kourilsky@mail.mcgill.ca   | 2001-01-29 | 5428624173109136
 253 | Janna Jones     | Jan280   | United States | Janna.Jones@gmail.com          | 2000-07-07 | 6073385773012968
  34 | Sam Nguyen      | Sam113   | United States | Sam.Nguyen@yahoo.com           | 2004-01-18 | 6439396058513541
 108 | Karim Porta     | Kar99    | Egypt         | Karim.Porta@gmail.com          | 1984-10-14 | 8701458748485604
 310 | Olive Wilson    | Oli140   | Mexico        | Olive.Wilson@gmail.com         | 1974-12-10 | 1779844453333643
 168 | Misgana Porta   | Mis298   | Vietnam       | Misgana.Porta@gmail.com        | 2007-07-02 | 8266570000088955
 185 | Max Williams    | Max186   | Italy         | Max.Williams@gmail.com         | 1977-11-16 | 2259058454161636
   5 | Nova Wilson     | Nov358   | Italy         | Nova.Wilson@yahoo.com          | 1959-11-16 | 4192954663623455
  84 | Olive Meijer    | Oli209   | England       | Olive.Meijer@mail.mcgill.ca    | 1978-05-16 | 7747111133317960
 128 | Markus Meijer   | Mar301   | Spain         | Markus.Meijer@yahoo.com        | 1997-05-03 | 2845996360532188
 100 | Theo Meijer     | The195   | England       | Theo.Meijer@gmail.com          | 1996-10-20 | 9266496703144215
 302 | Kian Patel      | Kia375   | Spain         | Kian.Patel@mail.mcgill.ca      | 1959-02-02 | 9683773525587129
 197 | Lili Ling       | Lil355   | China         | Lili.Ling@yahoo.com            | 1958-03-24 | 5358444561495649
 360 | Nova Giordano   | Nov29    | Canada        | Nova.Giordano@mail.mcgill.ca   | 1970-03-15 | 8326956840879192
 136 | Elio Blaese     | Eli350   | Russia        | Elio.Blaese@mail.mcgill.ca     | 1986-03-02 | 2762587828025372
 314 | Olive Giordano  | Oli234   | Bolivia       | Olive.Giordano@mail.mcgill.ca  | 1981-06-20 | 1835229111594634
 317 | Roshane Johnson | Ros91    | France        | Roshane.Johnson@mail.mcgill.ca | 1979-09-29 | 6599740358581336
(20 rows)
```

Output after

```
cs421=> SELECT * FROM users LIMIT 20 OFFSET 40;
 uid |      name       | username |    country    |             email              |    dob     |   credit_info
-----+-----------------+----------+---------------+--------------------------------+------------+------------------
 164 | Tara Markoski   | Tar227   | Russia        | Tara.Markoski@yahoo.com        | 1989-08-28 | 3101055029413001
 365 | Theo Wilson     | The376   | Bolivia       | Theo.Wilson@mail.mcgill.ca     | 1962-06-04 | 4889664053371458
 179 | Magnus Angelo   | Mag130   | Egypt         | Magnus.Angelo@mail.mcgill.ca   | 1993-01-10 | 6260165498455396
 320 | Sam Kourilsky   | Sam93    | China         | Sam.Kourilsky@mail.mcgill.ca   | 2001-01-29 | 5428624173109136
 253 | Janna Jones     | Jan280   | United States | Janna.Jones@gmail.com          | 2000-07-07 | 6073385773012968
  34 | Sam Nguyen      | Sam113   | United States | Sam.Nguyen@yahoo.com           | 2004-01-18 | 6439396058513541
 108 | Karim Porta     | Kar99    | Egypt         | Karim.Porta@gmail.com          | 1984-10-14 | 8701458748485604
 310 | Olive Wilson    | Oli140   | Mexico        | Olive.Wilson@gmail.com         | 1974-12-10 | 1779844453333643
 168 | Misgana Porta   | Mis298   | Vietnam       | Misgana.Porta@gmail.com        | 2007-07-02 | 8266570000088955
 185 | Max Williams    | Max186   | Italy         | Max.Williams@gmail.com         | 1977-11-16 | 2259058454161636
   5 | Nova Wilson     | *Nov358  | Italy         | Nova.Wilson@yahoo.com          | 1959-11-16 | 4192954663623455
  84 | Olive Meijer    | *Oli209  | England       | Olive.Meijer@mail.mcgill.ca    | 1978-05-16 | 7747111133317960
 128 | Markus Meijer   | *Mar301  | Spain         | Markus.Meijer@yahoo.com        | 1997-05-03 | 2845996360532188
 100 | Theo Meijer     | *The195  | England       | Theo.Meijer@gmail.com          | 1996-10-20 | 9266496703144215
 302 | Kian Patel      | *Kia375  | Spain         | Kian.Patel@mail.mcgill.ca      | 1959-02-02 | 9683773525587129
 197 | Lili Ling       | *Lil355  | China         | Lili.Ling@yahoo.com            | 1958-03-24 | 5358444561495649
 360 | Nova Giordano   | *Nov29   | Canada        | Nova.Giordano@mail.mcgill.ca   | 1970-03-15 | 8326956840879192
 136 | Elio Blaese     | *Eli350  | Russia        | Elio.Blaese@mail.mcgill.ca     | 1986-03-02 | 2762587828025372
 314 | Olive Giordano  | *Oli234  | Bolivia       | Olive.Giordano@mail.mcgill.ca  | 1981-06-20 | 1835229111594634
 317 | Roshane Johnson | *Ros91   | France        | Roshane.Johnson@mail.mcgill.ca | 1979-09-29 | 6599740358581336
(20 rows)
```

(Here the view is listed from row 40 to row 60 from the users table)

Q7. Views

1 - This table shows the 10 most famous artists so far on the platform. They are ranked in terms of the number of sales/downloads that have been made so far from the listeners. Specifically, this table displays the stage name of the artist and the count (ie number of articles ordered by listeners).

SQL Statement

```
CREATE VIEW Top10Artists AS
SELECT art.stage_name, COUNT(a.article_id)
FROM isAddedTo A, releases R, artists art
WHERE A.article_id = R.article_id AND art.uid = R.artist_id
GROUP BY art.stage_name
ORDER BY COUNT(a.article_id) DESC
LIMIT 10;
```

System response

```
    stage_name     | count
-------------------+-------
 How About Now     |    18
 Aya Nova          |    15
 Lil Diggity       |     7
 Purpleplum        |     7
 Alec Wood         |     6
 Zaryah            |     6
 Francois Delacour |     5
 Dutch Dabber      |     5
 NMN               |     4
 Emma Della        |     4
(10 rows)
```

2 - This table shows the country all the listeners are from in descending order.

SQL Statement

```
CREATE VIEW TopCountries AS
SELECT country, COUNT(users.uid)
FROM users, listeners
WHERE users.uid = listeners.uid
GROUP BY country
ORDER BY COUNT(users.uid) DESC;
```

System response

```
cs421=> SELECT * FROM topcountries;
    country    | count
---------------+-------
 Canada        |     8
 China         |     7
 Egypt         |     7
 United States |     6
 England       |     6
 Bolivia       |     6
 Russia        |     5
 Vietnam       |     5
 Italy         |     5
 Spain         |     5
 France        |     4
 Mexico        |     3
(12 rows)
```

SQL UPDATE

We tried running an UPDATE statement on both of our views. We got the following 2 error messages :

```
ERROR:  cannot update view "top10artists"
DETAIL:  Views containing GROUP BY are not automatically updatable.
HINT:  To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional ON UPDATE DO INSTEAD rule.
```

```
ERROR:  cannot update view "topcountries"
DETAIL:  Views containing GROUP BY are not automatically updatable.
HINT:  To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional ON UPDATE DO INSTEAD rule.
```

Our views are not updatable since they both contain group-by and aggregated fields, as well as the “top 10 artists” view is made from multiple tables.

In order to be able to update a view, the view needs to be made from a single table with its primary(/ies) key(s), and should not contain any group-by and aggregated fields.

Q8. Check Constraints

1. **Add a constraint restricting the maximum price of an article to 9.0.** This ensures that all our prices remain competitive while being in a reasonable range for independent artists.

Constraint Addition Query:

```
ALTER TABLE articles ADD CONSTRAINT maxPrice
CHECK (price <= 9.0);
```

Revised Schema:

```
            Table "cs421g40.articles"
    Column    |         Type          | Modifiers
--------------+-----------------------+-----------
 article_id   | integer               | not null
 title        | character varying(50) |
 release_date | date                  |
 price        | real                  |
 genre        | character varying(30) |
Indexes:
    "articles_pkey" PRIMARY KEY, btree (article_id)
Check constraints:
    "maxprice" CHECK (price <= 9.0::double precision)
Referenced by:
    TABLE "albums" CONSTRAINT "albums_article_id_fkey" FOREIGN KEY (article_id) REFERENCES articles(article_id)
    TABLE "contains" CONSTRAINT "contains_article_id_fkey" FOREIGN KEY (article_id) REFERENCES articles(article_id)
    TABLE "isaddedto" CONSTRAINT "isaddedto_article_id_fkey" FOREIGN KEY (article_id) REFERENCES articles(article_id)
    TABLE "releases" CONSTRAINT "releases_article_id_fkey" FOREIGN KEY (article_id) REFERENCES articles(article_id)
    TABLE "songs" CONSTRAINT "songs_article_id_fkey" FOREIGN KEY (article_id) REFERENCES articles(article_id)
```

Violating Update:

```
INSERT INTO articles(article_id,title,genre,release_date,price) values (989,'No Country for Old Men','Country','2019-10-19',9.55);
```

Response of database to modifications that violate constraints:

```
        ^
cs421=> INSERT INTO articles (article_id,title,genre,release_date,price) values (989,'No Country for Old Men','Country','2019-10-19',9.55);
ERROR:  new row for relation "articles" violates check constraint "maxprice"
DETAIL:  Failing row contains (989, No Country for Old Men, 2019-10-19, 9.55, Country).
```

2. **Add a constraint restricting minimum order total to 0 (no negative pricing).**

Constraint Addition Query:

```
ALTER TABLE shopping_carts ADD CONSTRAINT minPrice
CHECK (total_amount >= 0);
```

Revised Schema:

```
  Table "cs421g40.shopping_carts"
    Column     |  Type   | Modifiers
---------------+---------+-----------
 order_id      | integer | not null
 total_amount  | real    |
 num_articles  | integer |
Indexes:
    "shopping_carts_pkey" PRIMARY KEY, btree (order_id)
Check constraints:
    "minprice" CHECK (total_amount >= 0::double precision)
Referenced by:
    TABLE "contains" CONSTRAINT "contains_order_id_fkey" FOREIGN KEY (order_id) REFERENCES shopping_carts(order_id)
    TABLE "moneyearned" CONSTRAINT "moneyearned_order_id_fkey" FOREIGN KEY (order_id) REFERENCES shopping_carts(order_id)
    TABLE "uses" CONSTRAINT "uses_order_id_fkey" FOREIGN KEY (order_id) REFERENCES shopping_carts(order_id)
```

Violating Update:

```
INSERT INTO shopping_carts(order_id,total_amount,num_articles) values (98,-1,3);
```

Response of database to modifications that violate constraints:

```
cs421=> INSERT INTO shopping_carts(order_id,total_amount,num_articles) values (98,-1,3);
ERROR:  new row for relation "shopping_carts" violates check constraint "minprice"
DETAIL:  Failing row contains (98, -1, 3).
```

Q9. Creativity

- We wrote a script (See **generateData.py)** that generates data and insert statements. When being executed, the program displays a little menu where you can choose to either generate insert statements (for a specific table or for all tables at once) or to generate random user data. ArticleID generation is also available. The program does display a warning when someone asks to generate random user data as it will automatically update the csv datasets and could make the current insert statements obsolete.
- Our data for each dataset is very realistic as you may see in our *users*, *articles* and *artists* datasets. For the *users*, each data was generated using our script, with realistic names, emails, countries, etc. For *articles* and *artists*, all the song names are real and we took the liberty to personalize the artist names and album names to enhance that realness. The *users* and *articles* datasets both have <= 100 entries.
- We made sure that some of our queries would involve at least 3 tables and would use comparisons and more elaborate functions such as COUNT and SUM.