```
-- Ameya Gaitonde

-- NOTE: I couldn't get the "psql -a -f" command to work in my
Terminal, so I put
-- the results.txt file together manually. This file contains the SQL
commands with
-- their output from "\i hw1.sql" directly below the command. Thanks.

-- 1. Table Creation and Analysis --

-- SQL DDL statements

DROP SCHEMA IF EXISTS ZotMusic CASCADE;
NOTICE:  drop cascades to 10 other objects
DETAIL:  drop cascades to table zotmusic.users
drop cascades to table zotmusic.artists
drop cascades to table zotmusic.listeners
drop cascades to table zotmusic.records
drop cascades to table zotmusic.singles
drop cascades to table zotmusic.albums
drop cascades to table zotmusic.songs
drop cascades to table zotmusic.sessions
drop cascades to table zotmusic.reviews
drop cascades to table zotmusic.reviewlikes
DROP SCHEMA

CREATE SCHEMA ZotMusic;
CREATE SCHEMA

SET search_path TO ZotMusic;
SET

CREATE TABLE Users (
    user_id        text NOT NULL,
    email          text NOT NULL,
    joined_date    date NOT NULL,
    nickname       text NOT NULL,
    street         text,
    city           text,
    state          text,
    zip            text,
    genres         text,
    PRIMARY KEY (user_id)
);
CREATE TABLE

CREATE TABLE Artists (
    user_id        text,
    bio            text,
    stagename      text,
```

```
    PRIMARY KEY (user_id),
    FOREIGN KEY (user_id) REFERENCES Users (user_id) ON DELETE CASCADE
);
CREATE TABLE

CREATE TABLE Listeners (
    user_id        text,
    subscription   text,
    first_name     text NOT NULL,
    last_name      text NOT NULL,
    PRIMARY KEY (user_id),
    FOREIGN KEY (user_id) REFERENCES Users (user_id) ON DELETE
CASCADE,
    CHECK (subscription IN ('free', 'monthly', 'yearly'))
);
CREATE TABLE

CREATE TABLE Records (
    record_id      text NOT NULL,
    artist_user_id text NOT NULL,
    title          text NOT NULL,
    genre          text NOT NULL,
    release_date   date NOT NULL,
    PRIMARY KEY (record_id),
    FOREIGN KEY (artist_user_id) REFERENCES Artists (user_id) ON
DELETE CASCADE
);
CREATE TABLE

CREATE TABLE Singles (
    record_id      text NOT NULL,
    video_url      text,
    PRIMARY KEY (record_id),
    FOREIGN KEY (record_id) REFERENCES Records (record_id) ON DELETE
CASCADE
);
CREATE TABLE

CREATE TABLE Albums (
    record_id      text NOT NULL,
    description    text,
    PRIMARY KEY (record_id),
    FOREIGN KEY (record_id) REFERENCES Records (record_id) ON DELETE
CASCADE
);
CREATE TABLE

CREATE TABLE Songs (
    record_id      text NOT NULL,
    track_number   int NOT NULL,
```

```
    title          text NOT NULL,
    length         int NOT NULL,
    bpm            int,
    mood           text,
    PRIMARY KEY (record_id, track_number),
    FOREIGN KEY (record_id) REFERENCES Records (record_id) ON DELETE
CASCADE
);
CREATE TABLE

CREATE TABLE Sessions (
    session_id        text NOT NULL,
    user_id           text NOT NULL,
    record_id         text NOT NULL,
    track_number      int NOT NULL,
    initiate_at       timestamp NOT NULL,
    leave_at          timestamp NOT NULL,
    music_quality     text NOT NULL,
    device            text NOT NULL,
    remaining_time    int NOT NULL,
    replay_count      int,
    PRIMARY KEY (session_id),
    FOREIGN KEY (user_id) REFERENCES Listeners(user_id) ON DELETE
CASCADE,
    FOREIGN KEY (record_id, track_number) REFERENCES Songs(record_id,
track_number) ON DELETE CASCADE
);
CREATE TABLE

CREATE TABLE Reviews (
    review_id     text NOT NULL,
    user_id       text NOT NULL,
    record_id     text NOT NULL,
    rating        int NOT NULL,
    body          text,
    posted_at     timestamp NOT NULL,
    PRIMARY KEY (review_id),
    FOREIGN KEY (user_id) REFERENCES Listeners (user_id) ON DELETE
CASCADE,
    FOREIGN KEY (record_id) REFERENCES Records (record_id) ON DELETE
CASCADE
);
CREATE TABLE

CREATE TABLE ReviewLikes(
    user_id text NOT NULL,
    review_id text NOT NULL,
    PRIMARY KEY (user_id, review_id),
    FOREIGN KEY (user_id) REFERENCES Listeners(user_id) ON DELETE
CASCADE,
```

```
    FOREIGN KEY (review_id) REFERENCES Reviews(review_id) ON DELETE
CASCADE
);
CREATE TABLE


-- Songs table field design observations

/*
record_id: Part of primary key, so this must be NOT NULL. Stored as a
string so its datatype is "text".

track_number: Part of primary key, so this must be NOT NULL. Stored as
integer so its datatype is "int".

title: When a song exists, it must have a title, therefore this field
must be NOT NULL. Stored as a string, so its datatype is "text".

length: When a song exists, it must have a length, so this field must
be NOT NULL. Stored as an integer, so its datatype is "int".

bpm: Don't necessarily need to always know the song's bpm, so this
field can be null. Stored as an integer, so its datatype is "int".

mood: Don't necessarily need to always know the mood, so this can be
null. Stored as a string, so its datatype is "text".
*/


-- 2. Data Loading (COPY commands) --


COPY users (user_id, email, joined_date, nickname, street, city,
state, zip, genres)
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/Users.csv'
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 200




COPY artists (user_id, bio, stagename)
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/Artists.csv'
```

```
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 116



COPY listeners (user_id, subscription, first_name, last_name)
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/Listeners.csv'
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 184



COPY records (record_id, artist_user_id, title, genre, release_date)
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/Records.csv'
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 1000



COPY singles (record_id, video_url)
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/Singles.csv'
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 300



COPY albums (record_id, description)
```

```
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/Albums.csv'
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 700



COPY songs (record_id, track_number, title, length, bpm, mood)
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/Songs.csv'
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 6252



COPY sessions (session_id, user_id, record_id, track_number,
initiate_at, leave_at, music_quality,
                        device, remaining_time, replay_count)
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/Sessions.csv'
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 50000



COPY reviews (review_id, user_id, record_id, rating, body, posted_at)
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/Reviews.csv'
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 9499
```

```
COPY reviewlikes (user_id, review_id)
FROM '/Applications/PostgreSQL 17/Big Data HW 1/zot-music-dataset-
assignment1/ReviewLikes.csv'
WITH (
    FORMAT csv,
    HEADER true,
    DELIMITER ',',
    NULL ''
);
COPY 91325



-- 3. Query Answers --



-- Problem A --

select 'num_users' as table_name, COUNT(*) as totals
from users
union all
select 'num_records' as table_name, COUNT(*) as totals
from records
union all
select 'num_reviews' as table_name, COUNT(*) as totals
from reviews;

 table_name  | totals
-------------+--------
 num_users   |    200
 num_records |   1000
 num_reviews |   9499
(3 rows)


-- Problem B --

select users.user_id, users.email, users.nickname, users.zip
from users join artists on users.user_id = artists.user_id
        join listeners on users.user_id = listeners.user_id
where users.email like '%icloud.com%'
;
                  user_id                  |           email
|    nickname    |  zip
-------------------------------------------
+--------------------------+---------------+-------
```

```
 user_10dfa3b6-52b6-43a9-835a-ad110ad50ff2 | roberthammond@icloud.com
| roberthammond  | 54869
 user_4a9ffbf6-5430-45a4-b68e-0ae6f6737d8b | william09@icloud.com
| william09      | 55308
 user_457ad608-9661-4384-9919-1d89c52fd0de | danielharrison@icloud.com
| danielharrison | 23703
 user_d383327a-dd9c-4a4f-bbaf-262c7a6d90a0 | chelsealawson@icloud.com
| chelsealawson  | 10206
 user_83a40c0c-573e-44ec-8cac-b5951513f88b | turnerkayla@icloud.com
| turnerkayla    | 02182
 user_38eaa9f8-e8fc-4ce4-a8ae-ffb882c1786c | ryanmorgan@icloud.com
| ryanmorgan     | 95166
 user_3c5d30bc-0ac2-4df1-8574-892d2f666df6 | browncarrie@icloud.com
| browncarrie    | 23550
(7 rows)


-- Problem C --

select records.record_id, records.title, records.genre,
records.release_date
from records join artists on records.artist_user_id = artists.user_id
        join users on records.artist_user_id = users.user_id
        where users.email like '%fwilson@outlook.com%'
order by records.release_date asc
;
                  record_id                  |            title
|  genre  | release_date
---------------------------------------------
+----------------------------+---------+--------------
 record_91c6325d-b17f-4f4c-be6d-3517b2173a9f | Statement matter
| Country | 2020-01-12
 record_822961a3-946a-49ff-8173-74d4035286b9 | Apply size
| Gospel  | 2020-01-29
 record_cbf93efd-2deb-48ae-ad73-83aa088c6f13 | Democratic what
| Soul    | 2020-03-27
 record_2406e933-23e3-4db1-acf9-3c863d48bff6 | General job heavy
| Country | 2020-05-08
 record_57061d35-de20-4bf1-9aac-a689f0db7e16 | Would determine
| Soul    | 2020-06-07
 record_3e4ed054-cf1a-4a04-8e97-e0177c6d3575 | Summer civil political
beat | Folk    | 2021-03-31
 record_116fbdd6-e706-41f7-9809-12e174e48e8f | Discover rate
| Jazz    | 2021-09-09
 record_62389a63-e95f-43d1-acea-aa1bac0e0050 | Result guess for
| Gospel  | 2021-10-17
 record_5cbf14c7-7b54-4e32-bfce-cba507c7277f | Bar talk long
| Jazz    | 2021-10-23
(9 rows)
```

```
-- Problem D --

select users.user_id, records.genre,
    count(albums.record_id) as album_count,
    count(singles.record_id) as single_count
from records join artists on records.artist_user_id = artists.user_id
join users on records.artist_user_id = users.user_id
left join albums on records.record_id = albums.record_id
left join singles on records.record_id = singles.record_id
where users.email like '%fwilson@outlook.com%'
group by users.user_id, records.genre;

                 user_id                  |  genre  | album_count |
single_count
------------------------------------------+---------+-------------
+--------------
 user_6ac27408-a0a6-4c57-a025-7b6854f7a8e3 | Country |           1 |
1
 user_6ac27408-a0a6-4c57-a025-7b6854f7a8e3 | Folk    |           1 |
0
 user_6ac27408-a0a6-4c57-a025-7b6854f7a8e3 | Gospel  |           2 |
0
 user_6ac27408-a0a6-4c57-a025-7b6854f7a8e3 | Jazz    |           1 |
1
 user_6ac27408-a0a6-4c57-a025-7b6854f7a8e3 | Soul    |           1 |
1
(5 rows)


-- Problem E --

select artists.stagename, users.email
from users join artists on users.user_id = artists.user_id
        join records on artists.user_id = records.artist_user_id
group by artists.stagename, users.email
having count(distinct records.genre) >= 9;

  stagename    |             email
---------------+-------------------------------
 blakeshannon  | william09@icloud.com
 ehester       | william57@mail.com
 elizabeth55   | powerschristopher@foxmail.com
 khall         | lthompson@college.edu
 richardsbilly | robert92@outlook.com
 sandersallison | keith65@university.edu
(6 rows)


-- Problem F --
```

```
select users.user_id, users.email
from users
where
    users.genres like '%R&B%' and
    users.genres like '%Hip-Hop%' and
    users.genres not like '%Indie%' and
    users.genres not like '%Jazz%'
order by users.user_id asc;
```

```
                  user_id                  |        email
-------------------------------------------+----------------------
 user_27895cb3-a325-4d85-8e2f-f442847f56f1 | callen@icloud.com
 user_a729dc21-7341-44cd-ac50-5a43022216b5 | sjohnston@yahoo.com
 user_bf4b7630-b9dd-40f0-b534-ef841ea43194 | linda38@outlook.com
 user_d466a1ce-756e-497a-a329-96f5b5e815ad | dlawrence@icloud.com
(4 rows)
```

```
-- Problem G --

select users.email, users.nickname,
array_length(string_to_array(users.genres, ','), 1) as num_genres
from users
order by num_genres desc
limit 10;
```

```
            email             |   nickname    | num_genres
------------------------------+---------------+------------
 courtney36@protonmail.com    | courtney36    |         10
 charleslewis@university.edu  | charleslewis  |         10
 ewilliams@mail.com           | ewilliams     |         10
 zmason@gmail.com             | zmason        |         10
 gomezbrittany@foxmail.com    | gomezbrittany |          9
 bknapp@icloud.com            | bknapp        |          9
 ryanmorgan@icloud.com        | ryanmorgan    |          9
 edwardscindy@foxmail.com     | edwardscindy  |          8
 joel00@gmail.com             | joel00        |          8
 gclayton@protonmail.com      | gclayton      |          8
(10 rows)
```

```
-- Problem H --

select UNNEST(string_to_array(users.genres, ',')) as genre,
       COUNT(users.genres) as num_users
from users
group by genre
order by num_users desc;
```

```
   genre    | num_users
------------+-----------
 Soul       |        58
 Techno     |        56
 Indie      |        54
 Folk       |        53
 Blues      |        53
 Funk       |        49
 Country    |        49
 Classical  |        47
 R&B        |        47
 Metal      |        47
 Jazz       |        47
 Disco      |        46
 Hip-Hop    |        45
 Latin      |        45
 Pop        |        44
 Gospel     |        43
 Punk       |        42
 Electronic |        38
 Rock       |        33
 Reggae     |        27
(20 rows)


-- Problem I --

select count(distinct sessions.user_id) as num_listeners,
           sessions.track_number,
           songs.title as song_title,
           records.title as record_title
from sessions join songs on sessions.record_id = songs.record_id
                                                 and
sessions.track_number = songs.track_number
         join records on sessions.record_id = records.record_id
where sessions.replay_count >= 1 and sessions.remaining_time < 0.2 *
songs.length
group by sessions.track_number, songs.title, records.title
order by num_listeners desc
limit 10;

 num_listeners | track_number |      song_title      |
record_title
---------------+--------------+----------------------
+----------------------
             6 |            2 | Push ever blood      | Article bit its
edge
             5 |            4 | Fire sing            | Mother for day
             5 |            1 | People owner end     | People owner
end
```

```
             5 |           2 | Its thank dark       | Soldier couple
             5 |           3 | Enjoy major          | Mother for day
             5 |           3 | Service our although | Indicate
             5 |           2 | Area                 | Special
             5 |           1 | Success course       | Than then near
             5 |           2 | Education remain     | Eye TV
             5 |           5 | Last there look      | Tell live
(10 rows)



-- Problem J --

-- Created some supporting tables. The following tables calculate the
weights for the
-- weighted average as upvotes + 1. Then we join the tables as needed
to find the
-- relevant weighted averages.

CREATE TABLE reviewWeights AS
select COUNT(reviewlikes.user_id) + 1 as upvote_weight,
reviewlikes.review_id
from reviewlikes
group by reviewlikes.review_id;
SELECT 9301

CREATE TABLE sample as
select reviewWeights.upvote_weight, reviews.record_id, reviews.rating
from reviewWeights join reviews on reviewWeights.review_id =
reviews.review_id
;
SELECT 9301

CREATE TABLE RR AS
select
    sample.record_id,
    SUM(sample.upvote_weight * sample.rating) /
SUM(sample.upvote_weight) AS rating
from sample
group by
    sample.record_id;
SELECT 1000


-- View DDL:

CREATE VIEW RatedRecords as
select
        RR.record_id,
        RR.rating,
```

```
        sample.upvote_weight - 1 as num_reviews,
        records.title as title
from RR join sample on RR.record_id = sample.record_id
         join records on RR.record_id = records.record_id
;
CREATE VIEW

-- View test query:
select * from RatedRecords
where RatedRecords.num_reviews >= 5
order by RatedRecords.rating desc
limit 10;

                record_id                  |       rating       |
num_reviews |        title
-------------------------------------------+--------------------
+-------------+----------------------
 record_19c5ba97-db80-4c0b-ae07-097683659ef5 | 5.0000000000000000 |
19 | Want good
 record_39a5d646-1633-4505-827b-c7f0895571db | 5.0000000000000000 |
16 | Discuss effort
 record_4b3efe96-4c8d-4385-bf05-6588a72e8585 | 5.0000000000000000 |
5 | Pull coach
 record_fc42ae17-696f-4817-8026-32964f2f474c | 5.0000000000000000 |
14 | Provide dinner help
 record_60229089-69cc-48f9-8a4a-38a66a24e799 | 4.6842105263157895 |
15 | Surface thousand
 record_474e6dd0-6c15-4bbb-b453-4eeeb5d0f7ac | 4.4666666666666667 |
8 | Lawyer run beautiful
 record_474e6dd0-6c15-4bbb-b453-4eeeb5d0f7ac | 4.4666666666666667 |
15 | Lawyer run beautiful
 record_474e6dd0-6c15-4bbb-b453-4eeeb5d0f7ac | 4.4666666666666667 |
7 | Lawyer run beautiful
 record_474e6dd0-6c15-4bbb-b453-4eeeb5d0f7ac | 4.4666666666666667 |
11 | Lawyer run beautiful
 record_de69e2da-305e-4e14-be97-0b264fb061aa | 4.4117647058823529 |
9 | Reduce
(10 rows)


-- Problem K --

-- Table alteration DDL:
alter table records
add column rating DECIMAL(3, 2);
ALTER TABLE

-- Table update query:
update records
set rating = rrec.rating
```

```
from RatedRecords rrec
where records.record_id = rrec.record_id;
UPDATE 1000

-- Change verification query:
select r.rating, rrec.*
from records r join RatedRecords rrec on r.record_id = rrec.record_id
where rrec.num_reviews >= 5
order by rrec.rating desc
limit 10;

rating |                  record_id                  |       rating
| num_reviews |        title
--------+---------------------------------------------
+--------------------+------------+---------------------
   5.00 | record_19c5ba97-db80-4c0b-ae07-097683659ef5 |
5.0000000000000000 |          19 | Want good
   5.00 | record_39a5d646-1633-4505-827b-c7f0895571db |
5.0000000000000000 |          16 | Discuss effort
   5.00 | record_4b3efe96-4c8d-4385-bf05-6588a72e8585 |
5.0000000000000000 |           5 | Pull coach
   5.00 | record_fc42ae17-696f-4817-8026-32964f2f474c |
5.0000000000000000 |          14 | Provide dinner help
   4.68 | record_60229089-69cc-48f9-8a4a-38a66a24e799 |
4.6842105263157895 |          15 | Surface thousand
   4.47 | record_474e6dd0-6c15-4bbb-b453-4eeeb5d0f7ac |
4.4666666666666667 |           8 | Lawyer run beautiful
   4.47 | record_474e6dd0-6c15-4bbb-b453-4eeeb5d0f7ac |
4.4666666666666667 |          15 | Lawyer run beautiful
   4.47 | record_474e6dd0-6c15-4bbb-b453-4eeeb5d0f7ac |
4.4666666666666667 |           7 | Lawyer run beautiful
   4.47 | record_474e6dd0-6c15-4bbb-b453-4eeeb5d0f7ac |
4.4666666666666667 |          11 | Lawyer run beautiful
   4.41 | record_de69e2da-305e-4e14-be97-0b264fb061aa |
4.4117647058823529 |           9 | Reduce
(10 rows)



-- Problem L --

-- NOTE: I have included only a fraction of the output from the
EXPLAIN command.
-- The numbers from the outputs might not exactly match what I've
written in the
-- comment.

-- Query against view:
-- HashAggregate(cost = 570.94), performs sequential scan.
```

```
EXPLAIN
select artists.user_id, users.nickname, avg(rrec.rating) as
content_rating
from artists
join records on artists.user_id = records.artist_user_id
join RatedRecords rrec on records.record_id = rrec.record_id
join users on artists.user_id = users.user_id
group by artists.user_id, users.nickname
having avg(rrec.rating) >= 3.3;

GroupAggregate  (cost=16810.15..19355.83 rows=33942 width=96)
   Group Key: artists.user_id, users.nickname


select artists.user_id, users.nickname, avg(rrec.rating) as
content_rating
from artists
join records on artists.user_id = records.artist_user_id
join RatedRecords rrec on records.record_id = rrec.record_id
join users on artists.user_id = users.user_id
group by artists.user_id, users.nickname
having avg(rrec.rating) >= 3.3;

                 user_id                  |  nickname  |
content_rating
------------------------------------------+------------
+-------------------
 user_f921401a-7991-4db2-9491-2cb32b4146db | paynedavid |
3.4626999917785777
(1 row)


-- Index DDL:
create index index_rec_rating
on Records (rating);
CREATE INDEX


-- Query against materialized data:
-- HashAggregate(cost = 76.02)

EXPLAIN
select artists.user_id, users.nickname, avg(records.rating) as
content_rating
from artists
join records on artists.user_id = records.artist_user_id
join users on artists.user_id = users.user_id
group by artists.user_id, users.nickname
having avg(records.rating) >= 3.3;
```

```
                                    QUERY PLAN
-------------------------------------------------------------------------------
-------------
 HashAggregate  (cost=100.72..115.72 rows=333 width=96)



select artists.user_id, users.nickname, avg(records.rating) as
content_rating
from artists
join records on artists.user_id = records.artist_user_id
join users on artists.user_id = users.user_id
group by artists.user_id, users.nickname
having avg(records.rating) >= 3.3;

                 user_id                  |    nickname    |
content_rating
------------------------------------------+----------------
+--------------------
 user_f921401a-7991-4db2-9491-2cb32b4146db | paynedavid     |
3.4511111111111111
 user_827369a9-7b0e-4937-917e-632d1ed5620f | robertfigueroa |
3.3412500000000000
(2 rows)



/*
The query against the view takes an average of 120-130 msec to run,
while
the query against the materialized data takes an average of 100-110
msec
to run.

Using the EXPLAIN command, I found that the query on the view resulted
in a
HashAggregate cost = 570.94, while the query on the records resulted
in
HashAggregate cost = 76.02. This is expected because we've created an
index on
the Records table. This allows for faster join operations due to a
lack of need
for sequential row scanning.

If the database grows to have many more sellers, then queries on the
view will
take excessively long to run. As the database grows, we could possibly
improve the
query's performance by using table partitions. This is will further
reduce the amount of
```

```
data that needs to be searched using the index.
*/



-- Problem M --

select
    coalesce(music_quality, 'ALL') AS music_quality,
    coalesce(device, 'ALL') AS device,
    count(session_id) as num_session
from sessions
group by rollup (music_quality, device)
order by num_session desc;


music_quality |     device      | num_session
--------------+-----------------+-------------
 ALL          | ALL             |       50000
 lowest       | ALL             |        8432
 lossless     | ALL             |        8400
 normal       | ALL             |        8380
 low          | ALL             |        8358
 high         | ALL             |        8327
 Hi-Fi        | ALL             |        8103
 lowest       | mobile-app      |        2188
 normal       | desktop-browser |        2156
 high         | mobile-browser  |        2153
 normal       | mobile-app      |        2147
 lowest       | desktop-browser |        2126
 lossless     | desktop-app     |        2115
 lossless     | desktop-browser |        2112
 lossless     | mobile-app      |        2106
 low          | desktop-app     |        2102
 low          | mobile-app      |        2102
 Hi-Fi        | mobile-app      |        2100
 low          | desktop-browser |        2091
 normal       | mobile-browser  |        2087
 lowest       | mobile-browser  |        2075
 high         | desktop-app     |        2075
 lossless     | mobile-browser  |        2067
 low          | mobile-browser  |        2063
 high         | mobile-app      |        2051
 high         | desktop-browser |        2048
 lowest       | desktop-app     |        2043
 Hi-Fi        | mobile-browser  |        2026
 Hi-Fi        | desktop-app     |        2003
 normal       | desktop-app     |        1990
 Hi-Fi        | desktop-browser |        1974
(31 rows)
```