

audition: **what makes a playlist successful?**

Jake Hannan



# set list

- speaking of playlists!
- reviewing the gig
- stack and approach
- exploratory analysis
- putting it all together
- Q&A



# playlists?

- inspired by this challenge to finally analyze my own playlist
- have been curating since 2017
- just reached over 1000 tracks!
- began tracking squabbles (streams) via last.fm in mid 2019

COLLABORATIVE PLAYLIST

## Music I enjoy

Created by [jhannan13](#) • 1,082 songs, 75 hr 52 min

PAUSE



```
def getTrackIds(playlist_id):
    results = sp.playlist_items(playlist_id)
    ids = []
    track_ids = []
    while results['next']:
        results = sp.next(results)
        ids.extend(results['items'])
    for item in ids:
        track = item['track']
        track_ids.append(track['id'])
    return track_ids

track_ids = getTrackIds('18gV5mUAQPGR8tb85iaPrk')

def getTrackFeatures(id):
    meta = sp.track(id)
    features = sp.audio_features(id)

    # meta
    name = meta['name']
    album = meta['album']['name']
    artist = meta['album']['artists'][0]['name']
    release_date = meta['album']['release_date']
    length = meta['duration_ms']
    popularity = meta['popularity']

    # features
    acousticness = features[0]['acousticness']
    danceability = features[0]['danceability']
    energy = features[0]['energy']
    instrumentalness = features[0]['instrumentalness']
    liveness = features[0]['liveness']
    loudness = features[0]['loudness']
    speechiness = features[0]['speechiness']
    tempo = features[0]['tempo']
    time_signature = features[0]['time_signature']
```



125.6

BPM

0.76

Energy

0.09

Speechiness

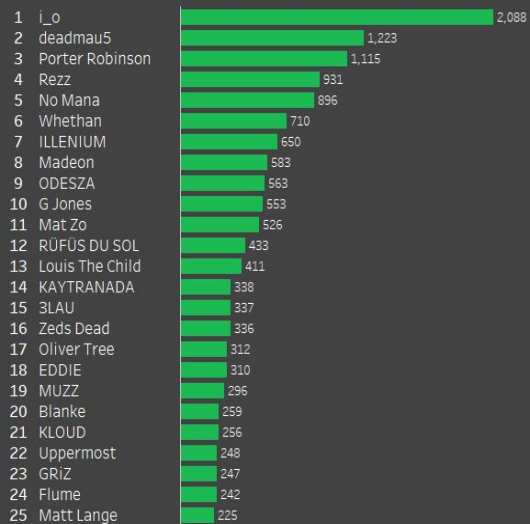
0.61

Danceability

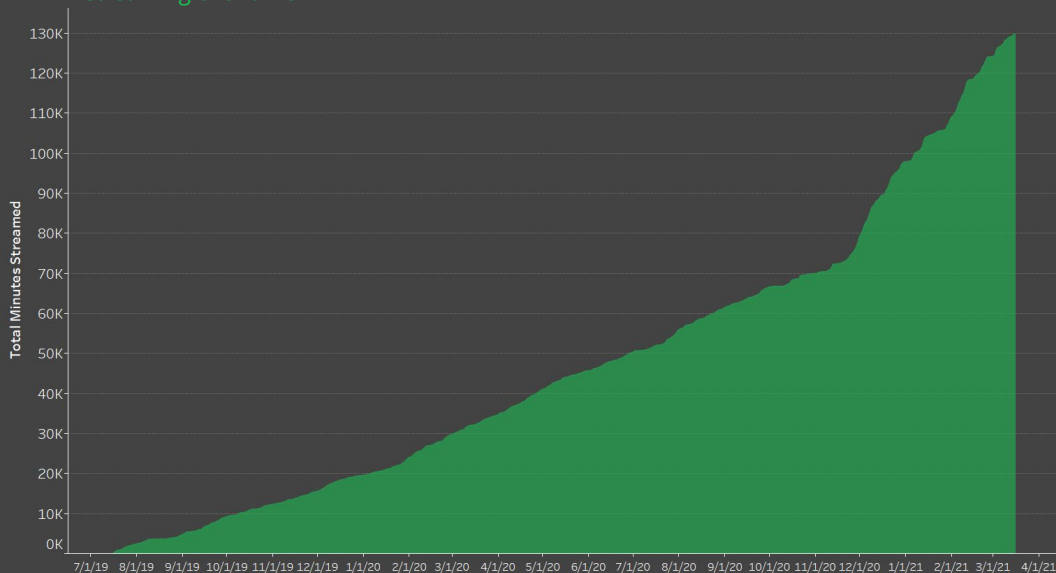
-5.9<sub>db</sub>

Loudness

### top 25 artists by streams (squabbles)



### streaming over time

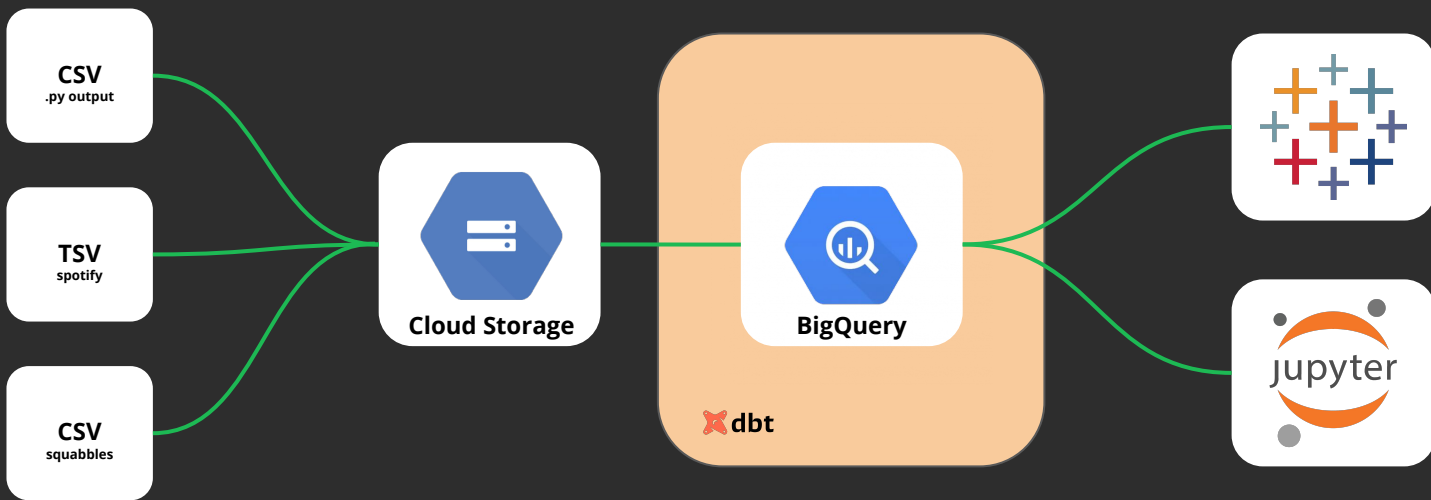


# the gig

You will work with data engineers and data scientists across Spotify, and partner with the FP&A team members, **to ensure that the right information is available, accessible and meaningful**. You will design and develop meaningful analytics, dashboards and executive reporting, and **ensure the FP&A team has access to accurate and impactful data in the Financial Data Lake**.



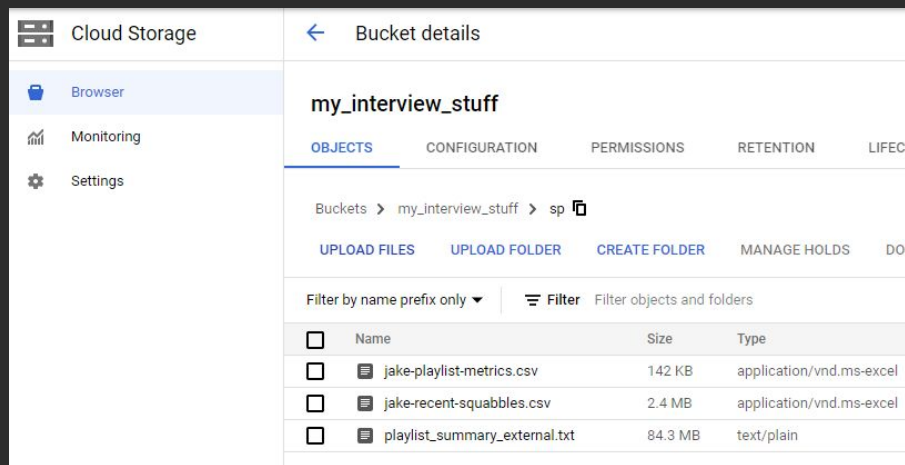
# stack and approach



# stage and load

- created a new bucket on cloud storage
- create bigquery tables from these storage csvs
- auto generate schemas
- confirm successful load!

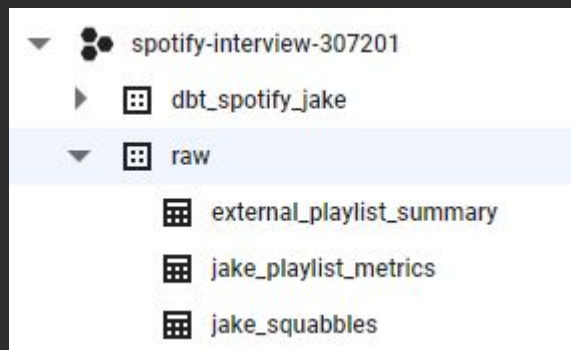
## loading to storage



The screenshot shows the Google Cloud Storage 'Bucket details' page for a bucket named 'my\_interview\_stuff'. The left sidebar contains navigation links: 'Cloud Storage', 'Browser' (selected), 'Monitoring', and 'Settings'. The main content area shows the bucket name and tabs for 'OBJECTS', 'CONFIGURATION', 'PERMISSIONS', 'RETENTION', and 'LIFECYCLE'. Below the tabs, there's a breadcrumb trail 'Buckets > my\_interview\_stuff > sp' and a list of actions: 'UPLOAD FILES', 'UPLOAD FOLDER', 'CREATE FOLDER', 'MANAGE HOLDS', and 'DO'. A filter section shows 'Filter by name prefix only' and a 'Filter' button. A table lists the objects in the bucket:

<input type="checkbox"/>	Name	Size	Type
<input type="checkbox"/>	jake-playlist-metrics.csv	142 KB	application/vnd.ms-excel
<input type="checkbox"/>	jake-recent-squabbles.csv	2.4 MB	application/vnd.ms-excel
<input type="checkbox"/>	playlist_summary_external.txt	84.3 MB	text/plain

## bigquery project structure

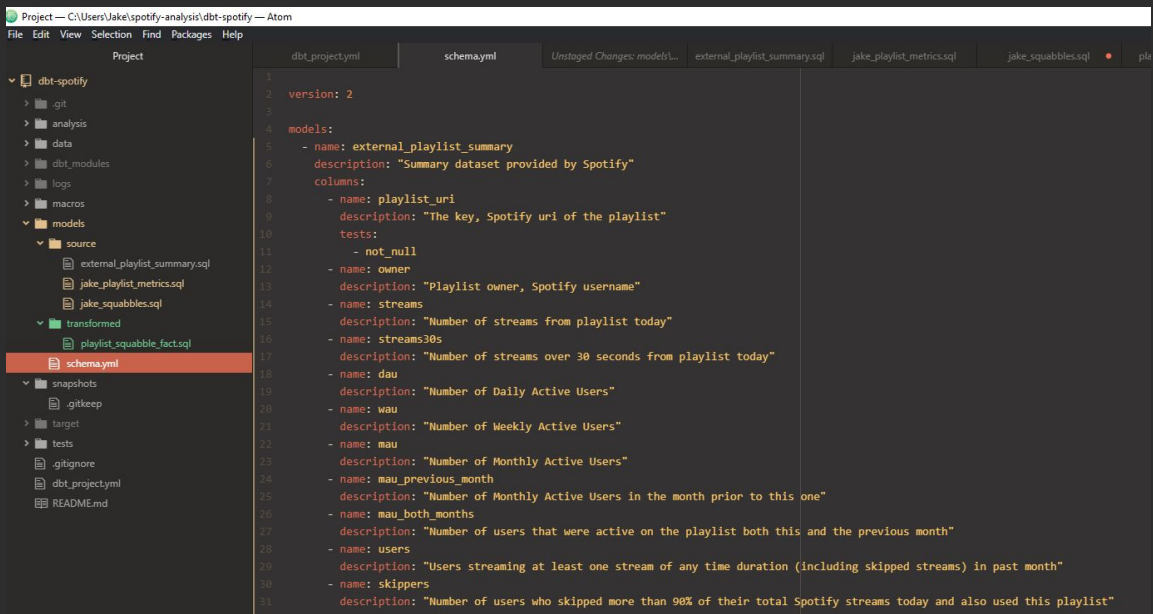


The screenshot shows the BigQuery project structure for 'spotify-interview-307201'. The project is expanded, showing a folder named 'dbt\_spotify\_jake' and a folder named 'raw'. The 'raw' folder is expanded, showing three tables: 'external\_playlist\_summary', 'jake\_playlist\_metrics', and 'jake\_squabbles'.



# trusted with dbt

- core theme - automate and create trust!
- dbt allows us to leverage software development principles with our data transformations (via SQL / YAML)
- ensure that data can be tested, validated, and easily consumed!



The screenshot shows the Atom editor interface with a project titled "Project — C:\Users\Jake\spotify-analysis\dbt-spotify — Atom". The left sidebar displays the project structure, including folders like .git, analysis, data, dbt\_modules, logs, macros, models, snapshots, target, tests, and files like .gitignore, dbt\_project.yml, and README.md. The "models" folder is expanded, showing subfolders "source" and "transformed", and files "external\_playlist\_summary.sql", "jake\_playlist\_metrics.sql", "jake\_squabbles.sql", and "playlist\_squabble\_fact.sql". The "schema.yml" file is selected and highlighted in orange. The main editor area displays the content of "schema.yml", which defines a model named "external\_playlist\_summary" with a description, columns, and tests.

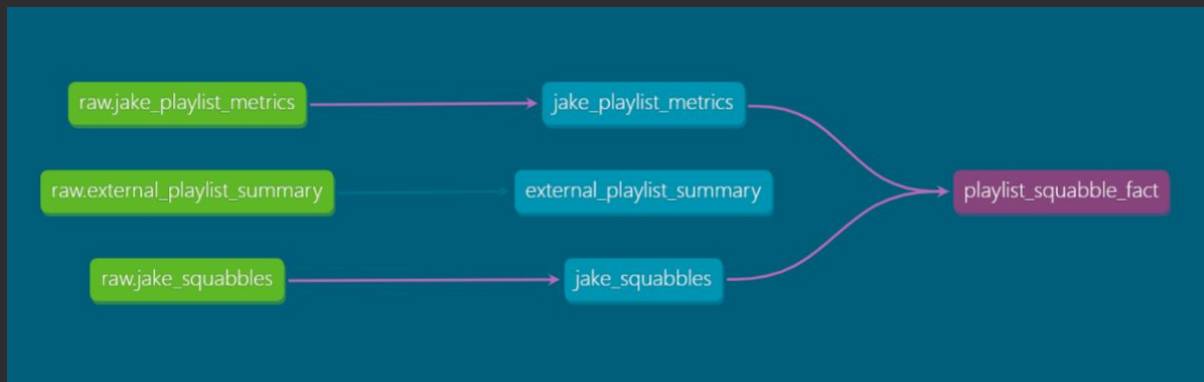
```
1
2 version: 2
3
4 models:
5   - name: external_playlist_summary
6     description: "Summary dataset provided by Spotify"
7     columns:
8       - name: playlist_uri
9         description: "The key, Spotify uri of the playlist"
10     tests:
11       - not_null
12   - name: owner
13     description: "Playlist owner, Spotify username"
14   - name: streams
15     description: "Number of streams from playlist today"
16   - name: streams30s
17     description: "Number of streams over 30 seconds from playlist today"
18   - name: dau
19     description: "Number of Daily Active Users"
20   - name: wau
21     description: "Number of Weekly Active Users"
22   - name: mau
23     description: "Number of Monthly Active Users"
24   - name: mau_previous_month
25     description: "Number of Monthly Active Users in the month prior to this one"
26   - name: mau_both_months
27     description: "Number of users that were active on the playlist both this and the previous month"
28   - name: users
29     description: "Users streaming at least one stream of any time duration (including skipped streams) in past month"
30   - name: skippers
31     description: "Number of users who skipped more than 90% of their total Spotify streams today and also used this playlist"
```





# trusted with dbt

- as we build data models, we can see the relationships between
  - great for understanding lineage!



# trusted with dbt

- by defining tests within our project, we can ensure anyone (me!) using these datasets have results that make sense and that are not prone to data errors

```
models:
- name: external_playlist_summary
  description: "Summary dataset provided by Spotify"
  columns:
    - name: playlist_uri
      description: "The key, Spotify uri of the playlist"
      tests:
        - not_null
```

```
C:\Users\Jake\spotify-analysis\dbt-spotify>dbt test
```



**so what makes a playlist successful?**



so what makes a playlist successful?

good music?



# exploratory analysis

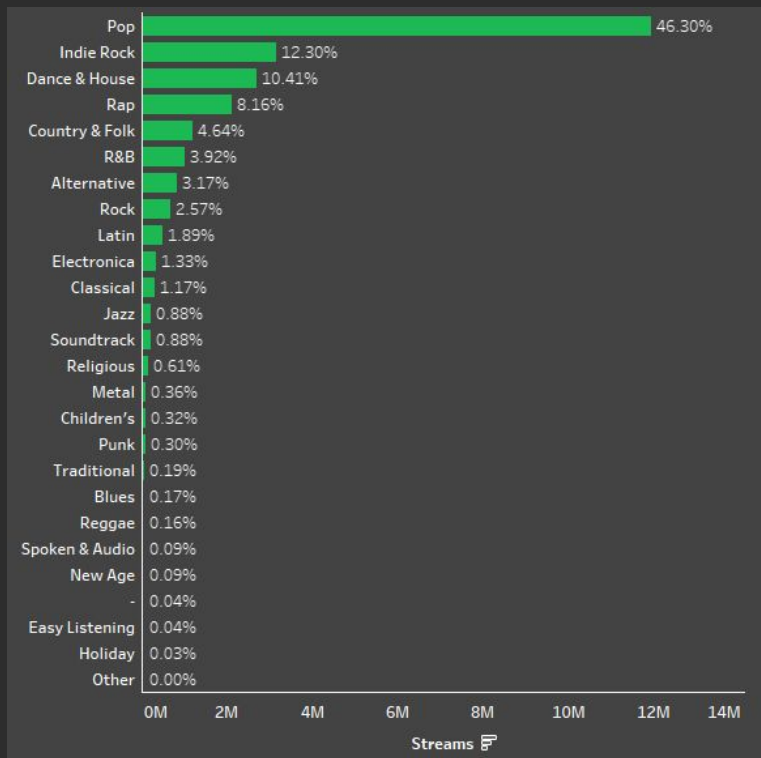
- refine the dataset by filtering out playlists with 0 streams of at least 30 seconds
- filter further for more usable dataset and add deciles to explore
  - ~30k playlists

```
1
2 select
3 *
4 , safe_divide(mau - mau_previous_month, mau_previous_month) as mau_change
5 , safe_divide(dau, users) as daily_per_users
6 , safe_divide(skippers, users) as skippers_users
7 , safe_divide(stream30s, streams) as stream30s_rate
8 , safe_divide(wau, users) as weekly_per_users
9 , ntile(10) over (order by streams asc) as streams_decile
10 , ntile(10) over (order by wau asc) as wau_decile
11 , ntile(10) over (order by safe_divide(dau, users) asc) as daily_per_tot_user_decile
12 , ntile(10) over (order by safe_divide(mau - mau_previous_month, mau_previous_month) asc) as mau_change_decile
13 , ntile(10) over (order by n_tracks asc) as tracks_decile
14 , ntile(10) over (order by n_artists asc) as artists_decile
15 , ntile(10) over (order by safe_divide(skippers, users) asc) as skippers_users_decile
16
17 from {{ ref('external_playlist_summary') }}
18
19 where stream30s > 0
20 and mau_previous_month > 10
21
22 order by safe_divide(mau - mau_previous_month, mau) desc
23
```



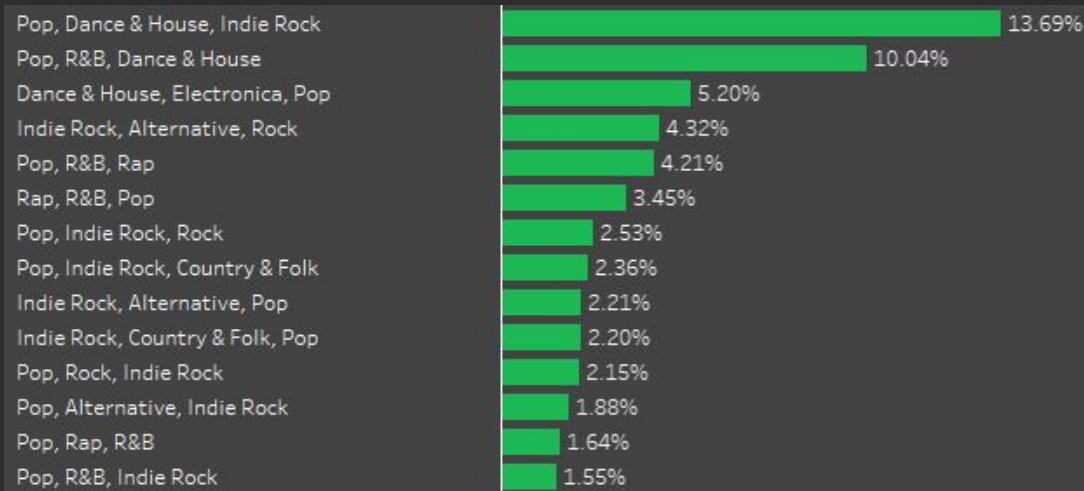
# exploratory analysis

- heavily pop-infused playlists (genre\_1) drive almost half of all streams within playlists



# exploratory analysis

- heavily pop-infused playlists (genre\_1) drive almost half of all streams within playlists

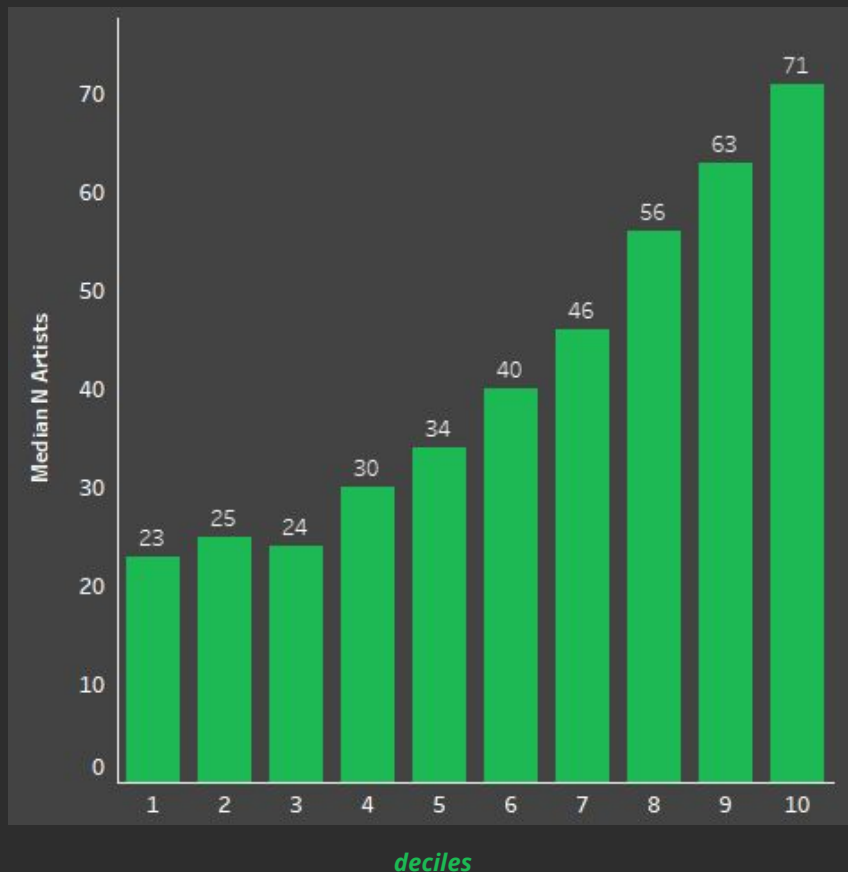


*total streams*



# exploratory analysis

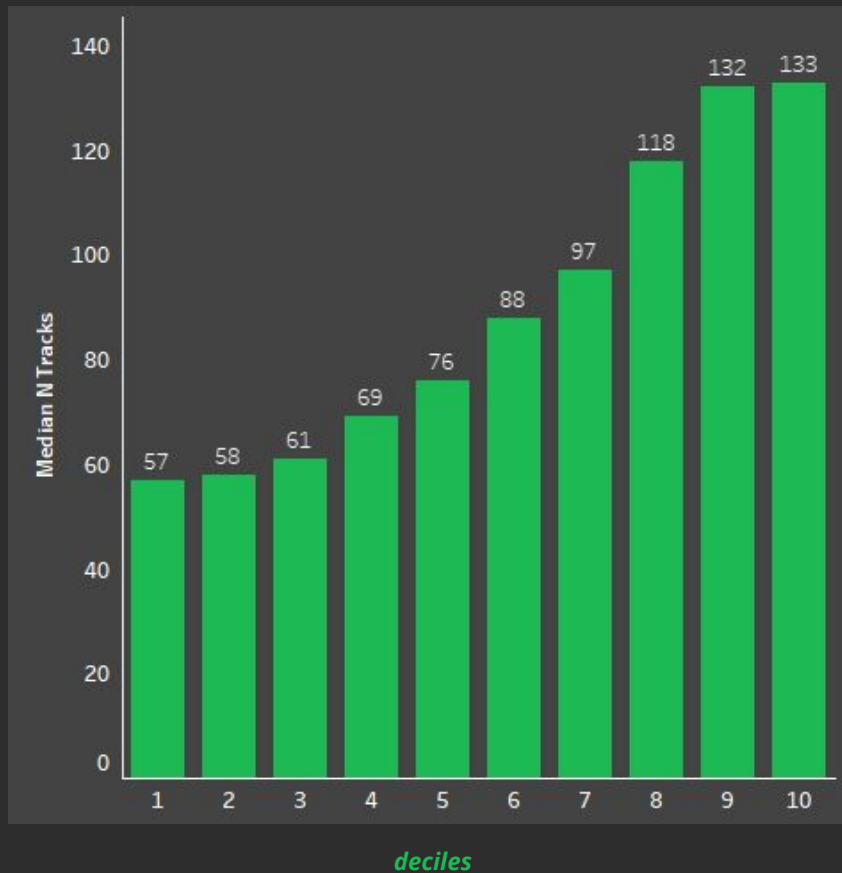
- by bucketing playlists into deciles from their stream counts, we can look to see if there are relations between specific metrics
- as median number of artists increases, streams of the specific playlist trend towards the higher percentiles (i.e., more plays)





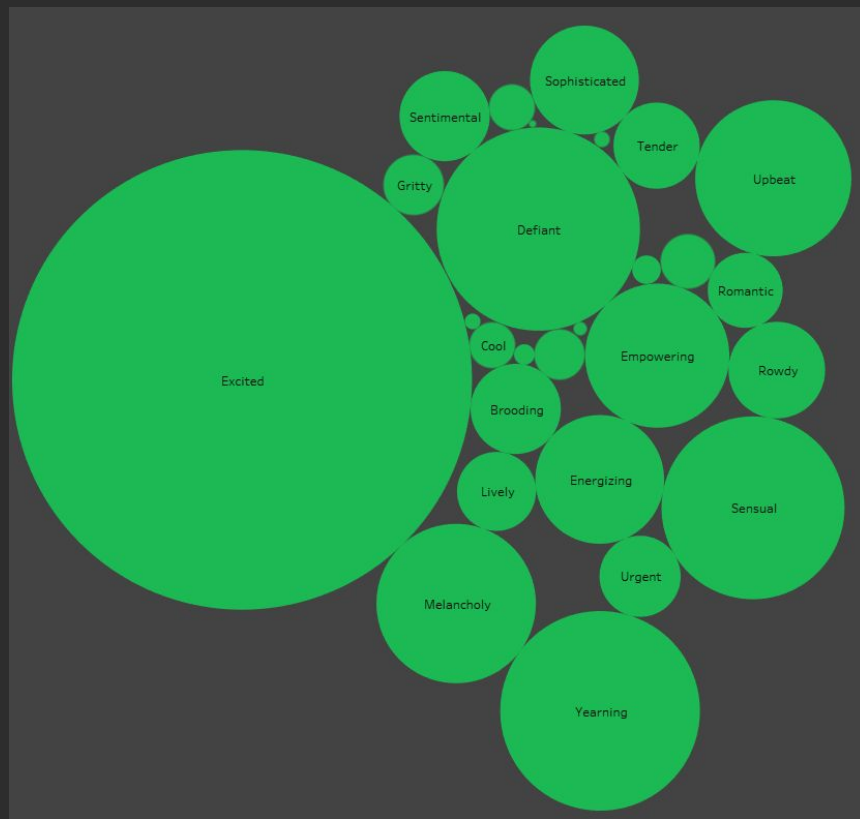
# exploratory analysis

- as median number of tracks per playlist increases, streams of the specific playlist trend towards the higher percentiles (i.e., more plays)



# exploratory analysis

- Focused on the top 10% of playlists by streams, the most streamed mood is excited

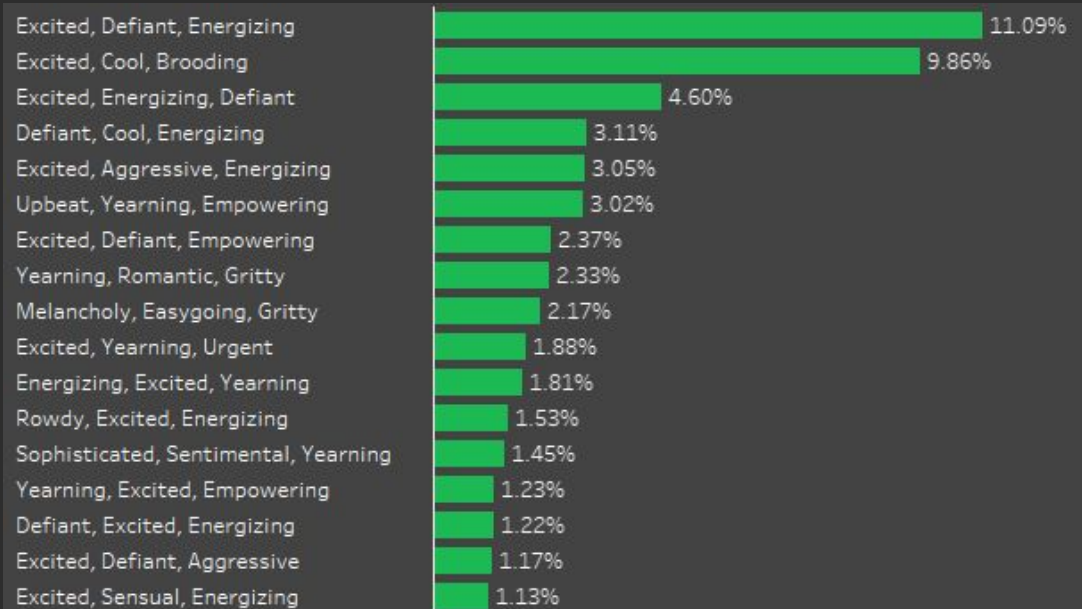


size --> 30s streams



# exploratory analysis

- zooming out to see all three moods combined, **excited** still dominates total streams with **energizing** and **cool** supporting moods



*total streams*



# key takeaways

- infusing your playlists with pop will help drive streams
- streams by users tend to increase as number of tracks and artists increases
- users typically enjoy excited and energizing music
  - me too!



# additional data points

- streams by time
  - are genres driven by certain hours of day?
- individual tracks & artists within playlists for more granularity
  - popular artists leading the charge?
- larger dataset with other countries to explore regional interests
- how playlists are accessed
  - search vs. share vs. home
- clean up the tokens column!



# Q&A



jhannan13/spotify-analysis

