**DATA MODEL**

**Nodi:**

*Creazione del nodo User.*

CREATE (u:User {{ nameUser: "{name}", id: "{id}", track: "{track}"}})

*Creazione del nodo Track.*

CREATE (t:Track {{nameTrack: "{tracks}", idTracks: "{idTracks}", artist: "{artist}", \

idArtist: "{idArtist}", nameAlbum: "{nameAlbum}", idAlbum: "{idAlbum}", danceability: {float(danceability)}, \

key: {int(key)}, loudness: {float(loudness)}, mode: {int(mode)}, speechiness: {float(speechiness)}, acousticness: {float(acousticness)}, \

instrumentalness: {float(instrumentalness)}, liveness: {float(liveness)}, valence: {float(valence)}, tempo: {float(tempo)}, \

duration\_ms: {int(duration\_ms)} }} )

*Creazione del nodo Artist.*

CREATE (a:Artist {{nameArtist: "{name}", id: "{id}", genre: "{genre}"}})

*Creazione del nodo Album.*

CREATE (al:Album {{nameAlbum: "{nameAlbum}", idAlbum: "{idAlbum}", typeAlbum: "{typeAlbum}",\

totalTracks: "{int(totalTracks)}", releaseDate: "{(releaseDate)}", artist: "{artist}", idArtist: "{idArtist}"}})

*Creazione del nodo Genre.*

CREATE (g:Genre {{genres: "{genres}"}})

**Relazioni:**

*Utilizzando la clausola "MATCH" e "WHERE" si crea una relazione tra un utente (u:User) e una traccia (t:Track) in cui l'utente ha ascoltato la traccia.*

MATCH (u:User), (t:Track) \

WHERE u.track CONTAINS t.idTracks \

CREATE (u)-[r:LISTENED\_TO]->(t)

*La relazione tra una traccia (t:Track) e un album (al:Album) viene creata utilizzando la clausola "MERGE" per evitare la creazione di relazioni duplicate.*

MATCH (t:Track), (al:Album) \

WHERE t.idAlbum = al.idAlbum \

MERGE (t)-[r:PART\_OF]->(al)

*La relazione tra una traccia (t:Track) e un artista (a:Artist) viene creata utilizzando la clausola "WHERE" per collegare la traccia all'artista che l'ha eseguita.*

MATCH (t:Track), (a:Artist) \

WHERE t.idArtist = a.id \

CREATE (t)-[r:PERFORMED\_BY]->(a)

*La relazione tra un artista (a:Artist) e un genere (g:Genre) viene creata utilizzando la clausola "WHERE" per collegare l'artista al genere a cui appartiene.*

MATCH (a:Artist), (g:Genre) \

WHERE a.genre CONTAINS g.genres \

CREATE (a)-[r:BELONGS\_TO]->(g)

*La relazione tra un album (al:Album) e un artista (a:Artist) viene creata utilizzando la clausola "WHERE" per collegare l'album all'artista che lo ha prodotto.*

MATCH (al:Album), (a:Artist) \

WHERE al.idArtist = a.id \

CREATE (al)-[r:BY\_ARTIST]->(a)

*La relazione tra due artisti (a1:Artist) e (a2:Artist) viene creata utilizzando la clausola "WHERE" per collegare gli artisti che hanno collaborato insieme su una traccia.*

MATCH (a1:Artist),(a2:Artist)

WHERE a1 <> a2

WITH a1, a2

MATCH (t:Track)

WHERE t.artist CONTAINS a1.nameArtist AND t.artist CONTAINS a2.nameArtist

CREATE (a1)-[r:COLLABORATED\_WITH]->(a2)

**QUERY CYPHER NEO4J**

1. **Mostra lo schema del Data Model:**

CALL db.schema.visualization()

1. **Stabilire quale sia la canzone più presente nei Wrapped 2022:**

MATCH (t:Track)<-[:LISTENED\_TO]-(u:User)

RETURN t.nameTrack

LIMIT 1

1. **Stabilire quali siano le canzoni in comune tra gli utenti, quanti ascoltatori hanno avuto e quali sono:**

MATCH (u1:User)-[:LISTENED\_TO]->(t:Track)<-[:LISTENED\_TO]-(u2:User)

WHERE u1 <> u2

RETURN t.nameTrack, COUNT(DISTINCT u1) as numUsers, COLLECT(DISTINCT u1.nameUser) as commonUsers

ORDER BY numUsers DESC, SIZE(commonUsers) DESC

1. **Stabilire quale sia l'artista più presente nei Wrapped 2022:**

MATCH (u:User)-[:LISTENED\_TO]->(t:Track)-[:PERFORMED\_BY]->(a:Artist)

WITH a.nameArtist as artist, count(a) as count

ORDER BY count DESC

LIMIT 1

RETURN artist

1. **Per ogni utente, stabilire quale sia l'artista più presente nel proprio Wrapped :**

MATCH (u:User)-[:LISTENED\_TO]->(t:Track)-[:PERFORMED\_BY]->(a:Artist)-[:BELONGS\_TO]->(g:Genre)

WHERE u.nameUser = $nameUser

RETURN a.nameArtist as artist, count(a) as count

ORDER BY count DESC

LIMIT 1", nameUser=nameUser

MATCH (u:User)-[:LISTENED\_TO]->(t:Track)-[:PERFORMED\_BY]->(a:Artist)

WITH u, a, COUNT(t) as count

ORDER BY count DESC

WITH u, COLLECT({art: a, count: count}) as a

RETURN u.nameUser, a[0].art.nameArtist as mostListenedArtist

1. **Stabilire per ogni artista quante collaborazioni ha avuto e con quali artisti all’interno delle tracce del Wrapped:**

MATCH (a1:Artist)-[r:COLLABORATED\_WITH]->(a2:Artist)

WITH a1.nameArtist as name,

COLLECT(DISTINCT a2.nameArtist) as collaborators,

SIZE(COLLECT(DISTINCT a2.nameArtist)) as collaborations

RETURN name, collaborations, collaborators

ORDER BY collaborations DESC

1. **Stabilire quale sia l'album che ha più tracce nei Wrapped:**

MATCH (al:Album)<-[:PART\_OF]-(t:Track)<-[:LISTENED\_TO]-(u:User)

WITH al.nameAlbum AS album, COUNT(u) as numListeners

ORDER BY numListeners DESC

LIMIT 1

RETURN album

1. **Per ogni utente, stabilire quale sia l'album che ha più tracce all’interno del proprio Wrapped:**

MATCH (u:User)-[:LISTENED\_TO]->(t:Track)-[:PART\_OF]->(al:Album)

WITH u, al, COUNT(t) AS numTracks

ORDER BY numTracks DESC

WITH u, COLLECT(al)[0] AS al, MAX(numTracks) as maxTracks

RETURN DISTINCT u.nameUser, al.nameAlbum

1. **Stabilire quale sia il genere più ascoltato tra le canzoni all’interno del Wrapped:**

MATCH (g:Genre)<-[:BELONGS\_TO]-(a:Artist)<-[:PERFORMED\_BY]-(t:Track)<-[:LISTENED\_TO]-(u:User)

WITH g.genres AS genre, COUNT(u) as numListeners

ORDER BY numListeners DESC

LIMIT 1

RETURN genre

1. **Per ogni utente, stabilire quale sia il genere più ascoltato all’interno del proprio Wrapped:**

MATCH (u:User)-[:LISTENED\_TO]->(t:Track)-[:PERFORMED\_BY]->(a:Artist)-[:BELONGS\_TO]->(g:Genre)

WHERE u.nameUser = $nameUser

RETURN g.genres as genre, count(g) as count

ORDER BY count DESC

LIMIT 1", nameUser=nameUser

ORDER BY numUsers DESC, SIZE(commonUsers) DESC

MATCH (u:User)-[:LISTENED\_TO]->(t:Track)-[:PERFORMED\_BY]->(a:Artist)-[:BELONGS\_TO]->(g:Genre)

WITH u, g, COUNT(t) as count

ORDER BY count DESC

WITH u, COLLECT({gen: g, count: count}) as g

RETURN u.nameUser, g[0].gen.genres as mostListenedGenre

1. **Trovare quali siano gli utenti più simili per quanto riguarda le tracce del Wrapped:**

MATCH (u1:User)-[:LISTENED\_TO]->(t:Track)<-[:LISTENED\_TO]-(u2:User)

WHERE u1.nameUser <> u2.nameUser AND u1.nameUser < u2.nameUser

WITH DISTINCT u1.nameUser AS user1, u2.nameUser AS user2, collect(DISTINCT t.nameTrack) AS common\_tracks

RETURN user1, user2, common\_tracks, size(common\_tracks) AS n\_tracks

ORDER BY n\_tracks DESC

1. **Stabilire quale siano gli utenti che hanno più canzoni in comune con tutti gli altri utenti**

MATCH (u1:User)-[:LISTENED\_TO]->(t:Track)<-[:LISTENED\_TO]-(u2:User)

WITH u1, size(collect(DISTINCT t.nameTrack)) AS common\_track

ORDER BY common\_track DESC

RETURN u1.nameUser as user, common\_track

1. **Trovare il percorso più vicino tra gli utenti**

MATCH p=shortestPath((u1:User)-[\*..]-(u2:User))

WHERE u1 <> u2 and u1>u2

RETURN u1.nameUser AS user1, u2.nameUser AS user2, length(p) as distance

ORDER BY distance DESC