Name: Ameya Gaitonde

Student ID: 95552640

Q1:

A:

i Loading entities queries

```
LOAD CSV WITH HEADERS FROM
'file:////Users/ameya/Desktop/BD_HW4/csv_files/Users.csv' AS row
CREATE (u:Users {
user_id: row.user_id,
email: row.email,
joined_date: date(row.joined_date),
nickname: row.nickname,
street: row.street,
city: row.city,
state: row.state,
zip: toFloat(row.zip),
genres: row.genres
});

LOAD CSV WITH HEADERS FROM
'file:////Users/ameya/Desktop/BD_HW4/csv_files/Artists.csv' AS row
MATCH (u:Users {user_id: row.user_id})
MERGE (a:Artists {user_id: row.user_id})
SET a.bio = row.bio;

LOAD CSV WITH HEADERS FROM
'file:////Users/ameya/Desktop/BD_HW4/csv_files/Listeners.csv' AS row
MATCH (u:Users {user_id: row.user_id})
MERGE (l:Listeners {user_id: row.user_id})
```

```
SET l.subscription = row.subscription,
l.first_name = row.first_name,
l.last_name = row.last_name;

LOAD CSV WITH HEADERS FROM
'file:////Users/ameya/Desktop/BD_HW4/csv_files/Records.csv' AS row
CREATE (r:Records {record_id: row.record_id, title: row.title, genre: row.genre});

LOAD CSV WITH HEADERS FROM
'file:////Users/ameya/Desktop/BD_HW4/csv_files/Albums.csv' AS row
MATCH (r:Records {record_id: row.record_id})
MERGE (a:Albums {record_id: row.record_id})
SET a.description = row.description;

LOAD CSV WITH HEADERS FROM
'file:////Users/ameya/Desktop/BD_HW4/csv_files/Reviews.csv' AS row
CREATE (r:Reviews {review_id: row.review_id, rating: toInteger(row.rating), body:
row.body});

LOAD CSV WITH HEADERS FROM
'file:////Users/ameya/Desktop/BD_HW4/csv_files/Singles.csv' AS row
MATCH (r:Records {record_id: row.record_id})
MERGE (s:Singles {record_id: row.record_id})
SET s.video_url = row.video_url;

LOAD CSV WITH HEADERS FROM
'file:////Users/ameya/Desktop/BD_HW4/csv_files/Upvotes.csv' AS row
CREATE (uv:Upvotes {user_id: row.user_id, review_id: row.review_id})
```

Ii Creating indexes queries

CREATE INDEX user_id_index FOR (u:Users) ON (u.user_id);

CREATE INDEX record_id_index FOR (r:Records) ON (r.record_id);

Iii Loading relationships commands

LOAD CSV WITH HEADERS FROM 'file:////Users/ameya/Desktop/BD_HW4/csv_files/Artists_Releases_Record.csv' AS row

MATCH (a:Artists {user_id: row.artist_user_id})

MATCH (r:Records {record_id: row.record_id})

MERGE (a)-[rel:Artists_Releases_Record]->(r)

SET rel.release_date = date(row.release_date);

LOAD CSV WITH HEADERS FROM 'file:////Users/ameya/Desktop/BD_HW4/csv_files/Reviews_About_Record.csv' AS row

MATCH (r:Reviews {review_id: row.review_id})

MATCH (rec:Records {record_id: row.record_id})

MERGE (r)-[rel:Reviews_About_Record]->(rec)

SET rel.review_id = row.review_id, rel.record_id = row.record_id;

LOAD CSV WITH HEADERS FROM

‘file:////Users/ameya/Desktop/BD_HW4/csv_files/Users_Post_Reviews.csv’ AS row

MATCH (u: Users {user_id: row.user_id})

MATCH (r: Reviews {review_id: row.review_id}

MERGE (u)-[rel:Users_Post_Reviews]->(r)

SET rel.user_id = row.user_id, rel.review_id = row.review_id, rel.posted_at = row.posted_at

B:

I

```
neo4j$ CALL db.relationshipTypes;
```

Table

Text

Code

relationshipType
"Artists_Releases_Record"
"Reviews_About_Record"
"Users_Post_Reviews"

li

```
neo4j$ CALL db.labels;
```

Table

Text

Code

label
"Users"
"Artists"
"Listeners"
"Records"
"Albums"
"Singles"
"Reviews"
"Upvotes"

Q2:

Query:

```
MATCH (r: Records)
WITH r LIMIT 1000
UNWIND (keys(r)) AS recordKeys
RETURN DISTINCT recordKeys;
```

Results (screenshot below):

recordKeys
"record_id"
"genre"
"title"

Q3 A:

Query:

```
MATCH (u:Users)
RETURN u
ORDER BY u.joined_date DESC
LIMIT 5;
```

Results (screenshot below):

```
u
(:Users {joined_date: "2022-12-18",user_id: "user_nyIhrD99",genres: "Disco,R&B,Classical",street: "28448 Smith Causeway Apt. 891",nickname: "josephrangel",state: "Virginia",email: "josephrangel@college.edu"})
(:Users {joined_date: "2022-12-12",city: "Ramseyside",user_id: "user_cWptaV9B",genres: "Latin,Hip-Hop,Country,Gospel",street: "03158 Evans Via Suite 281",nickname: "duranjoshua",state: "Virginia",email: "duranjoshua@hotmail.com"})
(:Users {zip: 30240.0,joined_date: "2022-12-10",city: "Simonshire",user_id: "user_OldjtiH5",genres: "Disco,Metal,Rock,Techno,Soul,Jazz,Blues,Hip-Hop",street: "2402 Anderson Glens Apt. 163",nickname: "milesjoshua",state: "Montana",email: "milesjoshua@protonmail.com"})
(:Users {zip: 85977.0,joined_date: "2022-11-16",city: "South Debrafurt",user_id: "user_EtKq93oS",genres: "Rock,R&B",street: "755 Carrillo Spur",nickname: "barroyo",state: "West Virginia",email: "barroyo@protonmail.com"})
(:Users {zip: 21224.0,joined_date: "2022-11-06",city: "Timothyberg",user_id: "user_geMLolwT",genres: "Rock,Latin,R&B,Pop,Jazz,Reggae",street: "7038 Sellers Gardens Apt. 132",nickname: "smithhelen",state: "Louisiana",email: "smithhelen@protonmail.com"})
```

Q3 B:

Query:

```
MATCH (a:Artists)-[rela:Artists_Releases_Record]->(rec:Records)
WHERE a.user_id = 'user_WKWGxlZa'
RETURN a, rec, rela.release_date
ORDER BY rec.record_id ASC;
```

Results (screenshot below):

a	rec	rela.release_date
(:Artists {user_id: "user_WKWGxlZa",bio: "Arm shake him what other yard people. Firm leader each medical action goal training. Until suddenly visit myself his because.\nQuite specific table we PM realize. Kind final coach."})	(:Records {record_id: "record_1487gZZ9",genre: "Indie",title: "Civil government find"})	"2020-08-08"
(:Artists {user_id: "user_WKWGxlZa",bio: "Arm shake him what other yard people. Firm leader each medical action goal training. Until suddenly visit myself his because.\nQuite specific table we PM realize. Kind final coach."})	(:Records {record_id: "record_696_41KT",genre: "Techno",title: "Hope there scientist"})	"2022-12-26"
(:Artists {user_id: "user_WKWGxlZa",bio: "Arm shake him what other yard people. Firm leader each medical action goal training. Until suddenly visit myself his because.\nQuite specific table we PM realize. Kind final coach."})	(:Records {record_id: "record_9PGDade0",genre: "Gospel",title: "Bring official"})	"2020-07-04"

COLUMN WIDTH:

Q3 C:

Query:

```
// 3C
// First, we match all Listeners nodes, then we match the nodes whose user_id
// does not exist in the Users_Post_Reviews relationship, effectively isolating
// the users who have not posted a review. We then count the number of distinct
// such user_id values.

MATCH (lis:Listeners)
WHERE NOT EXISTS {
MATCH (lis)-[:Users_Post_Reviews]->()
}
RETURN COUNT(DISTINCT lis.user_id) AS num_nonreview_listeners;
```

Results (screenshot below):

num_nonreview_listeners
184

Q3 D:

Query:

```
MATCH (r:Reviews)-[:Reviews_About_Record]->(rec:Records)
WHERE rec.title = "Audience star apply" AND r.rating = 5
WITH r.review_id AS review_id
MATCH (u:Users)-[rel:Users_Post_Reviews]->(revtwo:Reviews)
WHERE revtwo.review_id = review_id
WITH u.user_id AS user_id
MATCH (lis:Listeners {user_id: user_id})
RETURN lis.first_name AS first_name, lis.last_name AS last_name
ORDER BY lis.first_name ASC;
```

Results (screenshot below):

	first_name	last_name
1	"Doris"	"Baker"
2	"Mark"	"Allen"

Q3 E:

Query:

```
MATCH ()-[r:Reviews_About_Record]->()
WITH r.record_id AS record_id, COUNT(r) AS review_count
WHERE review_count > 25
MATCH (rec:Records) WHERE rec.record_id = record_id
RETURN record_id, rec.title, review_count
ORDER BY review_count DESC;
```

Results (screenshot below):

record_id	rec.title	review_count
"record_51zx5w5v"	"Focus idea defense"	26

Q3 F:

Query:

```
MATCH (u:Users)
MATCH (u)-[:Users_Post_Reviews]->(r:Reviews)
MATCH (up:Upvotes {user_id: u.user_id})
WITH u, COUNT(r) AS reviewCount, COUNT(up) AS upvoteCount
WHERE reviewCount > 60 AND upvoteCount > 500
RETURN u.user_id AS userId
ORDER BY u.user_id ASC
LIMIT 5;
```

Results (screenshot below):

userId
"user_-6xZN_3p"
"user_04JO2Lh1"
"user_07TZ_5bz"
"user_0I-IPVxu"
"user_0geeGahd"

Q3 G:

Query:

```
MATCH (r1:Reviews)-[:Reviews_About_Record]->(rec:Records),
(r2:Reviews)-[:Reviews_About_Record]->(rec),
(u1:Users)-[:Users_Post_Reviews]->(r1),
(u2:Users)-[:Users_Post_Reviews]->(r2)
WHERE r1.rating = r2.rating
AND id(r1) < id(r2)
WITH rec, r1.rating AS rating, u1.user_id AS user_id_1, u2.user_id AS user_id_2
// Now that the relevant user_id values have been extracted, we can match
// the Listeners based on user_id vals.
MATCH (l1:Listeners {user_id: user_id_1}),
(l2:Listeners {user_id: user_id_2})
RETURN rec.record_id AS record_id,
l1.last_name AS last_name_1,
l2.last_name AS last_name_2
```

```
ORDER BY record_id ASC
LIMIT 10;
```

Results (screenshot below):

record_id	last_name_1	last_name_2
"record_--4_Ht0N"	"Sanders"	"Gallegos"
"record_--4_Ht0N"	"Sanders"	"Myers"
"record_--4_Ht0N"	"Gallegos"	"Myers"
"record_--4_Ht0N"	"Sherman"	"Mullins"
"record_--4_Ht0N"	"Young"	"Wyatt"
"record_--4_Ht0N"	"Sherman"	"Copeland"
"record_--4_Ht0N"	"Mullins"	"Copeland"
"record_-2WQnPOg"	"Morgan"	"Evans"
"record_-2WQnPOg"	"Taylor"	"Allen"
"record_-2WQnPOg"	"Moore"	"Morgan"

Q3 H:

Query:

// first_user releases reviewA on recordA, and second_user releases reviewB on recordB. The idea is that recordA was released by second user, while recordB was released by first_user. Then, we use the CASE WHEN section to prevent logical duplicates.

```
MATCH (first_user:Users)-[:Users_Post_Reviews]->(reviewA:Reviews)-[:Reviews_About_Record]->(recordA:Records)

MATCH (second_user:Users)-[:Users_Post_Reviews]->(reviewB:Reviews)-[:Reviews_About_Record]->(recordB:Records)

WHERE EXISTS {

MATCH (artistB:Artists)-[:Artists_Releases_Record]->(recordA)

} AND EXISTS {

MATCH (artistA:Artists)-[:Artists_Releases_Record]->(recordB)

}

WITH first_user, second_user

WHERE first_user <> second_user

WITH DISTINCT

CASE WHEN first_user.nickname < second_user.nickname THEN first_user.nickname + '-' + second_user.nickname

ELSE second_user.nickname + '-' + first_user.nickname END AS user_pair,

first_user.nickname AS first_user_nickname, second_user.nickname AS second_user_nickname

ORDER BY first_user_nickname

RETURN first_user_nickname, second_user_nickname

LIMIT 10;
```

Results (screenshot below):

first_user_nickname	second_user_nickname
"aaron76"	"ryanmorgan"
"aaron76"	"hollowayjonathan"
"aaron76"	"jeanettehoffman"
"aaron76"	"randallclark"
"aaron76"	"westchristopher"
"aaron76"	"sherry98"
"aaron76"	"courtney36"
"aaron76"	"tina44"
"aaron76"	"eheath"
"aaron76"	"rjones"

Q3 I:

i

Query:

```
MATCH (originalArtist:Artists {user_id: 'user_0ZIeALBX'}),
(finalArtist:Artists)
WHERE originalArtist <> finalArtist
MATCH path = shortestPath((originalArtist)-[*]-(finalArtist))
RETURN length(path) AS length_of_shortest_path
LIMIT 1;
```

Results (screenshot below):

length_of_shortest_path
6

ii

Query:

```
MATCH (originalArtist:Artists {user_id: 'user_0ZIeALBX'}),
(finalArtist:Artists)
WHERE originalArtist <> finalArtist
MATCH path = shortestPath((originalArtist)-[*]-(finalArtist))
WHERE length(path) = 6
RETURN finalArtist.user_id AS artistId
ORDER BY artistId ASC
LIMIT 5;
```

Results (screenshot below):

artistId
"user_-2Ijew-e"
"user_-6xZN_3p"
"user_-EzOnF9o"
"user_0I-IPVxu"
"user_1CNoW-_M"