Interview tasks of Muhammad Azhar Shaikh

Task 1

Technical details:

- Used Hasura Cloud :
  - Project ID : d41de11d-de2d-458f-81bd-b00a72285e60
  - GraphQL API : https://azharshaikh-task-1.hasura.app/v1/graphql
  - Domain : azharshaikh-task-1.hasura.app

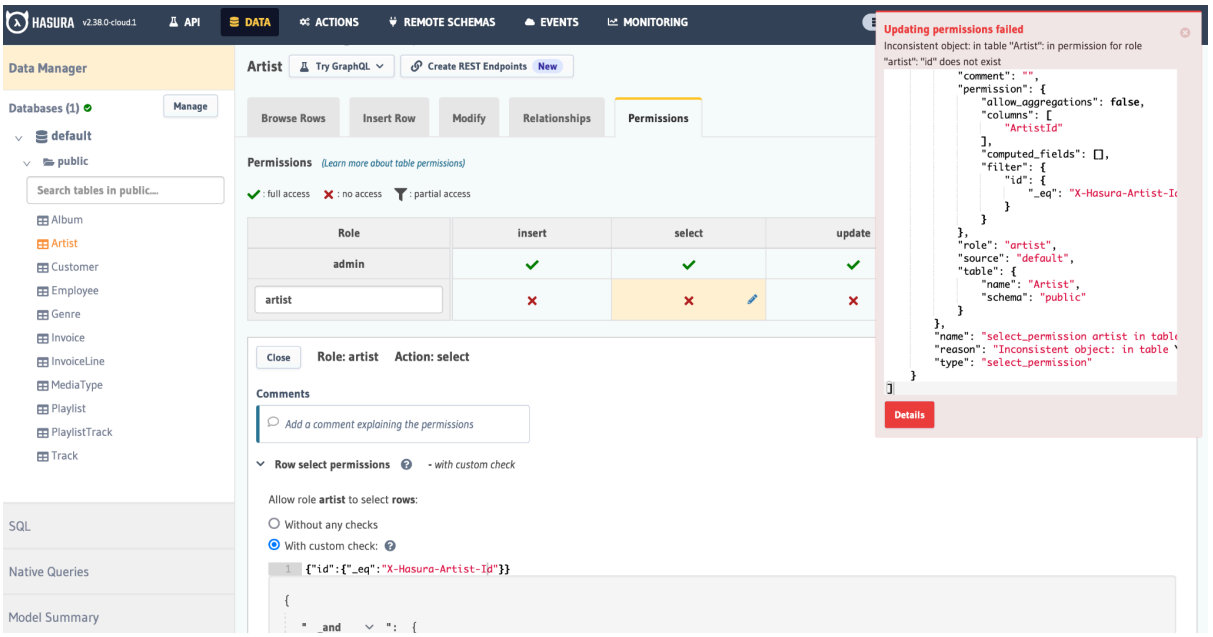
- Used neon Postgres as the data source
  - PG database URL : polished-truth-20269338.us-west-2.aws.neon.tech

- Used the Chinook data set.

- Added additional key in the Request header “x-hasura-artist-id” for identifying the Artist

Followed the document and added ENV VAR “HASURA_GRAPHQL_JWT_SECRET”

Was not able to resolve below error;



- Artists should not be allowed to access albums that do not belong to them.

I need more clarity on this question. The Database is imported from Chinook.
Where exactly on the Hasura side can these changes be made?

Objectives:

Configure a GraphQL API that can answer the following statements via GraphQL:

1. How many artists are in the database?

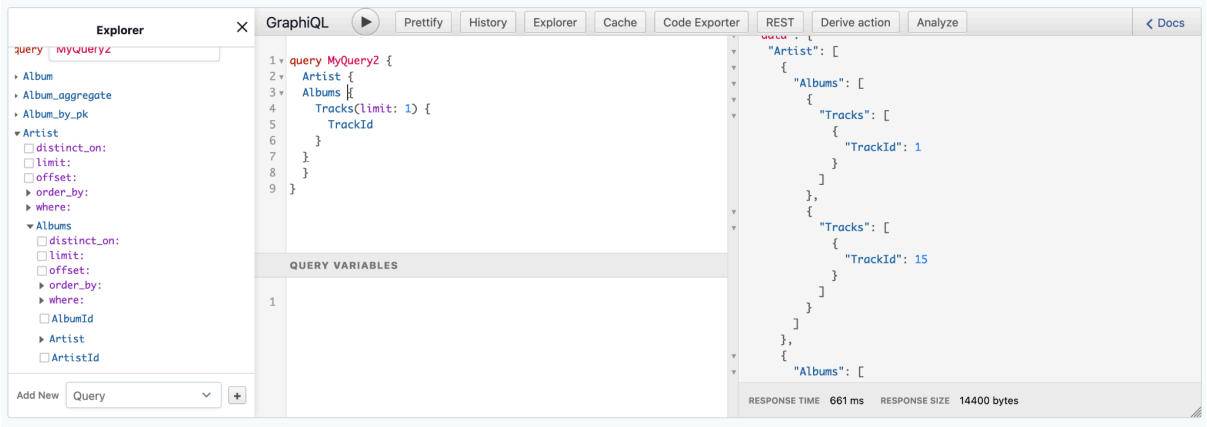
```
query MyQuery {
  Artist_aggregate {
      aggregate {
      count
      }
  }
}
```

Output:

```
{
  "data": {
      "Artist_aggregate": {
      "aggregate": {
      "count": 275
      }
      }
  }
}
```

2. List the first track of every album by every artist in ascending order.

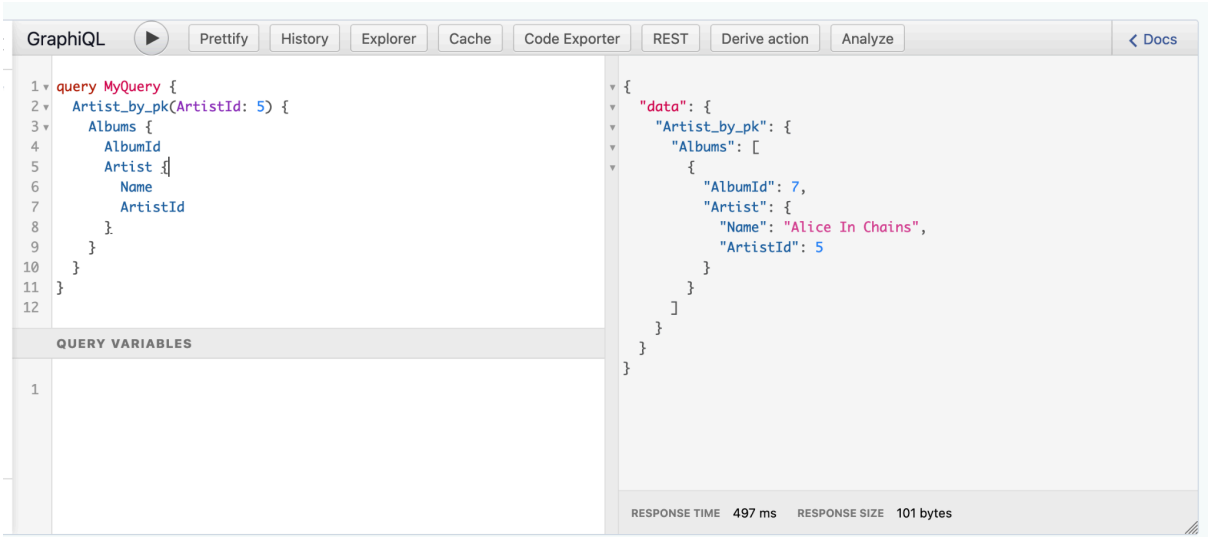
```
query MyQuery2 {
  Artist {
      Albums {
      Tracks(limit: 1) {
      TrackId
      }
      }
  }
}
```

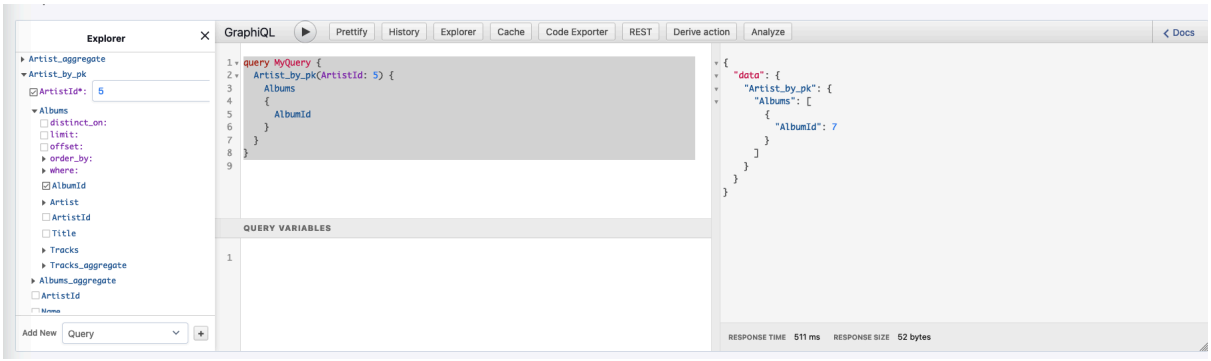


3. Get all albums for artist id = 5 without specifying a where clause.

```
query MyQuery {
  Artist_by_pk(ArtistId: 5) {
      Albums {
      AlbumId
      Artist {
      Name
      ArtistId
      }
      }
  }
```

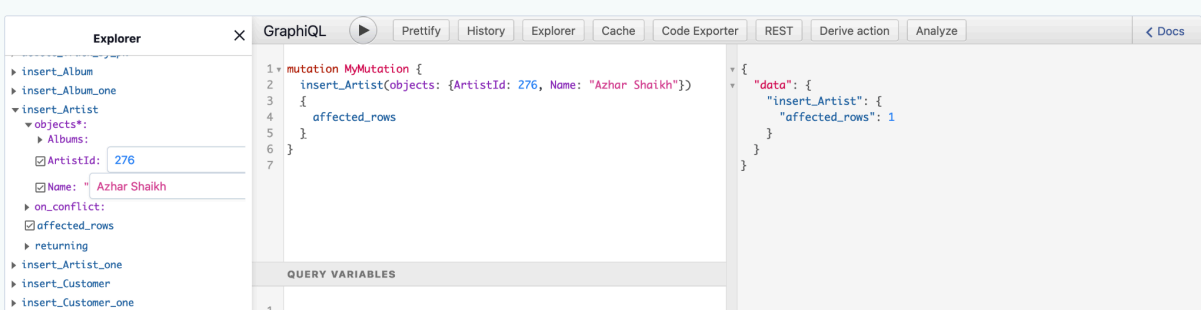
```
}
```



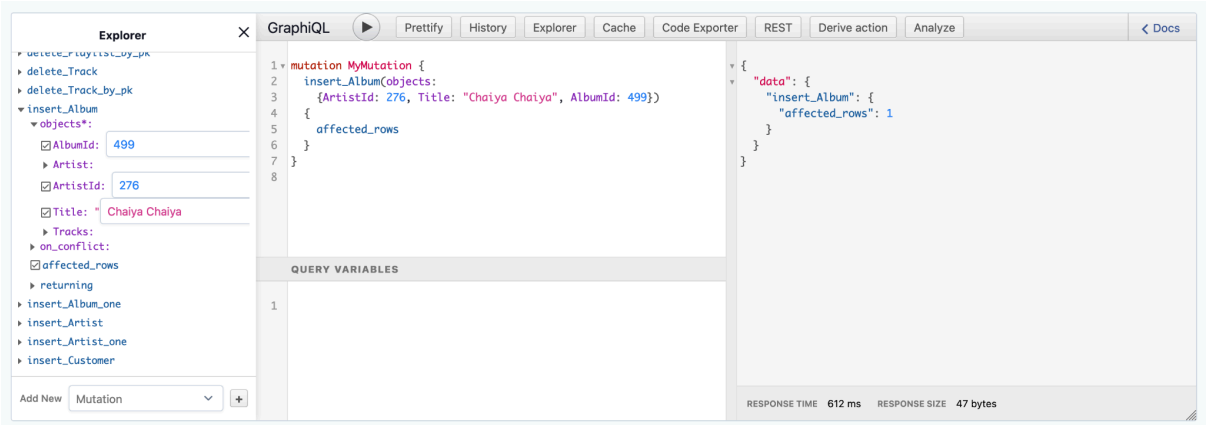


4. Using a GraphQL mutation, add your favorite artist and one of their albums that isn't in the dataset.

There are a total 275 artists as per earlier queries.
Added artist id 276



Added Album Id 499 and linked with Artist Id 276



Verified whether data is correctly inserted

Referred : https://hasura.io/docs/latest/mutations/postgres/insert/

5. How did you identify which ID to include in the mutation?

There are a total 275 artists as per earlier queries. Also, it will fail when inserting a new Artist with the value of Artist id between 1 to 275.

So added artist id 276

Using a Postgres client, configure a SQL statement to retrieve the below information.

Connected the postgres database

```
]# brew install postgresql@15
```

```
]# psql -h polished-truth-20269338.us-west-2.aws.neon.tech -d
thorough-herring-67_db_8114218 -U neondb_owner -p 5432
--set=sslmode=require

Password for user neondb_owner:

SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits:
256, compression: off)
Type "help" for help.
```

Tried all available commands of psql to understand DB details completely. Command output is mentioned in a [separate text file](#).
Get a list of available databases.

```
thorough-herring-67_db_8114218=> \l
                                                  List of databases
           Name              |     Owner       | Encoding | Collate |
Ctype |     Access privileges
-----------------------------+-----------------+----------+---------+---
----+-----------------------------------
 neondb                      | neondb_owner | UTF8   | C         | C   |
=Tc/neondb_owner             +
                             |              |          |           |     |
neondb_owner=CTc/neondb_owner  +
                             |              |          |           |     |
neon_superuser=CTc/neondb_owner
 postgres                    | cloud_admin  | UTF8   | C         | C   |
 sharp-honeybee-24_db_5865181   | neondb_owner | UTF8       | C         |
C     | =Tc/neondb_owner             +
                             |              |          |           |     |
neondb_owner=CTc/neondb_owner  +
                             |              |          |           |     |
neon_superuser=CTc/neondb_owner
 task-1_db_3779131           | neondb_owner | UTF8   | C         | C   |
=Tc/neondb_owner             +
                             |              |          |           |     |
neondb_owner=CTc/neondb_owner  +
                             |              |          |           |     |
neon_superuser=CTc/neondb_owner
 task-1_db_4444083           | neondb_owner | UTF8   | C         | C   |
=Tc/neondb_owner             +
                             |              |          |           |     |
```

```
neondb_owner=CTc/neondb_owner  +
                               |           |           |           |     |
neon_superuser=CTc/neondb_owner
 task-1_db_4496277             | neondb_owner | UTF8   | C         | C   |
=Tc/neondb_owner               +
                               |           |           |           |     |
neondb_owner=CTc/neondb_owner  +
                               |           |           |           |     |
neon_superuser=CTc/neondb_owner
 template0                     | cloud_admin  | UTF8   | C         | C   |
=c/cloud_admin                 +
                               |           |           |           |     |
cloud_admin=CTc/cloud_admin
 template1                     | cloud_admin  | UTF8   | C         | C   |
=c/cloud_admin                 +
                               |           |           |           |     |
cloud_admin=CTc/cloud_admin
 thorough-herring-67_db_8114218 | neondb_owner | UTF8       | C         |
C     | =Tc/neondb_owner            +
                               |           |           |           |     |
neondb_owner=CTc/neondb_owner  +
                               |           |           |           |     |
neon_superuser=CTc/neondb_owner
(9 rows)
```

```
thorough-herring-67_db_8114218=> \dt
            List of relations
 Schema |   Name         | Type  |   Owner
--------+----------------+-------+--------------
 public | Album          | table | neondb_owner
 public | Artist         | table | neondb_owner
 public | Customer       | table | neondb_owner
 public | Employee       | table | neondb_owner
 public | Genre          | table | neondb_owner
 public | Invoice        | table | neondb_owner
 public | InvoiceLine    | table | neondb_owner
 public | MediaType      | table | neondb_owner
 public | Playlist       | table | neondb_owner
 public | PlaylistTrack | table | neondb_owner
 public | Track          | table | neondb_owner
(11 rows)
```

1. Return the artist with the most number of albums

Checked the schema of “Album” Table

```
thorough-herring-67_db_8114218=> \d "Album";
                        Table "public.Album"
  Column  |       Type        | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 AlbumId  | integer               |           | not null |
 Title    | character varying(160) |           | not null |
 ArtistId | integer               |           | not null |
Indexes:
      "PK_Album" PRIMARY KEY, btree ("AlbumId")
      "IFK_AlbumArtistId" btree ("ArtistId")
Foreign-key constraints:
      "FK_AlbumArtistId" FOREIGN KEY ("ArtistId") REFERENCES
"Artist"("ArtistId")
Referenced by:
      TABLE ""Track"" CONSTRAINT "FK_TrackAlbumId" FOREIGN KEY
("AlbumId") REFERENCES "Album"("AlbumId")
```

Checked the schema of “Artist “ Table.

```
thorough-herring-67_db_8114218=> \d "Artist";
                        Table "public.Artist"
  Column  |       Type        | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 ArtistId | integer               |           | not null |
 Name     | character varying(120) |           |          |
Indexes:
      "PK_Artist" PRIMARY KEY, btree ("ArtistId")
Referenced by:
      TABLE ""Album"" CONSTRAINT "FK_AlbumArtistId" FOREIGN KEY
("ArtistId") REFERENCES "Artist"("ArtistId")
```

It was giving an error while using the Table name directly for DML queries.

```
thorough-herring-67_db_8114218=> SELECT * FROM Album;
ERROR:  relation "album" does not exist
LINE 1: SELECT * FROM Album;
                      ^
```

Table name is case sensitive and using double quotes “” solves the error.

thorough-herring-67_db_8114218=> SELECT * FROM "Album";

Solution :

2. Return the top three genres found in the dataset in descending order;

Verified schema of “Genre” Table.

```
 thorough-herring-67_db_8114218=> \d "Genre";
                      Table "public.Genre"
 Column  |        Type         | Collation | Nullable | Default
---------+-----------------------+-----------+----------+---------
 GenreId | integer               |           | not null |
 Name | character varying(120) |          |          |
Indexes:
      "PK_Genre" PRIMARY KEY, btree ("GenreId")
Referenced by:
      TABLE ""Track"" CONSTRAINT "FK_TrackGenreId" FOREIGN KEY
("GenreId") REFERENCES "Genre"("GenreId")
```

Solution : Top three genres found in the dataset in descending order based on Genre name.

```
thorough-herring-67_db_8114218=> SELECT * FROM "Genre" ORDER BY "Name"
DESC;
 GenreId |        Name
---------+--------------------
      16 | World
      19 | TV Shows
      10 | Soundtrack
      18 | Science Fiction
      20 | Sci Fi & Fantasy
      5 | Rock And Roll
      1 | Rock
      8 | Reggae
      14 | R&B/Soul
```

```
      9 | Pop
      25 | Opera
      3 | Metal
      7 | Latin
      2 | Jazz
      17 | Hip Hop/Rap
      13 | Heavy Metal
      15 | Electronica/Dance
      12 | Easy Listening
      21 | Drama
      22 | Comedy
      24 | Classical
      11 | Bossa Nova
      6 | Blues
      4 | Alternative & Punk
      23 | Alternative
(25 rows)
```

Top three genres found in the dataset in descending order based on Genre ID.

```
thorough-herring-67_db_8114218=> SELECT * FROM "Genre" ORDER BY
"GenreId" DESC;
 GenreId |        Name
---------+--------------------
      25 | Opera
      24 | Classical
      23 | Alternative
      22 | Comedy
      21 | Drama
      20 | Sci Fi & Fantasy
      19 | TV Shows
      18 | Science Fiction
      17 | Hip Hop/Rap
      16 | World
      15 | Electronica/Dance
      14 | R&B/Soul
      13 | Heavy Metal
      12 | Easy Listening
      11 | Bossa Nova
      10 | Soundtrack
      9 | Pop
      8 | Reggae
      7 | Latin
      6 | Blues
```

```
      5 | Rock And Roll
      4 | Alternative & Punk
      3 | Metal
      2 | Jazz
      1 | Rock
(25 rows)
```

3. Return the number of tracks and average run time for each media type.

```
thorough-herring-67_db_8114218=> \d "Track";
                        Table "public.Track"
      Column      |           Type        | Collation | Nullable |
Default
--------------+------------------------+-----------+----------+---------
 TrackId      | integer                |           | not null |
 Name         | character varying(200) |           | not null |
 AlbumId      | integer                |           |          |
 MediaTypeId  | integer                |           | not null |
 GenreId      | integer                |           |          |
 Composer     | character varying(220) |           |          |
 Milliseconds | integer                |           | not null |
 Bytes        | integer                |           |          |
 UnitPrice    | numeric(10,2)          |           | not null |
Indexes:
      "PK_Track" PRIMARY KEY, btree ("TrackId")
      "IFK_TrackAlbumId" btree ("AlbumId")
      "IFK_TrackGenreId" btree ("GenreId")
      "IFK_TrackMediaTypeId" btree ("MediaTypeId")
Foreign-key constraints:
      "FK_TrackAlbumId" FOREIGN KEY ("AlbumId") REFERENCES
"Album"("AlbumId")
      "FK_TrackGenreId" FOREIGN KEY ("GenreId") REFERENCES
"Genre"("GenreId")
      "FK_TrackMediaTypeId" FOREIGN KEY ("MediaTypeId") REFERENCES
"MediaType"("MediaTypeId")
Referenced by:
      TABLE ""InvoiceLine"" CONSTRAINT "FK_InvoiceLineTrackId" FOREIGN
KEY ("TrackId") REFERENCES "Track"("TrackId")
      TABLE ""PlaylistTrack"" CONSTRAINT "FK_PlaylistTrackTrackId"
FOREIGN KEY ("TrackId") REFERENCES "Track"("TrackId")
```

Deliverables:

● Please remove any sensitive information/secrets from the below deliverables before sharing.
● A directory in a GitHub repo with the following contents:

```
hasura init azharshaikh-task-1 --endpoint https://azharshaikh-task-1.hasura.app
--admin-secret xxxxxxxxxxxxx
```

○ Briefly describe the steps taken to configure your Hasura GraphQL Engine
○ GraphQL query and results set for each of the above statements
○ Metadata in YAML format
○ Describe any challenges you encountered and your troubleshooting steps to address them.
○ SQL statements used directly against Postgres

All artifacts uploaded to Github repo :https://github.com/azharali49/azharshaikh-task1

Task 2:

Resolve configuration issues with a provided Hasura GraphQL environment. Do not use the environment from Task One to complete this task.
Technical details:
● Use Docker Compose and the provided docker-compose.yaml to instantiate the Hasura GraphQL Engine

curl -L https://github.com/hasura/graphql-engine/raw/stable/cli/get.sh | bash

Installed docker
brew install --cask docker

$ docker version
Client: Docker Engine - Community
 Version: 26.1.0
 API version: 1.45
 Go version: go1.22.2
 Git commit: 9714adc6c7

Built:	Mon Apr 22 17:00:04 2024
OS/Arch:	darwin/arm64
Context:	default

(base) azharshaikh-macbookpro:task-two-artifacts azharshaikh$ docker-compose up -d docker-compose.yaml
WARN[0000] /Users/azharshaikh/Downloads/tech-eval/task-two-artifacts/docker-compose.yaml: `version` is obsolete
no such service: docker-compose.yaml

Checked the file contents for version details
(base) azharshaikh-macbookpro:task-two-artifacts azharshaikh$ vi docker-compose.yaml

Made a copy of non-working file.
(base) azharshaikh-macbookpro:task-two-artifacts azharshaikh$ cp docker-compose.yaml docker-compose-not-working.yaml

Edited the version by replacing double quotes with single quotes.

(base) azharshaikh-macbookpro:task-two-artifacts azharshaikh$ vi docker-compose.yaml

Now it gives different errors.
(base) azharshaikh-macbookpro:task-two-artifacts azharshaikh$ docker-compose up -d
WARN[0000] /Users/azharshaikh/Downloads/tech-eval/task-two-artifacts/docker-compose.yaml: `version` is obsolete
[+] Running 3/3
 ✘ redis Error context canceled 1.9s
 ✘ graphql-engine Error failed to resolve reference "docker.io/hasura/graphql-engine:v2.48.2-pro": docke... 1.9s
 ✘ postgres Error context canceled 1.9s
Error response from daemon: failed to resolve reference "docker.io/hasura/graphql-engine:v2.48.2-pro": docker.io/hasura/graphql-engine:v2.48.2-pro: not found

The image “docker.io/hasura/graphql-engine:v2.48.2-pro” does not exist.
Corrected the version number in docker-compose.yaml file

Reference : https://hub.docker.com/r/hasura/graphql-engine/tags

Detailed output of all docker commands are mentioned in the file “docker command outputs”

The HAsura console was not available because HASURA_GRAPHQL_ENABLE_CONSOLE was set to FALSE and had to be changed to TRUE.
https://hasura.io/docs/latest/deployment/graphql-engine-flags/reference/#enable-console

Edited the value and restarted the container.

(base) azharshaikh-macbookpro:task-two-artifacts azharshaikh$ docker-compose up -d
WARN[0000]
/Users/azharshaikh/Downloads/tech-eval/task-two-artifacts/docker-compose.yaml: `version` is obsolete
[+] Running 3/3
✔ Container task-two-artifacts-redis-1 Running
0.0s
✔ Container task-two-artifacts-postgres-1 Running
0.0s
✔ Container task-two-artifacts-graphql-engine-1 Started
0.1s

The ENV VAR “PG_DATABASE_URL” is backward compatible. So no need to correct it to “HASURA_PG_DATABASE_URL”

● Use Postgres as the data source - localhost:5432
● Use the Chinook data set found here (hint: look into init.sql to make this easier)
● Use the provided metadata to configure the Hasura GraphQL Engine
● Assume the client will use the header x-hasura-artist-id for identifying the Artist
● Artists should not be allowed to access albums that do not belong to them
● Artists should not be allowed to access leverage aggregate queries

NOTE: I was not able to connect to the GRAPHQL console for docker based installation. All containers are UP and running but the console page was not loading.

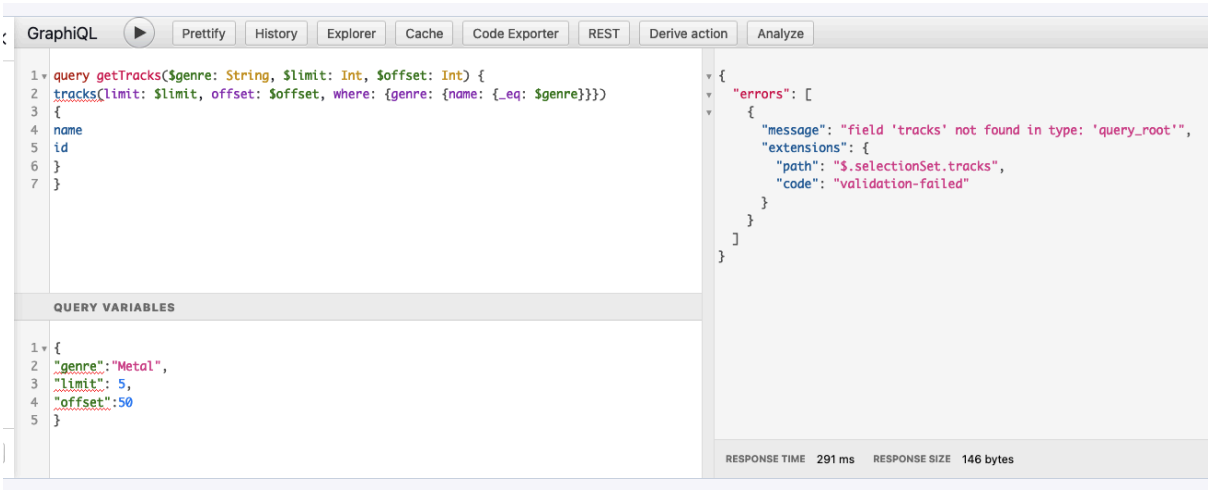
Objective:
1. Share your query, the headers used and the results of the following three queries:

```
# Execute as an administrator
query getTracks($genre: String, $limit: Int, $offset: Int) {
tracks(limit: $limit, offset: $offset, where: {genre: {name: {_eq:
$genre}}})
{
name
id
}
```

```
}
---
Query variables:
{
"genre":"Metal",
"limit": 5,
"offset":50
}
```

Output

```
{
  "errors": [
      {
      "message": "field 'tracks' not found in type: 'query_root'",
      "extensions": {
      "path": "$.selectionSet.tracks",
      "code": "validation-failed"
      }
      }
  ]
}
```




```
# Execute as an Artist
query getAlbumsAsArtist{
albums {
title
}
}
```

```
# Execute as an Artist
query trackValue {
tracks_aggregate {
aggregate {
sum {
unit_price
}
}
}
}
```

2. Execute a complex query of your choice, with and without caching. Share the query, the response and the response time for each.

Deliverables:
● Please remove any sensitive information/secrets from the below deliverables before sharing.
● A directory in the same GitHub repo with the following contents:
○ Describe any issues you discovered with the shared deployment artifacts and the steps you took to remediate the issue(s).
○ Include the requested artifacts from each above question
○ docker-compose.yaml for your working environment
○ Metadata in YAML format for your working environment
○ Describe any challenges you encountered when executing the above queries and your troubleshooting steps to address them.