Task One:

Setup an Hasura GraphQL instance with a working GraphQL API that includes correct permissions to return appropriate result sets.

Technical details:

- Use Docker Compose or Hasura Cloud
- Use Postgres as the data source
- Use the Chinook data set found here
- 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

Objective:

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

1. How many artists are in the database?
2. List the first track of every album by every artist in ascending order.
3. Get all albums for artist id = 5 without specifying a where clause.
4. Using a GraphQL mutation, add your favorite artist and one of their albums that isn't in the dataset.
5. How did you identify which ID to include in the mutation?

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

1. Return the artist with the most number of albums
2. Return the top three genres found in the dataset in descending order
3. Return the number of tracks and average run time for each media type

Deliverables:

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

Task Two:

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
- Use Postgres as the data source
- 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

Objective:

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

```
Unset
# 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
}
```

```
Unset
# Execute as an Artist

query getAlbumsAsArtist{
  albums {
    title
  }
}
```

```
Unset
# 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.