GraphQL Queries, Mutations & Subscriptions

For a comprehensive guide on GraphQL core concepts, visit:
https://hasura.io/learn/graphql/intro-graphql/core-concepts/

Login here to obtain a token and practice GraphQL queries and mutations:
https://hasura.io/learn/graphql/graphiql

Post Request URL:
https://hasura.io/learn/graphql

GraphQL Queries

GraphQL queries enable data retrieval with precision by specifying the structure of the response.

1. Fetch Users and Their Todos

Retrieve a list of users and the titles of their associated todos.

```
query {
   users {
     name
     todos {
       title
     }
   }
 }
```

2. Query with an Argument: Fetch Limited Todos

Limit the query to fetch only 10 todos instead of all available todos.

```
query {
  todos(limit: 10) {
    id
    title
  }
}
```

3. Query with Multiple Arguments

Fetch 2 users and their 5 most recent todos, sorted by descending creation date.

```
query {
  users(limit: 2) {
    id
    name
    todos(order_by: {created_at: desc}, limit: 5) {
      id
      title
    }
  }
}
```

4. Query with Variables

Specify the number of todos to fetch using a query variable.

```
query ($limit: Int!) {
  todos(limit: $limit) {
    id
    title
  }
}
```

Variables:

```
{
   "limit": 10
}
```

5. Query with Filter (Where Clause)

Filter todos to retrieve only public todos along with their title, public status, and completion status.

```
query {
  todos(where: {is_public: {_eq: true}}) {
    title
    is_public
    is_completed
  }
}
```

GraphQL Mutations

Mutations are used to perform data-modifying operations, such as inserting, updating, or deleting records.

Insert Operations

1. Simple Insert

Insert a todo with a specific title.

```
mutation {
  insert_todos(objects: [{title: "sdet"}]) {
    affected_rows
    returning {
      id
      created_at
      title
    }
  }
}
```

2. Insert Using Query Variables

Insert a todo object dynamically using variables.

```
mutation ($todo: todos_insert_input!) {
  insert_todos(objects: [$todo]) {
    affected_rows
    returning {
      id
      created_at
      title
    }
  }
}
```

Variables:

```
{
  "todo": {
    "title": "sdetautomation"
  }
}
```

Update Operations

1. Simple Update

Update a todo by modifying its title and marking it as completed based on its ID.

```
mutation {
  update_todos(where: {id: {_eq: 74533}}, _set: {title: "sdetqa",
is_completed: true}) {
    affected_rows
    returning {
      id
      title
      is_completed
    }
  }
}
```

2. Update Using Query Variables

Update a todo using variables for its ID, title, and completion status.

```
mutation ($id: Int, $title: String, $is_completed: Boolean) {
  update_todos(where: {id: {_eq: $id}}, _set: {title: $title,
is_completed: $is_completed}) {
    affected_rows
    returning {
      id
      title
      is_completed
    }
  }
}
```

Variables:

```
{
  "id": 74522,
  "title": "sdetqaautoamtion",
  "is_completed": true
}
```

Delete Operations

1. Simple Delete

Delete a todo by specifying its ID.

```
mutation {
  delete_todos(where: {id: {_eq: 74533}}) {
    affected_rows
    returning {
      title
    }
  }
}
```

GraphQL Subscriptions

Subscriptions enable real-time updates by listening to changes in data.

Example: Realtime Updates

```
subscription Greetings {
    greetings
}
```

Subscription URL: `https://graphql.postman-echo.com/graphql`

Notes:

- GraphQL Queries are flexible and allow precise data fetching using arguments, variables, and filters.
- Mutations provide powerful mechanisms for modifying data, including insert, update, and delete operations.
- Subscriptions enable real-time data updates, useful for live applications.
- Use variables to make queries and mutations dynamic and reusable.
- Always refer to the GraphQL schema to understand available fields and operations for queries and mutations.