GraphQL

GraphQL

- GraphQL is a Query Language
- Alternative to using a REST API

```
Query {
  books {
    title,
    author,
    price
  }
}
```

REST API

Over fetching:

Getting back more data than we need

mysite.com/api/courses


```
{
  "id": "1",
  "title": "Thud",
  "author": {...},
  "price": "10.99",
  "thumbnail_url": "...",
  "video_url": "...",
}
```

REST API

Under fetching:

Getting back less data than we need

mysite.com/api/courses/1 →

```
{
  "id": "1",
  "title": "Thud",
  "author": {...},
  "price": "10.99",
  "thumbnail_url": "...",
  "video_url": "...",
}
```

Additional request:

mysite.com/api/author/1

GraphQL

Single Endpoint

mygrahqlsite.com/graphql

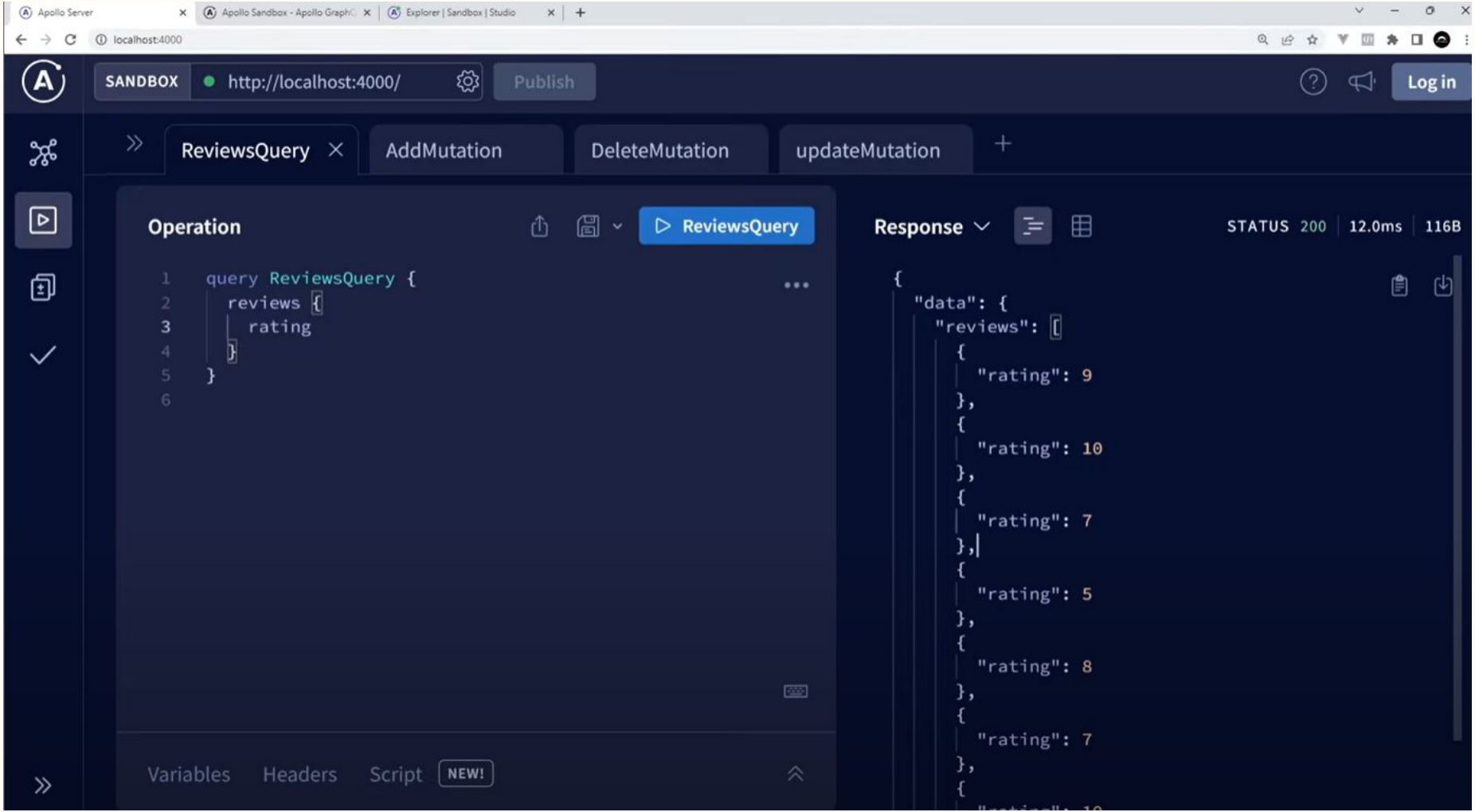
```
Query {
  courses {
    id,
    title,
    thumbnail_url
  }
}
```

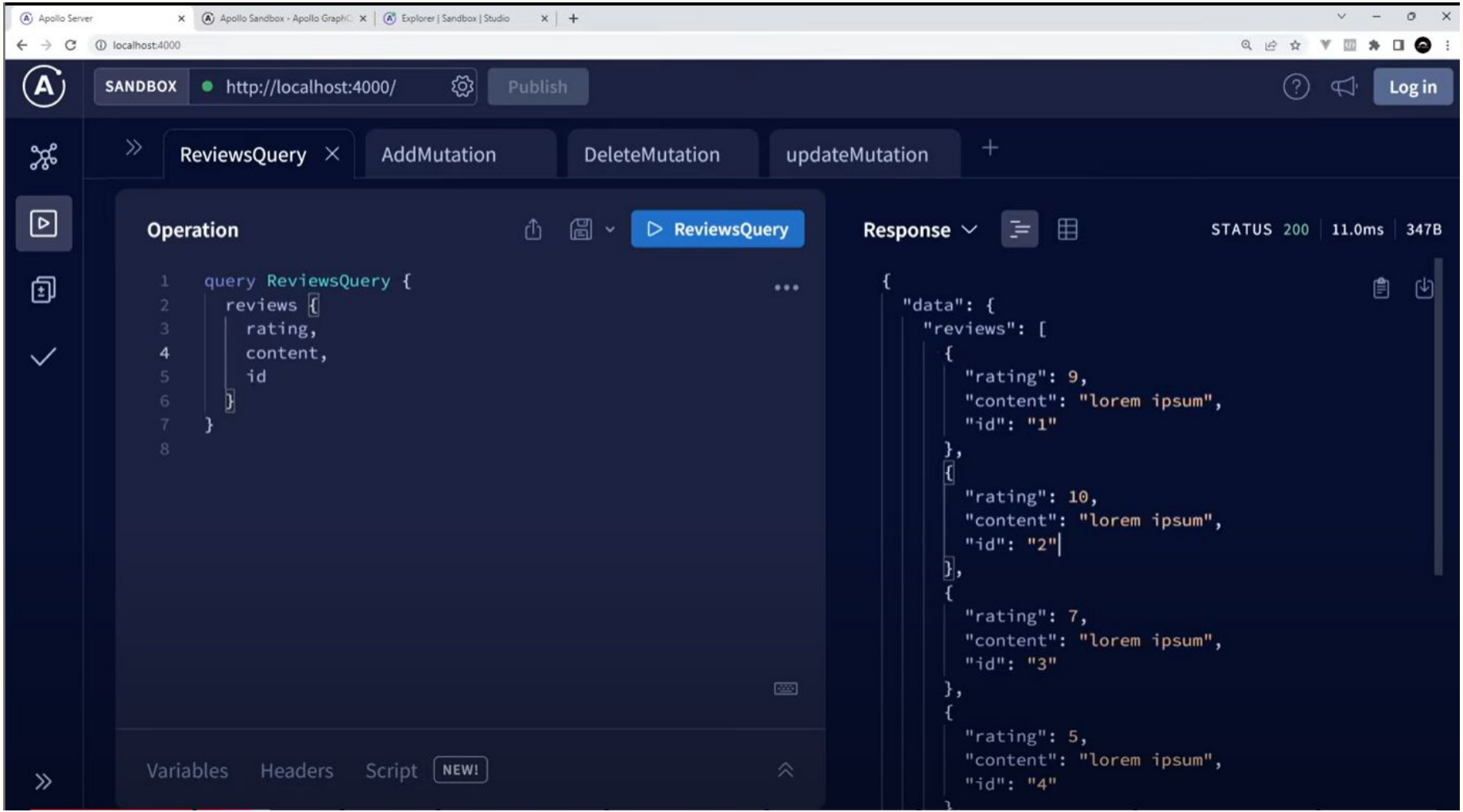
GraphQL

Single Endpoint

mygrahqlsite.com/graphql

```
Query {
  course(id: "1") {
    id,
    title,
    thumbnail_url,
    author {
      name,
      id,
      courses {
        id,
        title,
        thumbnail_url
      }
    }
  }
}
```


Apollo Server
Apollo Sandbox - Apollo GraphQ
Explorer | Sandbox | Studio
localhost:4000
SANDBOX
http://localhost:4000/
Publish
Log in
ReviewsQuery
AddMutation
DeleteMutation
updateMutation
Operation
ReviewsQuery
query ReviewsQuery {
reviews {
rating
}
}
Variables
Headers
Script
NEW!
Response
STATUS 200
12.0ms
116B
{
"data": {
"reviews": [
{
"rating": 9
},
{
"rating": 10
},
{
"rating": 7
},
{
"rating": 5
},
{
"rating": 8
},
{
"rating": 7
},
{

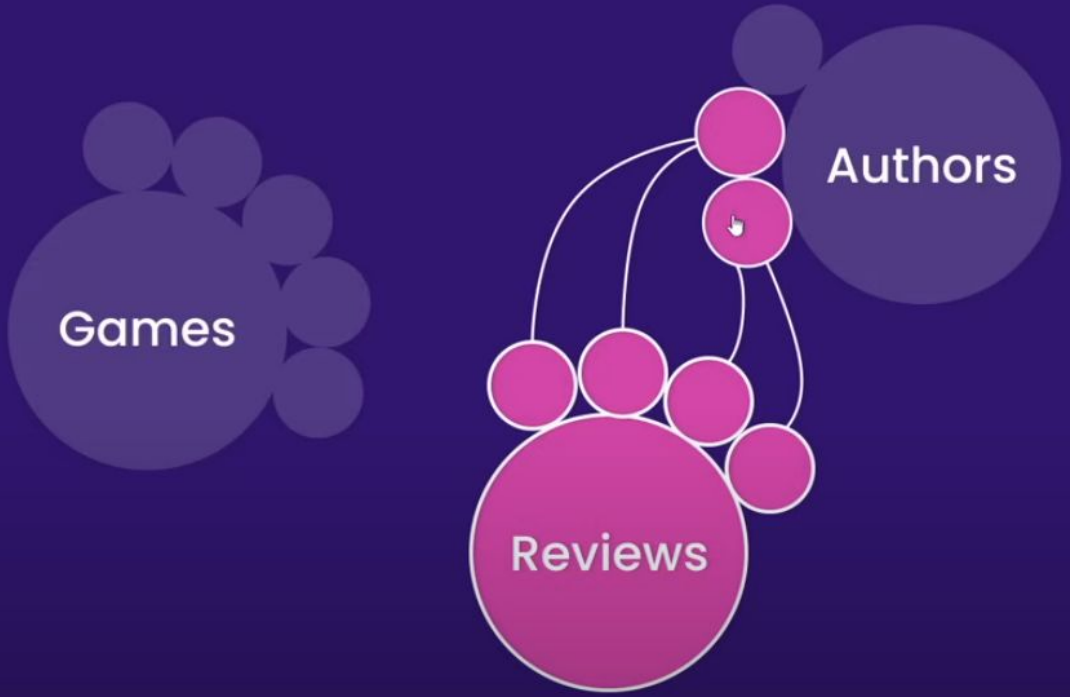

Apollo Server
Apollo Sandbox - Apollo Graph
Explorer | Sandbox | Studio
localhost:4000
SANDBOX
http://localhost:4000/
Publish
Log in
ReviewsQuery
AddMutation
DeleteMutation
updateMutation
Operation
ReviewsQuery
query ReviewsQuery {
reviews {
rating,
content,
id
}
}
Variables
Headers
Script
NEW!
Response
STATUS 200
11.0ms
347B
{
"data": {
"reviews": [
{
"rating": 9,
"content": "lorem ipsum",
"id": "1"
},
{
"rating": 10,
"content": "lorem ipsum",
"id": "2"
},
{
"rating": 7,
"content": "lorem ipsum",
"id": "3"
},
{
"rating": 5,
"content": "lorem ipsum",
"id": "4"

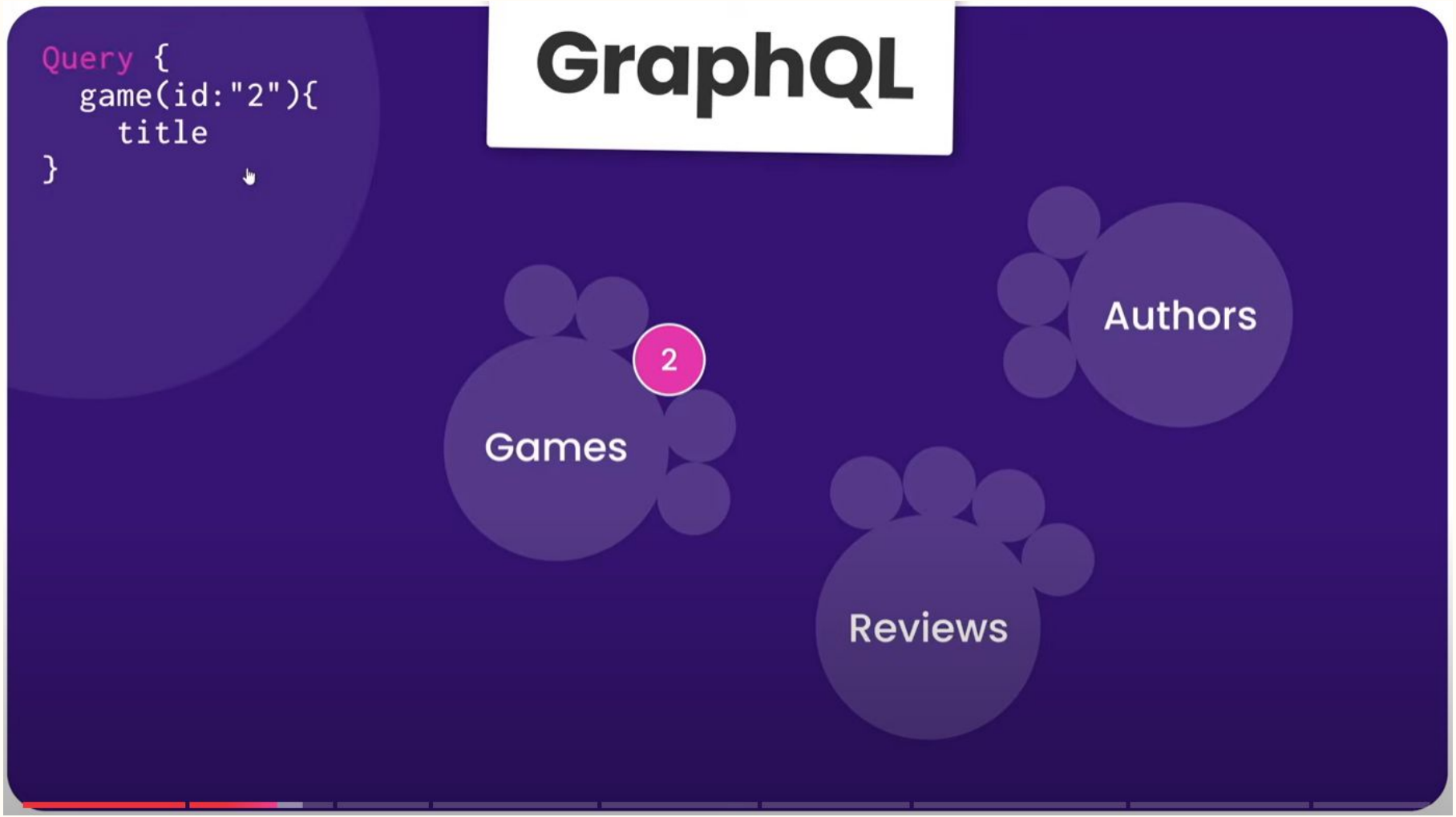
GraphQL

```
Query {
  reviews {
    rating
  }
}
```

GraphQL

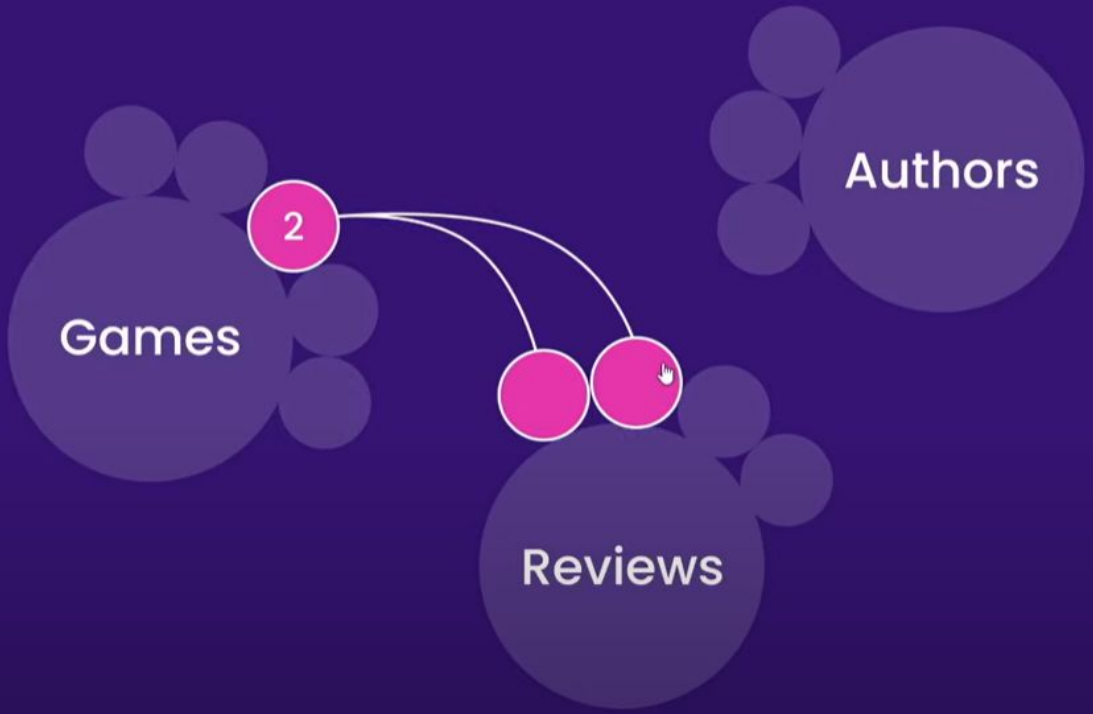
```
Query {
  reviews {
    rating
    author {
      name
    }
  }
}
```




Query {
game(id:"2"){
title
}
GraphQL
2
Games
Authors
Reviews

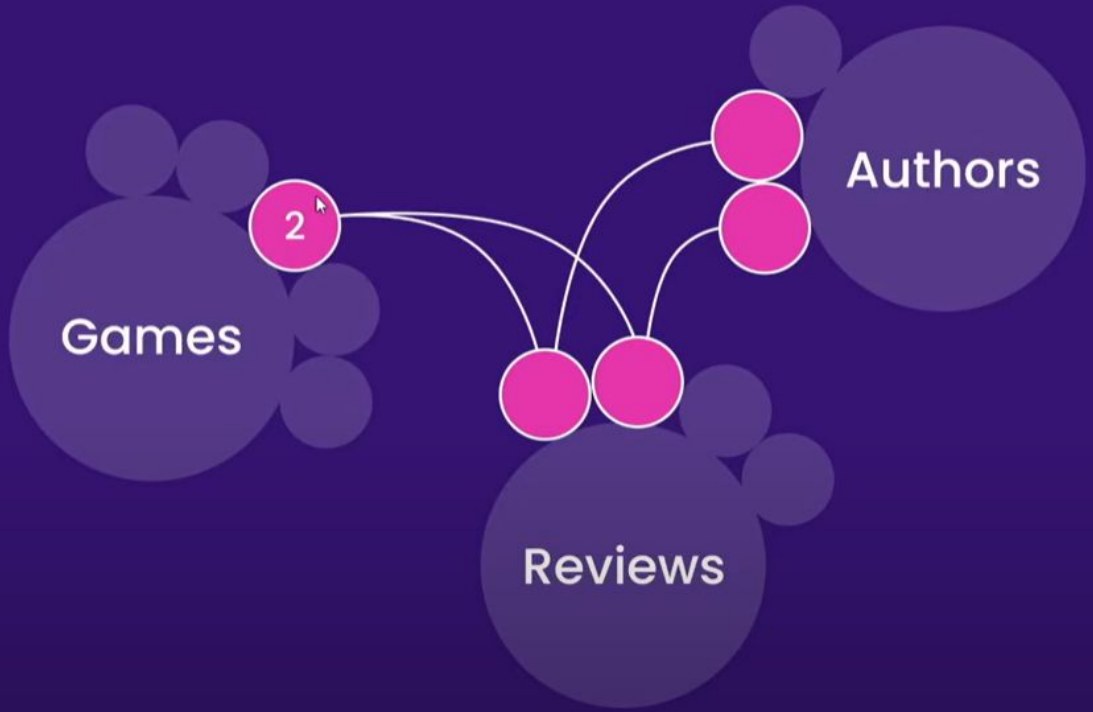
```
Query {
  game(id:"2"){
    title,
    review {
      rating
    },
}
```

GraphQL



GraphQL

```
Query {
  game(id:"2"){
    title,
    review {
      rating,
      author {
        name
      }v
    },
}
```



index.js | package.json

index.js > ...

```
1   import { ApolloServer } from '@apollo/server';
2   import { startStandaloneServer } from '@apollo/server/standalone';
3
4   //server setup
5   const server = new ApolloServer({
6       // typeDefs - descriptions of datatypes and their relationships with other datatypes
7       // resolvers - functions that determine how we respond to queries for different data on the graph
8   })
9
10  const {url} = await startStandaloneServer(server, {
11      listen: {port: 4000}
12  })
13
14  console.log('Server read at port', 4000);
```

index.js | schema.js ●

schema.js > ...

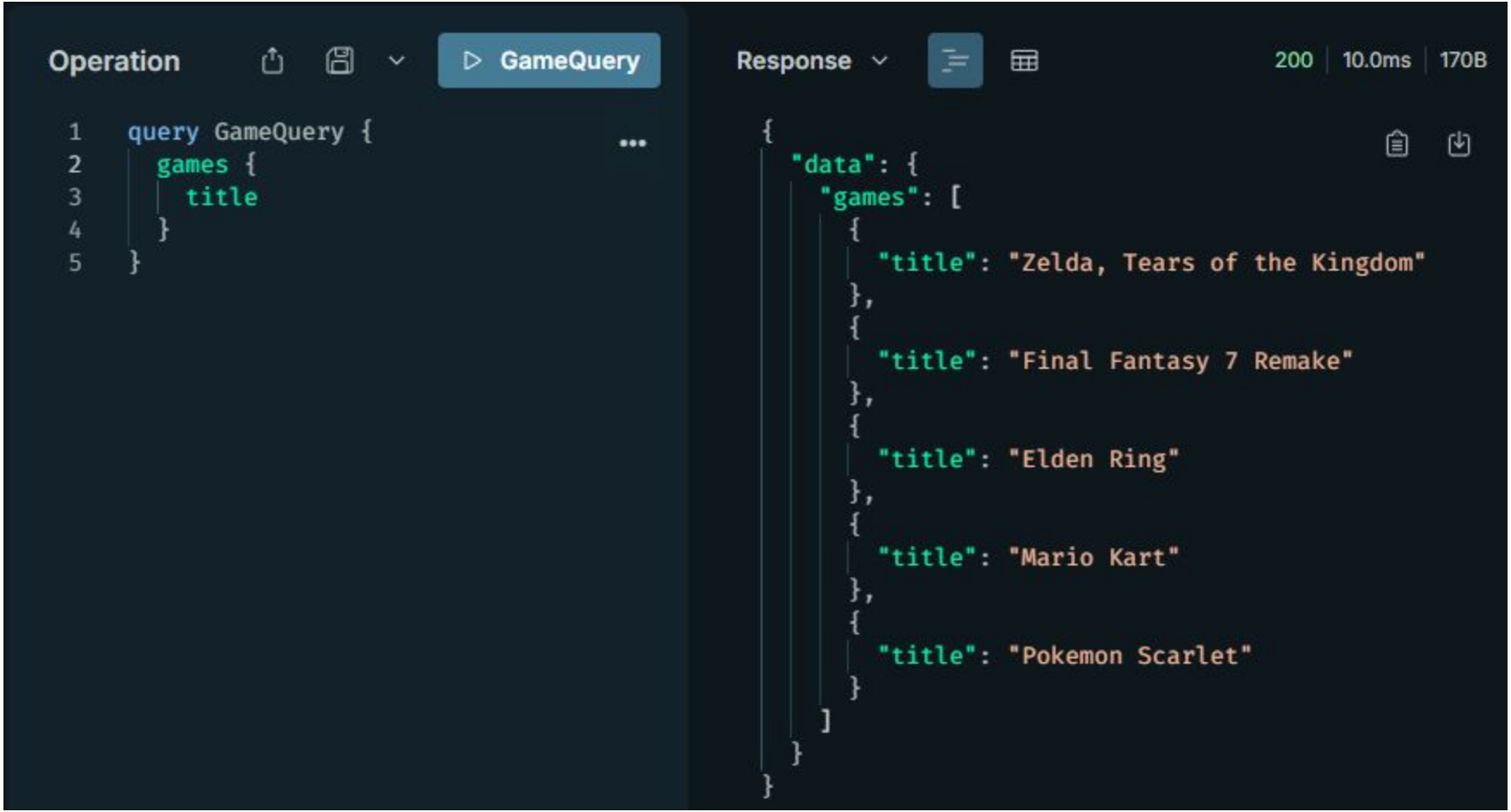
```graphql
export const typeDefs = `#graphql
    type Game {
        id: ID!
        title: String!
        platform: [String!]!
    }
    type Review {
        id: ID!
        rating: Int!,
        content: String!
    }
    type Author {
        id: ID!
        name: String!
        verified: Boolean!
    }
    type Query {
        reviews: [Reviews]
        games: [Game]
        authors: [Author]
    }
`
// Types - Int, float, String, boolean, ID
// Add a "!" to a type to make it non-nullable
```

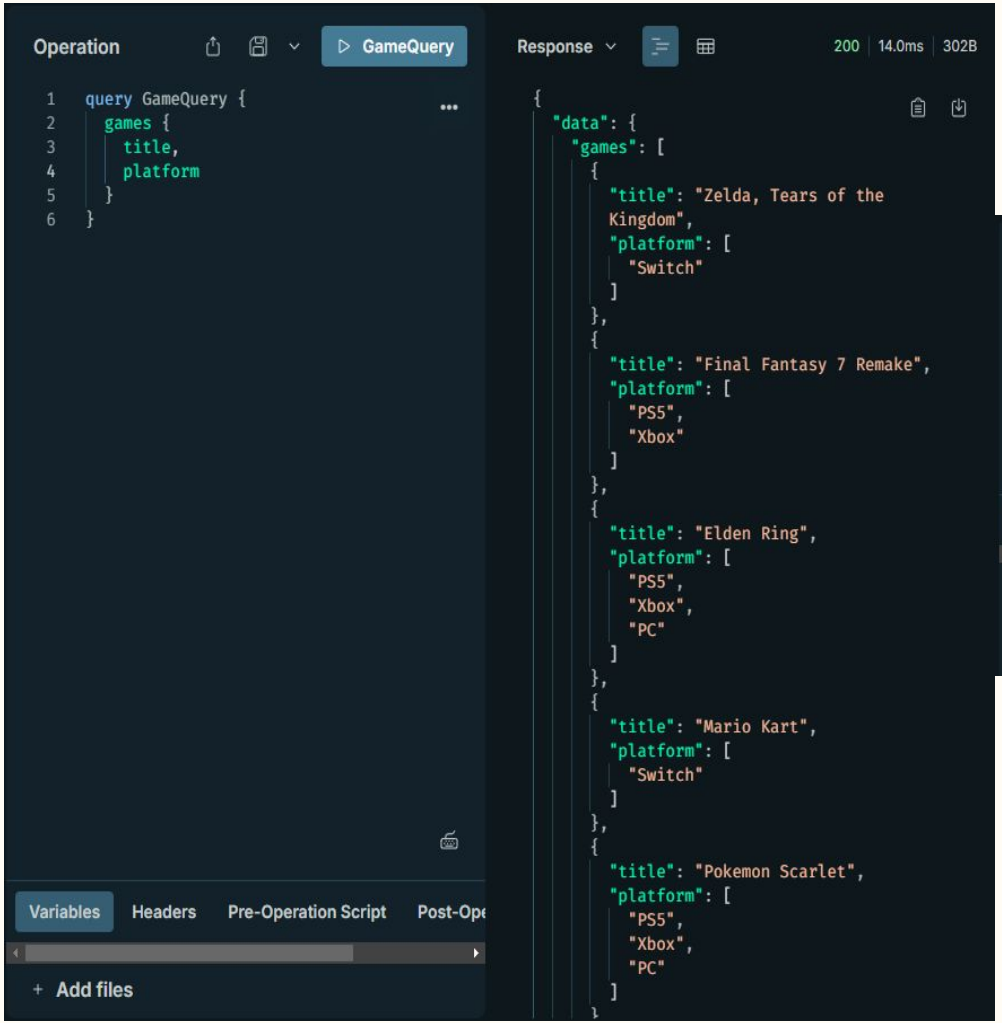
```
const resolvers = {
    Query:{
        Codeium: Refactor | Explain | Generate JSDoc | X
        games() {
            return db.games;
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        reviews() {
            return db.reviews;
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        authors() {
            return db.authors;
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        review(_, args, context) {
            return db.reviews.find((review)=> review.id === args.id);
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        game(_, args, context){
            return db.games.find((game)=> game.id === args.id);
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        author(_, args, context){
            return db.authors.find((author) => author.id === args.id);
        }
    }
}

```

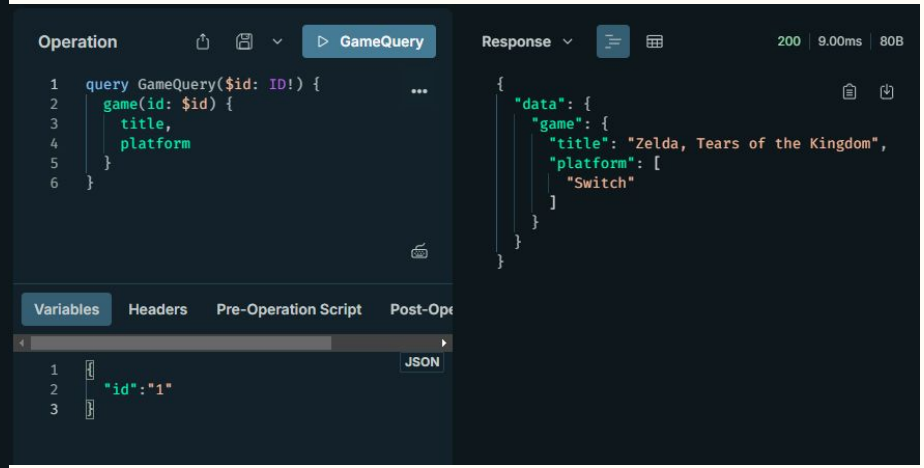
Resolver in index.js

Schema.js

```
type Query {
    reviews: [Review]
    review(id: ID!): Review
    games: [Game]
    game(id: ID!): Game
    authors: [Author]
    author(id: ID!): Author
}
```


Operation
GameQuery
Response
200 | 10.0ms | 170B
query GameQuery {
games {
title
}
}
{
"data": {
"games": [
{
"title": "Zelda, Tears of the Kingdom"
},
{
"title": "Final Fantasy 7 Remake"
},
{
"title": "Elden Ring"
},
{
"title": "Mario Kart"
},
{
"title": "Pokemon Scarlet"
}
]
}
}


Operation
GameQuery
Response
200
14.0ms
302B
query GameQuery {
games {
title,
platform
}
}
{
"data": {
"games": [
{
"title": "Zelda, Tears of the Kingdom",
"platform": [
"Switch"
]
},
{
"title": "Final Fantasy 7 Remake",
"platform": [
"PS5",
"Xbox"
]
},
{
"title": "Elden Ring",
"platform": [
"PS5",
"Xbox",
"PC"
]
},
{
"title": "Mario Kart",
"platform": [
"Switch"
]
},
{
"title": "Pokemon Scarlet",
"platform": [
"PS5",
"Xbox",
"PC"
]
}
Variables
Headers
Pre-Operation Script
Post-Ope
Add files

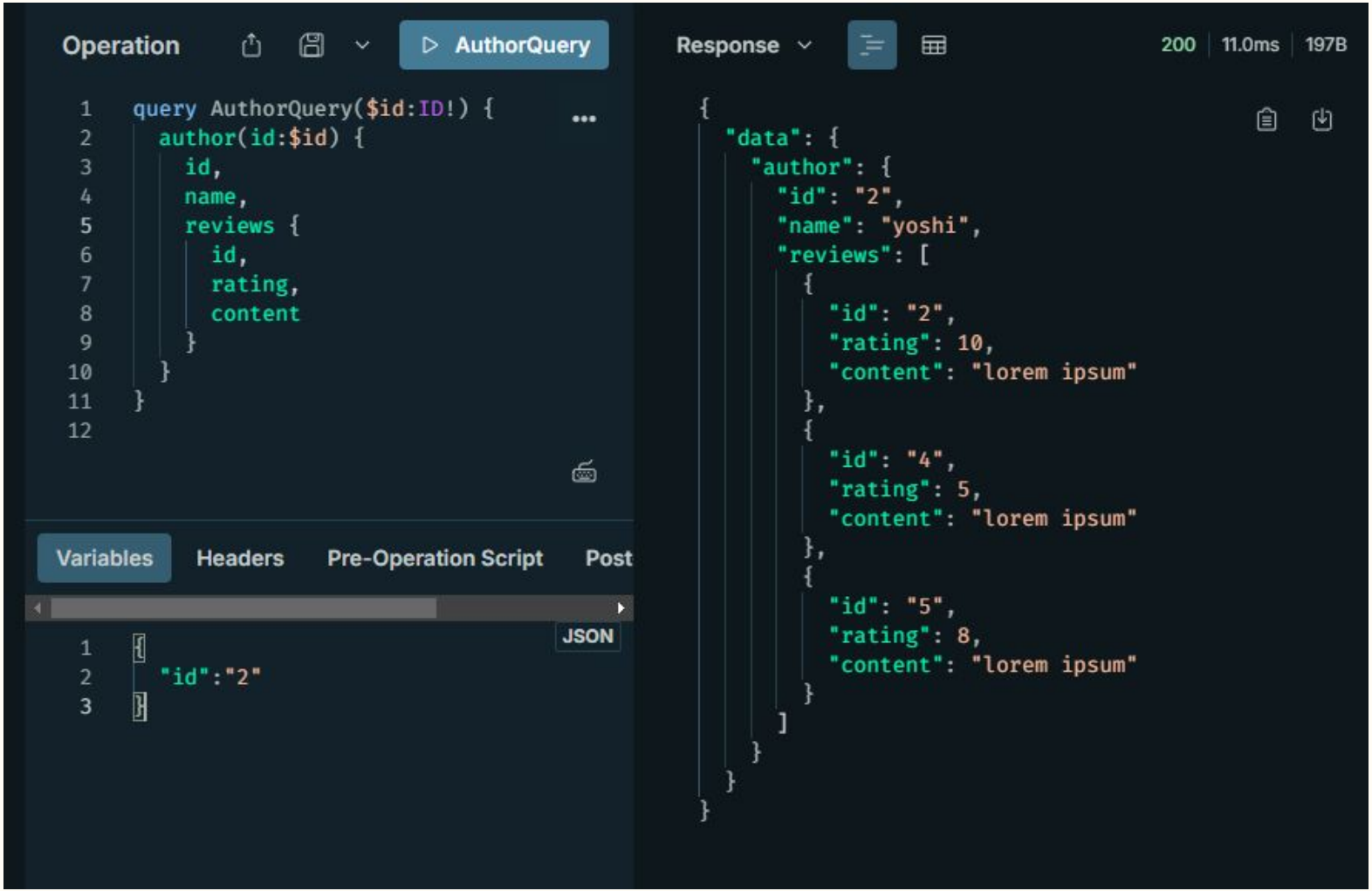

Operation
GameQuery
Response
200
9.00ms
80B
query GameQuery($id: ID!) {
game(id: $id) {
title,
platform
}
}
{
"data": {
"game": {
"title": "Zelda, Tears of the Kingdom",
"platform": [
"Switch"
]
}
}
}
Variables
Headers
Pre-Operation Script
Post-Ope
JSON
{
"id":"1"
}

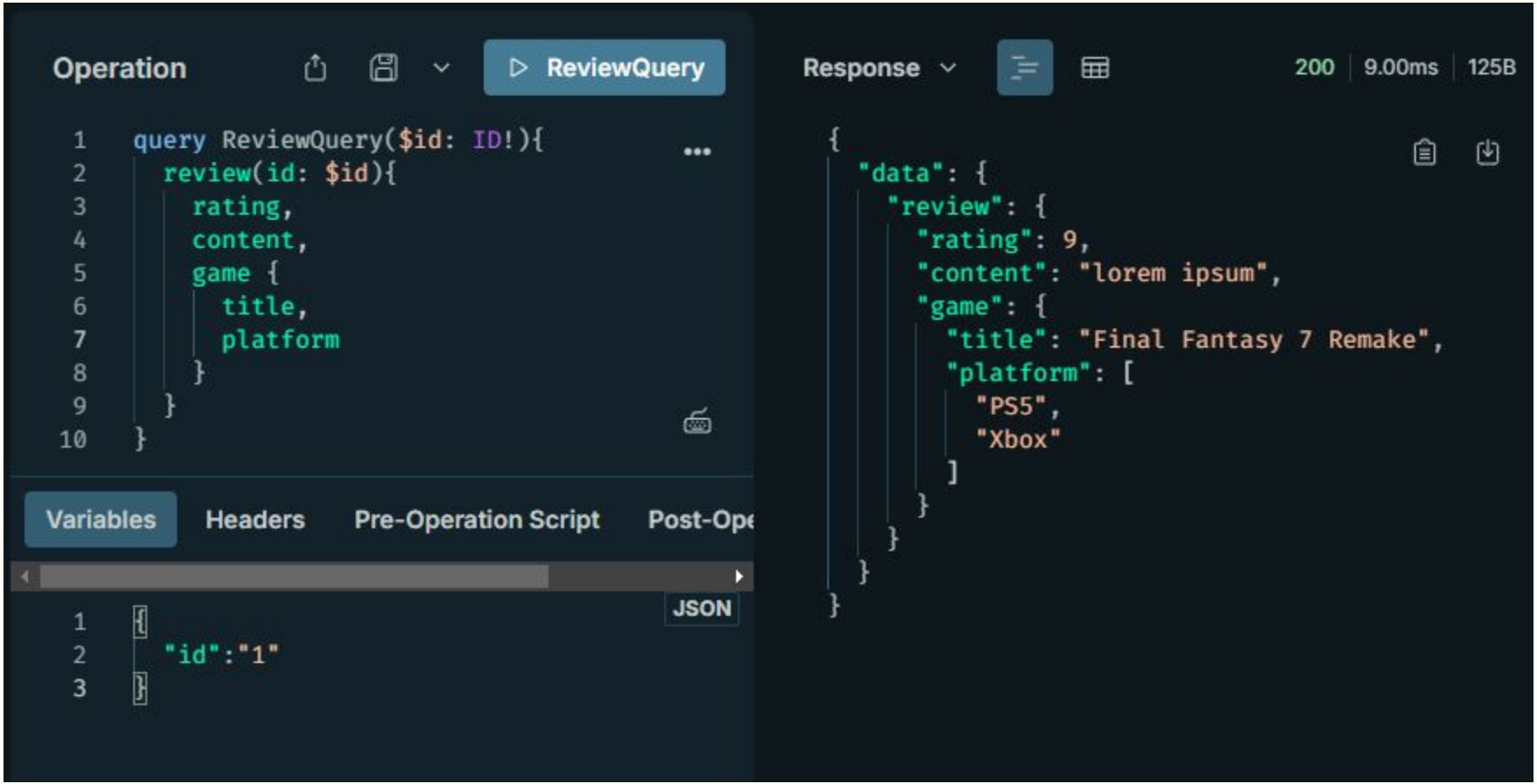
Properties in resolvers

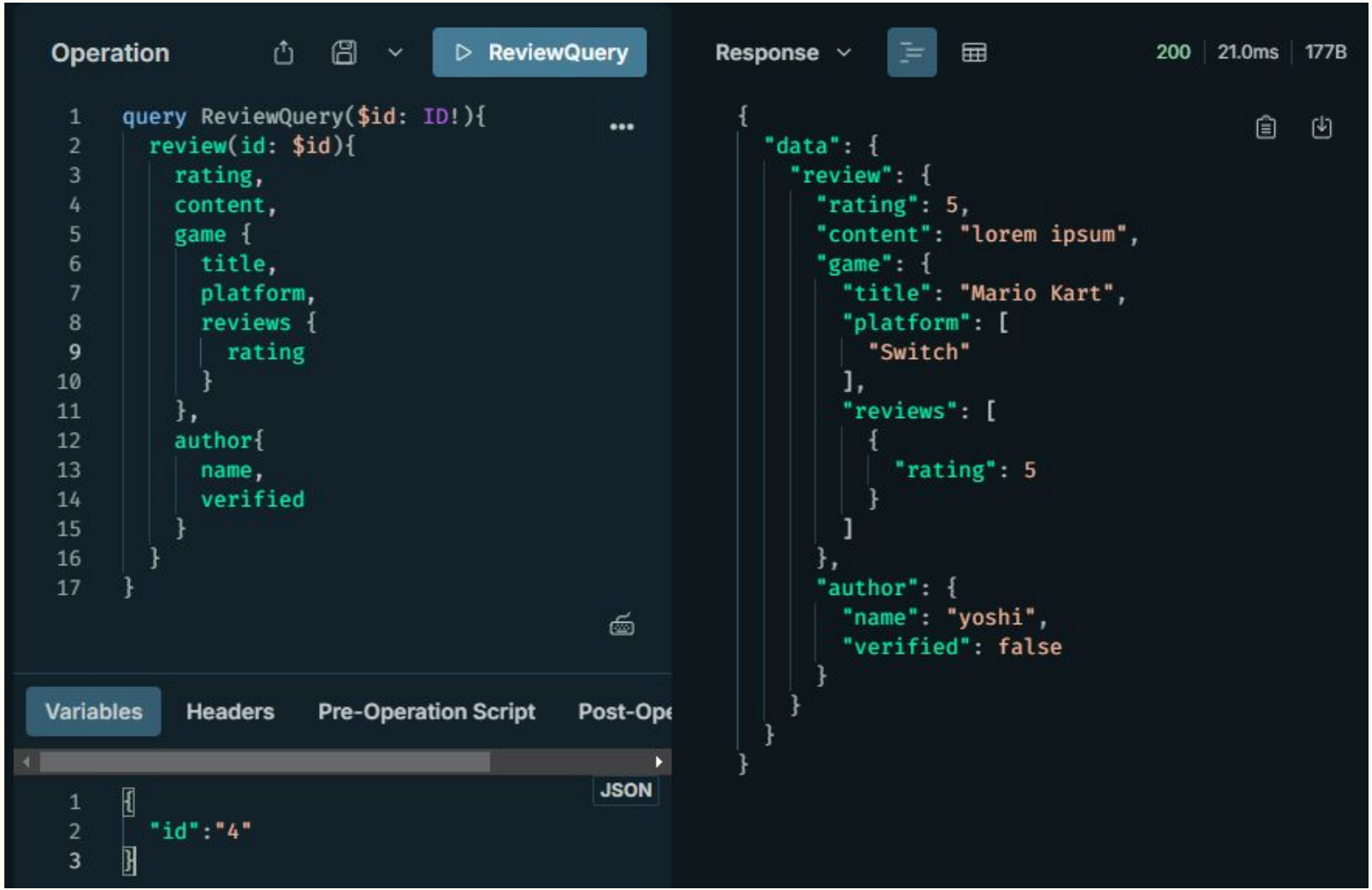
```
const resolvers = {
    Query:{
        Codeium: Refactor | Explain | Generate JSDoc | X
        games() {
            return db.games;
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        reviews() {
            return db.reviews;
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        authors() {
            return db.authors;
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        review(_, args, context) {
            return db.reviews.find((review)=> review.id === args.id);
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        game(_, args, context){
            return db.games.find((game)=> game.id === args.id);
        },
        Codeium: Refactor | Explain | Generate JSDoc | X
        author(_, args, context){
            return db.authors.find((author) => author.id === args.id);
        }
    },
    Game: {
        Codeium: Refactor | Explain | Generate JSDoc | X
        reviews(parent) {
            return db.reviews.filter((review)=>review.game_id === parent.id);
        }
    }
}
```

```
29      Game: {
            Codeium: Refactor | Explain | Generate JSDoc | X
30          reviews(parent) {
31              return db.reviews.filter((review)=>review.game_id === parent.id);
32          }
33      },
34      Author: {
            Codeium: Refactor | Explain | Generate JSDoc | X
35          reviews(parent) {
36              return db.reviews.filter((review)=>review.author_id === parent.id);
37          }
38      },
39      Review: {
            Codeium: Refactor | Explain | Generate JSDoc | X
40          game(parent) {
41              return db.games.find((game)=> game.id === parent.game_id);
42          },
            Codeium: Refactor | Explain | Generate JSDoc | X
43          author(parent) {
44              return db.authors.find((author)=> author.id === parent.author_id);
45          }
46      }
```

```
1   export const typeDefs = `#graphql
2       type Game {
3           id: ID!
4           title: String!
5           platform: [String!]!
6           reviews: [Review!]
7       }
8       type Review {
9           id: ID!
10          rating: Int!,
11          content: String!,
12          author: Author!,
13          game: Game!
14      }
15      type Author {
16          id: ID!
17          name: String!
18          verified: Boolean!
19          reviews: [Review!]
20      }
21      type Query {
22          reviews: [Review]
23          review(id: ID!): Review
24          games: [Game]
25          game(id: ID!): Game
26          authors: [Author]
27          author(id: ID!): Author
28      }
29  `
```


Operation
AuthorQuery
query AuthorQuery($id:ID!) {
author(id:$id) {
id,
name,
reviews {
id,
rating,
content
}
}
}
Variables
Headers
Pre-Operation Script
Post
JSON
{
"id":"2"
}
Response
200
11.0ms
197B
{
"data": {
"author": {
"id": "2",
"name": "yoshi",
"reviews": [
{
"id": "2",
"rating": 10,
"content": "lorem ipsum"
},
{
"id": "4",
"rating": 5,
"content": "lorem ipsum"
},
{
"id": "5",
"rating": 8,
"content": "lorem ipsum"
}
]
}
}
}


Operation
ReviewQuery
query ReviewQuery($id: ID!){
review(id: $id){
rating,
content,
game {
title,
platform
}
}
}
Variables
Headers
Pre-Operation Script
Post-Ope
JSON
{
"id":"1"
}
Response
200
9.00ms
125B
{
"data": {
"review": {
"rating": 9,
"content": "lorem ipsum",
"game": {
"title": "Final Fantasy 7 Remake",
"platform": [
"PS5",
"Xbox"
]
}
}
}
}


Operation
ReviewQuery
query ReviewQuery($id: ID!){
review(id: $id){
rating,
content,
game {
title,
platform,
reviews {
rating
}
},
author{
name,
verified
}
}
}
Variables
Headers
Pre-Operation Script
Post-Ope
JSON
{
"id":"4"
}
Response
200
21.0ms
177B
{
"data": {
"review": {
"rating": 5,
"content": "lorem ipsum",
"game": {
"title": "Mario Kart",
"platform": [
"Switch"
],
"reviews": [
{
"rating": 5
}
]
},
"author": {
"name": "yoshi",
"verified": false
}
}
}
}

```
47      Mutation: {
            Codeium: Refactor | Explain | Generate JSDoc | X
48          deleteGame(_, args, context) {
49              db.games = db.games.filter((game)=> game.id !== args.id);
50              return db.games;
51          },
            Codeium: Refactor | Explain | Generate JSDoc | X
52          addGame(_, args, context) {
53              let game = {
54                  ...args.game,
55                  id: Math.floor(Math.random()*10000).toString()
56              }
57              db.games.push(game);
58              return game;
59          },
            Codeium: Refactor | Explain | Generate JSDoc | X
60          updateGame(_, args) {
61              db.games = db.games.map((game)=>{
62                  if(game.id === args.id) {
63                      return {
64                          ...game,
65                          ...args.edits
66                      }
67                  }
68                  return game;
69              })
70              return db.games.find((game)=>game.id === args.id);
71          }
72      }
```

Add mutations to your resolvers and schema like this to perform create, update and delete operations

```
29      type Mutation {
30          deleteGame(id: ID!): [Game]
31          addGame(game: AddGameInput!): Game
32          updateGame(id: ID!, edits: EditGameInput!): Game
33      }
34      input AddGameInput {
35          title: String!
36          platform: [String!]
37      }
38
39      input EditGameInput {
40          title: String
41          platform: [String!]
42      }
```

Operation

AddGame

```
mutation AddGame($game: AddGameInput!) {
  addGame(game: $game){
    id,
    title,
    platform
  }
}
```

Variables | Headers | Pre-Operation Script | Post-Operation Script

JSON

```
{
  "game":{
    "title": "Rocket League",
    "platform": ["PS5", "Switch", "Xbox"],
  }
}
```

Response

```
{
  "data": {
    "addGame": {
      "id": "3339",
      "title": "Rocket League",
      "platform": [
        "PS5",
        "Switch",
        "Xbox"
      ]
    }
  }
}
```

Operation

UpdateGame

```
mutation UpdateGame($id: ID!, $edits: EditGameInput!) {
  updateGame(id: $id, edits: $edits){
    id,
    title,
    platform
  }
}
```

Variables | Headers | Pre-Operation Script | Post-Operation Script

JSON

```
{
  "id":"1",
  "edits": {
    "title":"Zelda",
    "platform":["PS5"]
  }
}
```

Response

```
{
  "data": {
    "updateGame": {
      "id": "1",
      "title": "Zelda",
      "platform": [
        "PS5"
      ]
    }
  }
}
```