

MOVIE TICKET RESERVATION SYSTEM

P4

| Team Members | NUID |
|------------------------|-----------|
| Krutik Kanakia | 002787847 |
| Kumar Mehul | 002761391 |
| Mansi Sanjeev Upadhyay | 002766397 |
| Tanuj Verma | 002726506 |

Data Model: Document (NoSQL)

Target Platform: Arango DB

Objective/Scope:

- Create a Database System to store movie reservation system information.
- Implement Data Validation to ensure that the data entered in the database is accurate and consistent.
- Use indexing to improve the performance and scalability of the database.
- Implement security measures to protect the sensitive information of the customers.
- Use visualizations to discover the trends and movie popularity among the customers.

Visualization Tool: Tableau

DATA REFRESH

We have used python to implement data refresh.

Whenever either of the node files are changed, this script will implement the made changes into our Arango DB web UI.

The json files of all the collections as per our code exists in path:

`"/Users/tanujverma/Desktop/NEU/ADBMS/ArangoDB`

We are monitoring our json files in this particular path for any changes

Implementation Example

We have created below : **ONGOING DATA REFRESH**

(we can create a script of below code and run it via terminal for constant monitoring)

```
import json
import time
from arango import ArangoClient
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

class MyHandler(FileSystemEventHandler):
    def on_modified(self, event):
        if event.src_path.endswith('.json'):
            update_arango_db(event.src_path)

def update_arango_db(json_file_path):
    file_name = json_file_path.split('/')[-1].split('.')[0]

    if file_name in ['Movie', 'Theater', 'ShowTime', 'Reservation', 'Customer']:
        update_collection(json_file_path)
        print(f"Collection updated: {file_name}")

def update_collection(json_file_path):
    collection_name = json_file_path.split('/')[-1].split('.')[0]

    # Connect to the ArangoDB server
    client = ArangoClient(hosts='http://localhost:8529')
    db = client.db('_system', username='root', password='')

    # Clear the existing collection
    if db.has_collection(collection_name):
        collection = db.collection(collection_name)
        collection.truncate()
    else:
        # Create the collection if it doesn't exist
        db.create_collection(collection_name)
        collection = db.collection(collection_name)

    # Import data from the JSON file
```

```

with open(json_file_path, 'r') as f:
    data = json.load(f)
    for document in data:
        collection.insert(document)

def main():
    path = "/Users/tanujverma/Desktop/NEU/ADBMS/ArangoDB" # Set your path to the JSON files
    event_handler = MyHandler()
    observer = Observer()
    observer.schedule(event_handler, path, recursive=False)
    observer.start()

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()

    observer.join()

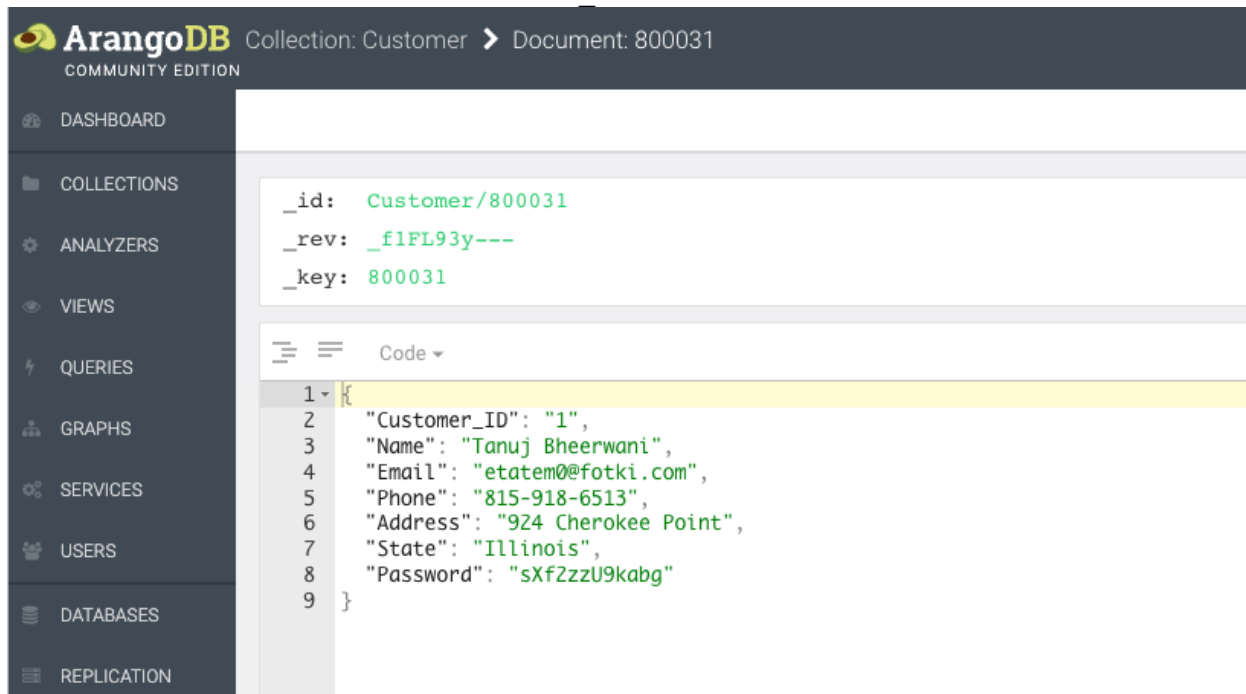
if __name__ == "__main__":
    main()

```

Now when ever we change any attributes in the code that particular collection will be reflected

Example of Implementation

Below are the customer details for Customer ID 1 :



The screenshot shows the ArangoDB web interface. The top header indicates the collection is 'Customer' and the document is '800031'. The left sidebar contains navigation options: DASHBOARD, COLLECTIONS, ANALYZERS, VIEWS, QUERIES, GRAPHS, SERVICES, USERS, DATABASES, and REPLICATION. The main content area displays the document details for 'Customer/800031'.

Document Details:

- _id: Customer/800031
- _rev: _f1FL93y---
- _key: 800031

The document is displayed in a JSON format in the 'Code' view:

```

1 {
2   "Customer_ID": "1",
3   "Name": "Tanuj Bheerwani",
4   "Email": "etatem0@fotki.com",
5   "Phone": "815-918-6513",
6   "Address": "924 Cherokee Point",
7   "State": "Illinois",
8   "Password": "sXf2zzU9kabg"
9 }

```

We are going to change the Customer Name

Now we make change into the JSON file of Customer

Originally in our json file , we have the name as “Tanuj Bheerwani”

```
Untitled-1.ipynb • {} Customer.json •
Users > tanujverma > Desktop > NEU > ADBMS > ArangoDB > {} Customer.json > ...
1 [{"_key":"800031",
2   "_id":"Customer/800031",
3   "_rev":"_f1E45iu---",
4   "Customer_ID":"1",
5   "Name":"Tanuj Bheerwani",
6   "Email":"etatem0@fotki.com",
7   "Phone":"815-918-6513",
8   "Address":"924 Cherokee Point",
9   "State":"Illinois",
10  "Password":"sXf2zzU9kabg"},
11
12  {"_key":"800033","_id":"Customer/800033","_rev":"_f1E45jG---","Customer_ID":"3","Name":"Donnie Turnor","Email":"dturnor@fotki.com","Phone":"815-918-6513","Address":"924 Cherokee Point","State":"Illinois","Password":"sXf2zzU9kabg"}]
```

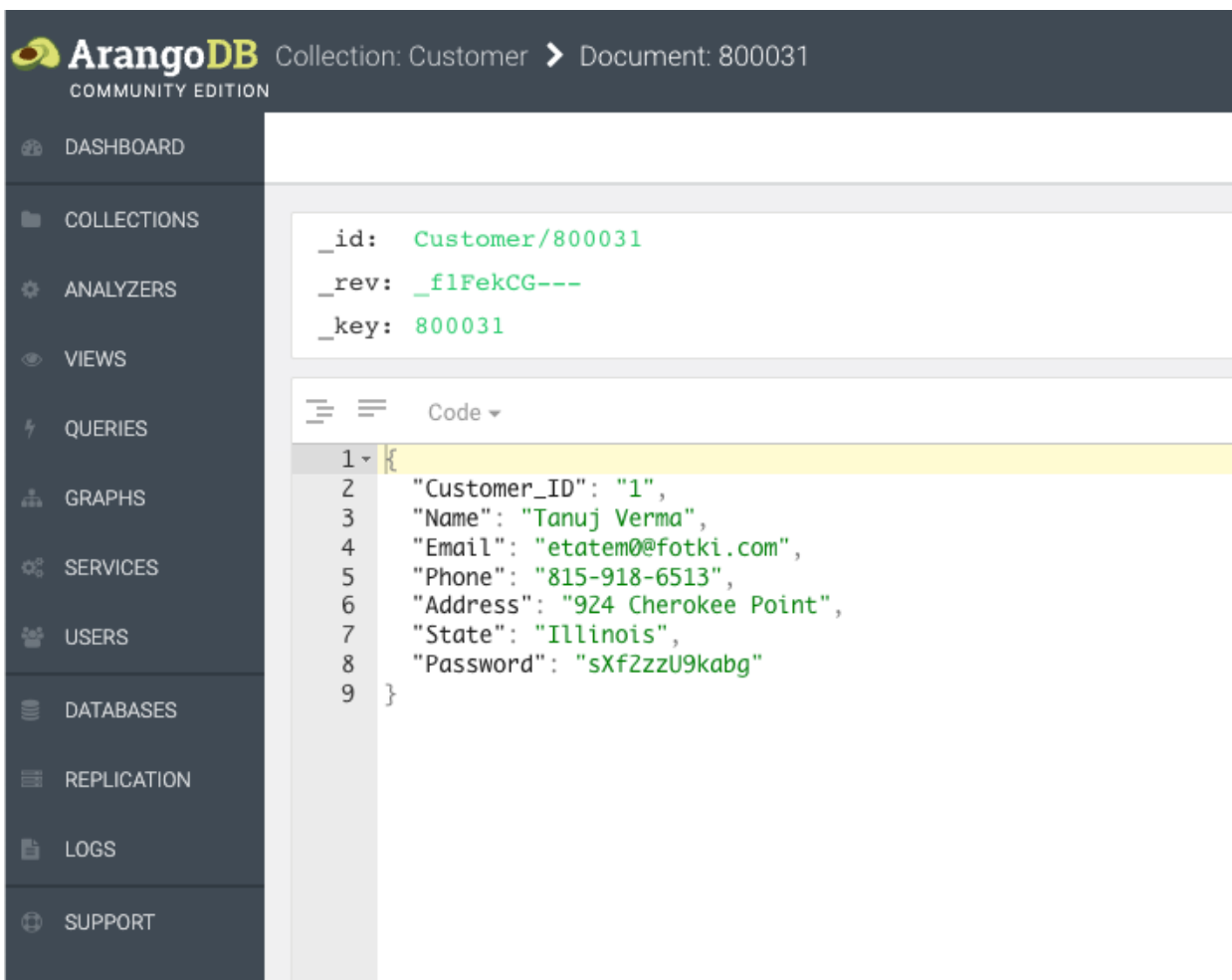
We change it to “Tanuj Verma” and save it

```
Untitled-1.ipynb • {} Customer.json •
Users > tanujverma > Desktop > NEU > ADBMS > ArangoDB > {} Customer.json > {} 0 > [Name]
1 [{"_key":"800031",
2   "_id":"Customer/800031",
3   "_rev":"_f1E45iu---",
4   "Customer_ID":"1",
5   "Name":"Tanuj Verma",
6   "Email":"etatem0@fotki.com",
7   "Phone":"815-918-6513",
8   "Address":"924 Cherokee Point",
9   "State":"Illinois",
10  "Password":"sXf2zzU9kabg"},
11
12  {"_key":"800033","_id":"Customer/800033","_rev":"_f1E45jG---","Customer_ID":"3","Name":"Donnie Turnor","Email":"dturnor@fotki.com","Phone":"815-918-6513","Address":"924 Cherokee Point","State":"Illinois","Password":"sXf2zzU9kabg"}]
```

Our Python script immediately found which collection was updated and pushed that change into the web UI

```
[3] 10.2s
... Collection updated: Customer
    Collection updated: Customer
```

Now we check. Our webUI



The screenshot shows the ArangoDB web interface. The top header displays the ArangoDB logo, 'COMMUNITY EDITION', and the current view: 'Collection: Customer > Document: 800031'. A left sidebar contains navigation links: DASHBOARD, COLLECTIONS, ANALYZERS, VIEWS, QUERIES, GRAPHS, SERVICES, USERS, DATABASES, REPLICATION, LOGS, and SUPPORT. The main content area shows the details of document 800031. It lists metadata: `_id: Customer/800031`, `_rev: _f1FekCG---`, and `_key: 800031`. Below this, a code editor shows the document's JSON structure:

```
1 {
2   "Customer_ID": "1",
3   "Name": "Tanuj Verma",
4   "Email": "etatem0@fotki.com",
5   "Phone": "815-918-6513",
6   "Address": "924 Cherokee Point",
7   "State": "Illinois",
8   "Password": "sXf2zzU9kabg"
9 }
```

We can see that Name is automatically updated. That's how we implemented ongoing data refresh.

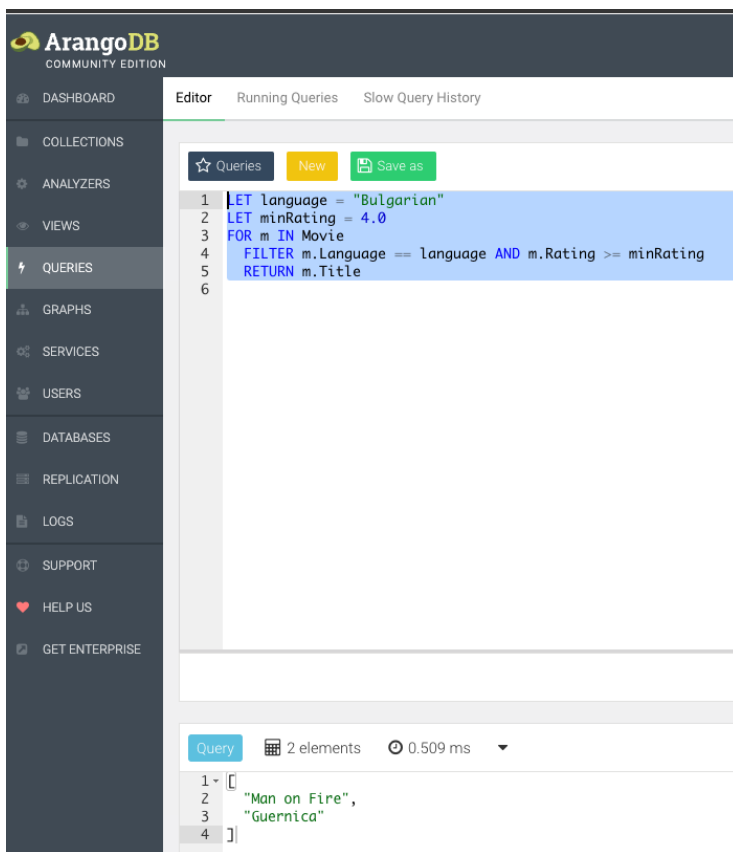
QUERIES WHICH WE EXECUTED ON OUR IMPLEMENTED DATABASE

1. Find all movies in a specific language: “Bulgarian” with a minimum rating:

Query :

```
LET language = "Bulgarian"  
LET minRating = 4.0  
FOR m IN Movie  
  FILTER m.Language == language AND m.Rating >= minRating  
RETURN m.Title
```

Output :



The screenshot displays the ArangoDB Community Edition web interface. On the left is a dark sidebar with navigation links: DASHBOARD, COLLECTIONS, ANALYZERS, VIEWS, QUERIES (highlighted with a lightning bolt icon), GRAPHS, SERVICES, USERS, DATABASES, REPLICATION, LOGS, SUPPORT, HELP US, and GET ENTERPRISE. The main area is titled 'Editor' and contains a query editor with a blue background. The query is as follows:

```
1 LET language = "Bulgarian"  
2 LET minRating = 4.0  
3 FOR m IN Movie  
4   FILTER m.Language == language AND m.Rating >= minRating  
5   RETURN m.Title  
6
```

Below the editor, the results are shown in a table format. The table has a header 'Query' and a status bar indicating '2 elements' and a execution time of '0.509 ms'. The results are:

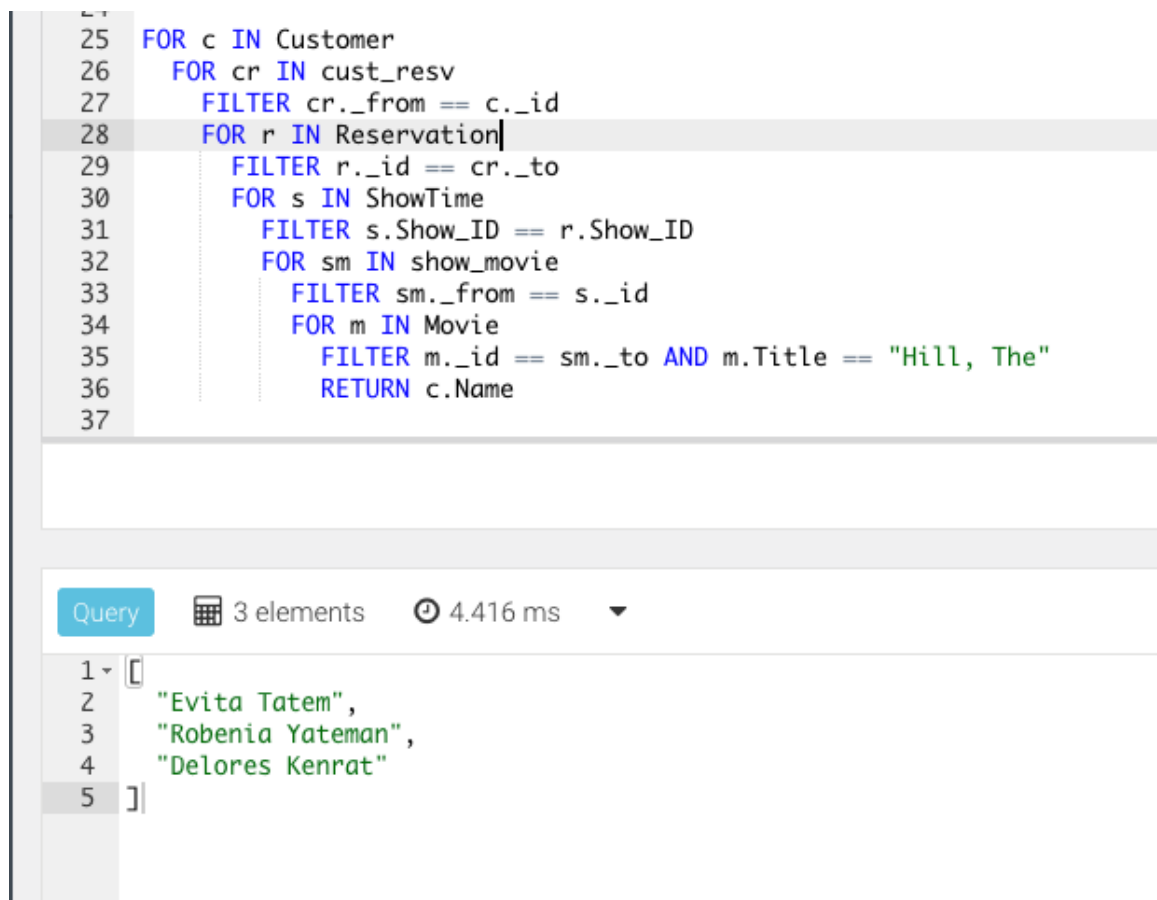
| Query |
|------------------|
| 1 - [|
| 2 "Man on Fire", |
| 3 "Guernica" |
| 4] |

2. Find customers who have made reservations for a specific movie: "Hill, The"

Query :

```
FOR c IN Customer
  FOR cr IN cust_resv
    FILTER cr._from == c._id
  FOR r IN Reservation
    FILTER r._id == cr._to
  FOR s IN ShowTime
    FILTER s.Show_ID == r.Show_ID
  FOR sm IN show_movie
    FILTER sm._from == s._id
  FOR m IN Movie
    FILTER m._id == sm._to AND m.Title == "Hill, The"
  RETURN c.Name
```

Output :



The screenshot shows a database query editor with a query written in a SQL-like syntax. The query is designed to find customers who have made reservations for a specific movie, "Hill, The". The query is as follows:

```
25 FOR c IN Customer
26   FOR cr IN cust_resv
27     FILTER cr._from == c._id
28   FOR r IN Reservation
29     FILTER r._id == cr._to
30   FOR s IN ShowTime
31     FILTER s.Show_ID == r.Show_ID
32   FOR sm IN show_movie
33     FILTER sm._from == s._id
34   FOR m IN Movie
35     FILTER m._id == sm._to AND m.Title == "Hill, The"
36   RETURN c.Name
37
```

Below the query editor, the results of the query are displayed. The results show three customer names: "Evita Tatem", "Robenia Yateman", and "Delores Kenrat". The results are presented in a table format with a column index on the left and the customer names in the right column.

| 1 | 2 |
|---|--------------------|
| 1 | "Evita Tatem", |
| 2 | "Robenia Yateman", |
| 3 | "Delores Kenrat" |
| 4 | |
| 5 | |

3.Show count of reservations made by each customer :

Query :

```
FOR c IN Customer
  LET reservationCount = (
    FOR cr IN cust_resv
      FILTER cr._from == c._id
    FOR r IN Reservation
      FILTER r._id == cr._to
    RETURN 1
  )
RETURN {"CustomerName": c.Name, "Reservations": SUM(reservationCount)}
```

Output :

| | |
|--------------------------------------|--|
| ArangoDB COMMUNITY EDITION | |
| DASHBOARD | |
| COLLECTIONS | |
| ANALYZERS | |
| VIEWS | |
| QUERIES | |
| GRAPHS | |
| SERVICES | |
| USERS | |
| DATABASES | |
| REPLICATION | |
| LOGS | |
| SUPPORT | |
| HELP US | |
| GET ENTERPRISE | |

```
18
19
20 FOR c IN Customer
21   LET reservationCount = (
22     FOR cr IN cust_resv
23       FILTER cr._from == c._id
24     FOR r IN Reservation
25       FILTER r._id == cr._to
26     RETURN 1
27   )
28   RETURN {"CustomerName": c.Name, "Reservations": SUM(reservationCount)}
29
```

Query 12 elements 1.073 ms

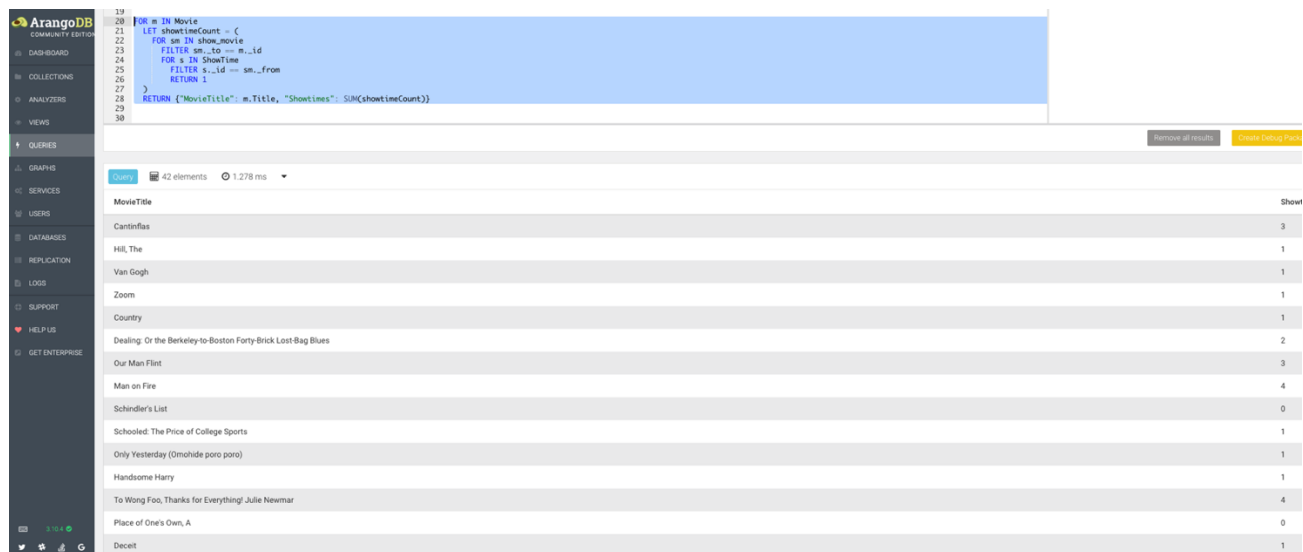
| CustomerName | Reservations |
|-----------------|--------------|
| Evita Tatem | 4 |
| Donnie Turnor | 2 |
| Tore Bennit | 2 |
| Dame Bracey | 1 |
| Hussein Danford | 1 |
| Harrietta Pardy | 0 |
| Haydon Kenford | 2 |
| Robenia Yateman | 1 |
| Delores Kenrat | 3 |
| Catlaina Duthy | 0 |
| Melly Oles | 1 |
| Edy Raspel | 0 |

4. show count of showtimes for all the movies

Query :

```
FOR m IN Movie
LET showtimeCount = (
  FOR sm IN show_movie
  FILTER sm._to == m._id
  FOR s IN ShowTime
  FILTER s._id == sm._from
  RETURN 1
)RETURN {"MovieTitle": m.Title, "Showtimes": SUM(showtimeCount)}
```

Output :



The screenshot shows the ArangoDB Compass interface. The left sidebar contains navigation options: DASHBOARD, COLLECTIONS, ANALYZERS, VIEWS, QUERIES, GRAPHS, SERVICES, USERS, DATABASES, REPLICATION, LOGS, SUPPORT, HELP US, and GET ENTERPRISE. The main area displays a query in a light blue editor. Below the editor, a status bar indicates '42 elements' and '1.278 ms'. The results are shown in a table with two columns: 'MovieTitle' and 'Showtimes'.

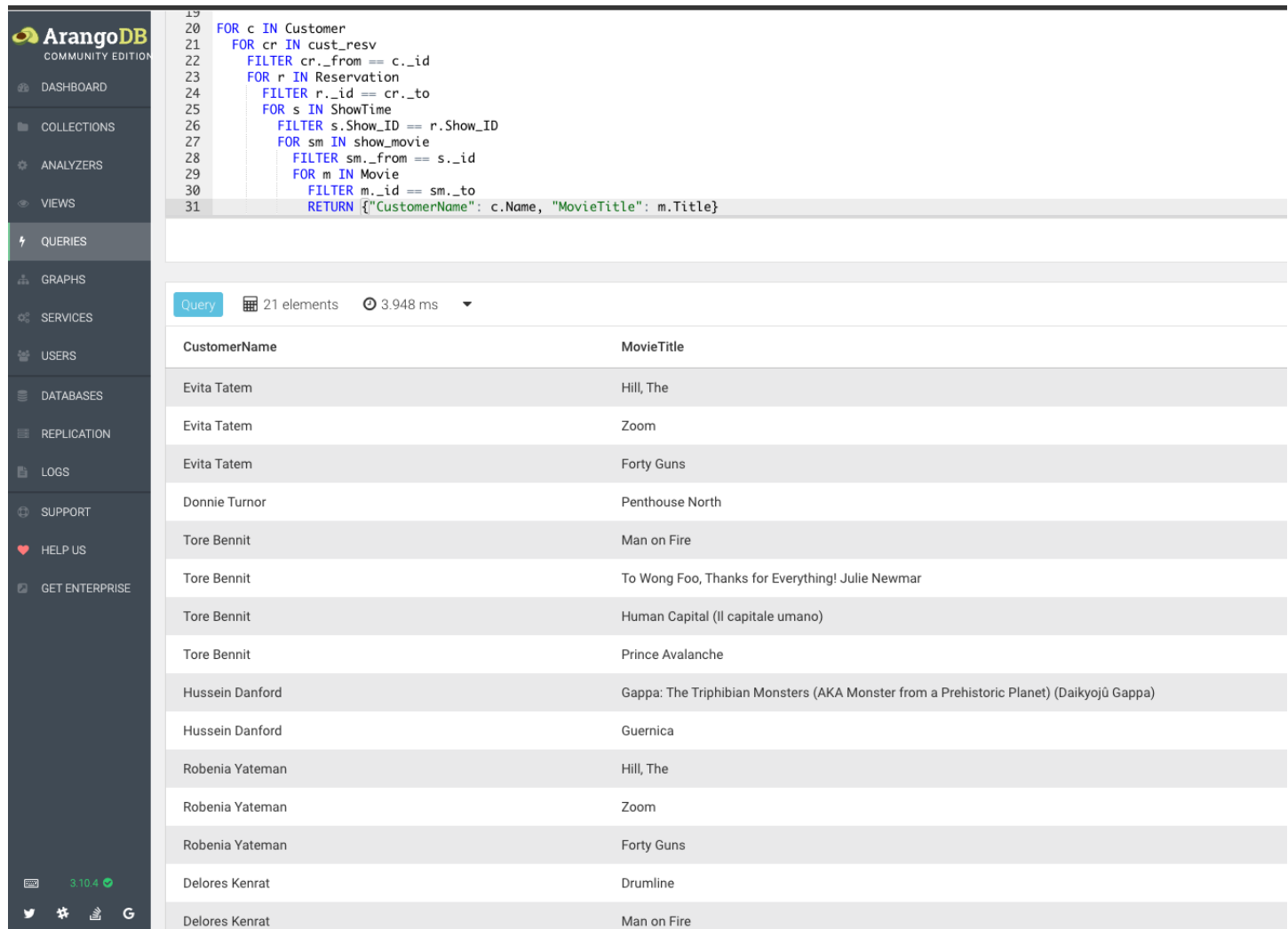
| MovieTitle | Showtimes |
|---|-----------|
| Cantinflas | 3 |
| Hill, The | 1 |
| Van Gogh | 1 |
| Zoom | 1 |
| Country | 1 |
| Dealing: Or the Berkeley-to-Boston Forty-Brick Lost-Bag Blues | 2 |
| Our Man Flint | 3 |
| Man on Fire | 4 |
| Schindler's List | 0 |
| Schooled: The Price of College Sports | 1 |
| Only Yesterday (Omohide poro poro) | 1 |
| Handsome Harry | 1 |
| To Wong Foo, Thanks for Everything! Julie Newmar | 4 |
| Place of One's Own, A | 0 |
| Deceit | 1 |

5. show all the customers who have booked movies

Query :

```
FOR c IN Customer
  FOR cr IN cust_resv
    FILTER cr._from == c._id
  FOR r IN Reservation
    FILTER r._id == cr._to
  FOR s IN ShowTime
    FILTER s.Show_ID == r.Show_ID
  FOR sm IN show_movie
    FILTER sm._from == s._id
  FOR m IN Movie
    FILTER m._id == sm._to
  RETURN {"CustomerName": c.Name, "MovieTitle": m.Title}
```

Output:



The screenshot shows the ArangoDB Community Edition interface. On the left is a sidebar with navigation links: DASHBOARD, COLLECTIONS, ANALYZERS, VIEWS, QUERIES (highlighted), GRAPHS, SERVICES, USERS, DATABASES, REPLICATION, LOGS, SUPPORT, HELP US, and GET ENTERPRISE. The main area displays a query in the editor, which is the same query as shown in the previous block. Below the editor, the results are shown in a table format. The table has two columns: CustomerName and MovieTitle. There are 21 rows of data, representing 21 elements. The execution time is 3.948 ms.

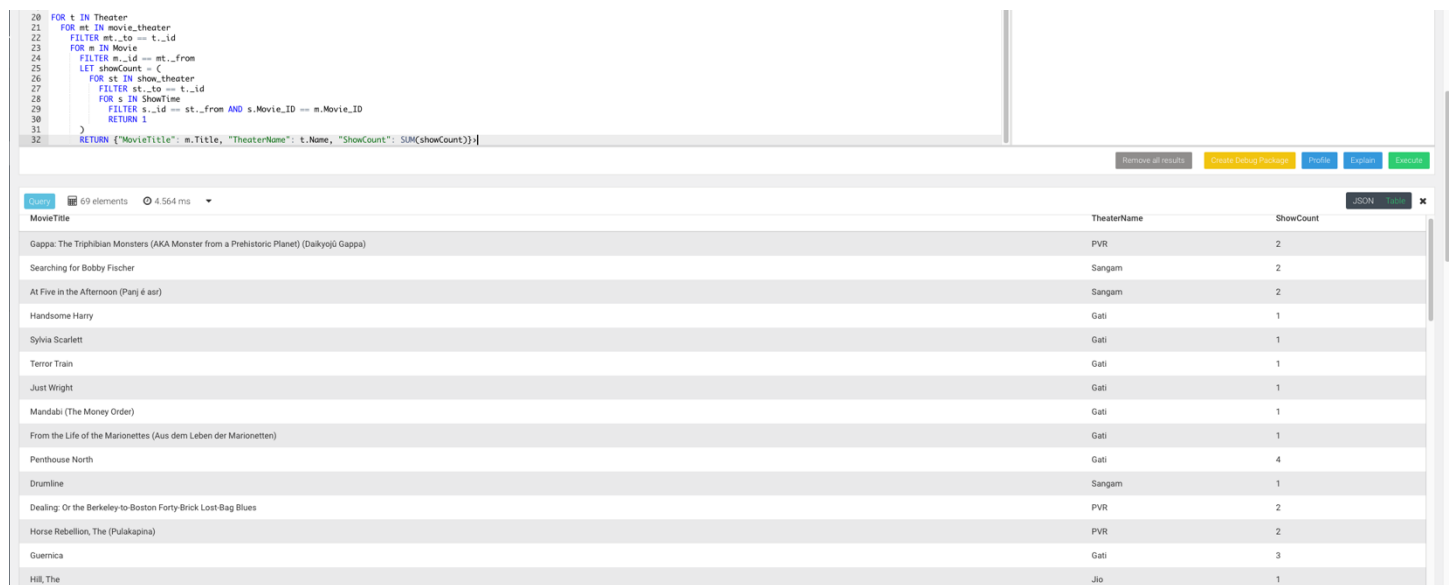
| CustomerName | MovieTitle |
|-----------------|---|
| Evita Tatem | Hill, The |
| Evita Tatem | Zoom |
| Evita Tatem | Forty Guns |
| Donnie Turnor | Penthouse North |
| Tore Bennit | Man on Fire |
| Tore Bennit | To Wong Foo, Thanks for Everything! Julie Newmar |
| Tore Bennit | Human Capital (Il capitale umano) |
| Tore Bennit | Prince Avalanche |
| Hussein Danford | Gappa: The Triphibian Monsters (AKA Monster from a Prehistoric Planet) (Daikyojû Gappa) |
| Hussein Danford | Guernica |
| Robenia Yateman | Hill, The |
| Robenia Yateman | Zoom |
| Robenia Yateman | Forty Guns |
| Delores Kenrat | Drumline |
| Delores Kenrat | Man on Fire |

6. show all the movies running in a theater with the movie name, theater name, and the count of shows in that theater

Query :

```
FOR t IN Theater
  FOR mt IN movie_theater
    FILTER mt._to == t._id
    FOR m IN Movie
      FILTER m._id == mt._from
      LET showCount = (
        FOR st IN show_theater
          FILTER st._to == t._id
          FOR s IN ShowTime
            FILTER s._id == st._from AND s.Movie_ID == m.Movie_ID
          RETURN 1
        )
      RETURN {"MovieTitle": m.Title, "TheaterName": t.Name, "ShowCount": SUM(showCount)}>
```

Output :



The screenshot shows a database query interface. At the top, a query is entered in a text area. Below the query, there are buttons for 'Remove all results', 'Create Catalog Package', 'Profile', 'Explain', and 'Execute'. The results are displayed in a table with columns 'MovieTitle', 'TheaterName', and 'ShowCount'. The table contains 16 rows of data. The interface also shows '69 elements' and '4.564 ms' for the query execution.

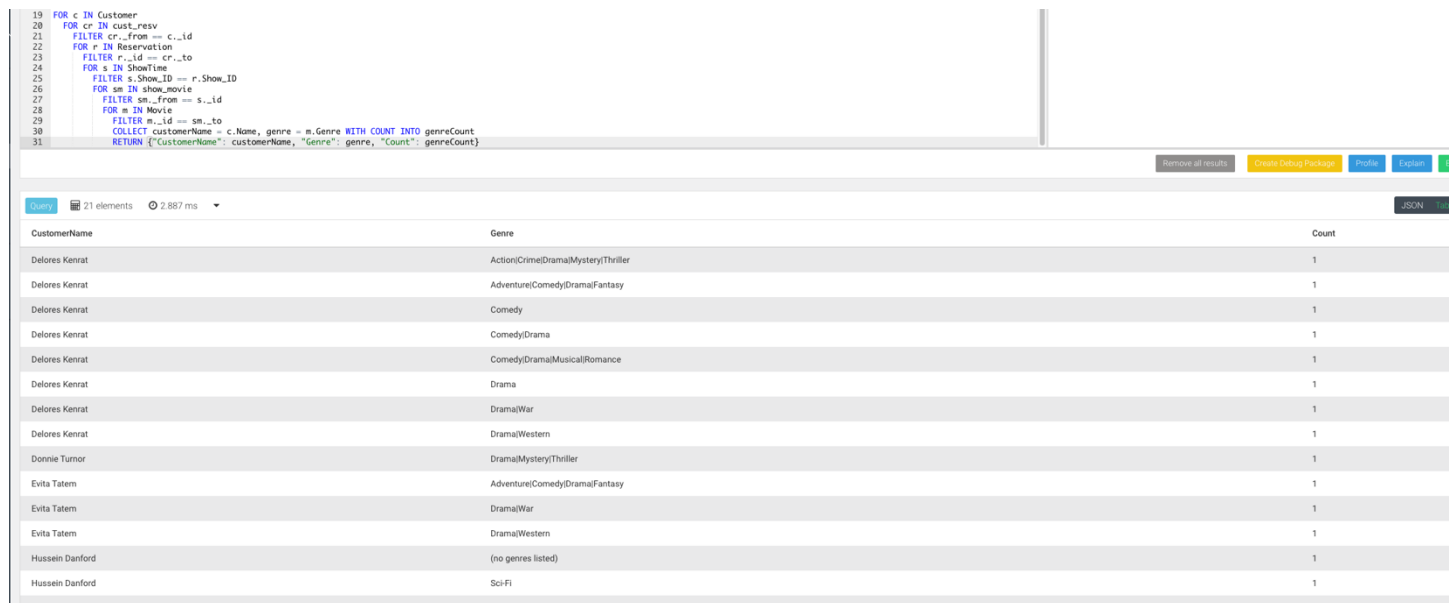
| MovieTitle | TheaterName | ShowCount |
|--|-------------|-----------|
| Gappa: The Triphibian Monsters (AKA Monster from a Prehistoric Planet) (Dakyojû Gappa) | PVR | 2 |
| Searching for Bobby Fischer | Sangam | 2 |
| At Five in the Afternoon (Parî é asr) | Sangam | 2 |
| Handsome Harry | Gati | 1 |
| Sylvia Scarlett | Gati | 1 |
| Terror Train | Gati | 1 |
| Just Wright | Gati | 1 |
| Mandabi (The Money Order) | Gati | 1 |
| From the Life of the Marionettes (Aus dem Leben der Marionetten) | Gati | 1 |
| Penthouse North | Gati | 4 |
| Drumline | Sangam | 1 |
| Dealing: Or the Berkeley-to-Boston Forty-Brick Lost Bag Blues | PVR | 2 |
| Horse Rebellion, The (Pulakapina) | PVR | 2 |
| Guernica | Gati | 3 |
| Hill, The | Jio | 1 |

7. show the customer name, watched genres, and count of times they watched those genres

Query :

```
FOR c IN Customer
  FOR cr IN cust_resv
    FILTER cr._from == c._id
  FOR r IN Reservation
    FILTER r._id == cr._to
  FOR s IN ShowTime
    FILTER s.Show_ID == r.Show_ID
  FOR sm IN show_movie
    FILTER sm._from == s._id
  FOR m IN Movie
    FILTER m._id == sm._to
  COLLECT customerName = c.Name, genre = m.Genre WITH COUNT INTO genreCount
  RETURN {"CustomerName": customerName, "Genre": genre, "Count": genreCount}
```

Output:



The screenshot shows a query execution interface. At the top, a query is entered in a text area. Below the query, there are buttons for "Remove all results", "Create Catalog Package", "Profile", and "Explain". The results are displayed in a table with the following columns: CustomerName, Genre, and Count. The table contains 14 rows of data, showing the customer name, the genres they watched, and the count of times they watched those genres.

| CustomerName | Genre | Count |
|-----------------|-------------------------------------|-------|
| Delores Kenrat | Action Crime Drama Mystery Thriller | 1 |
| Delores Kenrat | Adventure Comedy Drama Fantasy | 1 |
| Delores Kenrat | Comedy | 1 |
| Delores Kenrat | Comedy Drama | 1 |
| Delores Kenrat | Comedy Drama Musical Romance | 1 |
| Delores Kenrat | Drama | 1 |
| Delores Kenrat | Drama War | 1 |
| Delores Kenrat | Drama Western | 1 |
| Donnie Tumor | Drama Mystery Thriller | 1 |
| Evita Tatern | Adventure Comedy Drama Fantasy | 1 |
| Evita Tatern | Drama War | 1 |
| Evita Tatern | Drama Western | 1 |
| Hussein Danford | (no genres listed) | 1 |
| Hussein Danford | Sci-Fi | 1 |

8. show all the movies watched by the customer whose duration is more that 1 hour

Query :

```
FOR c IN Customer
  FOR cr IN cust_resv
    FILTER cr._from == c._id
  FOR r IN Reservation
    FILTER r._id == cr._to
  FOR s IN ShowTime
    FILTER s.Show_ID == r.Show_ID
  FOR sm IN show_movie
    FILTER sm._from == s._id
  FOR m IN Movie
    FILTER m._id == sm._to AND m.Duration_hr > 1
  RETURN {"CustomerName": c.Name, "WatchedMovie": m.Title, "Duration_hr": m.Duration_hr}
```

Output :

19 FOR c IN Customer

20 FOR cr IN cust_resv

21 FILTER cr._from == c._id

22 FOR r IN Reservation

23 FILTER r._id == cr._to

24 FOR s IN ShowTime

25 FILTER s.Show_ID == r.Show_ID

26 FOR sm IN show_movie

27 FILTER sm._from == s._id

28 FOR m IN Movie

29 FILTER m._id == sm._to AND m.Duration_hr > 1

30 RETURN {"CustomerName": c.Name, "WatchedMovie": m.Title, "Duration_hr": m.Duration_hr}

Remove all results

Create Debug Package

Profile

Log

21 elements

3.904 ms

| CustomerName | WatchedMovie | Duration_hr |
|-----------------|--|-------------|
| Evita Tatern | Hill, The | 2.67 |
| Evita Tatern | Zoom | 2.93 |
| Evita Tatern | Forty Guns | 2.84 |
| Donnie Turnor | Penthouse North | 2.97 |
| Tore Bennit | Man on Fire | 2.8 |
| Tore Bennit | To Wong Foo, Thanks for Everything! Julie Newmar | 1.44 |
| Tore Bennit | Human Capital (Il capitale umano) | 2.8 |
| Tore Bennit | Prince Avalanche | 1.27 |
| Hussein Danford | Gappa: The Trighbian Monsters (AKA Monster from a Prehistoric Planet) (Daikyoji Gappa) | 2.48 |
| Hussein Danford | Guernica | 2.01 |
| Robenia Yateman | Hill, The | 2.67 |
| Robenia Yateman | Zoom | 2.93 |
| Robenia Yateman | Forty Guns | 2.84 |
| Delores Kenrat | Drumline | 1.83 |
| Delores Kenrat | Man on Fire | 2.8 |

9.Show movies running in all theaters

Query :

```
FOR m IN Movie
  COLLECT movieID = m.Movie_ID INTO moviesInAnyTheater = m
  LET theaters = (
    FOR mt IN movie_theater
      FILTER mt._from == moviesInAnyTheater[0]._id
    FOR t IN Theater
      FILTER t._id == mt._to
    RETURN t.Name
  )
  FILTER LENGTH(theaters) > 0
RETURN {"MovieTitle": moviesInAnyTheater[0].Title, "Theaters": theaters}
```

Output :

18

19

20

21

22

23

24

25

26

27

28

29

30

```
FOR m IN Movie
COLLECT movieID = m.Movie_ID INTO moviesInAnyTheater = m
LET theaters = (
  FOR mt IN movie_theater
    FILTER mt._from == moviesInAnyTheater[0]._id
  FOR t IN Theater
    FILTER t._id == mt._to
  RETURN t.Name
)
FILTER LENGTH(theaters) > 0
RETURN {"MovieTitle": moviesInAnyTheater[0].Title, "Theaters": theaters}
```

Remove all results

Query

13 elements

4.358 ms

| MovieTitle | Theaters |
|---|--|
| Hill, The | ['Jio'] |
| Country | ['Santosh'] |
| Van Gogh | ['Bahar'] |
| Searching for Bobby Fischer | ['Sangam','Gati','Gati','Jio','Jio','Santosh','PVR'] |
| Handsome Harry | ['Gati','Jio'] |
| First Position | ['AMC'] |
| Guernica | ['Gati','AMC'] |
| Penthouse North | ['Gati','Jio','Bahar'] |
| Schindler's List | ['Sangam','Jio','Gati'] |
| Drumline | ['Sangam','Santosh','Jio'] |
| Gappa: The Triphibian Monsters (AKA Monster from a Prehistoric Planet) (Daikyoji Gappa) | ['PVR','Jio','Bahar'] |
| Chicago Massacre: Richard Speck | ['PVR'] |
| Dealing: Or the Berkeley-to-Boston Forty-Brick Lost-Bag Blues | ['PVR','Jio','Sangam','PVR'] |

10. Show movies not running in any theater

Query :

```
FOR m IN Movie
  FILTER m.Movie_ID NOT IN (
    FOR mt IN movie_theater
      RETURN mt._from
  )
RETURN {"Title": m.Title, "Genre": m.Genre}
```

Output :

25 FOR m IN Movie

26 FILTER m.Movie_ID NOT IN (

27 FOR mt IN movie_theater

28 RETURN mt._from

29)

30 RETURN {"Title": m.Title, "Genre": m.Genre}

31

32

Remove all results

Create Debug Pack

Query

42 elements

3.668 ms

| Title | Genre |
|---|-------------------------------------|
| Cantinflas | Drama |
| Hill, The | Drama War |
| Van Gogh | Drama |
| Zoom | Adventure Comedy Drama Fantasy |
| Country | Drama |
| Dealing: Or the Berkeley-to-Boston Forty-Brick Lost-Bag Blues | Comedy Drama Thriller |
| Our Man Flint | Adventure Comedy Sci-Fi |
| Man on Fire | Action Crime Drama Mystery Thriller |
| Schindler's List | Drama War |
| Schooled: The Price of College Sports | Documentary |
| Only Yesterday (Omohide poro poro) | Animation Drama |
| Handsome Harry | Crime Drama |
| To Wong Foo, Thanks for Everthino! Julie Newmar | Comedy |