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)

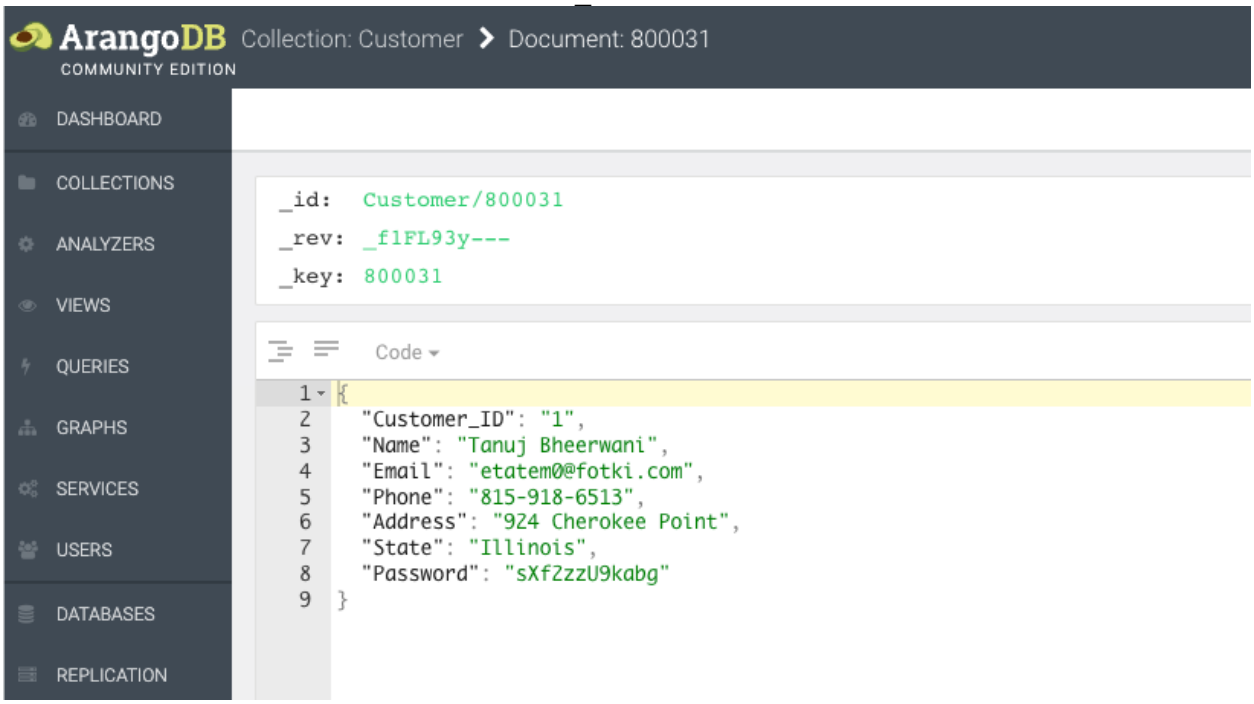
```
import json
import time
from arango import ArangoClient
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler


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
```

```python
    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 :



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 > ...


```
[{"_key":"800031",
"_id":"Customer/800031",
"_rev":"_f1E45iu---",
"Customer_ID":"1",
"Name":"Tanuj Bheerwani",
"Email":"etatem0@fotki.com",
"Phone":"815-918-6513",
"Address":"924 Cherokee Point",
"State":"Illinois",
"Password":"sXf2zzU9kabg"},

{"_key":"800033","_id":"Customer/800033","_rev":"_f1E45jG---","Customer_ID":"3","Name":"Don
```


We change it to “Tanuj Verma” and save it

Untitled-1.ipynb ● {} Customer.json ●

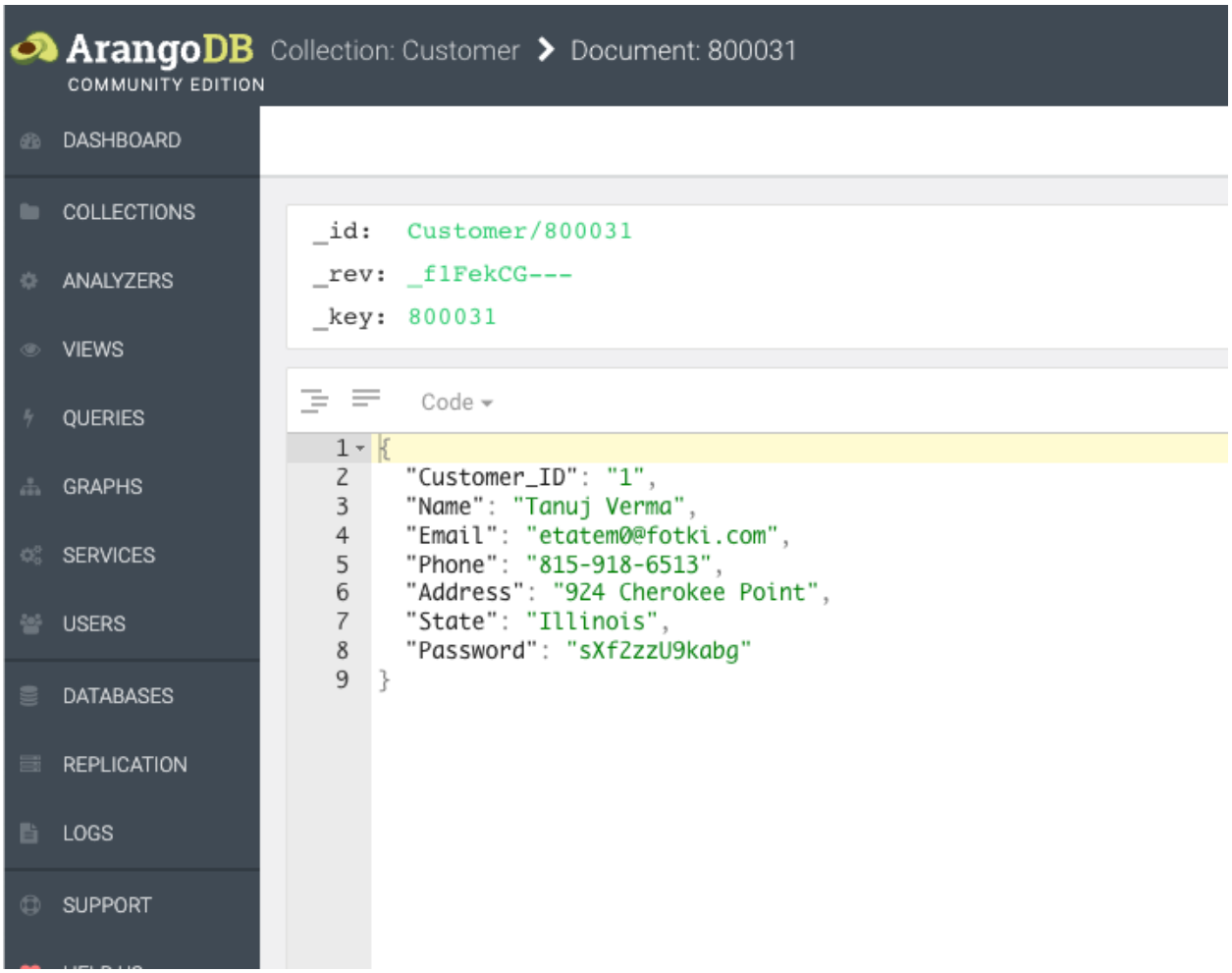
Users > tanujverma > Desktop > NEU > ADBMS > ArangoDB > {} Customer.json > {} 0 > Name


```
[{"_key":"800031",
"_id":"Customer/800031",
"_rev":"_f1E45iu---",
"Customer_ID":"1",
"Name":"Tanuj Verma",
"Email":"etatem0@fotki.com",
"Phone":"815-918-6513",
"Address":"924 Cherokee Point",
"State":"Illinois",
"Password":"sXf2zzU9kabg"},

{"_key":"800033","_id":"Customer/800033","_rev":"_f1E45jG---","Customer_ID":"3","Name":"Donnie Turnor","Email":"dturn
```

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



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

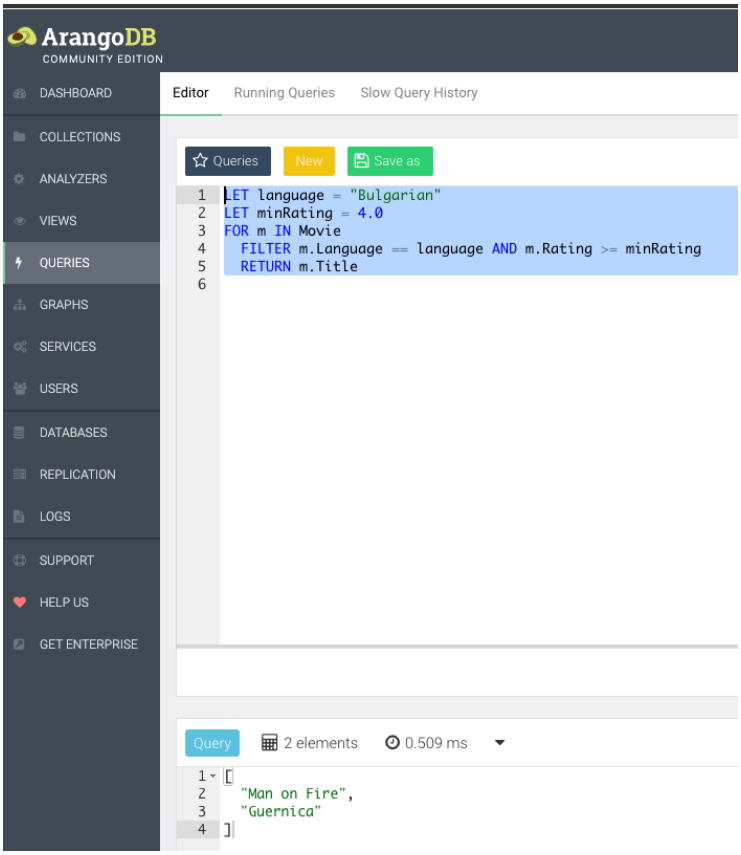
QUERIES WHICH WE EXECUTED ON OUR IMPLEMENTED DATABASE

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 :



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 :

```
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
```

Query 3 elements 4.416 ms

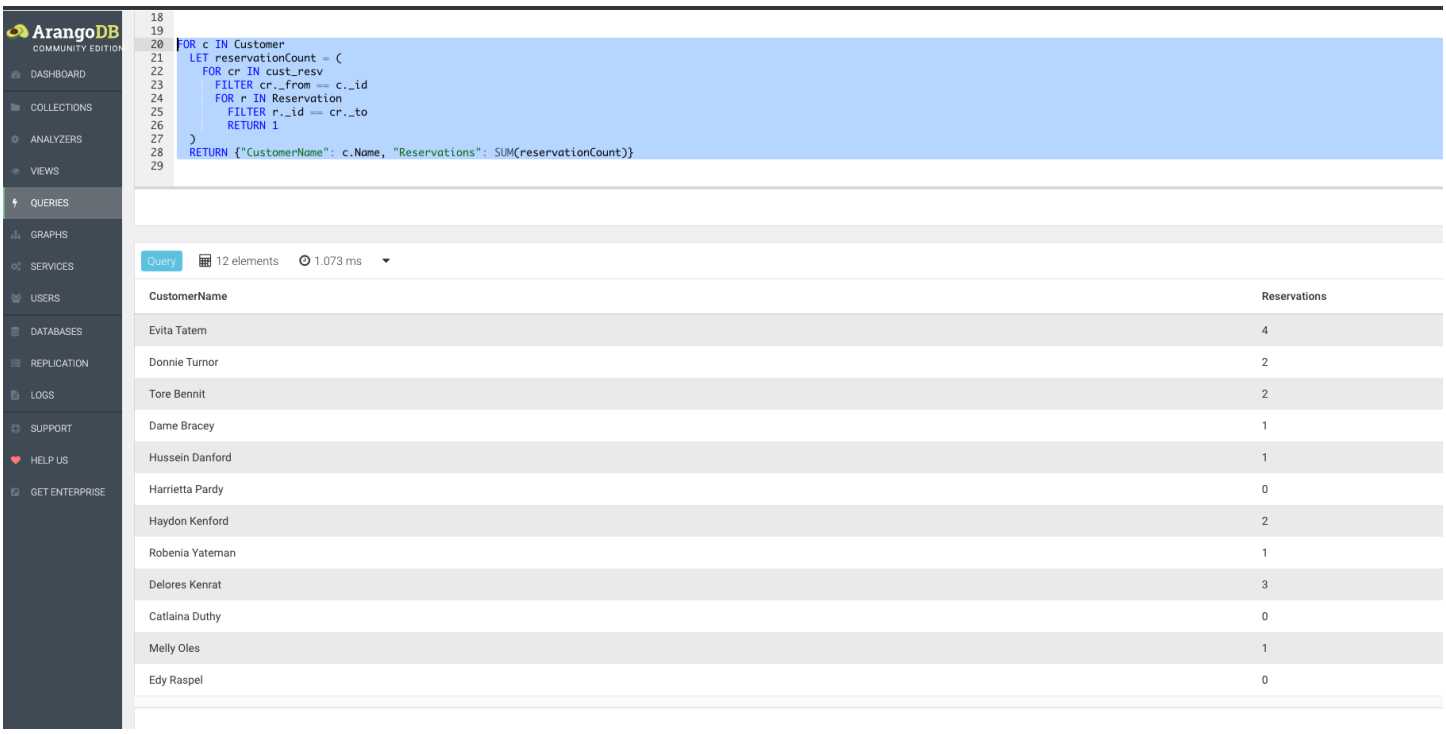
```
[
  "Evita Tatem",
  "Robenia Yateman",
  "Delores Kenrat"
]
```

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 :



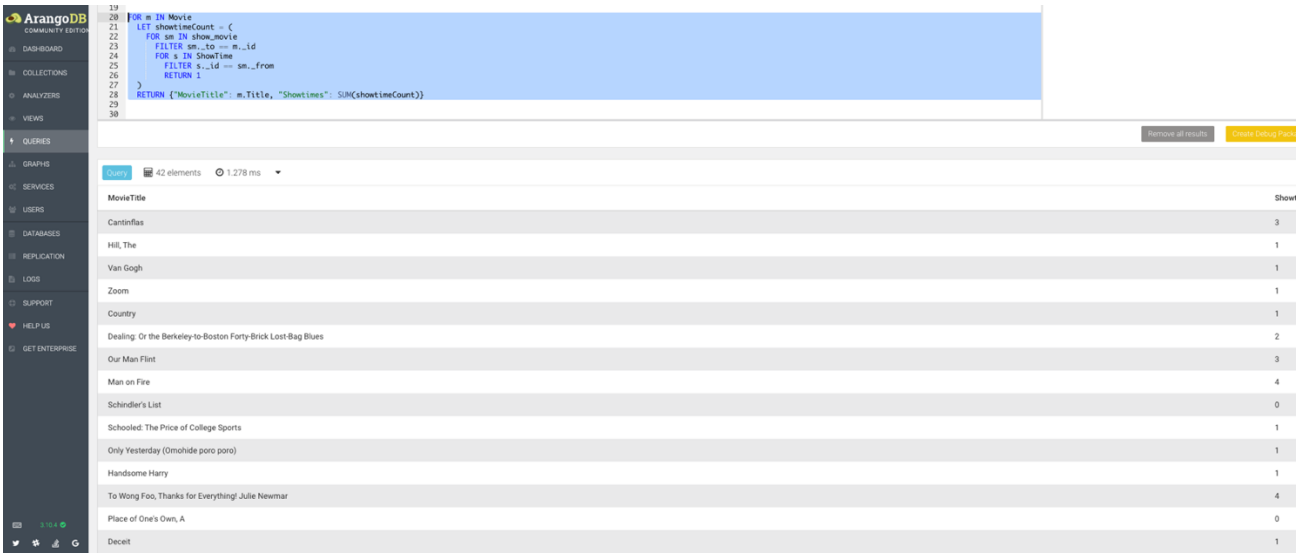
CustomerName	Reservations
Evita Tatem	4
Donnie Turnor	2
Tore Bennit	2
Dame Bracey	1
Hussein Danford	1
Harrietta Pardy	0
Haydon Kenford	2
Robenia Yateman	1
Delores Kenrat	3
Catlaina Duthy	0
Melly Oles	1
Edy Raspel	0

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 :

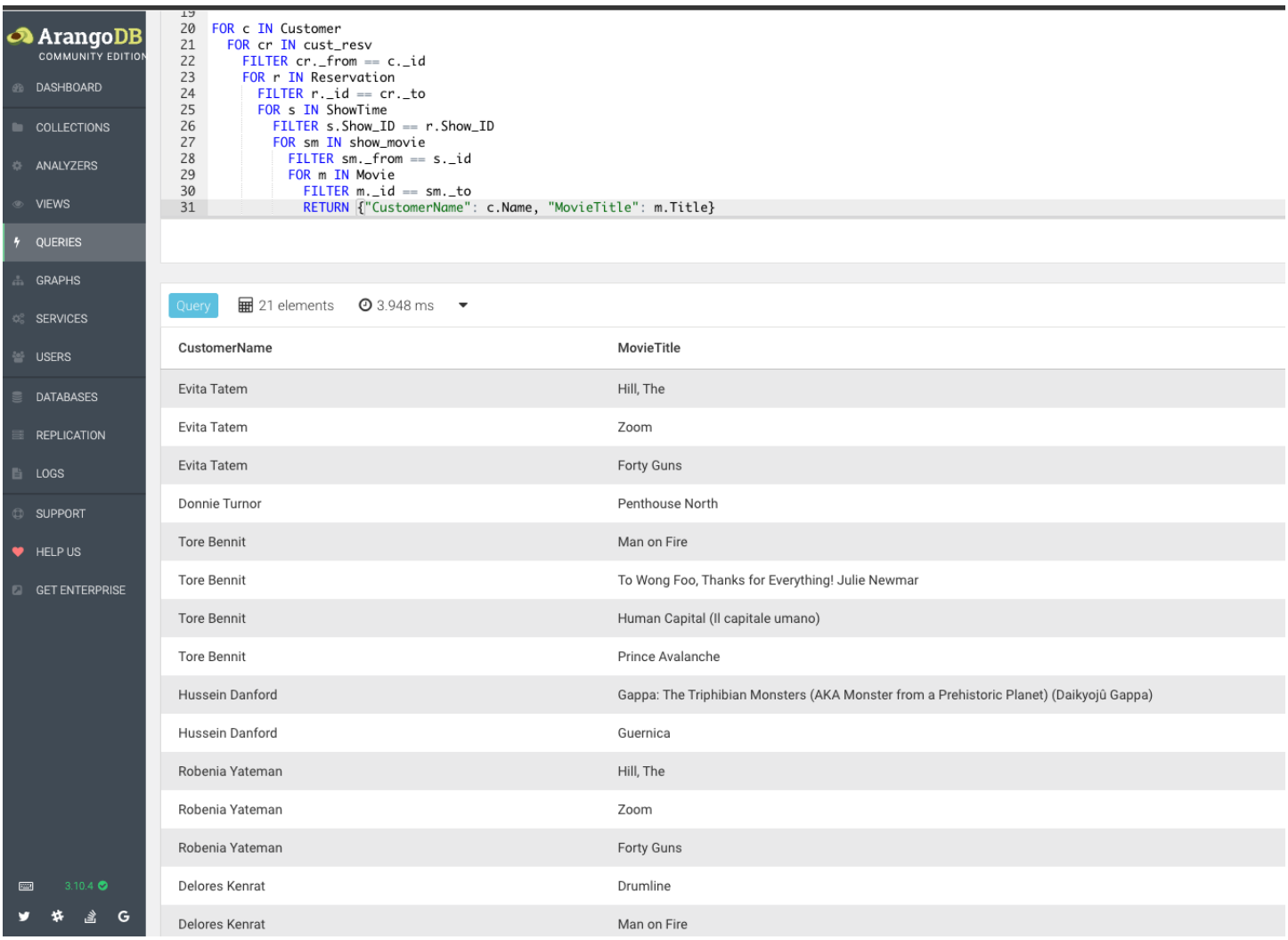


5. show all the customers who have booked movies

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:



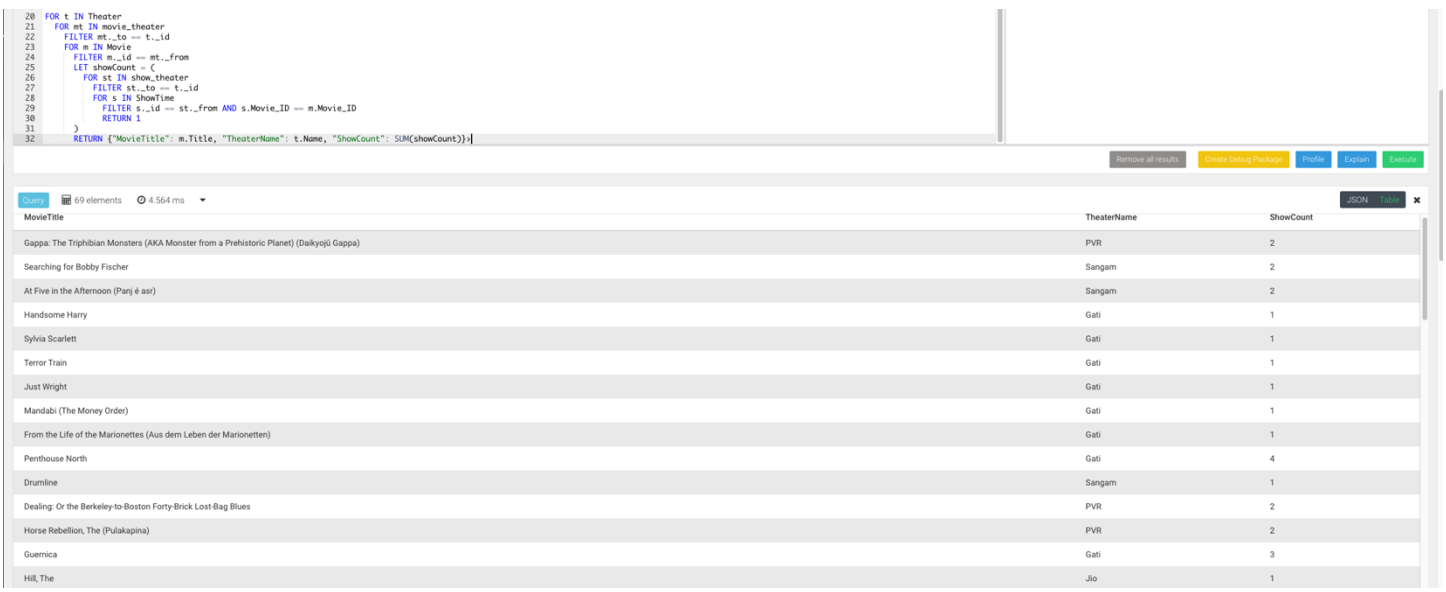
CustomerName	MovieTitle
Evita Tatem	Hill, The
Evita Tatem	Zoom
Evita Tatem	Forty Guns
Donnie Turnor	Penthouse North
Tore Bennit	Man on Fire
Tore Bennit	To Wong Foo, Thanks for Everything! Julie Newmar
Tore Bennit	Human Capital (Il capitale umano)
Tore Bennit	Prince Avalanche
Hussein Danford	Gappa: The 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 :



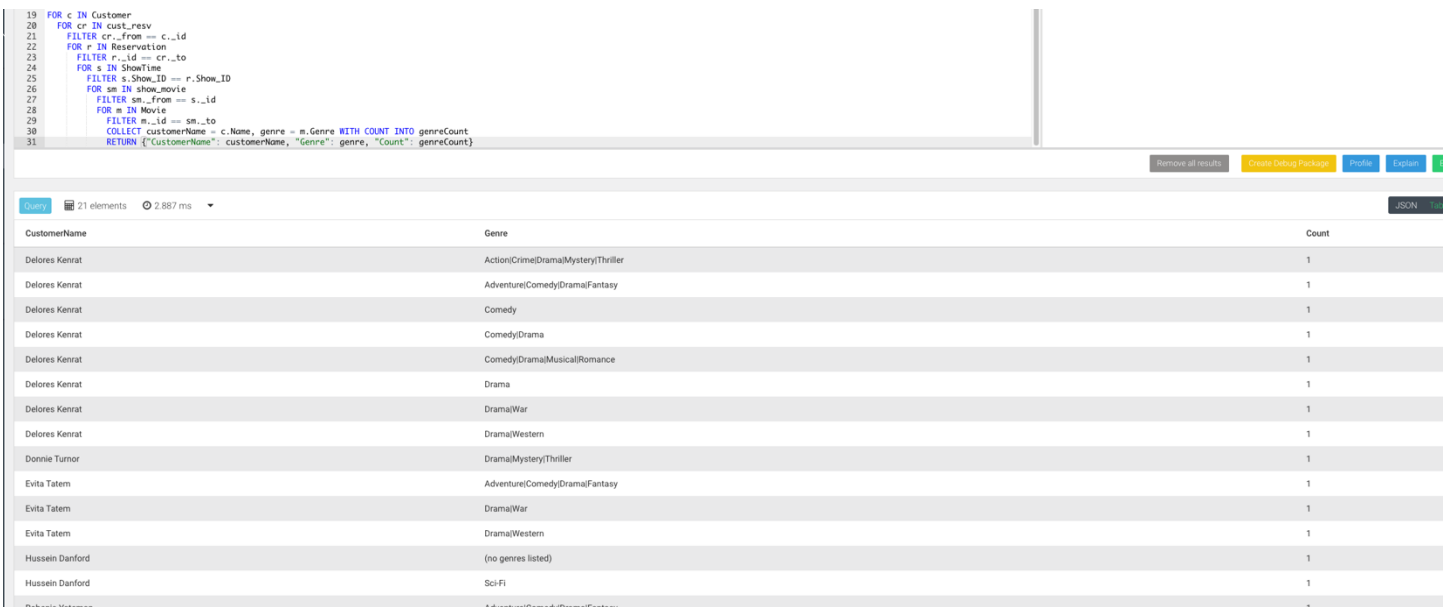
MovieTitle	TheaterName	ShowCount
Gappa: The Triphibian Monsters (AKA Monster from a Prehistoric Planet) (Daikyojū Gappa)	PVR	2
Searching for Bobby Fischer	Sangam	2
At Five in the Afternoon (Panj é asr)	Sangam	2
Handsome Harry	Gati	1
Sylvia Scarlett	Gati	1
Terror Train	Gati	1
Just Wright	Gati	1
Mandabi (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:



Remove all results | Create Debug Package | Profile | Explain

Query | 21 elements | 2.887 ms | JSON

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 Turnor	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 :

```
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}
```

Remove all results | Create Debug Package | Profile | Exp

Query | 21 elements | 3.904 ms | JS

CustomerName	WatchedMovie	Duration_hr
Evita Tatem	Hill, The	2.67
Evita Tatem	Zoom	2.93
Evita Tatem	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 Triphibian Monsters (AKA Monster from a Prehistoric Planet) (Daikyojû 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 :

```
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) (Daikyojû 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 :

```
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}
```

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 Everything! Julie Newmar	Comedy