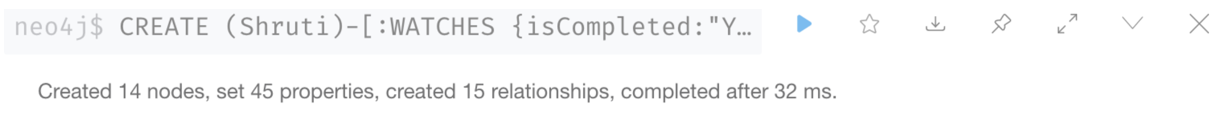
NEO4J-CASE STUDY

OTT System

Develop a graph for your own case study using Neo4j and perform CRUD operations on it.

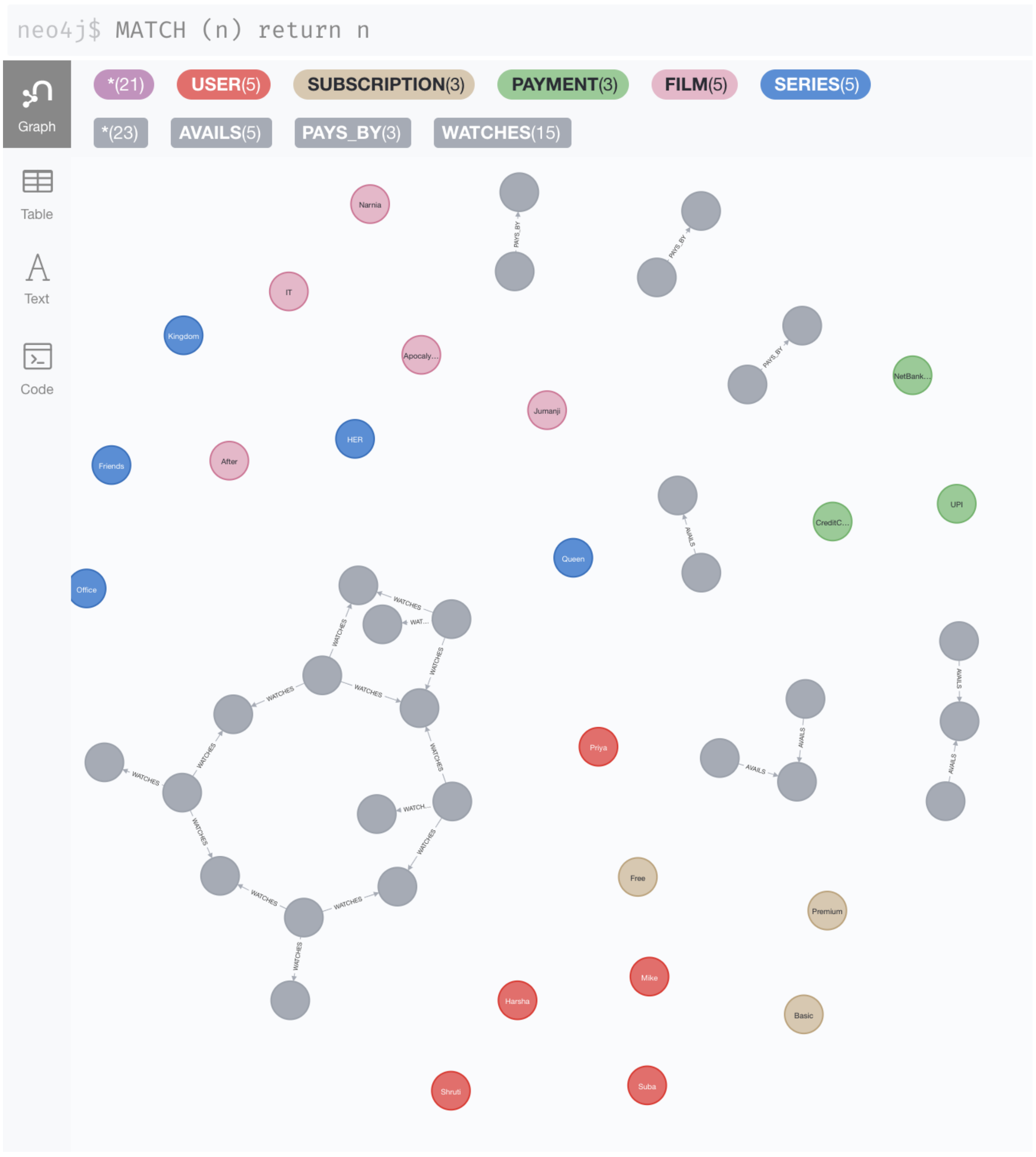
1. **Create the nodes and relationships**

```
CREATE
(Shruti:USER{name: "Shruti",age:22,Phone:"9585848203",city:"Coimbatore"
}),
(Priya:USER{name: "Priya",age:32,Phone:"9343458123",city:"Pollachi" }),
(Suba:USER{name: "Suba ",age:22,Phone:"9585841111",city:"Palakkad" }),
(Harsha:USER{name: "Harsha",age:25,Phone:"9523841111",city:"Kotagiri" }),
(Mike:USER{name: "Mike",age:41,Phone:"9115841111",city:"New Jersey" })

CREATE
(Free:SUBSCRIPTION {name: "Free", amount:0}),
(Basic:SUBSCRIPTION {name: "Basic", amount:500}),
(Premium:SUBSCRIPTION { name: "Premium", amount:1000})

 CREATE
 (Shruti)-[:AVAILS {start_date:"22/07/2019",end_date:"22/07/2020"}]->(Free),
(Harsha)-[:AVAILS {start_date:"22/09/2019",end_date:"22/09/2020"}]->(Free),
(Suba)-[:AVAILS {start_date:"23/07/2020",end_date:"23/07/2021"}]->(Basic),
(Priya)-[:AVAILS {start_date:"01/01/2021",end_date:"01/01/2022"}]->(Basic),
(Mike)-[:AVAILS {start_date:"22/03/2021",end_date:"22/03/2022"}]->(Premium)

CREATE
(UPI:PAYMENT {name: "UPI"}),
(CreditCard:PAYMENT {name: "CreditCard"}),
(NetBanking:PAYMENT { name: "NetBanking"})

CREATE
(Suba)-[:PAYS_BY {payment_date:"23/07/2019",paid_amount:500}]->(UPI),
(Priya)-[:PAYS_BY
{payment_date:"01/01/2020",paid_amount:500}]->(CreditCard),
(Mike)-[:PAYS_BY
{payment_date:"22/03/2020",paid_amount:1000}]->(NetBanking)

CREATE
(Jumanji:FILM {name:
"Jumanji",Genre:["Thriller","Fantasy"],RuntimeMinutes:120,Rating:9}),
(Narnia:FILM {name:
"Narnia",Genre:["Thriller","Fantasy"],RuntimeMinutes:100,Rating:8}),
(Apocalypse:FILM { name:
"Apocalypse",Genre:["Thriller","Horror"],RuntimeMinutes:90,Rating:8}),
(IT:FILM {name:
"IT",Genre:["Thriller","Horror"],RuntimeMinutes:120,Rating:7}),
(After:FILM { name:
"After",Genre:["Romance","Comedy"],RuntimeMinutes:100,Rating:9})

CREATE
(Kingdom:SERIES {name:
"Kingdom",Genre:["Thriller","Fantasy"],RuntimeMinutes:30,Rating:9}),
(Queen:SERIES {name:
"Queen",Genre:["Thriller","Fantasy"],RuntimeMinutes:40,Rating:8}),
(Friends:SERIES { name:
"Friends",Genre:["Comedy"],RuntimeMinutes:20,Rating:10}),
(HER:SERIES {name:
"HER",Genre:["Thriller","Horror"],RuntimeMinutes:20,Rating:8}),
(Office:SERIES { name:
"Office",Genre:["Romance","Comedy"],RuntimeMinutes:40,Rating:9})


CREATE
(Shruti)-[:WATCHES
{isCompleted:"Yes",NoOfViews:100,remainingWatchTime:0}]->(Queen),
(Shruti)-[:WATCHES
{isCompleted:"No",NoOfViews:2,remainingWatchTime:10}]->(After),
(Shruti)-[:WATCHES
{isCompleted:"No",NoOfViews:4,remainingWatchTime:20}]->(Narnia),
(Harsha)-[:WATCHES
{isCompleted:"Yes",NoOfViews:10,remainingWatchTime:0}]->(IT),
(Harsha)-[:WATCHES
{isCompleted:"Yes",NoOfViews:20,remainingWatchTime:30}]->(HER),
(Harsha)-[:WATCHES
{isCompleted:"Yes",NoOfViews:200,remainingWatchTime:23}]->(Friends),
(Suba)-[:WATCHES
{isCompleted:"No",NoOfViews:1,remainingWatchTime:21}]->(Queen),
(Suba)-[:WATCHES
{isCompleted:"No",NoOfViews:2,remainingWatchTime:10}]->(Office),
(Suba)-[:WATCHES
{isCompleted:"No",NoOfViews:4,remainingWatchTime:70}]->(Kingdom),
(Priya)-[:WATCHES
{isCompleted:"Yes",NoOfViews:10,remainingWatchTime:20}]->(IT),
(Priya)-[:WATCHES
{isCompleted:"No",NoOfViews:20,remainingWatchTime:14}]->(Kingdom),
(Priya)-[:WATCHES
{isCompleted:"Yes",NoOfViews:30,remainingWatchTime:26}]->(Apocalypse),
(Mike)-[:WATCHES
{isCompleted:"No",NoOfViews:10,remainingWatchTime:50}]->(Queen),
(Mike)-[:WATCHES
{isCompleted:"No",NoOfViews:20,remainingWatchTime:16}]->(Narnia),
(Mike)-[:WATCHES
{isCompleted:"Yes",NoOfViews:30,remainingWatchTime:20}]->(Friends)
```



```
neo4j$ CREATE (Shruti)-[:WATCHES {isCompleted:"Y…
```

```
Created 14 nodes, set 45 properties, created 15 relationships, completed after 32 ms.
```



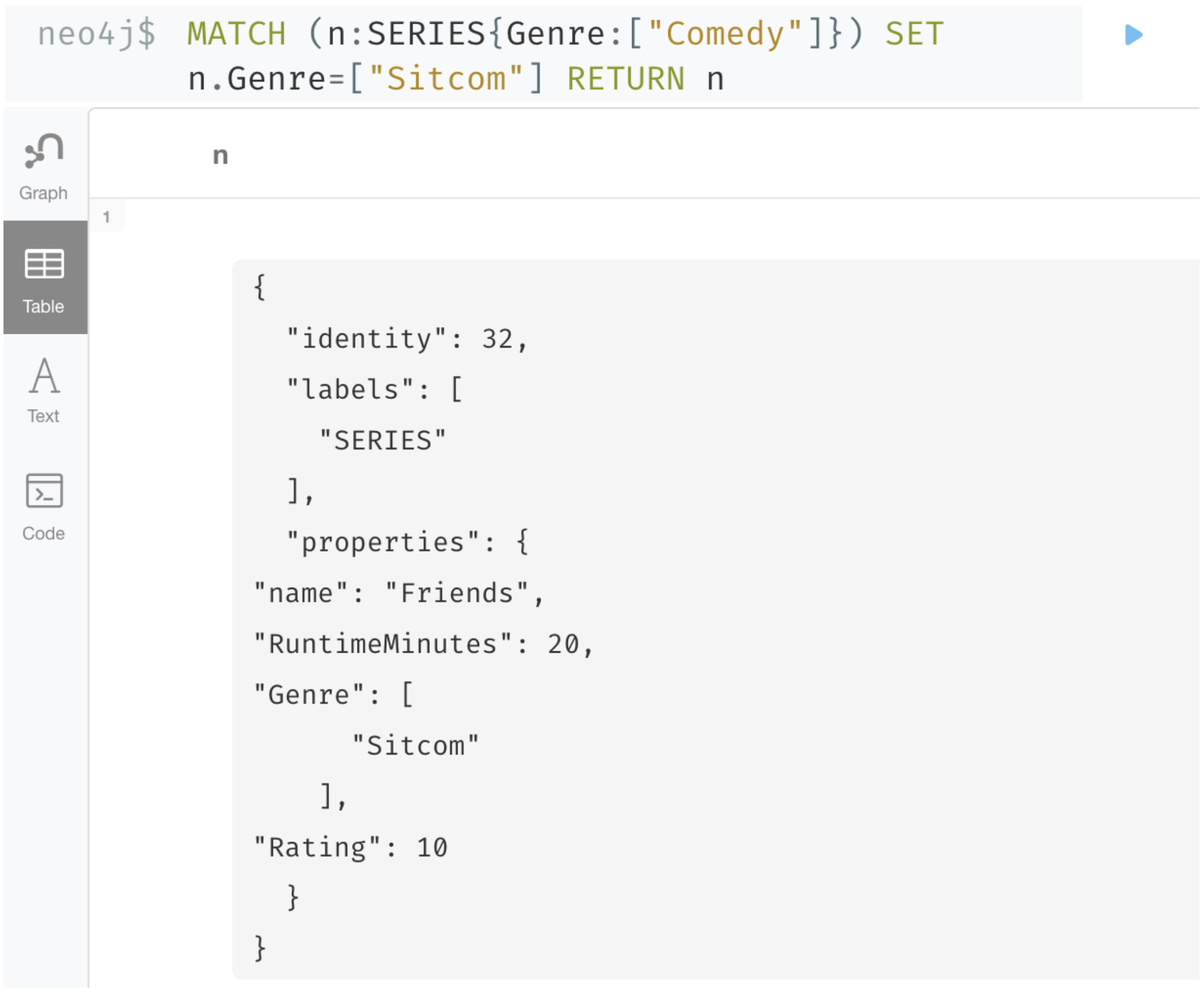
```
neo4j$ MATCH (n) return n
```

2. Practice the following commands

a. UPDATE

Updating the SERIES with the only GENRE as Comedy to Sitcom

MATCH (n:SERIES{Genre:["Comedy"]}) SET n.Genre=["Sitcom"] RETURN n



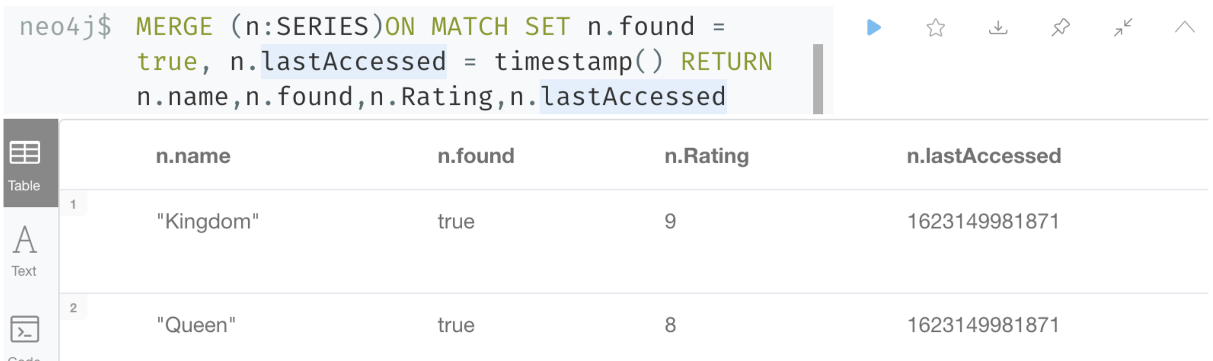
```
neo4j$ MATCH (n:SERIES{Genre:["Comedy"]}) SET
       n.Genre=["Sitcom"] RETURN n
```

```
n
1
{
  "identity": 32,
  "labels": [
    "SERIES"
  ],
  "properties": {
"name": "Friends",
"RuntimeMinutes": 20,
"Genre": [
      "Sitcom"
    ],
"Rating": 10
  }
}
```

b. MERGE

Merge with ON MATCH setting multiple properties by matching the Series and setting the timestamp value of last accessed along with found boolean set as true

MERGE (n:SERIES)ON MATCH SET n.found = true, n.lastAccessed = timestamp() RETURNn.name,n.found,n.Rating,n.lastAccessed



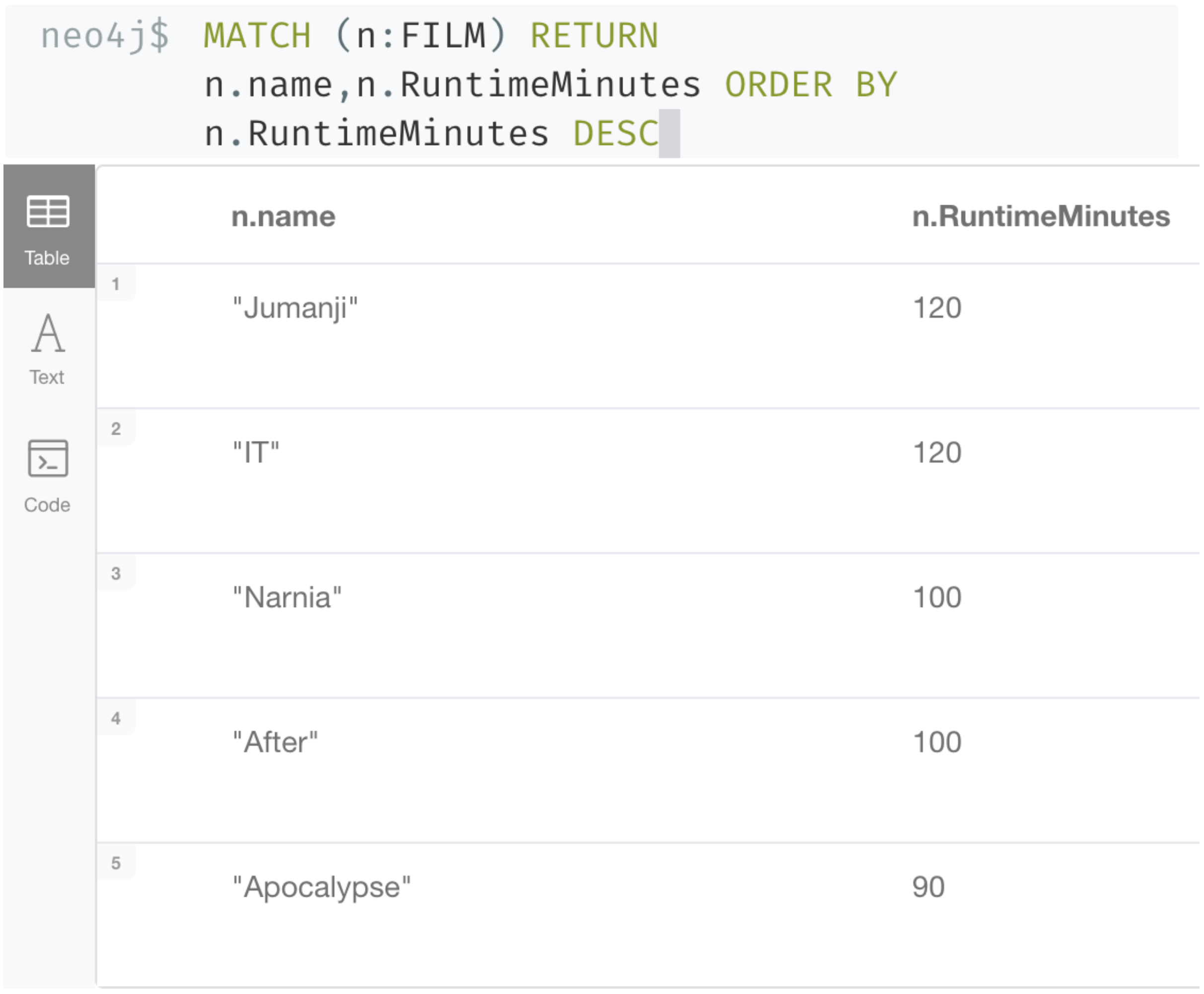
```
neo4j$ MERGE (n:SERIES)ON MATCH SET n.found =
       true, n.lastAccessed = timestamp() RETURN
       n.name,n.found,n.Rating,n.lastAccessed
```

	n.name	n.found	n.Rating	n.lastAccessed
1	"Kingdom"	true	9	1623149981871
2	"Queen"	true	8	1623149981871

c. ORDER BY

Ordering nodes by property by matching the FILMS and ordering them in the descending order based upon their Runtime in minutes .

```
MATCH (n:FILM) RETURN n.name,n.RuntimeMinutes ORDER BY n.RuntimeMinutes DESC
```

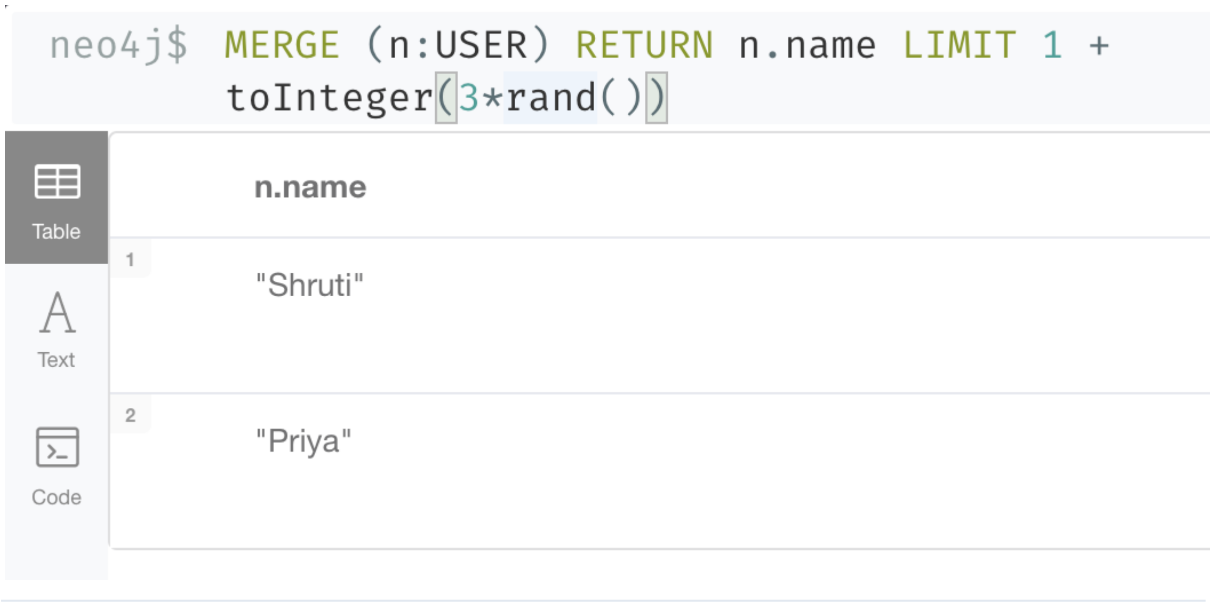


```
neo4j$ MATCH (n:FILM) RETURN
       n.name,n.RuntimeMinutes ORDER BY
       n.RuntimeMinutes DESC
```

	n.name	n.RuntimeMinutes
1	"Jumanji"	120
2	"IT"	120
3	"Narnia"	100
4	"After"	100
5	"Apocalypse"	90

d. LIMIT

Using an expression with LIMIT to return a subset of the rows . Displaying the USERS name . Limit 1 row plus randomly 0, 1, or 2. So randomly limit to 1, 2, or 3 rows.



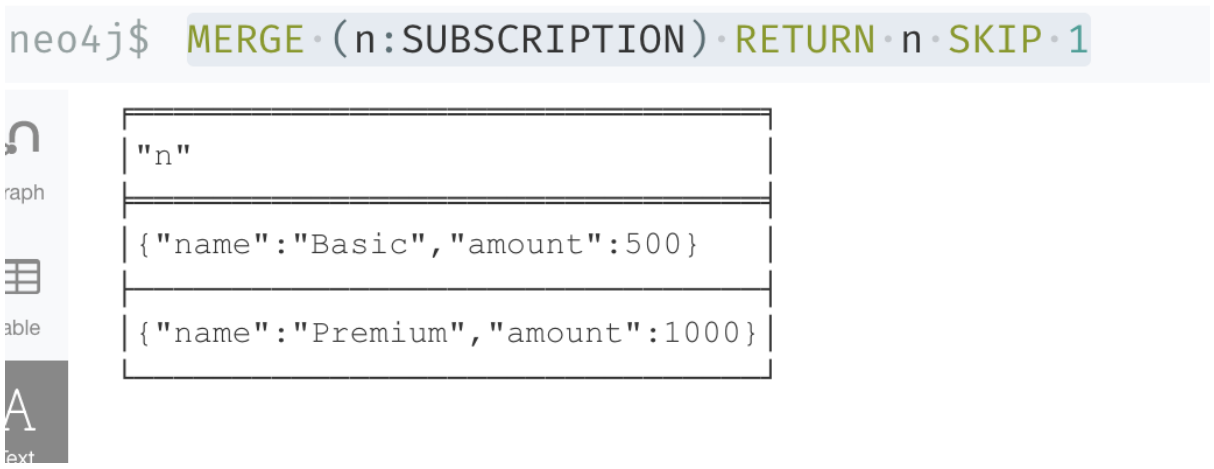
```
neo4j$ MERGE (n:USER) RETURN n.name LIMIT 1 + toInteger(3*rand())
```

```
n.name
1 "Shruti"
2 "Priya"
```

e. **SKIP**

Skipping the first row in SUBSCRIPTION and displaying the rest

MERGE (n:SUBSCRIPTION) RETURN n SKIP 1



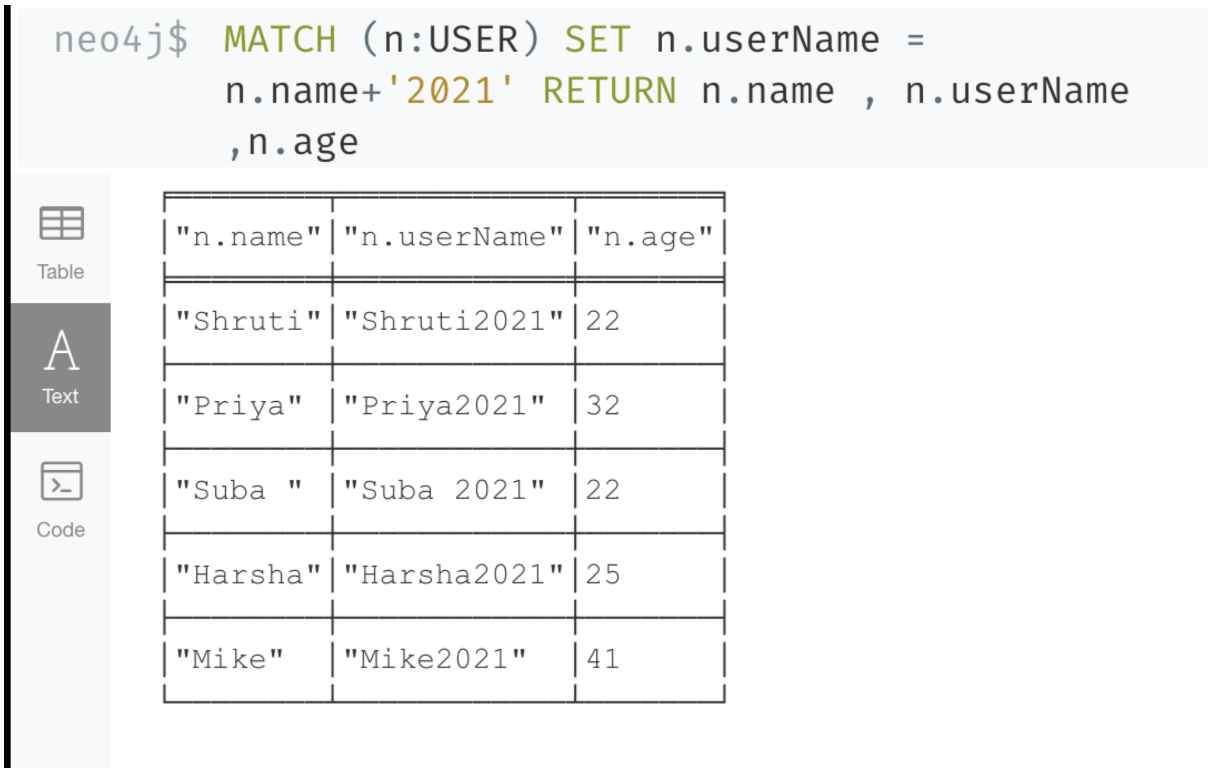
```
neo4j$ MERGE (n:SUBSCRIPTION) RETURN n SKIP 1
```

```
╒══════════════════════════════════╕
│"n"                               │
╞══════════════════════════════════╡
│{"name":"Basic","amount":500}     │
├──────────────────────────────────┤
│{"name":"Premium","amount":1000}  │
└──────────────────────────────────┘
```

f. **WITH STRING FUNCTIONS**

Using the string concatenation operator + and creating User Names by appending the year 2021 with their names .

MATCH (n:USER) SET n.userName = n.name+'2021' RETURN n.name , n.userName ,n.age

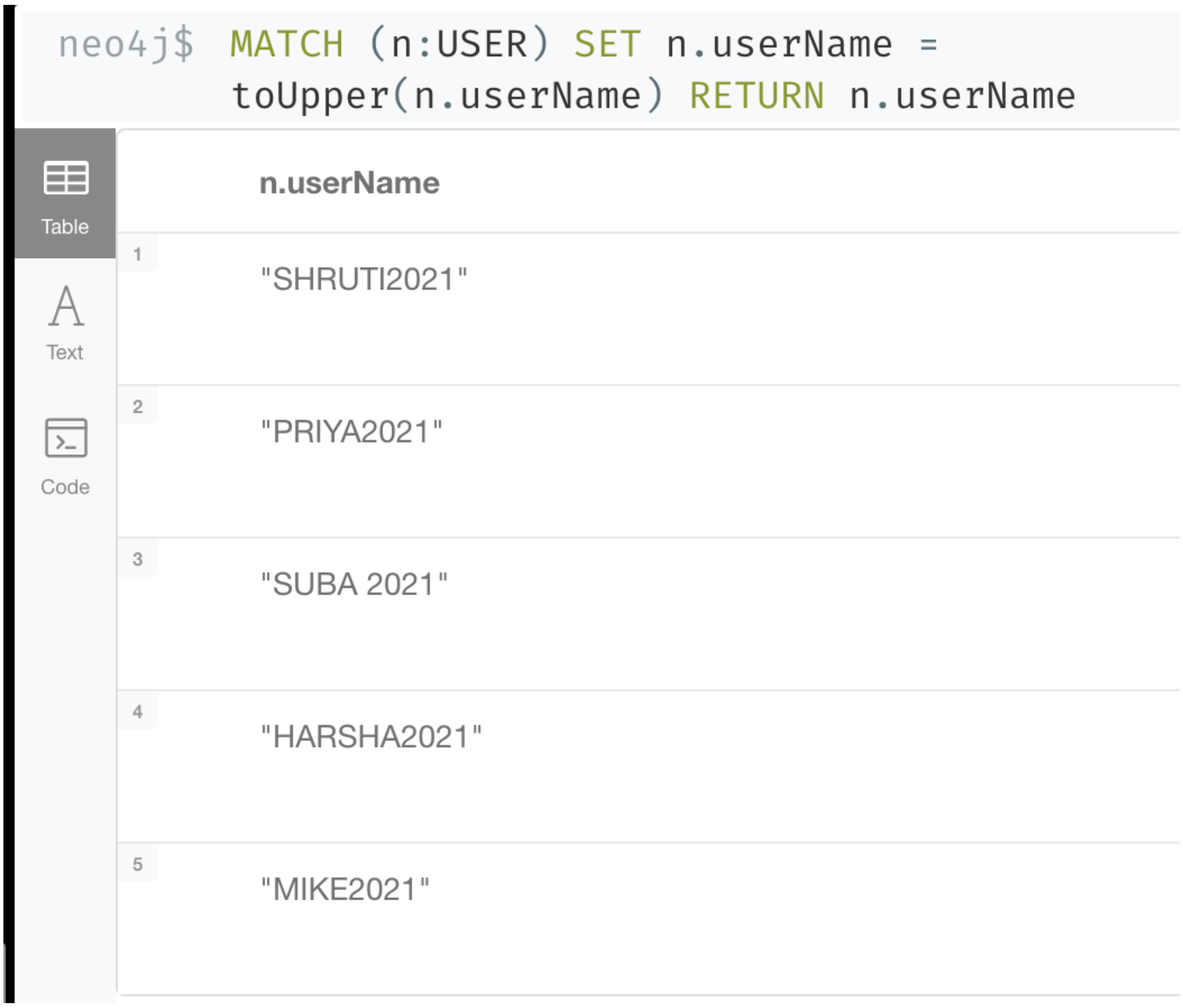


```
neo4j$ MATCH (n:USER) SET n.userName =
       n.name+'2021' RETURN n.name , n.userName
       ,n.age
```

```
╒══════════╤══════════════╤═══════╕
│"n.name"  │"n.userName"  │"n.age"│
╞══════════╪══════════════╪═══════╡
│"Shruti"  │"Shruti2021"  │22     │
├──────────┼──────────────┼───────┤
│"Priya"   │"Priya2021"   │32     │
├──────────┼──────────────┼───────┤
│"Suba "   │"Suba 2021"   │22     │
├──────────┼──────────────┼───────┤
│"Harsha"  │"Harsha2021"  │25     │
├──────────┼──────────────┼───────┤
│"Mike"    │"Mike2021"    │41     │
└──────────┴──────────────┴───────┘
```

Using the toUpper() function , the string is converted to uppercase .

```
MATCH (n:USER) SET n.userName = toUpper(n.userName) RETURN
n.userName
```



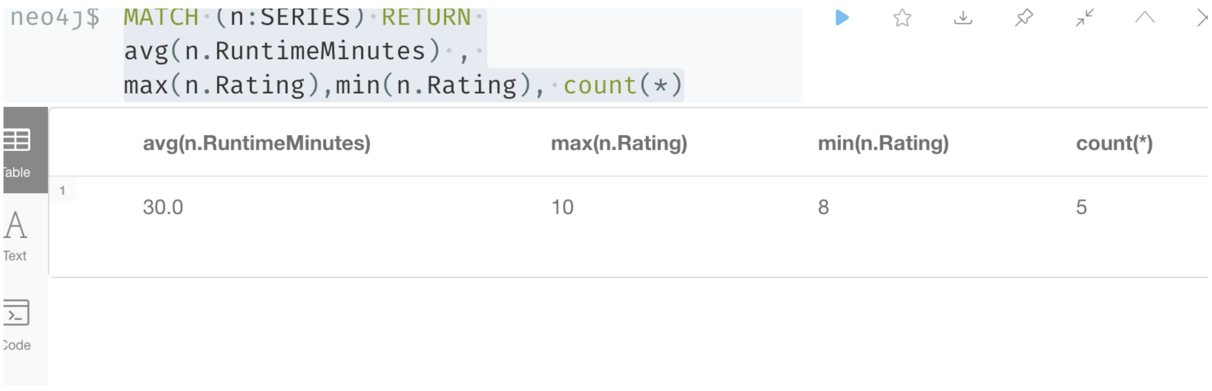
```
neo4j$ MATCH (n:USER) SET n.userName =
       toUpper(n.userName) RETURN n.userName
```

	n.userName
1	"SHRUTI2021"
2	"PRIYA2021"
3	"SUBA 2021"
4	"HARSHA2021"
5	"MIKE2021"

g. **AGGREGATE FUNCTIONS**

Calculating the average runtime , maximum rating , minimum rating and count of the series

```
MATCH (n:SERIES) RETURN avg(n.RuntimeMinutes) ,
max(n.Rating),min(n.Rating), count(*)
```



```
neo4j$ MATCH (n:SERIES) RETURN
avg(n.RuntimeMinutes) ,
max(n.Rating),min(n.Rating), count(*)
```

	avg(n.RuntimeMinutes)	max(n.Rating)	min(n.Rating)	count(*)
1	30.0	10	8	5