

NoSQL Morning Assignment

Queries:

Query for Q1:

```
Select * from couchbasedemo where MovieName="Name";
```

Query for Q2:

```
Select * from couchbasedemo where "CityName" in city;
```

Query for Q3:

Insertion
From
Couchbase:-

```
insert into couchbasedemo(key,value)
values("couchbasedemo",{
    "movieName": "Your_Movie_Name",
    "city": ["City 1", "City2", "City3"],
    "date": "Release data",
    "id": "Movie ID",
    "genres": ["Genre1", "Genre2", "Genre3"]
});
```

Postman
Raw-Text:

```
{
    "movieName": "Your Movie Name",
    "city": ["City 1", "City2", "City3"],
    "date": "Release date",
    "id": "MovieID",
    "genres": ["Genre1", "Genre2", "Genre3"]
}
```

Query for Q4:

Deletion via ID:

```
delete from couchbasedemo where id="MovieID";
```

Deletion via Name:

```
delete from couchbasedemo where MovieName="Name";
```

Deletion via City:

```
delete from couchbasedemo where "CityName" in city;
```

Deletion via Genres:

```
delete from couchbasedemo where "GenreName" in Genres;
```

SCREENSHOTS:

The screenshot shows a 'Query Editor' interface with the following details:

- Query Editor** tab is active.
- Query text: `1 select * from couchbasedemo where MovieName="Upload";`
- Status bar: `< history (47/47) >`
- Execution status: `Execute` button is blue, `Explain` button is white. `External Query Advisor` shows success, just now, elapsed: 2ms, execution: 2ms, docs: 1, size: 415 bytes.
- Format: `format` button.
- Query Results** tab is active, showing a JSON document structure.
- JSON document content (lines 1-18):

```
1 - [
2 - {
3 -   "couchbasedemo": {
4 -     "MovieName": "Upload",
5 -     "_class": "com.example.au.couchbasedemo.model.Blogs",
6 -     "city": [
7 -       "Delhi/NCR",
8 -       "Kanpur",
9 -       "Lucknow",
10 -      "Kolkata"
11 -    ],
12 -    "date": 1609977600000,
13 -    "genres": [
14 -      "Sci-Fi",
15 -      "Fiction"
16 -    ]
17 -  }
18 - }
```
- Result format options: Table, JSON (selected), Tree, Plan, Plan Text.

Figure 1 Screenshot for Question 1

```
1 select * from couchbasedemo where "Ranchi" in city;
```

Execute Explain External Query Advisor ✓ success 27 min ago | elapsed: 10ms | execution: 10ms | docs: 2 | size: 954 bytes format

Query Results ✓ 🔍

Table JSON Tree Plan Plan Text

```
2 ✘ {
3 ✘   "couchbasedemo": {
4 ✘     "MovieName": "Two Broke Girls",
5 ✘     "_class": "com.example.au.couchbasedemo.model.Blogs",
6 ✘     "city": [
7 ✘       "Delhi/NCR",
8 ✘       "Kanpur",
9 ✘       "Lucknow",
10      "Kolkata",
11      "Bhubaneswar",
12      "Ranchi",
13      "Mumbai"
14    ],
15    "date": 1615075200000,
16    "genres": [
17      "Comedy"
18    ]
19  }
20 }
```

Figure 2 Screenshot for Question 2

Query Editor < history (48/48) >

```
1 insert into couchbasedemo(key,value)
2 values("couchbasedemo",{"movieName": "Upload",
3 "city": ["Jaipur","Chennai","Kerela"],
4 "date": "2021-10-07",
5 "id": "Movies1",
6 "genres": ["Comedy","Fiction"]});
```

Execute Explain External Query Advisor ✓ success just now | elapsed: 997.3µs | execution: 997.3µs | mutations: 1 format

Query Results ✓ 🔍

Table JSON Tree Plan Plan Text

```
1 ✘ {
2 ✘   "results": []
3 }
```

Figure 3 Screenshot for Question 3

The screenshot shows the Apache CouchDB Query Editor interface. In the top left, it says "Query Editor". In the top right, there's a "history (45/48)" link. Below the editor area, there are two buttons: "Execute" (which is blue) and "Explain". To the right of these buttons, it says "External Query Advisor" with a green checkmark, "success", "21 min ago", "elapsed: 4ms", "execution: 4ms", and "mutations: 1". On the far right, there's a "format" button. Below the editor area, there are tabs for "Query Results" (with icons for CSV and JSON), "Table", "JSON" (which is selected and highlighted in blue), "Tree", "Plan", and "Plan Text". The main content area shows the following JSON response:

```
1 ✓ {
2   "results": []
3 }
```

Figure 4 Screenshot for Question 4

Question 2

Inserting Records:

Postman Script to enter records:

```
{
  "id": "Player2",
  "name": "Lionel Messi",
  "goals": 618
}
```

Insertion from couchbase:

```
insert into couchbasedemo(key,value)
values("couchbasedemo",{
  "id": "Player3",
  "name": "Pele",
  "goals": 643
});
```

Table used for insertion:

The screenshot shows a table from the FIFA.com News Centre. The table has columns for Player, Club, Goals, Appearances, and Ratio. The data includes Pele, Lionel Messi, Gerd Muller, Fernando Poyet, Josef Bican, Jimmy McGrory, Jimmy Jones, Uwe Seeler, Eusebio, and Cristiano Ronaldo.

Player	Club	Goals	Appearances	Ratio
Pele	Santos	643	659	0.98
Lionel Messi	Barcelona	618	705	0.88
Gerd Muller	Bayern Munich	564	605	0.93
Fernando Poyet	Sporting	544	334	1.63
Josef Bican	Slavia Prague	534	274	1.95
Jimmy McGrory	Celtic	522	501	1.04
Jimmy Jones	Glenavon	517	419	1.23
Uwe Seeler	Hamburg	507	587	0.86
Eusebio	Benfica	473	440	1.08
Cristiano Ronaldo	Real Madrid	450	438	1.03

Figure 5 Table containing data inserted to couchbase

Fetching Data:

The screenshot shows the Apache CouchDB Query Editor. A query is run: `1 select * from couchbasedemo;`. The results are displayed in JSON format, showing 10 documents. Each document has a '_id' field and a '_value' field. The '_value' field contains a document with fields: '_class': 'com.example.au.couchbasedemo.model.Blogs', 'goals': 450, and 'name': 'Cristiano Ronaldo'. Another document has '_value' with '_class': 'com.example.au.couchbasedemo.model.Blogs', 'goals': 473, and 'name': 'Eusebio'.

```
1 select * from couchbasedemo;
2 [
3     {
4         "couchbasedemo": {
5             "_class": "com.example.au.couchbasedemo.model.Blogs",
6             "goals": 450,
7             "name": "Cristiano Ronaldo"
8         }
9     },
10    {
11        "couchbasedemo": {
12            "_class": "com.example.au.couchbasedemo.model.Blogs",
13            "goals": 473,
14            "name": "Eusebio"
15        }
16    },
17    {
18        "couchbasedemo": {
19            "_class": "com.example.au.couchbasedemo.model.Blogs",
20            "goals": 473,
21            "name": "Eusebio"
22        }
23    }
24 ]
```

Figure 6 Screenshot showing all records fetched

Fetching A Particular Player's Goals:

The screenshot shows the Apache CouchDB Futon interface. In the 'Query Editor' tab, a query is entered: `select goals from couchbasedemo where name="Eusebio";`. Below the editor, the status bar indicates success with a green checkmark, execution time of 22ms, and a single document returned. The 'Query Results' section displays the JSON response:

```
1 ↴ [  
2 ↴   {  
3 ↴     "goals": 473  
4 ↴   }  
5 ↴ ]
```

 The results are also shown in Table, JSON, Tree, Plan, and Plan Text formats.

Figure 7 Query to fetch number of goals by a player

Query:

```
select goals from couchbasedemo where name="PlayerName";
```

Players having more than 10 Goals:

Query:

```
select name from couchbasedemo where goals>10;
```

Creating Primary Index:

The screenshot shows the Apache CouchDB Futon interface. In the 'Query Editor' tab, a query is entered: `create primary index 'id' on 'couchbasedemo' using GSI
2 with {"defer_build":true};`. Below the editor, the status bar indicates success with a green checkmark, execution time of 400ms, and a single document returned. The 'Query Results' section is empty.

Figure 8 Query to create primary index on "ID"