

2023

# SQL and Extensions

DEEPANKAR RIJAL

6 OCTOBER 2023

## Executive Summary

A tourism track recommender system is the main topic of the project. The goal is to develop a system that uses information from social media to track potential tourists and suggest nearby tourism walks. To do this, I have used a number of database systems and extensions, including PostgreSQL, PostGIS spatial, Neo4j graph, and MongoDB document databases. These databases will keep track of details about individuals, their social networks, locations, track logs, reviews, and friendship ties.

Four database components make up the project:

**Postgres Database:** Using tables for users, their connections, and location information, it simulates social networking data. It is necessary to retrieve the coordinates and social relationships of specific persons.

**PostGIS Spatial Database:** This technique uses PostGIS to manage spatial data, including tracklogs, in the PostgreSQL database. When there is no exact match, spatial searches can locate the closest tracklog or identify tracklogs that are located within a certain range of coordinates.

**MongoDB Document Database:** Managing reviews of tourist walks as JSON documents. This involves adding reviews and retrieving the one with the best rating.

**Neo4j Graph Database:** Making a graph database in Neo4j to represent individuals and their friendship connections. This component includes retrieving all related persons for a specific person as well as adding new people and relationships.

## Table of Contents

<b>Executive Summary.....</b>	<b>1</b>
<b>1. Postgres Database (PostgreSQL).....</b>	<b>5</b>
<b>1.1 Tables Setup and Data .....</b>	<b>5</b>
1.1.1 Postgres database creation .....	5
1.1.2 Tables creation.....	5
1.1.3 Data insertion .....	9
<b>1.2 Retrieving coordinates for a specific person.....</b>	<b>13</b>
<b>1.3 Retrieving friendship/social media links for a specific person.....</b>	<b>16</b>
<b>2. PostGIS Spatial Database .....</b>	<b>21</b>
<b>2.1 Tables Setup and Data .....</b>	<b>21</b>
<b>2.2 Retrieving a Tracklog for a given set of point coordinates – within polygon .....</b>	<b>23</b>
<b>2.2 Retrieving a Tracklog for a given set of point coordinates – nearest polygon.....</b>	<b>27</b>
<b>3. Document NoSQL database (MongoDB).....</b>	<b>31</b>
<b>3.1 Database Setup, Collection, and Document.....</b>	<b>31</b>
<b>3.1.1 Database Setup and Collection.....</b>	<b>31</b>
<b>3.1.2 Documents (Reviews) of Initial 5 people .....</b>	<b>33</b>
<b>3.2 Inserting new review into database .....</b>	<b>38</b>
<b>3.3 Retrieving highest rated review .....</b>	<b>43</b>
<b>4. Graph Database (Neo4j).....</b>	<b>47</b>
<b>4.1 Neo4j Database Creation .....</b>	<b>47</b>
4.1.1 Assumptions .....	47
4.1.2 Database Creation Images.....	47
<b>4.2 Inserting Nodes and Links .....</b>	<b>53</b>
4.2.1 Inserting Person (Nodes).....	53
4.2.2 Inserting NewPerson (Nodes).....	54
4.2.3 Inserting links(relationships) between Person and NewPerson nodes .....	55
4.2.4 Inserting Location Nodes.....	57
4.2.5 Inserting links(relationships) between Person and Location Nodes .....	58
<b>4.3 Retrieving all connected people for a specific person.....</b>	<b>60</b>
4.3.1 For John Doe .....	60
4.3.2 For Bishow Budhamagar .....	60
4.3.3 For Michael Johnson .....	61
4.3.4 For Saurav Acharya.....	61
4.3.5 For David Brown .....	62
<b>5. Critical Reflection.....</b>	<b>63</b>
<b>5.1 Proposed extensions to the system – Spatial .....</b>	<b>63</b>
<b>5.2 NoSQL.....</b>	<b>63</b>
<b>5.3 Graph.....</b>	<b>63</b>
<b>5.4 How Could I Improve: .....</b>	<b>63</b>
<b>6. Conclusion:.....</b>	<b>64</b>
Figure 1 Postgres database creation (1) .....	5

Figure 2 Postgres database creation (2) .....	5
Figure 3 Profiles table creation .....	6
Figure 4 Location table creation.....	7
Figure 5 Links table creation .....	8
Figure 6 Friendships table creation .....	9
Figure 7 Profiles data insertion .....	10
Figure 8 Location data insertion.....	11
Figure 9 Links data insertion .....	12
Figure 10 Friendships data insertion.....	13
Figure 11 Retrieving coordinates for John Doe.....	13
Figure 12 Retrieving coordinates for Bishow Budhamagar .....	14
Figure 13 Retrieving coordinates for Michael Johnson.....	15
Figure 14 Retrieving coordinates for Saurav Acharya .....	15
Figure 15 Retrieving coordinates for David Brown .....	16
Figure 16 Retrieving friendship/social media links for John Doe.....	17
Figure 17 Retrieving friendship/social media links for Bishow Budhamagar .....	18
Figure 18 Retrieving friendship/social media links for Michael Johnson.....	19
Figure 19 Retrieving friendship/social media links for Saurav Acharya .....	20
Figure 20 Retrieving friendship/social media links for David Brown .....	20
Figure 21 Creating PostGIS Extension (1).....	21
Figure 22 Creating PostGIS Extension (2).....	21
Figure 23 Spatial Tables(tracklogs) creation .....	22
Figure 24 Spatial Data insertion.....	23
Figure 25 Retrieving a Tracklog within Taronga Zoo Sydney.....	24
Figure 26 Retrieving a Tracklog within Sydney Opera House .....	25
Figure 27 Retrieving a Tracklog within South Cronulla Beach .....	25
Figure 28 Retrieving a Tracklog within Sea Cliff Bridge .....	26
Figure 29 Retrieving a Tracklog within Bondi Beach.....	27
Figure 30 Retrieving a Tracklog nearest to Taronga Zoo Sydney .....	28
Figure 31 Retrieving a Tracklog nearest to Sydney Opera House .....	29
Figure 32 Retrieving a Tracklog nearest to South Cronulla Beach.....	29
Figure 33 Retrieving a Tracklog nearest to Sea Cliff Bridge .....	30
Figure 34 Retrieving a Tracklog nearest to Bondi Beach .....	30
Figure 35 Database Setup and Collection (1) .....	31
Figure 36 Database Setup and Collection (2) .....	31
Figure 37 Database Setup and Collection (3) .....	32
Figure 38 Database Setup and Collection (4) .....	32
Figure 39 Database Setup and Collection (5) .....	33
Figure 40 Documents (Reviews) of John Doe.....	34
Figure 41 Documents (Reviews) of Bishow Budhamagar.....	35
Figure 42 Documents (Reviews) of Michael Johnson.....	36
Figure 43 Documents (Reviews) of Saurav Acharya .....	37
Figure 44 Documents (Reviews) of David Brown .....	38
Figure 45 Inserting new review into database of Emily Wilson .....	39
Figure 46 Inserting new review into database of Benjamin Davis.....	40
Figure 47 Inserting new review into database of Chloe Harris .....	41
Figure 48 Inserting new review into database of James Smith .....	42
Figure 49 Inserting new review into database of Sophia Jackson .....	43
Figure 50 Retrieving highest rated review for Taronga Zoo Sydney .....	43
Figure 51 Retrieving highest rated review for Sydney Opera House.....	44

Figure 52 Retrieving highest rated review for South Cronulla Beach.....	45
Figure 53 Retrieving highest rated review for Sea Cliff Bridge .....	45
Figure 54 Retrieving highest rated review for Bondi Beach .....	46
Figure 55 Database Creation Image (1) .....	47
Figure 56 Database Creation Image (2) .....	48
Figure 57 Database Creation Image (3) .....	48
Figure 58 Database Creation Image (4) .....	49
Figure 59 Database Creation Image (5) .....	49
Figure 60 Database Creation Image (6) .....	50
Figure 61 Database Creation Image (7) .....	50
Figure 62 Database Creation Image (8) .....	51
Figure 63 Database Creation Image (9) .....	51
Figure 64 Screenshot to show it is connected to my database ITECH2004_30397109_SPACIAL_DB.....	52
Figure 65 Inserting Person nodes .....	53
Figure 66 Graph representation of Person Nodes .....	53
Figure 67 Inserting NewPerson nodes .....	54
Figure 68 Graph representation of NewPerson Nodes.....	54
Figure 69 Inserting links(relationships) between Person and NewPerson nodes .....	55
Figure 70 Graph representation of relationships between Person and NewPerson nodes.....	56
Figure 71 Inserting Location Nodes .....	57
Figure 72 Graph representation of Location Nodes .....	57
Figure 73 Inserting links(relationships) between Person and Location Nodes .....	58
Figure 74 Graph representation of relationships between Person and Location Nodes.....	59
Figure 75 Retrieving all the friends of John Doe .....	60
Figure 76 Retrieving all the friends of Bishow Budhamagar .....	60
Figure 77 Retrieving all the friends of Michael Johnson .....	61
Figure 78 Retrieving all the friends of Saurav Acharya.....	61
Figure 79 Retrieving all the friends of David Brown .....	62

# 1. Postgres Database (PostgreSQL)

## 1.1 Tables Setup and Data

### 1.1.1 Postgres database creation

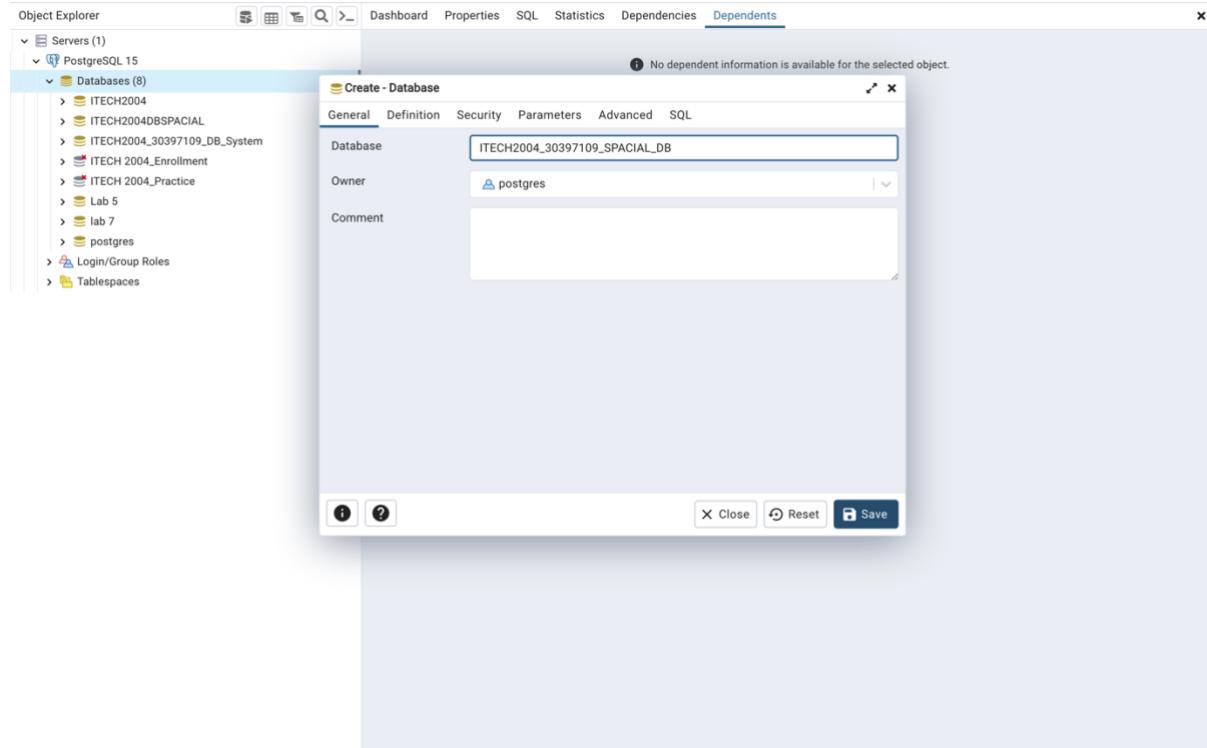


Figure 1 Postgres database creation (1)

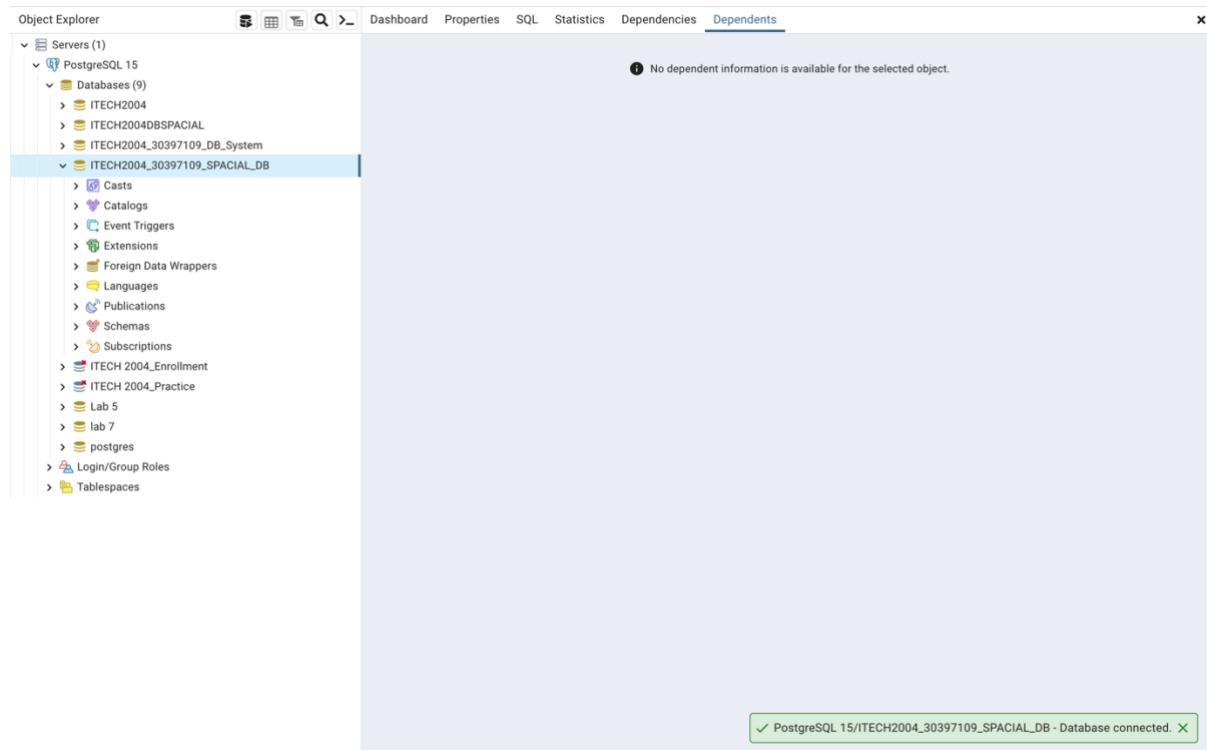


Figure 2 Postgres database creation (2)

### 1.1.2 Tables creation

## profiles

START TRANSACTION;

```
CREATE TABLE profiles (
    profile_id          INTEGER,
    fname               VARCHAR(255) NOT NULL,
    lname               VARCHAR(255) NOT NULL,
    username            VARCHAR(255) UNIQUE NOT NULL,
    bio                 TEXT,
    date_of_birth       DATE DEFAULT CURRENT_TIMESTAMP NOT NULL,
    profile_image_url   VARCHAR(255),
    PRIMARY KEY (profile_id)
);
```

COMMIT;

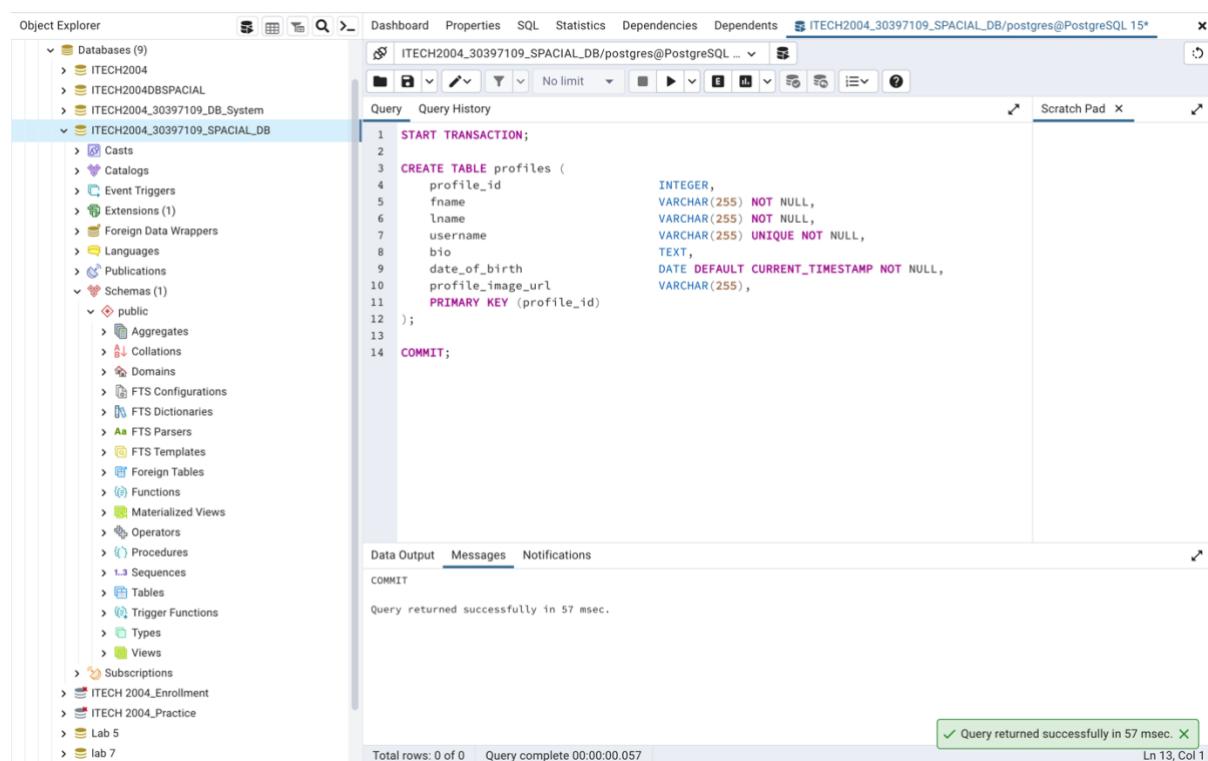


Figure 3 Profiles table creation

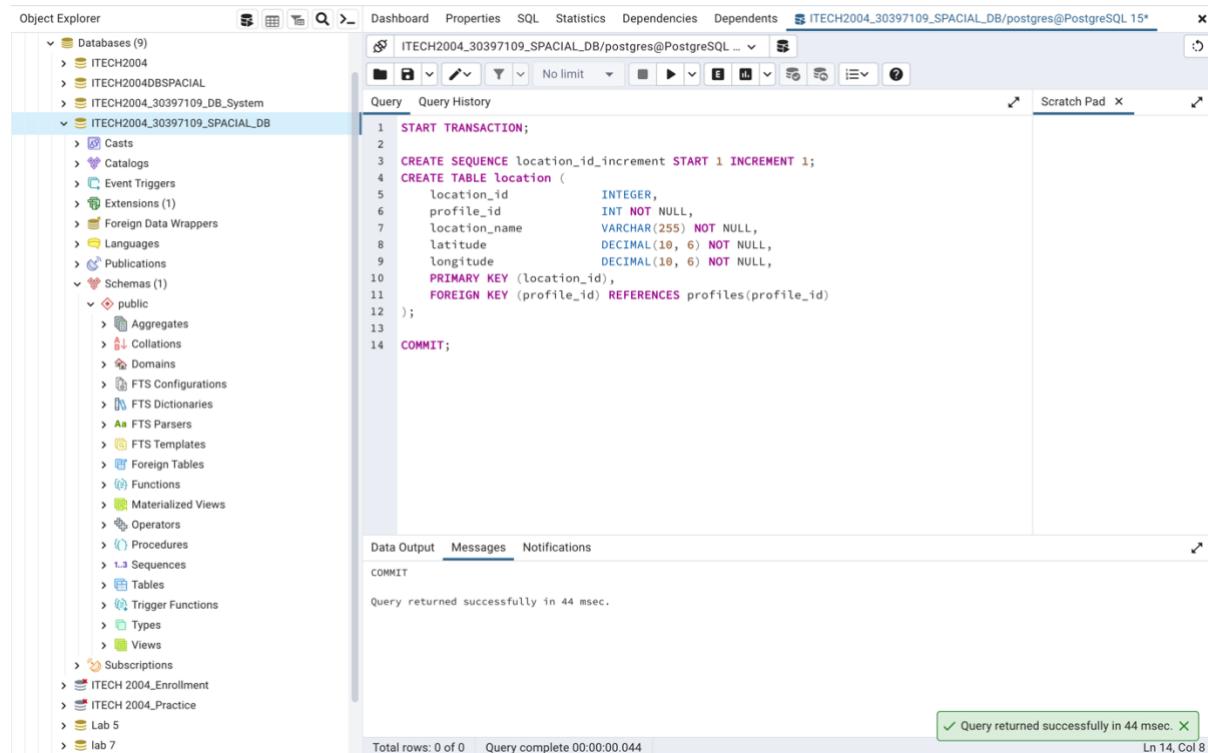
## location

START TRANSACTION;

```
CREATE SEQUENCE location_id_increment START 1 INCREMENT 1;
CREATE TABLE location (
    location_id          INTEGER,
    profile_id           INT NOT NULL,
    location_name        VARCHAR(255) NOT NULL,
    latitude              DECIMAL(10, 6) NOT NULL,
    longitude             DECIMAL(10, 6) NOT NULL,
    PRIMARY KEY (location_id),
    FOREIGN KEY (profile_id) REFERENCES profiles(profile_id)
```

);

COMMIT;



The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer pane, which lists databases, schemas, and various database objects like tables, functions, and triggers. The current schema selected is 'ITECH2004\_30397109\_SPACIAL\_DB'. In the center is the SQL editor window containing the following SQL code:

```
1 START TRANSACTION;
2
3 CREATE SEQUENCE location_id_increment START 1 INCREMENT 1;
4 CREATE TABLE location (
5     location_id      INTEGER,
6     profile_id       INT NOT NULL,
7     location_name    VARCHAR(255) NOT NULL,
8     latitude          DECIMAL(10, 6) NOT NULL,
9     longitude         DECIMAL(10, 6) NOT NULL,
10    PRIMARY KEY (location_id),
11    FOREIGN KEY (profile_id) REFERENCES profiles(profile_id)
12 );
13
14 COMMIT;
```

Below the SQL editor, the Messages tab shows the output: "Query returned successfully in 44 msec." and "Total rows: 0 of 0 Query complete 00:00:00.044". A green status bar at the bottom right indicates "Query returned successfully in 44 msec. Ln 14, Col 8".

Figure 4 Location table creation

## links

START TRANSACTION;

```
CREATE TABLE links (
    profile_id           INT NOT NULL,
    location_id          INT NOT NULL,
    ratings              INT NOT NULL,
    likings              VARCHAR(25) NOT NULL,
    PRIMARY KEY (profile_id, location_id),
    FOREIGN KEY (profile_id) REFERENCES profiles(profile_id),
    FOREIGN KEY (location_id) REFERENCES location(location_id)
);
```

COMMIT;

```

1 START TRANSACTION;
2
3 CREATE TABLE links (
4     profile_id           INT NOT NULL,
5     location_id          INT NOT NULL,
6     ratings              INT NOT NULL,
7     likings               VARCHAR(25) NOT NULL,
8     PRIMARY KEY (profile_id, location_id),
9     FOREIGN KEY (profile_id) REFERENCES profiles(profile_id),
10    FOREIGN KEY (location_id) REFERENCES location(location_id)
11 );
12
13 COMMIT;

```

Query returned successfully in 59 msec.

Total rows: 0 of 0    Query complete 00:00:00.059

✓ Query returned successfully in 59 msec. X  
Ln 13, Col 8

Figure 5 Links table creation

## friendships

START TRANSACTION;

CREATE SEQUENCE friendship\_id\_increment START 1 INCREMENT 1;  
 CREATE TABLE friendships (

```

friendship_id           INTEGER,
profile_id1             INTEGER NOT NULL,
profile_id2             INTEGER NOT NULL,
friendship_strength     INTEGER NOT NULL,
PRIMARY KEY (friendship_id),
FOREIGN KEY (profile_id1) REFERENCES profiles(profile_id),
FOREIGN KEY (profile_id2) REFERENCES profiles(profile_id)
);
```

COMMIT;

```

1 START TRANSACTION;
2
3 CREATE SEQUENCE friendship_id_increment START 1 INCREMENT 1;
4 CREATE TABLE friendships (
5     friendship_id      INTEGER,
6     profile_id1        INTEGER NOT NULL,
7     profile_id2        INTEGER NOT NULL,
8     friendship_strength INTEGER NOT NULL,
9     PRIMARY KEY (friendship_id),
10    FOREIGN KEY (profile_id1) REFERENCES profiles(profile_id),
11    FOREIGN KEY (profile_id2) REFERENCES profiles(profile_id)
12 );
13
14 COMMIT;

```

Query returned successfully in 50 msec.

Total rows: 0 of 0    Query complete 00:00:00.050

✓ Query returned successfully in 50 msec. X  
Ln 14, Col 8

Figure 6 Friendships table creation

### 1.1.3 Data insertion

#### profiles

START TRANSACTION;

```

/* profiles rows      */
INSERT INTO profiles VALUES(1, 'John', 'Doe', 'john_doe123', 'I love exploring new
places.', '1990-05-15', 'https://example.com/john_profile.jpg');
INSERT INTO profiles VALUES(2, 'Bishow', 'Budhamagar', 'bishow_magar456',
'Travel enthusiast and photographer.', '1985-08-22',
'https://example.com/bishow_profile.jpg');
INSERT INTO profiles VALUES(3, 'Michael', 'Johnson', 'mike_j', 'Adventurer and
nature lover.', '1982-02-10', 'https://example.com/mike_profile.jpg');
INSERT INTO profiles VALUES(4, 'Saurav', 'Acharya', 'acharya_saurav', 'Always
looking for the next adventure.', '1995-11-30',
'https://example.com/saurav_profile.jpg');
INSERT INTO profiles VALUES(5, 'David', 'Brown', 'david_b', 'Exploring the world,
one step at a time.', '1988-07-18', 'https://example.com/david_profile.jpg');

COMMIT;

```

```

1 START TRANSACTION;
2
3 /* profiles rows */
4 INSERT INTO profiles VALUES(1, 'John', 'Doe', 'john_doe123', 'I love exploring new places.', '1990-05-15', 'https://www.example.com');
5 INSERT INTO profiles VALUES(2, 'Bishow', 'Budhamagar', 'bishow_magar456', 'Travel enthusiast and photographer.', '1985-08-20', 'http://travelphotographer.com');
6 INSERT INTO profiles VALUES(3, 'Michael', 'Johnson', 'mike_j', 'Adventurer and nature lover.', '1982-02-10', 'http://adventureandnature.com');
7 INSERT INTO profiles VALUES(4, 'Saurav', 'Acharya', 'acharya_saurav', 'Always looking for the next adventure.', '1988-07-15', 'http://nextadventure.com');
8 INSERT INTO profiles VALUES(5, 'David', 'Brown', 'david_b', 'Exploring the world, one step at a time.', '1988-07-15', 'http://onestepatime.com');

10 COMMIT;

```

Messages

Query returned successfully in 55 msec.

Total rows: 0 of 0    Query complete 00:00:00.055

Figure 7 Profiles data insertion

## location

START TRANSACTION;

```

/* location rows */
INSERT INTO location VALUES(1, 1, 'Taronga Zoo Sydney', -33.84086, 151.24141);
INSERT INTO location VALUES(2, 2, 'Sydney Opera House', -33.85453, 151.21610);
INSERT INTO location VALUES(3, 3, 'South Cronulla Beach', -34.05531, 151.15490);
INSERT INTO location VALUES(4, 4, 'Sea Cliff Bridge', -34.25290, 150.97351);
INSERT INTO location VALUES(5, 5, 'Bondi Beach', -33.89084, 151.27665);

```

COMMIT;

The screenshot shows the pgAdmin III application interface. On the left is the Object Explorer pane, which lists databases, schemas, tables, and other database objects. In the center is the Query pane, which contains a SQL script for inserting data into a 'location' table. The right side shows the Data Output pane, which displays the successful execution message: 'Query returned successfully in 52 msec.' Below the Data Output pane, it says 'Total rows: 0 of 0' and 'Query complete 00:00:00.052'. The status bar at the bottom right indicates 'Ln 5, Col 6'.

```

1 START TRANSACTION;
2
3 /* location rows */
4 INSERT INTO location VALUES(1, 1, 'Taronga Zoo Sydney', -33.84086, 151.24141);
5 INSERT INTO location VALUES(2, 2, 'Sydney Opera House', -33.85453, 151.21610);
6 INSERT INTO location VALUES(3, 3, 'South Cronulla Beach', -34.05531, 151.15490);
7 INSERT INTO location VALUES(4, 4, 'Sea Cliff Bridge', -34.25298, 150.97351);
8 INSERT INTO location VALUES(5, 5, 'Bondi Beach', -33.89084, 151.27665);
9
10 COMMIT;
11

```

Figure 8 Location data insertion

## links

START TRANSACTION;

```

/* links rows */
INSERT INTO links VALUES(1, 1, 4, 'Dislike');
INSERT INTO links VALUES(2, 2, 9, 'like');
INSERT INTO links VALUES(3, 3, 6, 'Favorite');
INSERT INTO links VALUES(4, 4, 6, 'Okay');
INSERT INTO links VALUES(5, 5, 8, 'Favorite');

```

COMMIT;

```

1 START TRANSACTION;
2
3 /* links rows */
4 INSERT INTO links VALUES(1, 1, 4, 'Dislike');
5 INSERT INTO links VALUES(2, 2, 9, 'like');
6 INSERT INTO links VALUES(3, 3, 6, 'Favorite');
7 INSERT INTO links VALUES(4, 4, 6, 'Okay');
8 INSERT INTO links VALUES(5, 5, 8, 'Favorite');
9
10 COMMIT;

```

Data Output    Messages    Notifications

Query returned successfully in 42 msec.

Ln 10, Col 8

Figure 9 Links data insertion

## friendships

START TRANSACTION;

```

/* friendships rows      */
INSERT INTO friendships VALUES(1, 1, 2, 8);
INSERT INTO friendships VALUES(2, 1, 3, 6);
INSERT INTO friendships VALUES(3, 2, 4, 7);
INSERT INTO friendships VALUES(4, 3, 5, 9);
INSERT INTO friendships VALUES(5, 4, 5, 5);

```

COMMIT;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the Object Explorer with the 'Databases' section expanded, showing 'ITECH2004\_30397109\_SPACIAL\_DB'. The main pane shows a query editor with the following SQL code:

```

1 START TRANSACTION;
2 /* friendships rows */
3 INSERT INTO friendships VALUES(1, 1, 2, 8);
4 INSERT INTO friendships VALUES(2, 1, 3, 6);
5 INSERT INTO friendships VALUES(3, 2, 4, 7);
6 INSERT INTO friendships VALUES(4, 3, 5, 9);
7 INSERT INTO friendships VALUES(5, 4, 5, 5);
8
9
10 COMMIT;
11

```

The 'Messages' tab shows the message 'Query returned successfully in 94 msec.' and the status bar indicates 'Total rows: 0 of 0 Query complete 00:00:00.094'.

Figure 10 Friendships data insertion

## 1.2 Retrieving coordinates for a specific person

### For John Doe

```

SELECT profiles.fname, profiles.lname, "location".latitude, "location".longitude,
"location".location_name
FROM "location" AS "location"
JOIN profiles AS profiles ON "location".profile_id = profiles.profile_id
WHERE profiles.username = 'john_doe123';

```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the Object Explorer with the 'Tables' section expanded, showing 'profiles'. The main pane shows a query editor with the following SQL code:

```

1
2 SELECT profiles.fname, profiles.lname, "location".latitude, "location".longitude,
3 FROM "location" AS "location"
4 JOIN profiles AS profiles ON "location".profile_id = profiles.profile_id
5 WHERE profiles.username = 'john_doe123';
6
7

```

The 'Data Output' tab shows the result of the query:

fname	lname	latitude	longitude	location_name
John	Doe	-33.840860	151.241410	Taronga Zoo Sydney

The status bar indicates 'Total rows: 1 of 1 Query complete 00:00:00.090' and a message 'Successfully run. Total query runtime: 90 msec. 1 rows affected.'

Figure 11 Retrieving coordinates for John Doe

## For Bishow Budhamagar

```
SELECT profiles.fname, profiles.lname, "location".latitude, "location".longitude,  
"location".location_name  
FROM "location" AS "location"  
JOIN profiles AS profiles ON "location".profile_id = profiles.profile_id  
WHERE profiles.profile_image_url = 'https://example.com/bishow_profile.jpg';
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** On the left, it lists various database objects including Schemas, Tables (with profiles selected), and Views.
- Query Editor:** The main area contains the SQL query provided above.
- Data Output:** Below the query editor, the results are displayed in a table:

	fname	lname	latitude	longitude	location_name
1	Bishow	Budhamagar	-33.854530	151.216100	Sydney Opera House

- Status Bar:** At the bottom right, it shows "Successfully run. Total query runtime: 101 msec. 1 rows affected."

Figure 12 Retrieving coordinates for Bishow Budhamagar

## For Michael Johnson

```
SELECT profiles.fname, profiles.lname, "location".latitude, "location".longitude,  
"location".location_name  
FROM "location" AS "location"  
JOIN profiles AS profiles ON "location".profile_id = profiles.profile_id  
WHERE profiles.username = 'mike_j';
```

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying various database objects like extensions, schemas, tables, and functions. The right pane contains a query editor window with the following SQL code:

```

1
2 SELECT profiles.fname, profiles.lname, "location".latitude, "location".longitude,
3 FROM "location" AS "location"
4 JOIN profiles AS profiles ON "location".profile_id = profiles.profile_id
5 WHERE profiles.username = 'mike_j';
6
7

```

Below the query editor is a Data Output grid showing the results of the query:

	fname	lname	latitude	longitude	location_name
1	Michael	Johnson	-34.055310	151.154900	South Cronulla Beach

At the bottom of the pgAdmin window, a status bar displays: "Total rows: 1 of 1 Query complete 00:00:00.092" and a message: "Successfully run. Total query runtime: 92 msec. 1 rows affected. Ln 5, Col 35".

Figure 13 Retrieving coordinates for Michael Johnson

## For Saurav Acharya

```

SELECT profiles.fname, profiles.lname, "location".latitude, "location".longitude,
"location".location_name
FROM "location" AS "location"
JOIN profiles AS profiles ON "location".profile_id = profiles.profile_id
WHERE profiles.bio = 'Always looking for the next adventure.';

```

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying various database objects. The right pane contains a query editor window with the following SQL code:

```

1
2 SELECT profiles.fname, profiles.lname, "location".latitude, "location".longitude,
3 FROM "location" AS "location"
4 JOIN profiles AS profiles ON "location".profile_id = profiles.profile_id
5 WHERE profiles.bio = 'Always looking for the next adventure.';
6
7

```

Below the query editor is a Data Output grid showing the results of the query:

	fname	lname	latitude	longitude	location_name
1	Saurav	Acharya	-34.252900	150.973510	Sea Cliff Bridge

At the bottom of the pgAdmin window, a status bar displays: "Total rows: 1 of 1 Query complete 00:00:00.093" and a message: "Successfully run. Total query runtime: 93 msec. 1 rows affected. Ln 5, Col 19".

Figure 14 Retrieving coordinates for Saurav Acharya

## For David Brown

```
SELECT profiles.fname, profiles.lname, "location".latitude, "location".longitude,  
"location".location_name  
FROM "location" AS "location"  
JOIN profiles AS profiles ON "location".profile_id = profiles.profile_id  
WHERE profiles.username = 'david_b';
```

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer tree, which includes Schemas, Tables (containing profiles), and other database objects. The main area is the Query Editor, displaying the SQL query for retrieving coordinates for David Brown. Below the editor is the Data Output pane, which shows a single row of results:

	fname	lname	latitude	longitude	location_name
1	David	Brown	-33.890840	151.276650	Bondi Beach

At the bottom of the interface, a message bar indicates: "Successfully run. Total query runtime: 97 msec. 1 rows affected." and "Ln 5, Col 36".

Figure 15 Retrieving coordinates for David Brown

## 1.3 Retrieving friendship/social media links for a specific person

### For John Doe

```
SELECT  
    CONCAT(p1.fname, ' ', p1.lname) AS profile_name,  
    CONCAT(p2.fname, ' ', p2.lname) AS friend_name,  
    f.friendship_strength  
FROM friendships f  
JOIN profiles p1 ON f.profile_id1 = p1.profile_id  
JOIN profiles p2 ON f.profile_id2 = p2.profile_id  
WHERE p1.profile_id = 1;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the Object Explorer, listing various database objects like extensions, schemas, tables, and functions. The main area is a query editor window with the following SQL code:

```

1 SELECT
2   CONCAT(p1.fname, ' ', p1.lname) AS profile_name,
3   CONCAT(p2.fname, ' ', p2.lname) AS friend_name,
4   f.friendship_strength
5 FROM friendships f
6 JOIN profiles p1 ON f.profile_id1 = p1.profile_id
7 JOIN profiles p2 ON f.profile_id2 = p2.profile_id
8 WHERE p1.profile_id = 1;

```

The Data Output tab shows the results of the query:

	profile_name	friend_name	friendship_strength
1	John Doe	Bishow Budhamagar	8
2	John Doe	Michael Johnson	6

At the bottom of the interface, a message indicates: "Successfully run. Total query runtime: 99 msec. 2 rows affected." and "Ln 8, Col 24".

Figure 16 Retrieving friendship/social media links for John Doe

## For Bishow Budhamagar

```

SELECT
  CONCAT(p1.fname, ' ', p1.lname) AS profile_name,
  CONCAT(p2.fname, ' ', p2.lname) AS friend_name,
  f.friendship_strength
FROM friendships f
JOIN profiles p1 ON f.profile_id1 = p1.profile_id
JOIN profiles p2 ON f.profile_id2 = p2.profile_id
WHERE p1.profile_id = 2;

```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the Object Explorer, listing various database objects like extensions, schemas, tables, and functions. The main area is a query editor window with the following SQL code:

```

1 SELECT
2   CONCAT(p1.fname, ' ', p1.lname) AS profile_name,
3   CONCAT(p2.fname, ' ', p2.lname) AS friend_name,
4   f.friendship_strength
5 FROM friendships f
6 JOIN profiles p1 ON f.profile_id1 = p1.profile_id
7 JOIN profiles p2 ON f.profile_id2 = p2.profile_id
8 WHERE p1.profile_id = 2;

```

The Data Output tab shows the results of the query:

	profile_name	friend_name	friendship_strength
1	Bishow Budhamagar	Saurav Acharya	7

At the bottom of the query editor, a message indicates: "Successfully run. Total query runtime: 102 msec. 1 rows affected." and "Ln 8, Col 24".

Figure 17 Retrieving friendship/social media links for Bishow Budhamagar

## For Michael Johnson

**SELECT**

```

CONCAT(p1.fname, ' ', p1.lname) AS profile_name,
CONCAT(p2.fname, ' ', p2.lname) AS friend_name,
f.friendship_strength
FROM friendships f
JOIN profiles p1 ON f.profile_id1 = p1.profile_id
JOIN profiles p2 ON f.profile_id2 = p2.profile_id
WHERE p1.profile_id = 3;

```

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Object Explorer' and lists various database objects: Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (1), public, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (4), friendships, links, location, profiles, Trigger Functions, Types, Views, Subscriptions, ITECH 2004\_Enrollment, ITECH 2004\_Practice, Lab 5, error, lab 7, postgres, Login/Group Roles, and Tablespaces. The 'profiles' table is selected.

The main window has tabs for 'Query', 'Query History', and 'Scratch Pad'. The 'Query' tab contains the following SQL code:

```

1 SELECT
2   CONCAT(p1.fname, ' ', p1.lname) AS profile_name,
3   CONCAT(p2.fname, ' ', p2.lname) AS friend_name,
4   f.friendship_strength
5 FROM friendships f
6 JOIN profiles p1 ON f.profile_id1 = p1.profile_id
7 JOIN profiles p2 ON f.profile_id2 = p2.profile_id
8 WHERE p1.profile_id = 3;

```

The 'Data Output' tab shows the results of the query:

	profile_name	friend_name	friendship_strength
1	Michael Johnson	David Brown	9

Below the table, the status bar indicates: Total rows: 1 of 1 Query complete 00:00:00.045. A green message box at the bottom right says: ✓ Successfully run. Total query runtime: 45 msec. 1 rows affected. Ln 8, Col 24.

Figure 18 Retrieving friendship/social media links for Michael Johnson

## For Saurav Acharya

**SELECT**

```

CONCAT(p1.fname, ' ', p1.lname) AS profile_name,
CONCAT(p2.fname, ' ', p2.lname) AS friend_name,
f.friendship_strength
FROM friendships f
JOIN profiles p1 ON f.profile_id1 = p1.profile_id
JOIN profiles p2 ON f.profile_id2 = p2.profile_id
WHERE p1.profile_id = 4;

```

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Object Explorer' and lists various database objects: Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (1), public, Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, Tables (4), friendships, links, location, profiles, Trigger Functions, Types, Views, Subscriptions, ITECH 2004\_Enrollment, ITECH 2004\_Practice, Lab 5, error, lab 7, postgres, Login/Group Roles, and Tablespaces.

The main area contains a query editor window with the following SQL code:

```

1 SELECT
2   CONCAT(p1.fname, ' ', p1.lname) AS profile_name,
3   CONCAT(p2.fname, ' ', p2.lname) AS friend_name,
4   f.friendship_strength
5 FROM friendships f
6 JOIN profiles p1 ON f.profile_id1 = p1.profile_id
7 JOIN profiles p2 ON f.profile_id2 = p2.profile_id
8 WHERE p1.profile_id = 4;

```

The 'Data Output' tab shows the results:

	profile_name	friend_name	friendship_strength
1	Saurav Acharya	David Brown	5

Below the table, the status bar indicates: Total rows: 1 of 1 Query complete 00:00:00.100. A message bar at the bottom right says: Successfully run. Total query runtime: 100 msec. 1 rows affected. Ln 8, Col 24.

Figure 19 Retrieving friendship/social media links for Saurav Acharya

## For David Brown

**SELECT**

```

CONCAT(p1.fname, ' ', p1.lname) AS profile_name,
CONCAT(p2.fname, ' ', p2.lname) AS friend_name,
f.friendship_strength

```

FROM friendships f

JOIN profiles p1 ON f.profile\_id1 = p1.profile\_id

JOIN profiles p2 ON f.profile\_id2 = p2.profile\_id

WHERE p1.profile\_id = 5;

The screenshot shows the pgAdmin 4 interface, identical to Figure 19, with the same Object Explorer sidebar and the same SQL query in the query editor. The 'Data Output' tab is empty, showing only the column headers: profile\_name, friend\_name, and friendship\_strength. Below the table, the status bar indicates: Total rows: 0 of 0 Query complete 00:00:00.098. A message bar at the bottom right says: Successfully run. Total query runtime: 98 msec. 0 rows affected. Ln 8, Col 24.

Figure 20 Retrieving friendship/social media links for David Brown

## 2. PostGIS Spatial Database

### 2.1 Tables Setup and Data

#### Extension creation

/\* Creating extension for Spatial Database Assignment \*/

CREATE extension postgis;

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying a tree structure of servers, databases, and various database objects like casts, catalogs, triggers, and extensions. The 'Extensions' node under the current database is selected. The right pane contains a query editor window with the following SQL code:

```
1 /* Creating extension for Spatial Database Assignment */
2 CREATE extension postgis;
```

Below the query editor, the 'Messages' tab is active, showing the output of the query:

CREATE EXTENSION

Query returned successfully in 1 secs 132 msec.

Total rows: 0 of 0    Query complete 00:00:01.132    Ln 1, Col 54

A green success message box is visible at the bottom right of the interface.

Figure 21 Creating PostGIS Extension (1)

This screenshot is nearly identical to Figure 21, showing the pgAdmin 4 interface with the same query and successful execution message. The main difference is in the Object Explorer tree, where the 'Extensions' node is now expanded, revealing two entries: 'plpgsql' and 'postgis'. This indicates that the PostGIS extension has been successfully created and is now listed among the available extensions in the database.

Figure 22 Creating PostGIS Extension (2)

## Spatial Tables(tracklogs) creation

START TRANSACTION;

```
CREATE TABLE tracklogs (
    tracklog_id INTEGER,
    profile_id INTEGER,
    tracklog_name VARCHAR(255) NOT NULL,
    tracklog_geom GEOMETRY(POLYGON, 4326) NOT NULL,
        PRIMARY KEY (tracklog_id),
    FOREIGN KEY (profile_id) REFERENCES profiles(profile_id)
);
```

COMMIT;

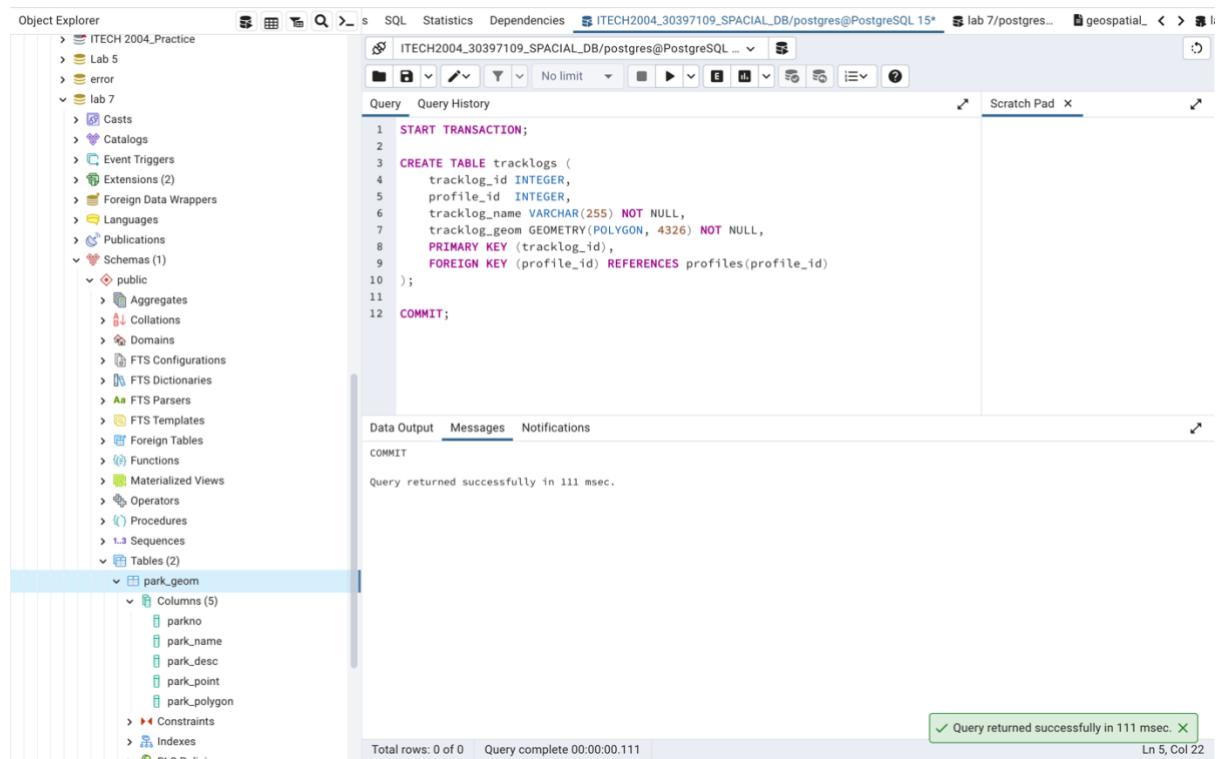


Figure 23 Spatial Tables(tracklogs) creation

## Spatial Data insertion

START TRANSACTION;

```
INSERT INTO tracklogs (tracklog_id, profile_id, tracklog_name, tracklog_geom)
VALUES
```

```
(1, 1,'Taroonga Zoo Sydney', ST_GeomFromText('POLYGON((151.2614984 - 33.8129410, 151.2617451 -33.8134223, 151.2618417 -33.8140641, 151.2622709 - 33.8148486, 151.2630004 -33.8147238, 151.2632579 -33.8148307, 151.2643952 - 33.8140106, 151.2642986 -33.8126914, 151.2630541 -33.8130212, 151.2614984 - 33.8129410)', 4326)),
```

```
(2, 2,'Sydney Opera House', ST_GeomFromText('POLYGON((151.2134082 - 33.8588087, 151.2140090 -33.8586973, 151.2145294 -33.8583276, 151.2152535 - 33.8582697, 151.2155486 -33.8574633, 151.2145884 -33.8568753, 151.2143148 -
```

```

33.8572094, 151.2140948 -33.8577930, 151.2139232 -33.8580959, 151.2134082 -
33.8588087)', 4326)),
(3, 3,'South Cronulla Beach', ST_GeomFromText('POLYGON((151.1550624 -
34.0542858, 151.1548800 -34.0547569, 151.1546869 -34.0551036, 151.1545796 -
34.0556013, 151.1549766 -34.0559036, 151.1554701 -34.0562769, 151.1552019 -
34.0557702, 151.1551375 -34.0548369, 151.1552341 -34.0545080, 151.1550624 -
34.0542858)', 4326)),
(4, 4,'Sea Cliff Bridge', ST_GeomFromText('POLYGON((150.9725450 -
34.2566740, 150.9731887 -34.2562483, 150.9743903 -34.2535171, 150.9758494 -
34.2510695, 150.9761928 -34.2490120, 150.9751628 -34.2501472, 150.9754203 -
34.2511404, 150.9749053 -34.2520627, 150.9731029 -34.2540491, 150.9725450 -
34.2566740)', 4326)),
(5, 5,'Bondi Beach', ST_GeomFromText('POLYGON((151.2741735 -
33.8944094, 151.2749245 -33.8933407, 151.2776282 -33.8916486, 151.2801388 -
33.8909361, 151.2816408 -33.8909361, 151.2824133 -33.8911677, 151.2814262 -
33.8902771, 151.2765339 -33.8908115, 151.2743666 -33.8922720, 151.2741735 -
33.8944094)', 4326))
;

```

COMMIT;

```

Object Explorer   ITECH2004_30397109_SPACIAL_DB/postgres@PostgreSQL 15*   lab 7/postgres...   geospatial...
ITECH 2004_Practice
> Lab 5
> error
Lab 7
> Casts
> Catalogs
> Event Triggers
> Extensions (2)
> Foreign Data Wrappers
> Languages
> Publications
Schemas (1)
public
> Aggregates
> Collations
> Domains
> FTS Configurations
> FTS Dictionaries
> FTS Parsers
> FTS Templates
> Foreign Tables
> Functions
> Materialized Views
> Operators
> Procedures
> Sequences
Tables (2)
park_geom
Columns (5)
parkno
park_name
park_desc
park_point
park_polygon
Constraints
Indexes

```

```

Query   Query History
1 START TRANSACTION;
2
3 INSERT INTO tracklogs (tracklog_id, profile_id, tracklog_name, tracklog_geom)
4 VALUES
5 (1, 1,'Taroonga Zoo Sydney', ST_GeomFromText('POLYGON((151.2614984 -33.812941,
6 (2, 2,'Sydney Opera House', ST_GeomFromText('POLYGON((151.2134082 -33.8588087
7 (3, 3,'South Cronulla Beach', ST_GeomFromText('POLYGON((151.1550624 -34.05428
8 (4, 4,'Sea Cliff Bridge', ST_GeomFromText('POLYGON((150.9725450 -34.2566740,
9 (5, 5,'Bondi Beach', ST_GeomFromText('POLYGON((151.2741735 -33.8944094, 151.2
10 ;
11
12 COMMIT;
13

Data Output   Messages   Notifications
COMMIT

Query returned successfully in 94 msec.
Total rows: 0 of 0   Query complete 00:00:00.094

```

✓ Query returned successfully in 94 msec. X  
Ln 11, Col 2

Figure 24 Spatial Data insertion

## 2.2 Retrieving a Tracklog for a given set of point coordinates – within polygon For Taroonga Zoo Sydney

WITH user\_location AS (

```

    SELECT ST_SetSRID(ST_MakePoint(151.2637, -33.8135), 4326) AS user_point
)
```

```

SELECT tracklog_id, tracklog_name
FROM tracklogs, user_location
WHERE ST_Within(user_location.user_point, tracklog_geom);

```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the Object Explorer, displaying the database structure. The main area is a query editor window with the following content:

```

1 WITH user_location AS (
2     SELECT ST_SetSRID(ST_MakePoint(151.2140, -33.8580), 4326) AS user_point
3 )
4
5 SELECT tracklog_id, tracklog_name
6 FROM tracklogs, user_location
7 WHERE ST_Within(user_location.user_point, tracklog_geom);
8

```

The Data Output tab shows a single row of results:

	tracklog_id	tracklog_name
1	[PK] integer	character varying (255)
1	1	Taronga Zoo Sydney

At the bottom of the query editor, a status bar indicates: "Successfully run. Total query runtime: 46 msec. 1 rows affected." and "Ln 7, Col 49".

Figure 25 Retrieving a Tracklog within Taronga Zoo Sydney

**For Sydney Opera House**

```

WITH user_location AS (
    SELECT ST_SetSRID(ST_MakePoint(151.2140, -33.8580), 4326) AS user_point
)
SELECT tracklog_id, tracklog_name
FROM tracklogs, user_location
WHERE ST_Within(user_location.user_point, tracklog_geom);

```

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying a tree structure of databases, schemas, and tables. The current database selected is 'ITECH2004\_30397109\_SPACIAL\_DB'. The right pane contains a query editor window with the following SQL code:

```

1 WITH user_location AS (
2     SELECT ST_SetSRID(ST_MakePoint(151.2140, -33.8580), 4326) AS user_point
3 )
4
5 SELECT tracklog_id, tracklog_name
6 FROM tracklogs, user_location
7 WHERE ST_Within(user_location.user_point, tracklog_geom);
8

```

Below the query editor is a Data Output tab showing a single row of results:

tracklog_id	tracklog_name
1	Sydney Opera House

At the bottom of the pgAdmin window, a status bar indicates: 'Successfully run. Total query runtime: 100 msec. 1 rows affected.' and 'Ln 3, Col 44'.

Figure 26 Retrieving a Tracklog within Sydney Opera House

**For South Cronulla Beach**  
**WITH user\_location AS (**  
**SELECT ST\_SetSRID(ST\_MakePoint(151.1550, -34.0550), 4326) AS user\_point**  
**)**  
**SELECT tracklog\_id, tracklog\_name**  
**FROM tracklogs, user\_location**  
**WHERE ST\_Within(user\_location.user\_point, tracklog\_geom);**

The screenshot shows the pgAdmin 4 interface, identical to Figure 26 but with a different query. The right pane contains the following SQL code:

```

1 WITH user_location AS (
2     SELECT ST_SetSRID(ST_MakePoint(151.1550, -34.0550), 4326) AS user_point
3 )
4
5 SELECT tracklog_id, tracklog_name
6 FROM tracklogs, user_location
7 WHERE ST_Within(user_location.user_point, tracklog_geom);
8

```

Below the query editor is a Data Output tab showing a single row of results:

tracklog_id	tracklog_name
1	South Cronulla Beach

At the bottom of the pgAdmin window, a status bar indicates: 'Successfully run. Total query runtime: 46 msec. 1 rows affected.' and 'Ln 3, Col 54'.

Figure 27 Retrieving a Tracklog within South Cronulla Beach

```

For Sea Cliff Bridge
WITH user_location AS (
    SELECT ST_SetSRID(ST_MakePoint(150.9735, -34.2550), 4326) AS user_point
)
SELECT tracklog_id, tracklog_name
FROM tracklogs, user_location
WHERE ST_Within(user_location.user_point, tracklog_geom);

```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** On the left, under the "Servers" node, the "PostgreSQL 15" database is selected. Under "ITECH2004\_30397109\_SPACIAL\_DB", the "Tables" node is expanded, showing five tables: friendships, links, location, profiles, and spatial\_ref\_sys.
- Query Editor:** The main area contains the SQL query provided above. The results pane shows a single row of data:

tracklog_id	tracklog.name
[PK] integer	character varying (255)
1	4
Sea Cliff Bridge	

- Status Bar:** At the bottom right, it says "Successfully run. Total query runtime: 60 msec. 1 rows affected." and "Ln 3, Col 44".

Figure 28 Retrieving a Tracklog within Sea Cliff Bridge

### For Bondi Beach

```

WITH user_location AS (
    SELECT ST_SetSRID(ST_MakePoint(151.2760, -33.8920), 4326) AS user_point
)
SELECT tracklog_id, tracklog_name
FROM tracklogs, user_location
WHERE ST_Within(user_location.user_point, tracklog_geom);

```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the Object Explorer, displaying the database structure. The main area is the Query Editor with the following SQL code:

```

1
2 WITH user_location AS (
3     SELECT ST_SetSRID(ST_MakePoint(151.2760, -33.8920), 4326) AS user_point
4 )
5 SELECT tracklog_id, tracklog_name
6 FROM tracklogs, user_location
7 WHERE ST_Within(user_location.user_point, tracklog_geom);
8

```

The Data Output tab shows the results of the query:

	tracklog_id	tracklog_name
1	[PK] integer	character varying (255)
1	5	Bondi Beach

At the bottom, a green message bar indicates: "Successfully run. Total query runtime: 40 msec. 1 rows affected." and "Ln 3, Col 54".

Figure 29 Retrieving a Tracklog within Bondi Beach

## 2.2 Retrieving a Tracklog for a given set of point coordinates – nearest polygon

### For Taroonga Zoo Sydney

```

WITH user_location AS (
    SELECT ST_SetSRID(ST_MakePoint(151.2625, -33.8145), 4326) AS user_point
)
SELECT tracklog_id, tracklog_name
FROM tracklogs, user_location
ORDER BY ST_Distance(user_location.user_point, tracklog_geom)
LIMIT 1;

```

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying a tree structure of databases, tables, and other database objects. The central pane is the Query Editor, containing the following SQL code:

```

1
2 WITH user_location AS (
3     SELECT ST_SetSRID(ST_MakePoint(151.2625, -33.8145), 4326) AS user_point
4 )
5 SELECT tracklog_id, tracklog_name
6 FROM tracklogs, user_location
7 ORDER BY ST_Distance(user_location.user_point, tracklog_geom)
8 LIMIT 1;

```

The Data Output pane shows the result of the query:

tracklog_id	tracklog_name
1	Taronga Zoo Sydney

At the bottom of the Query Editor, a status bar indicates: "Successfully run. Total query runtime: 127 msec. 1 rows affected. Ln 3, Col 44".

Figure 30 Retrieving a Tracklog nearest to Taroonga Zoo Sydney

## For Sydney Opera House

```

WITH user_location AS (
    SELECT ST_SetSRID(ST_MakePoint(151.2140, -33.8570), 4326) AS user_point
)
SELECT tracklog_id, tracklog_name
FROM tracklogs, user_location
ORDER BY ST_Distance(user_location.user_point, tracklog_geom)
LIMIT 1;

```

The screenshot shows the pgAdmin 4 interface, identical to Figure 30 but with a different query. The central pane is the Query Editor, containing the following SQL code:

```

1
2 WITH user_location AS (
3     SELECT ST_SetSRID(ST_MakePoint(151.2140, -33.8570), 4326) AS user_point
4 )
5 SELECT tracklog_id, tracklog_name
6 FROM tracklogs, user_location
7 ORDER BY ST_Distance(user_location.user_point, tracklog_geom)
8 LIMIT 1;

```

The Data Output pane shows the result of the query:

tracklog_id	tracklog_name
2	Sydney Opera House

At the bottom of the Query Editor, a status bar indicates: "Successfully run. Total query runtime: 122 msec. 1 rows affected. Ln 3, Col 44".

Figure 31 Retrieving a Tracklog nearest to Sydney Opera House

### For South Cronulla Beach

```
WITH user_location AS (
    SELECT ST_SetSRID(ST_MakePoint(151.1551, -34.0542), 4326) AS user_point
)
SELECT tracklog_id, tracklog_name
FROM tracklogs, user_location
ORDER BY ST_Distance(user_location.user_point, tracklog_geom)
LIMIT 1;
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** On the left, under "PostgreSQL 15", the "ITECH2004\_30397109\_SPACIAL\_DB" database is selected. Inside it, the "Tables" node is expanded, showing five tables: friendships, links, location, profiles, and spatial\_ref\_sys.
- Query Editor:** The main window contains the SQL query for retrieving the nearest tracklog to South Cronulla Beach. The result set is displayed in a table below the query.
- Data Output Table:**

	tracklog_id	tracklog_name
1	[PK] integer	character varying (255)
1	3	South Cronulla Beach
- Status Bar:** At the bottom right, it says "Successfully run. Total query runtime: 53 msec. 1 rows affected." and "Ln 3, Col 44".

Figure 32 Retrieving a Tracklog nearest to South Cronulla Beach

### For Sea Cliff Bridge

```
WITH user_location AS (
    SELECT ST_SetSRID(ST_MakePoint(150.9720, -34.2558), 4326) AS user_point
)
SELECT tracklog_id, tracklog_name
FROM tracklogs, user_location
ORDER BY ST_Distance(user_location.user_point, tracklog_geom)
LIMIT 1;
```

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying a tree structure of PostgreSQL objects. The current database selected is 'ITECH2004\_30397109\_SPACIAL\_DB'. The right pane contains a query editor window with the following SQL code:

```

1 WITH user_location AS (
2     SELECT ST_SetSRID(ST_MakePoint(150.9720, -34.2558), 4326) AS user_point
3 )
4
5 SELECT tracklog_id, tracklog_name
6 FROM tracklogs, user_location
7 ORDER BY ST_Distance(user_location.user_point, tracklog_geom)
8 LIMIT 1;

```

Below the query editor is a Data Output panel showing the results of the query:

tracklog_id	tracklog_name
1	Sea Cliff Bridge

At the bottom of the pgAdmin window, a status bar displays: 'Successfully run. Total query runtime: 104 msec. 1 rows affected.' and 'Ln 3, Col 44'.

Figure 33 Retrieving a Tracklog nearest to Sea Cliff Bridge

## For Bondi Beach

```

WITH user_location AS (
    SELECT ST_SetSRID(ST_MakePoint(151.2760, -33.8930), 4326) AS user_point
)
SELECT tracklog_id, tracklog_name
FROM tracklogs, user_location
ORDER BY ST_Distance(user_location.user_point, tracklog_geom)
LIMIT 1;

```

The screenshot shows the pgAdmin 4 interface, identical to Figure 33 but with a different query. The Object Explorer shows the same database structure. The query editor contains the following SQL code:

```

1 WITH user_location AS (
2     SELECT ST_SetSRID(ST_MakePoint(151.2760, -33.8930), 4326) AS user_point
3 )
4
5 SELECT tracklog_id, tracklog_name
6 FROM tracklogs, user_location
7 ORDER BY ST_Distance(user_location.user_point, tracklog_geom)
8 LIMIT 1;

```

The Data Output panel shows the results:

tracklog_id	tracklog_name
1	Bondi Beach

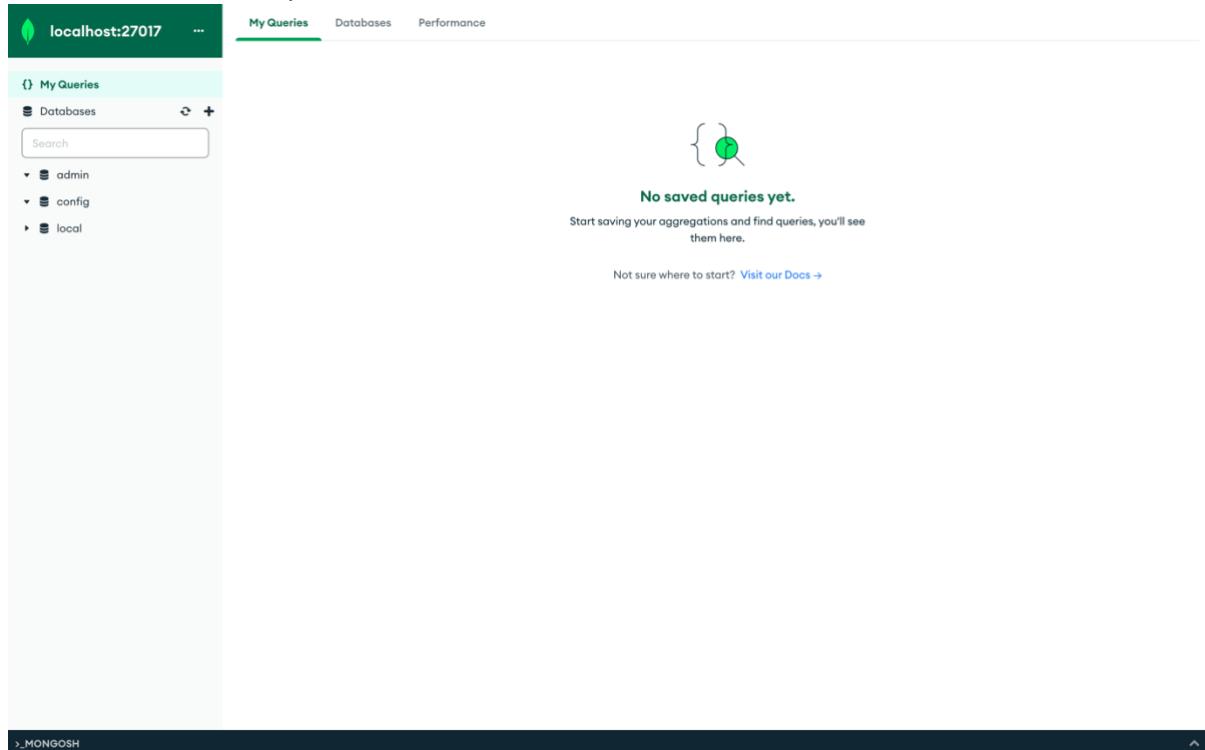
The status bar at the bottom of the pgAdmin window indicates: 'Successfully run. Total query runtime: 108 msec. 1 rows affected.' and 'Ln 3, Col 44'.

Figure 34 Retrieving a Tracklog nearest to Bondi Beach

### 3. Document NoSQL database (MongoDB)

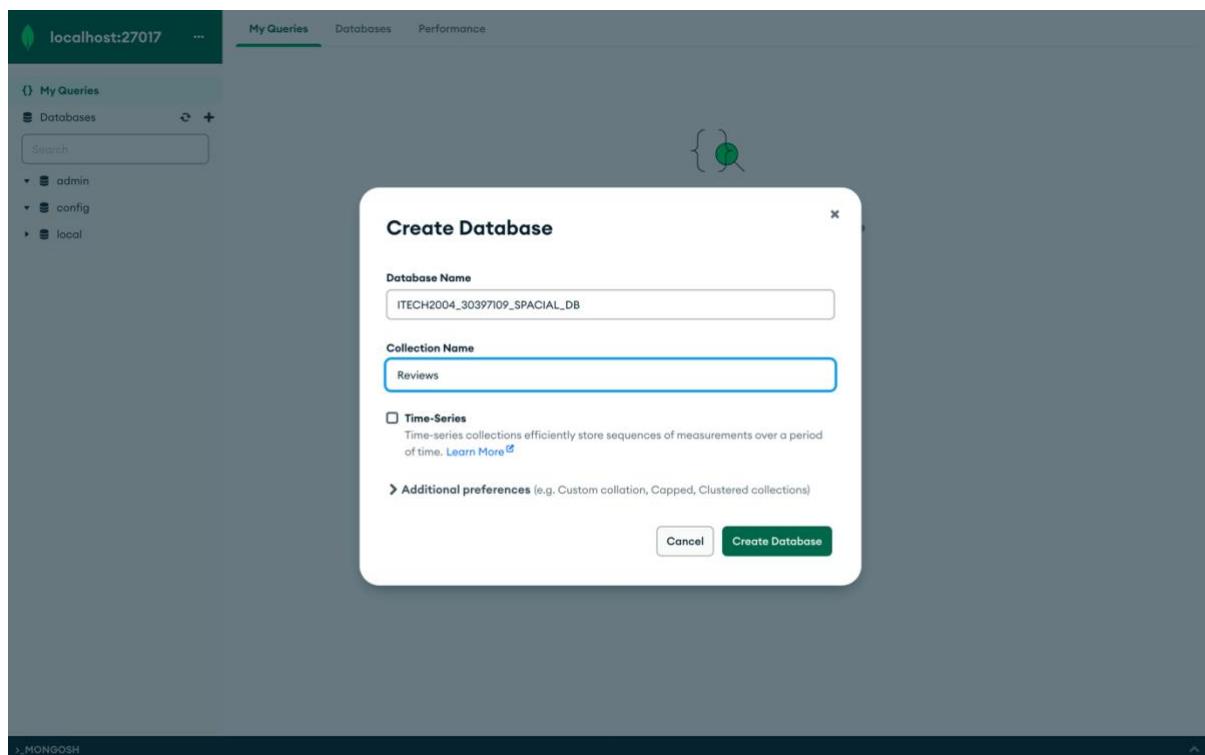
#### 3.1 Database Setup, Collection, and Document

##### 3.1.1 Database Setup and Collection



The screenshot shows the MONGOSH interface with the 'My Queries' tab selected. On the left, there's a sidebar with 'My Queries' and 'Databases' sections. Under 'Databases', 'admin', 'config', and 'local' are listed. A search bar is also present. The main area displays a green character icon and the message 'No saved queries yet.' Below it, a note says 'Start saving your aggregations and find queries, you'll see them here.' At the bottom, a link reads 'Not sure where to start? Visit our Docs →'.

Figure 35 Database Setup and Collection (1)



The screenshot shows the MONGOSH interface with the 'My Queries' tab selected. A 'Create Database' dialog box is open in the foreground. It has fields for 'Database Name' (set to 'ITECH2004\_30397109\_SPACIAL\_DB') and 'Collection Name' (set to 'Reviews'). There's a checkbox for 'Time-Series' which is unchecked. A note below it says 'Time-series collections efficiently store sequences of measurements over a period of time. Learn More'. A link 'Additional preferences (e.g. Custom collation, Capped, Clustered collections)' is also visible. At the bottom of the dialog are 'Cancel' and 'Create Database' buttons.

Figure 36 Database Setup and Collection (2)

localhost:27017

ITECH2004\_30397109\_SPACIAL\_DB.Reviews

0 DOCUMENTS 1 INDEXES

This collection has no data

It only takes a few seconds to import data from a JSON or CSV file.

Import Data

Figure 37 Database Setup and Collection (3)

Welcome to Studio 3T - Full Product Trial

You have explored 2 essential features out of 9. Learn more

Recent Connections

My license

Highlights

Figure 38 Database Setup and Collection (4)

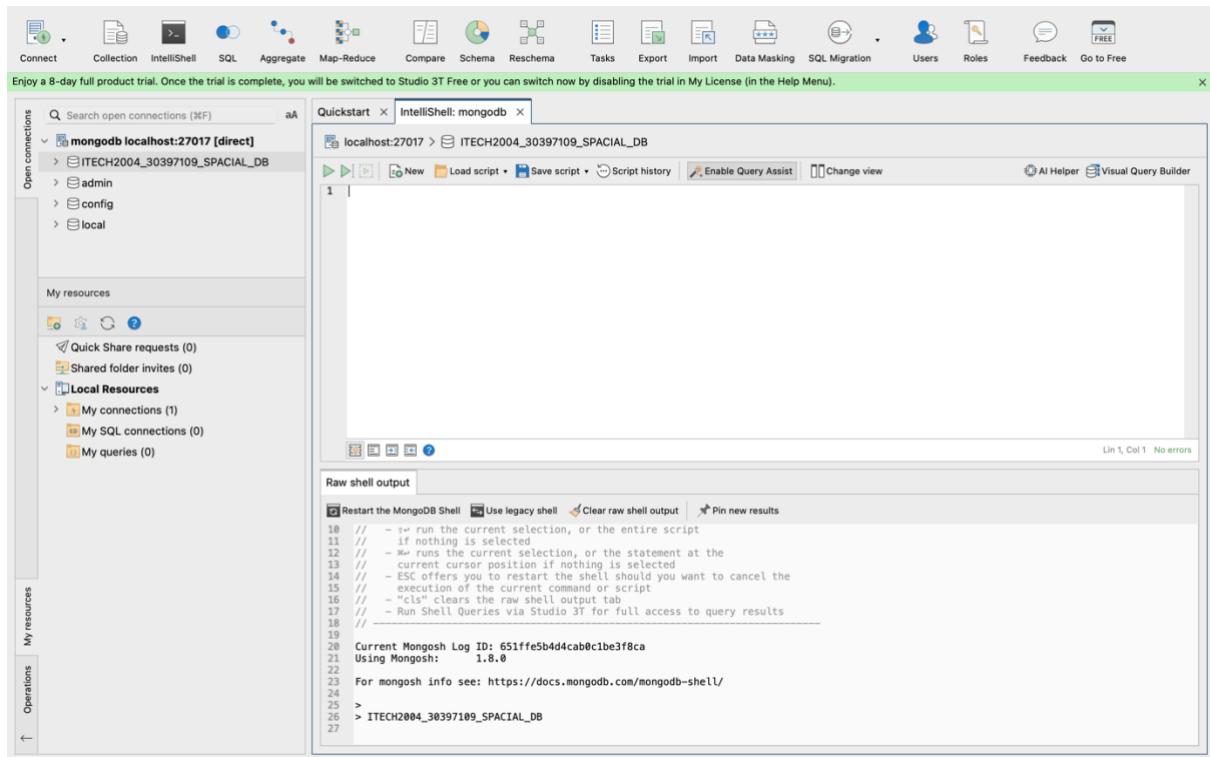


Figure 39 Database Setup and Collection (5)

### 3.1.2 Documents (Reviews) of Initial 5 people

#### For John Doe

```
db.Reviews.insert({
```

```
"Trip name/ID": "Taronga Zoo Sydney",
```

```
"Reviewer name/ID": "John Doe",
```

```
"Rating": 4,
```

```
"Review": "Taronga Zoo Sydney is a fantastic wildlife experience. Highly recommended!"
```

```
}
```

```
)
```

```
show collections
```

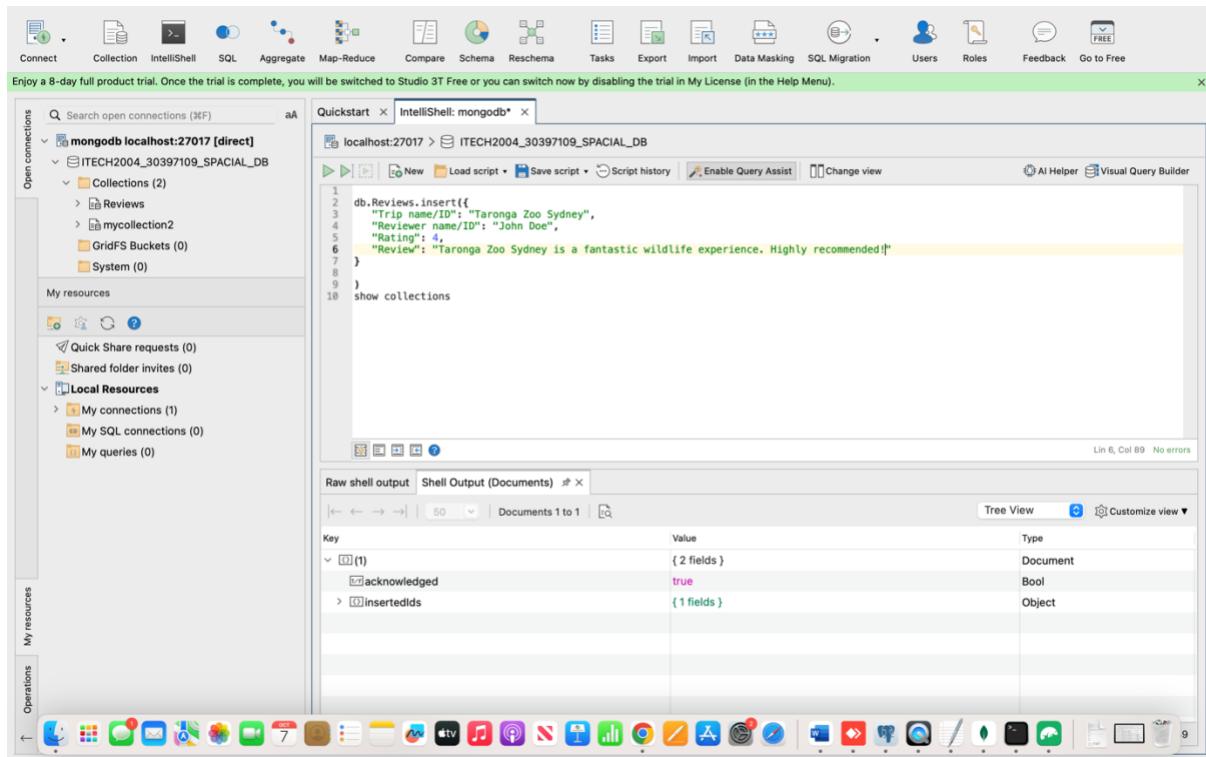


Figure 40 Documents (Reviews) of John Doe

## For Bishow Budhamagar

```
db.Reviews.insert({
  "Trip name/ID": "Sydney Opera House",
  "Reviewer name/ID": "Bishow Budhamagar",
  "Rating": 9,
  "Review": "Sydney Opera House is a remarkable architectural marvel. I was in awe
  of its grandeur."
})
```

show collections

```

db.Reviews.insert({
  "Trip name/ID": "Sydney Opera House",
  "Reviewer name/ID": "Bishow Budhamagar",
  "Rating": 9,
  "Review": "sydney Opera House is a remarkable architectural marvel. I was in awe of its grandeur."
})
show collections

```

```

{
  "acknowledged": true,
  "insertedIds": {
    "_id": ObjectId("652015cec04b27024363cd23")
  }
}
1 document selected

```

Figure 41 Documents (Reviews) of Bishow Budhamagar

## For Michael Johnson

```

db.Reviews.insert({
  "Trip name/ID": "South Cronulla Beach",
  "Reviewer name/ID": "Michael Johnson",
  "Rating": 6,
  "Review": "South Cronulla Beach is a lovely destination with a great atmosphere.
Enjoyed my time there."
}
)
show collections

```

The screenshot shows the MongoDB Studio 3T interface. The left sidebar displays 'Open connections' and 'My resources'. The main area has tabs for 'Quickstart' and 'IntelliShell: mongodbs\*'. In the 'IntelliShell' tab, a query is run against the 'Reviews' collection in the 'ITECH2004\_30397109\_SPACIAL\_DB' database. The query inserts a document with fields: 'Trip name/ID', 'Reviewer name/ID', 'Rating', and 'Review'. The 'Raw shell output' tab shows the inserted document in JSON format, which includes an '\_id' field generated by MongoDB.

```

db.Reviews.insert({
  "Trip name/ID": "South Cronulla Beach",
  "Reviewer name/ID": "Michael Johnson",
  "Rating": 6,
  "Review": "South Cronulla Beach is a lovely destination with a great atmosphere. Enjoyed my time there."
})
show collections

```

```

{
  "_id": ObjectId("65201805c04b27024363cd24"),
  "acknowledged": true,
  "insertedId": {
    "_id": ObjectId("65201805c04b27024363cd24")
  }
}

```

Figure 42 Documents (Reviews) of Michael Johnson

## For Saurav Acharya

```

db.Reviews.insert({
  "Trip name/ID": "Sea Cliff Bridge",
  "Reviewer name/ID": "Saurav Acharya",
  "Rating": 6,
  "Review": "Sea Cliff Bridge offers breathtaking coastal views. An unforgettable experience!"
}

)

```

show collections

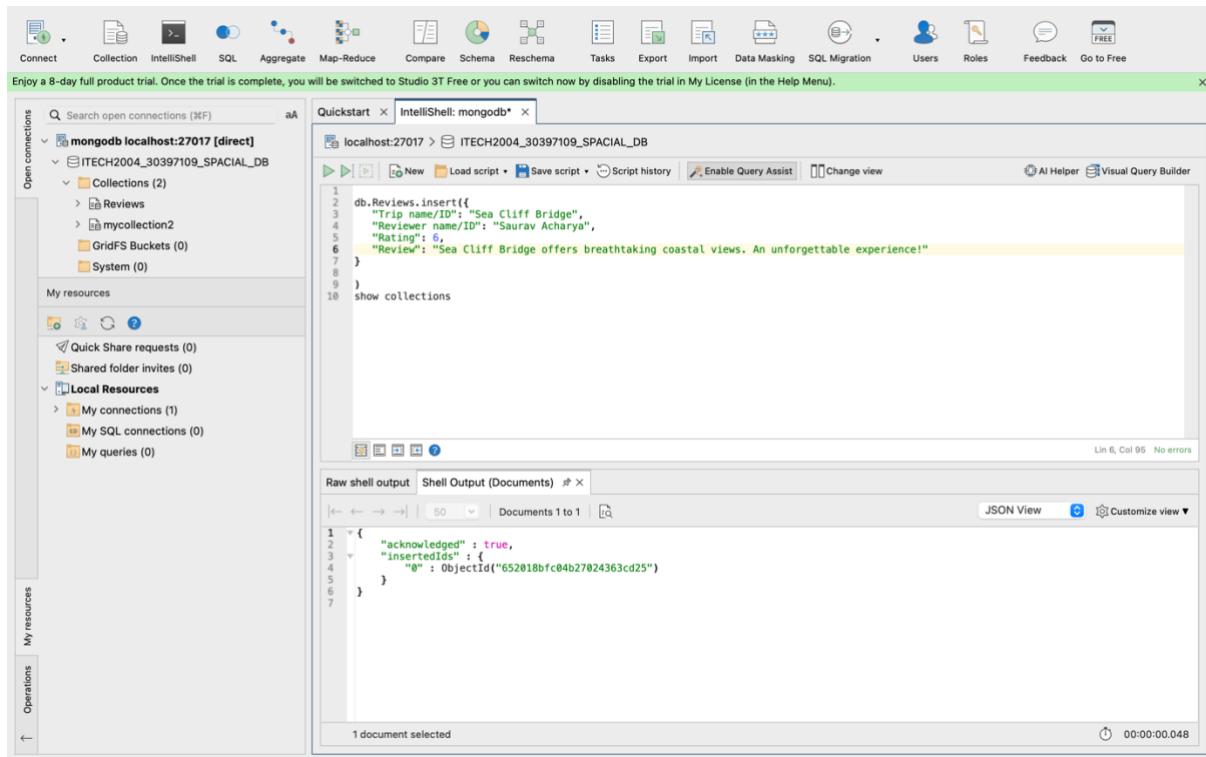


Figure 43 Documents (Reviews) of Saurav Acharya

## For David Brown

```

db.Reviews.insert({
  "Trip name/ID": "Bondi Beach",
  "Reviewer name/ID": "David Brown",
  "Rating": 8,
  "Review": "Bondi Beach is a beautiful place to visit. Loved the experience!"
}

)

```

**show collections**

The screenshot shows the Studio 3T interface with the following details:

- Top Bar:** Connect, Collection, IntelliShell, SQL, Aggregate, Map-Reduce, Compare, Schema, Reschema, Tasks, Export, Import, Data Masking, SQL Migration, Users, Roles, Feedback, Go to Free.
- Message Bar:** Enjoy a 8-day full product trial. Once the trial is complete, you will be switched to Studio 3T Free or you can switch now by disabling the trial in My License (in the Help Menu).
- Left Sidebar:**
  - Open connections: mongodb localhost:27017 [direct]
  - ITECH2004\_30397109\_SPACIAL\_DB (selected)
    - Collections (2): Reviews, mycollection2
    - GridFS Buckets (0)
    - System (0)
  - My resources: Quick Share requests (0), Shared folder invites (0), Local Resources (My connections (1), My SQL connections (0), My queries (0))
  - Operations: (refresh, help)
- IntelliShell:** mongoDB\* (localhost:27017 > ITECH2004\_30397109\_SPACIAL\_DB)
  - Script history: db.Reviews.insert({ "Trip name/ID": "Bondi Beach", "Reviewer name/ID": "David Brown", "Rating": 8, "Review": "Bondi Beach is a beautiful place to visit. Loved the experience!"})
  - Commands: show collections
- Raw shell output:**

Key	Value	Type
✓ (1)	{ 2 fields }	Document
✓ acknowledged	true	Bool
✓ insertedIds	{ 1 fields }	Object

0 items selected | 00:00:00.041

Figure 44 Documents (Reviews) of David Brown

### 3.2 Inserting new review into database For Emily Wilson

```
db.reviews.insert({
  "Trip name/ID": "Taronga Zoo Sydney",
  "Reviewer name/ID": "Emily Wilson",
  "Rating": 5,
  "Review": "This was an amazing experience!"
})
```

```
show collections
```

```

db.reviews.insert({
  "Trip name/ID": "Taronga Zoo Sydney",
  "Reviewer name/ID": "Emily Wilson",
  "Rating": 5,
  "Review": "This was an amazing experience!"
})
show collections

```

Raw shell output

Key	Value	Type
✓ {1}	{ 2 fields }	Document
✓ acknowledged	true	Bool
✓ insertedIds	{ 1 fields }	Object

Figure 45 Inserting new review into database of Emily Wilson

## For Benjamin Davis

```
db.reviews.insert({
```

"Trip name/ID": "Sydney Opera House",

"Reviewer name/ID": "Benjamin Davis",

"Rating": 5,

"Review": "This was an amazing experience!"

)

show collections

```

db.reviews.insert({
  "Trip name/ID": "Sydney Opera House",
  "Reviewer name/ID": "Benjamin Davis",
  "Rating": 5,
  "Review": "This was an amazing experience!"
})
show collections

```

Raw shell output

Key	Value	Type
✓ {1}	{ 2 fields }	Document
✓ acknowledged	true	Bool
✓ insertedIds	{ 1 fields }	Object

Figure 46 Inserting new review into database of Benjamin Davis

## For Chloe Harris

```

db.reviews.insert({
  "Trip name/ID": "South Cronulla Beach",
  "Reviewer name/ID": "Chloe Harris",
  "Rating": 8,
  "Review": "This was an amazing experience!"
})

```

```
show collections
```

The screenshot shows the Studio 3T interface with the MongoDB shell open. The left sidebar shows connections and resources. The main area has a script editor with the following code:

```

db.reviews.insert({
  "Trip name/ID": "South Cronulla Beach",
  "Reviewer name/ID": "Chloe Harris",
  "Rating": 8,
  "Review": "This was an amazing experience!"
})
show collections

```

The script output pane shows the inserted document details:

Key	Value	Type
acknowledged	true	Bool
insertedId	{ 1 fields }	Object

Figure 47 Inserting new review into database of Chloe Harris

## For James Smith

```

db.reviews.insert({
  "Trip name/ID": "Sea Cliff Bridge",
  "Reviewer name/ID": "James Smith",
  "Rating": 7,
  "Review": "This was an amazing experience!"
})

```

**show collections**

```

db.reviews.insert({
  "Trip name/ID": "Sea Cliff Bridge",
  "Reviewer name/ID": "James Smith",
  "Rating": 7,
  "Review": "This was an amazing experience!"
})

```

Key	Value	Type
acknowledged	true	Bool
insertedId	{ 1 fields }	Object

Figure 48 Inserting new review into database of James Smith

## For Sophia Jackson

```

db.reviews.insert({
  "Trip name/ID": "Bondi Beach",
  "Reviewer name/ID": "Sophia Jackson",
  "Rating": 6,
  "Review": "This was an amazing experience!"
})

```

show collections

```

db.reviews.insert({
  "Trip name/ID": "Bondi Beach",
  "Reviewer name/ID": "Sophia Jackson",
  "Rating": 6,
  "Review": "This was an amazing experience!"
})
show collections

```

Key	Type
1 (1)	Document
acknowledged	Bool
insertedId	Object

Figure 49 Inserting new review into database of Sophia Jackson

### 3.3 Retrieving highest rated review For Taronga Zoo Sydney

```
db.reviews.find({ "Trip name/ID": "Taronga Zoo Sydney" }).sort({ "Rating": -1 }).limit(1)
```

```

db.reviews.find({ "Trip name/ID": "Taronga Zoo Sydney" }).sort({ "Rating": -1 }).limit(1)

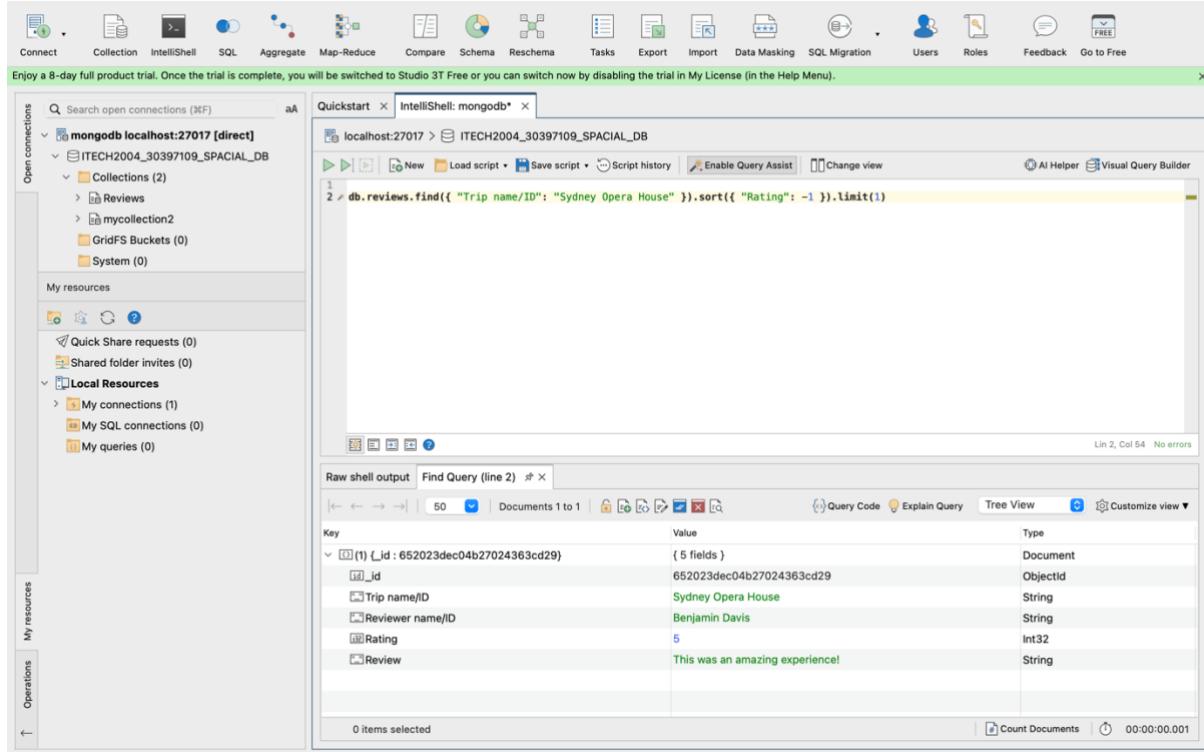
```

Key	Type
1 (1) _id : 65202399c04b27024363cd28	Document
_id	Objectid
Trip name/ID	String
Reviewer name/ID	String
Rating	Int32
Review	String

Figure 50 Retrieving highest rated review for Taronga Zoo Sydney

## For Sydney Opera House

```
db.reviews.find({ "Trip name/ID": "Sydney Opera House" }).sort({ "Rating": -1 }).limit(1)
```



The screenshot shows the Studio 3T interface. The left sidebar displays 'Open connections' with a connection to 'mongodb localhost:27017 [direct]' and 'My resources' section. The main area shows the 'IntelliShell: mongod\*' tab with the query:

```
1 db.reviews.find({ "Trip name/ID": "Sydney Opera House" }).sort({ "Rating": -1 }).limit(1)
```

The results pane below shows the output of the query:

Key	Type
✓ (1) _id : 652023dec04b27024363cd29	Document
✓ _id	ObjectId
✓ Trip name/ID	String
✓ Reviewer name/ID	String
✓ Rating	Int32
✓ Review	String

The result row contains one document with the following details:

- \_id: 652023dec04b27024363cd29
- Trip name/ID: Sydney Opera House
- Reviewer name/ID: Benjamin Davis
- Rating: 5
- Review: This was an amazing experience!

Figure 51 Retrieving highest rated review for Sydney Opera House

## For South Cronulla Beach

```
db.reviews.find({ "Trip name/ID": "South Cronulla Beach" }).sort({ "Rating": -1 }).limit(1)
```

The screenshot shows the MongoDB Studio 3T interface. The left sidebar displays 'Open connections' and 'My resources'. The main area shows a connection to 'mongodb localhost:27017 [direct]' with a database 'ITECH2004\_30397109\_SPACIAL\_DB'. A query is being run in the 'IntelliShell' tab:

```
db.reviews.find({ "Trip name/ID": "South Cronulla Beach" }).sort({ "Rating": -1 }).limit(1)
```

The results pane shows one document:

Key	Value	Type
<code>1 { _id : 652023fdc04b27024363cd2a }</code>	{ 5 fields }	Document
<code>1 _id</code>	652023fdc04b27024363cd2a	ObjectId
<code>1 Trip name/ID</code>	South Cronulla Beach	String
<code>1 Reviewer name/ID</code>	Chloe Harris	String
<code>1 Rating</code>	8	Int32
<code>1 Review</code>	This was an amazing experience!	String

Figure 52 Retrieving highest rated review for South Cronulla Beach

## For Sea Cliff Bridge

```
db.reviews.find({ "Trip name/ID": "Sea Cliff Bridge" }).sort({ "Rating": -1 }).limit(1)
```

The screenshot shows the MongoDB Studio 3T interface. The left sidebar displays 'Open connections' and 'My resources'. The main area shows a connection to 'mongodb localhost:27017 [direct]' with a database 'ITECH2004\_30397109\_SPACIAL\_DB'. A query is being run in the 'IntelliShell' tab:

```
db.reviews.find({ "Trip name/ID": "Sea Cliff Bridge" }).sort({ "Rating": -1 }).limit(1)
```

The results pane shows one document:

Key	Value	Type
<code>1 { _id : 65202427c04b27024363cd2b }</code>	{ 5 fields }	Document
<code>1 _id</code>	65202427c04b27024363cd2b	ObjectId
<code>1 Trip name/ID</code>	Sea Cliff Bridge	String
<code>1 Reviewer name/ID</code>	James Smith	String
<code>1 Rating</code>	7	Int32
<code>1 Review</code>	This was an amazing experience!	String

Figure 53 Retrieving highest rated review for Sea Cliff Bridge

## For Bondi Beach

```
db.reviews.find({ "Trip name/ID": "Bondi Beach" }).sort({ "Rating": -1 }).limit(1)
```

The screenshot shows the MongoDB Studio 3T interface. The top navigation bar includes options like Connect, Collection, IntelliShell, SQL, Aggregate, Map-Reduce, Compare, Schema, Reschema, Tasks, Export, Import, Data Masking, SQL Migration, Users, Roles, Feedback, and Go to Free. A trial message at the top states: "Enjoy a 8-day full product trial. Once the trial is complete, you will be switched to Studio 3T Free or you can switch now by disabling the trial in My License (in the Help Menu)." The left sidebar has sections for Open connections, My resources, and Operations. The main area shows an IntelliShell window titled "localhost:27017 > ITECH2004\_30397109\_SPACIAL\_DB". The query entered is: db.reviews.find({ "Trip name/ID": "Bondi Beach" }).sort({ "Rating": -1 }).limit(1). Below the query, the "Raw shell output" pane displays the results in a table:

Key	Type
✓ (1) { _id : 6520244cc04b27024363cd2c }	Document
✓ _id	ObjectId
✓ Trip name/ID	String
✓ Reviewer name/ID	String
✓ Rating	Int32
✓ Review	String

The result shows one document with the following values:

- \_id: 6520244cc04b27024363cd2c
- Trip name/ID: Bondi Beach
- Reviewer name/ID: Sophia Jackson
- Rating: 6
- Review: This was an amazing experience!

Figure 54 Retrieving highest rated review for Bondi Beach

## 4. Graph Database (Neo4j)

### 4.1 Neo4j Database Creation

#### 4.1.1 Assumptions

- Let's assume that the initial data collection process involves trawling social media platforms to identify and gather information about five distinct individuals. These individuals serve as the foundation for our data analysis.
- Let's gather 5 friendships links from 5 Person (1 each) trawled initially to recommend the same retrieved walk.
- In my assumption, Person Nodes are representation of people that are inserted in the table in first database (Postgres).
- Let's Insert 5 NewPerson Nodes and 5 Location Nodes.
- Let's create relationship among them.
- After creation of all the nodes and links, let's retrieve all the friends of each Person

#### 4.1.2 Database Creation Images

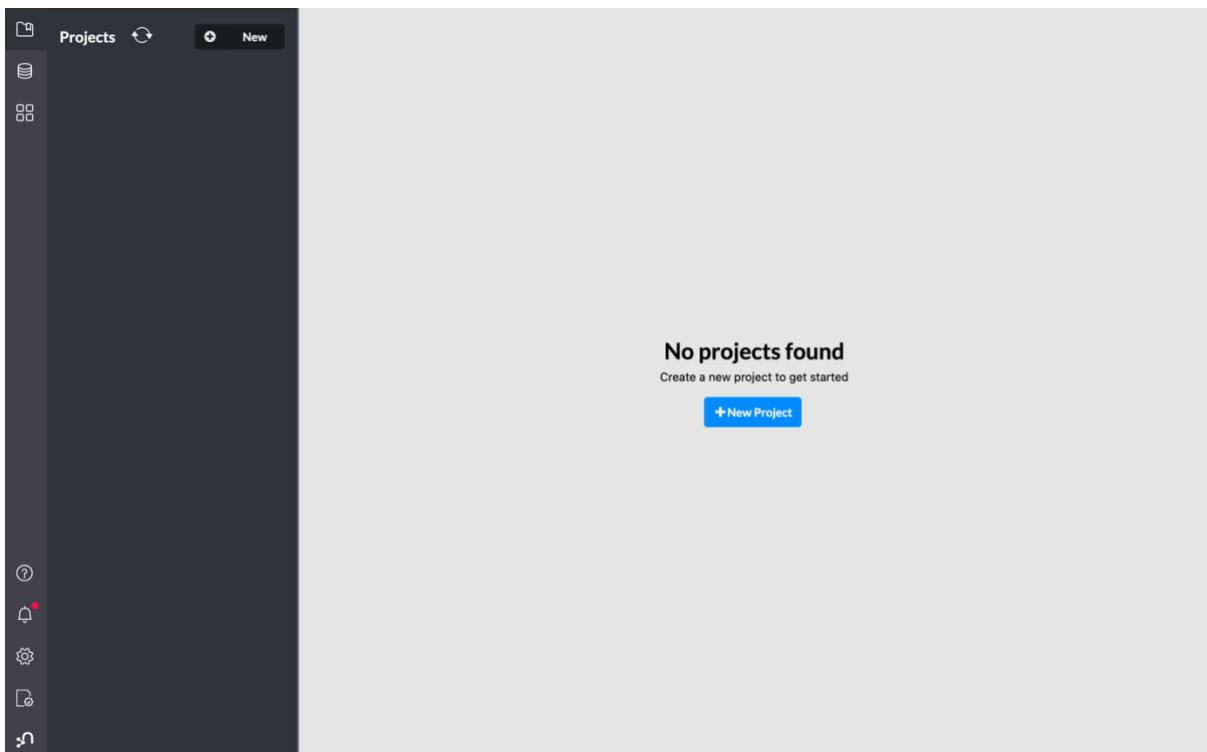


Figure 55 Database Creation Image (1)

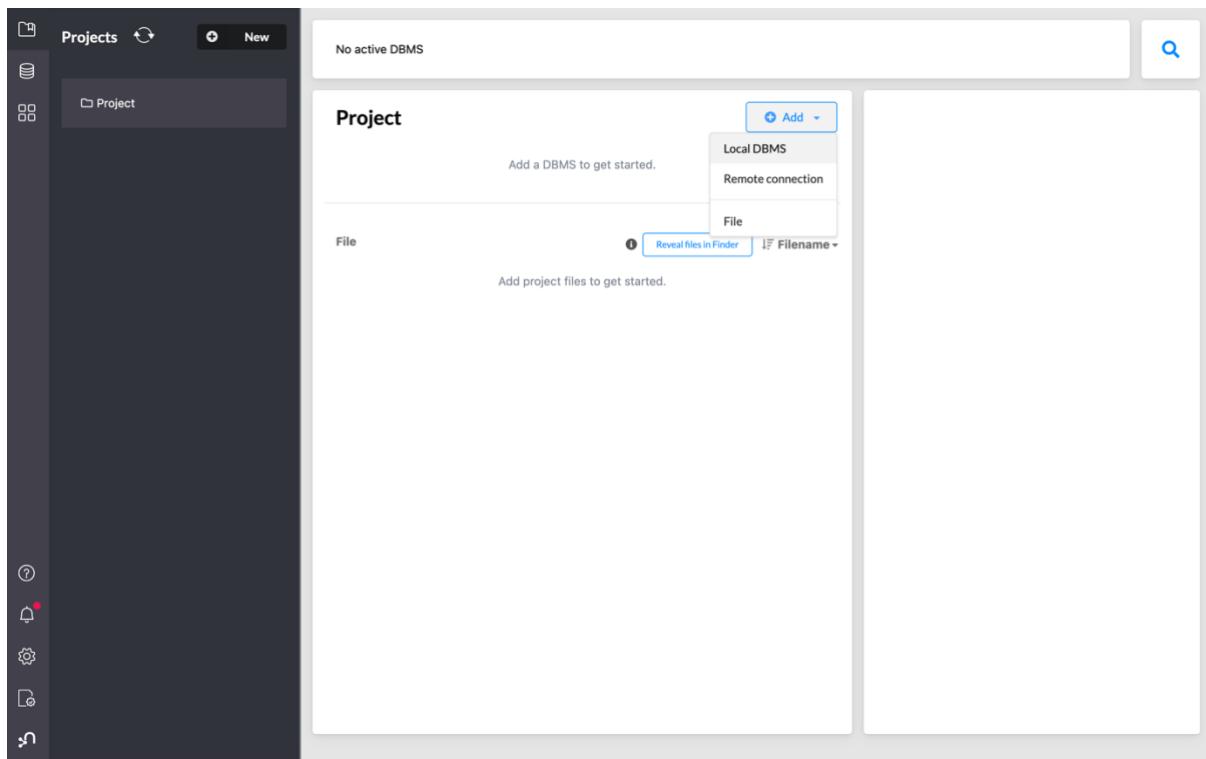


Figure 56 Database Creation Image (2)

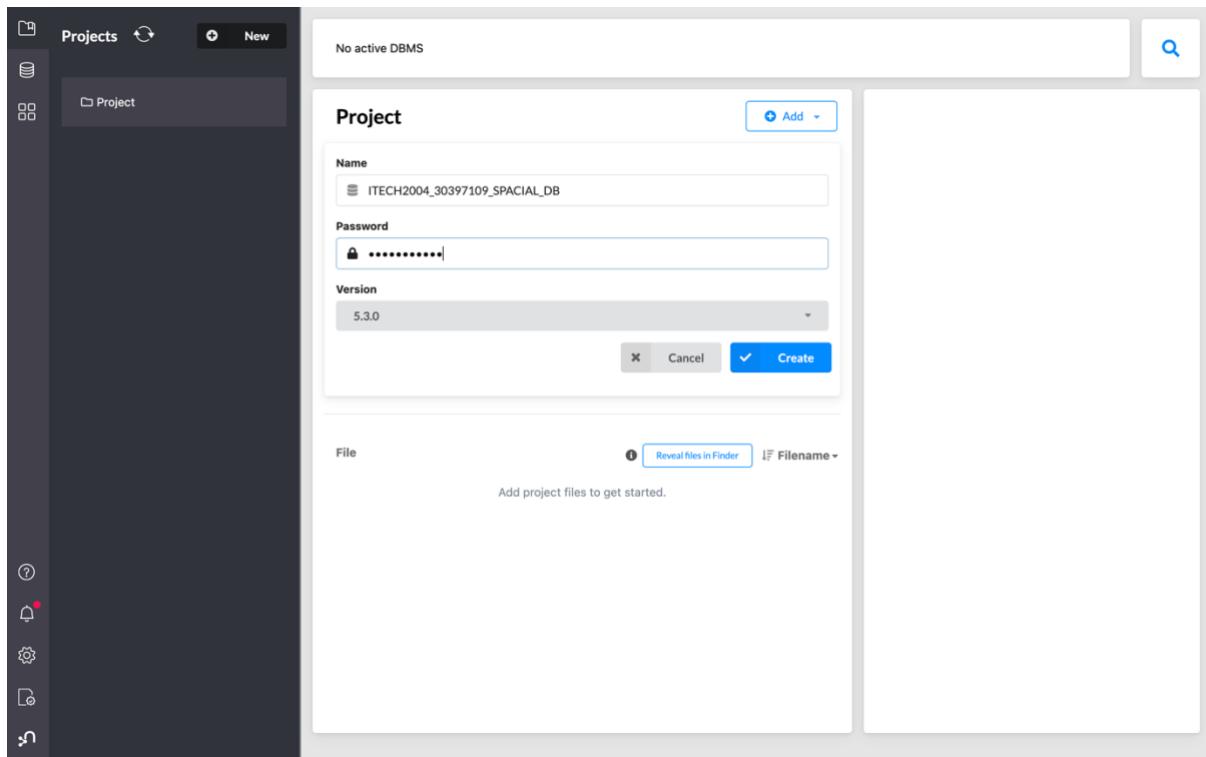


Figure 57 Database Creation Image (3)

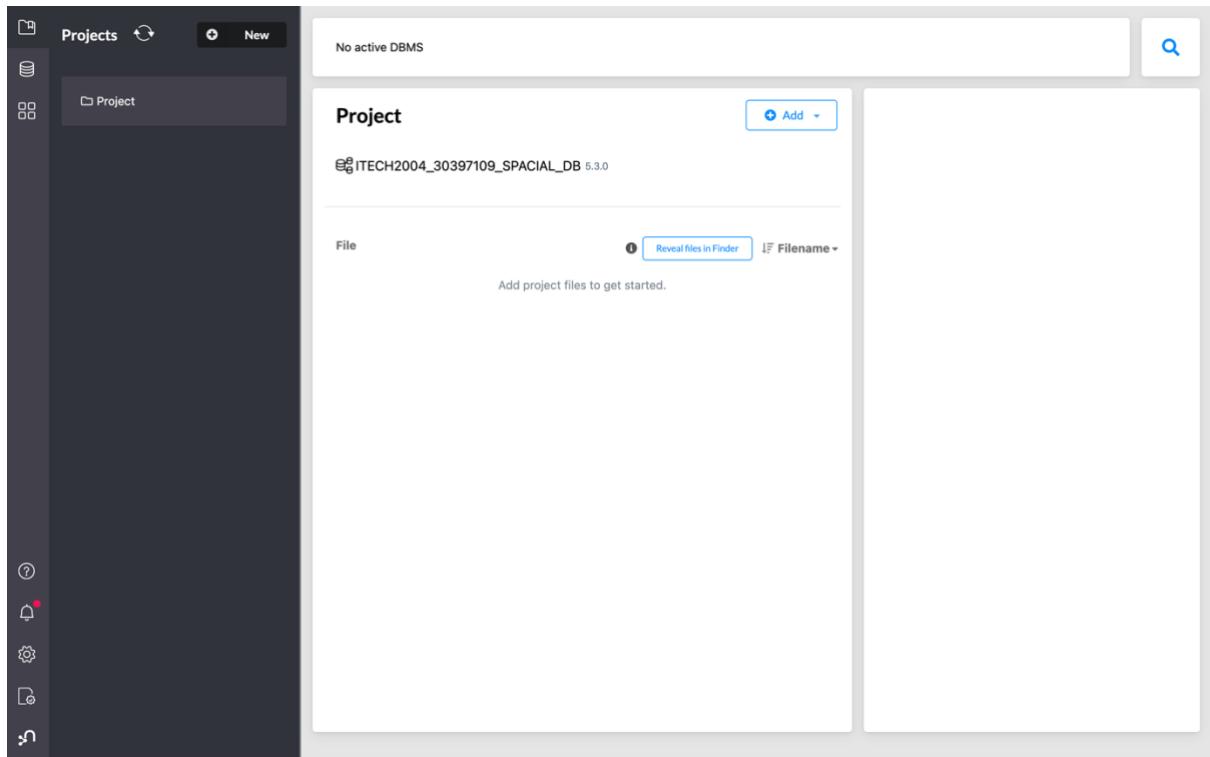


Figure 58 Database Creation Image (4)

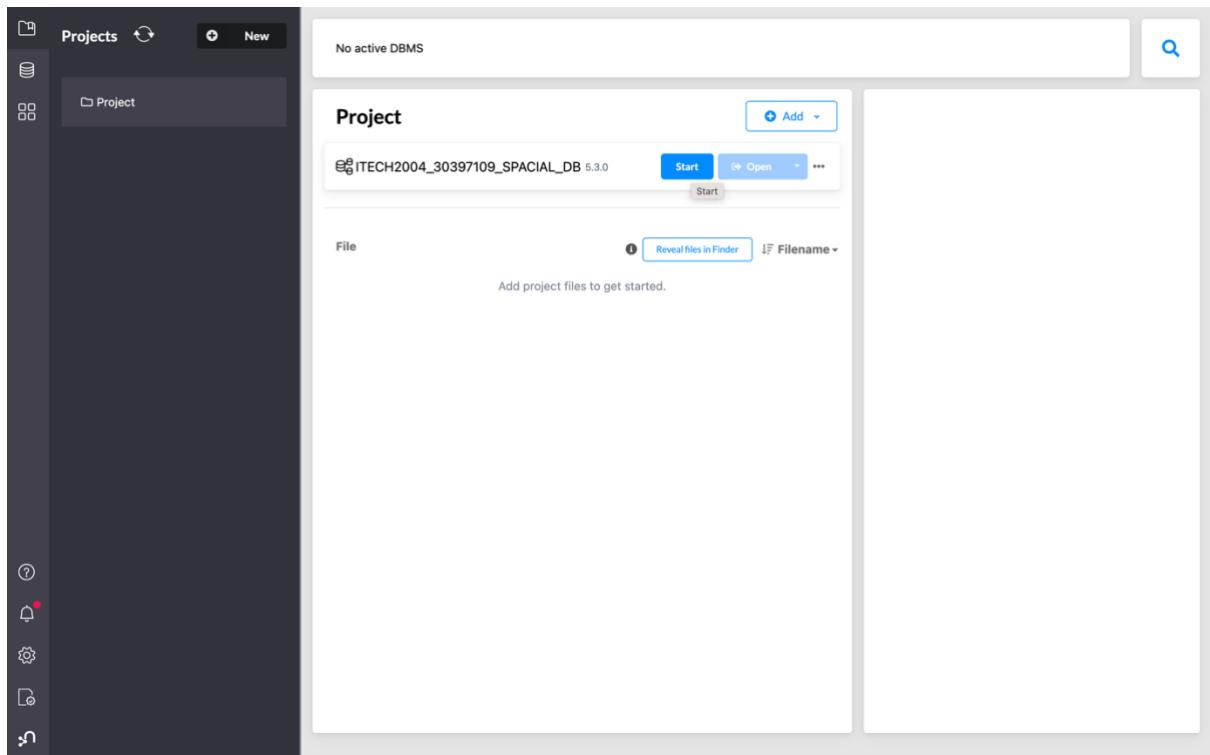


Figure 59 Database Creation Image (5)

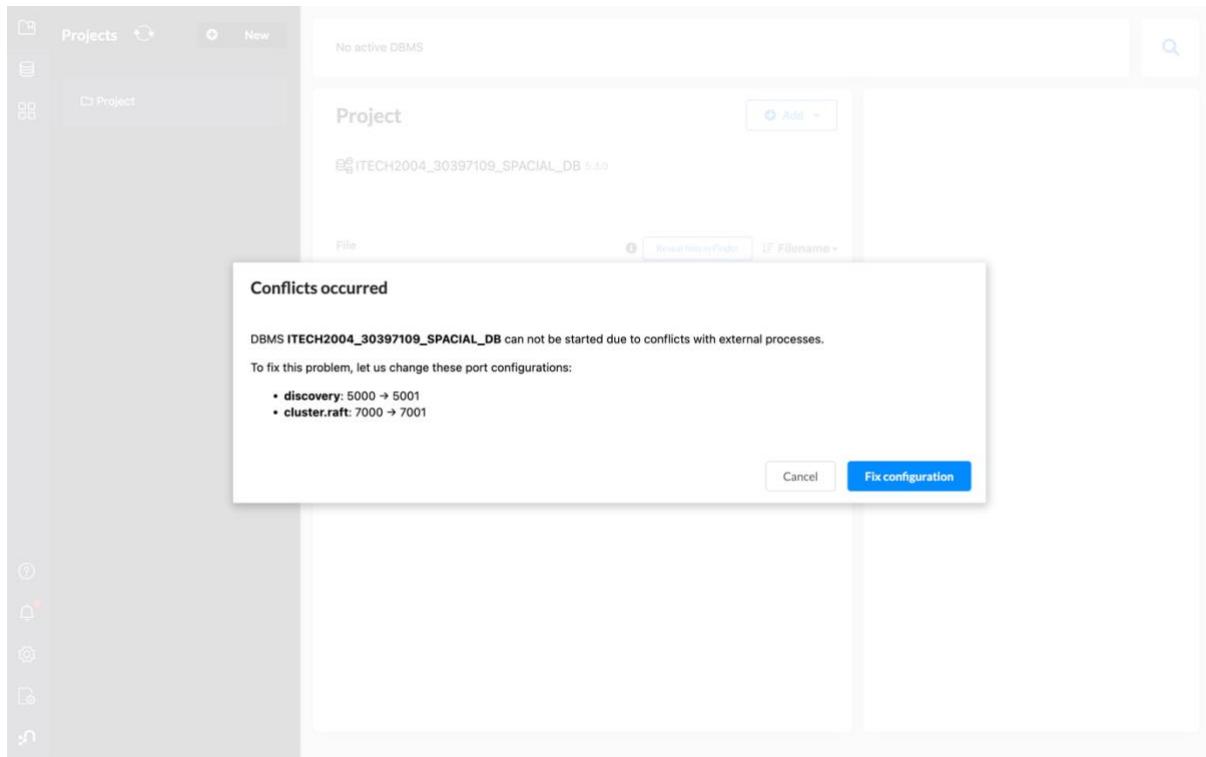


Figure 60 Database Creation Image (6)

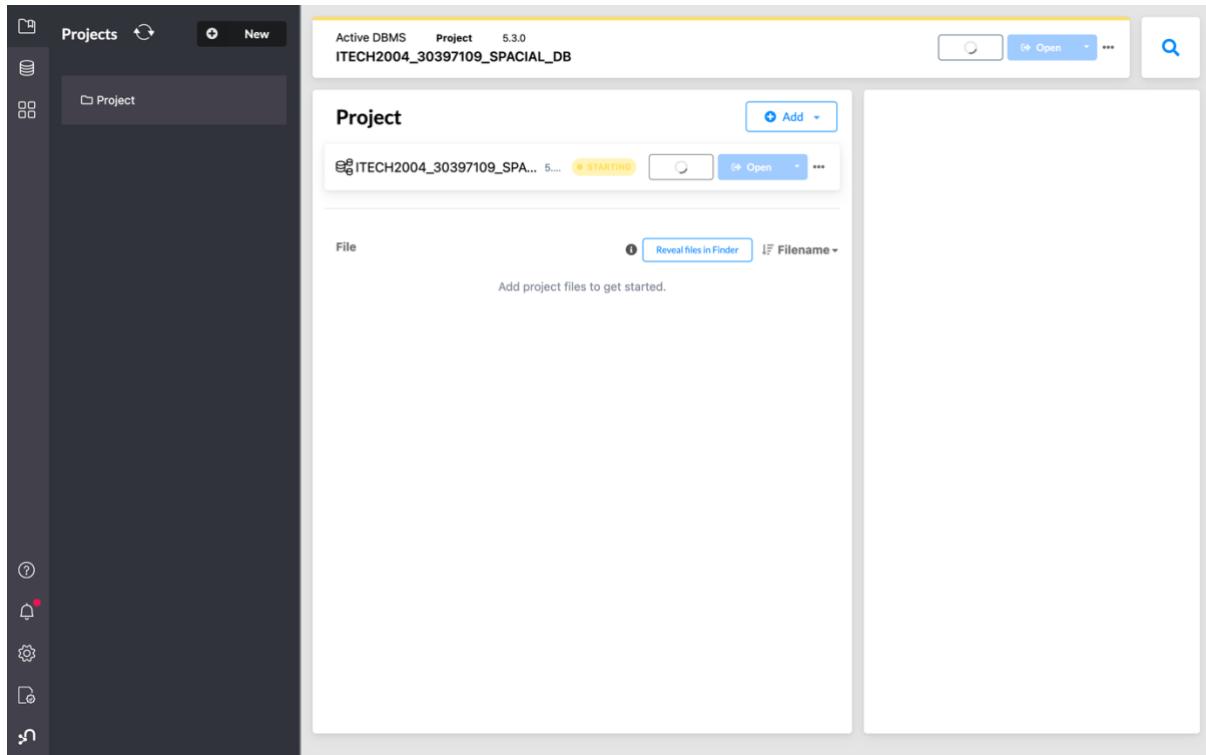


Figure 61 Database Creation Image (7)

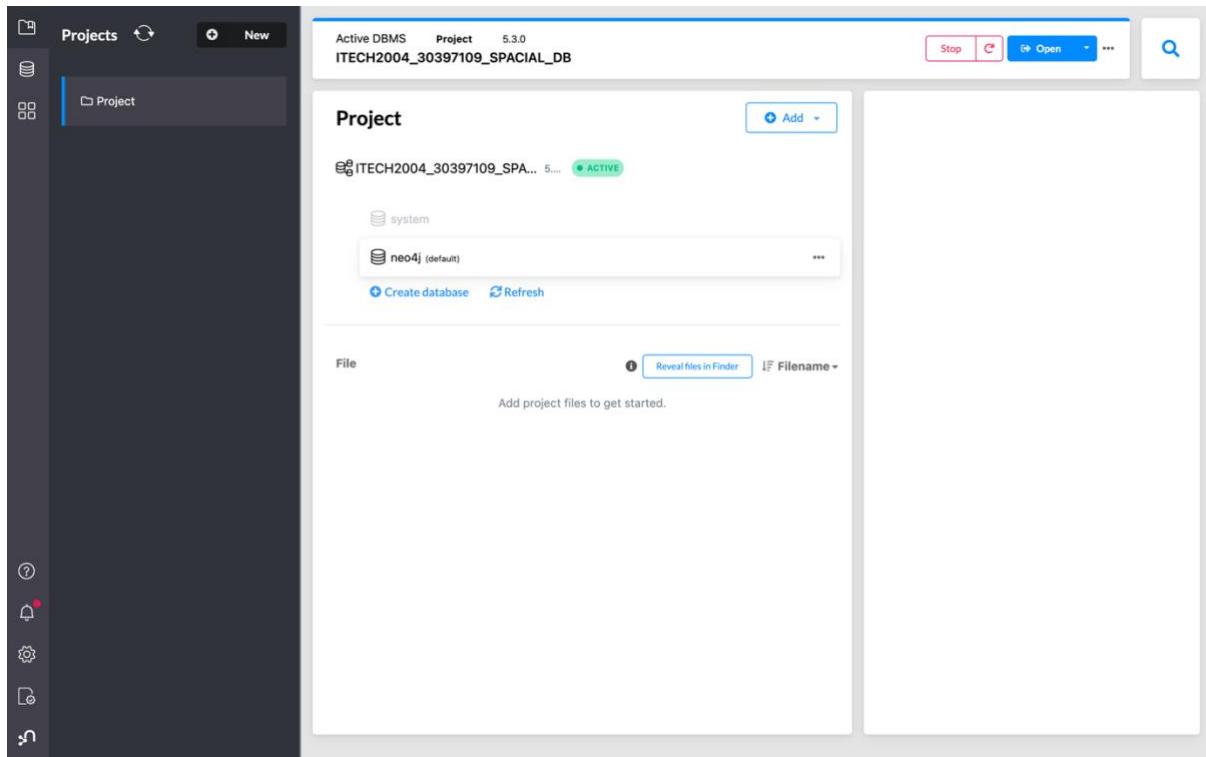


Figure 62 Database Creation Image (8)

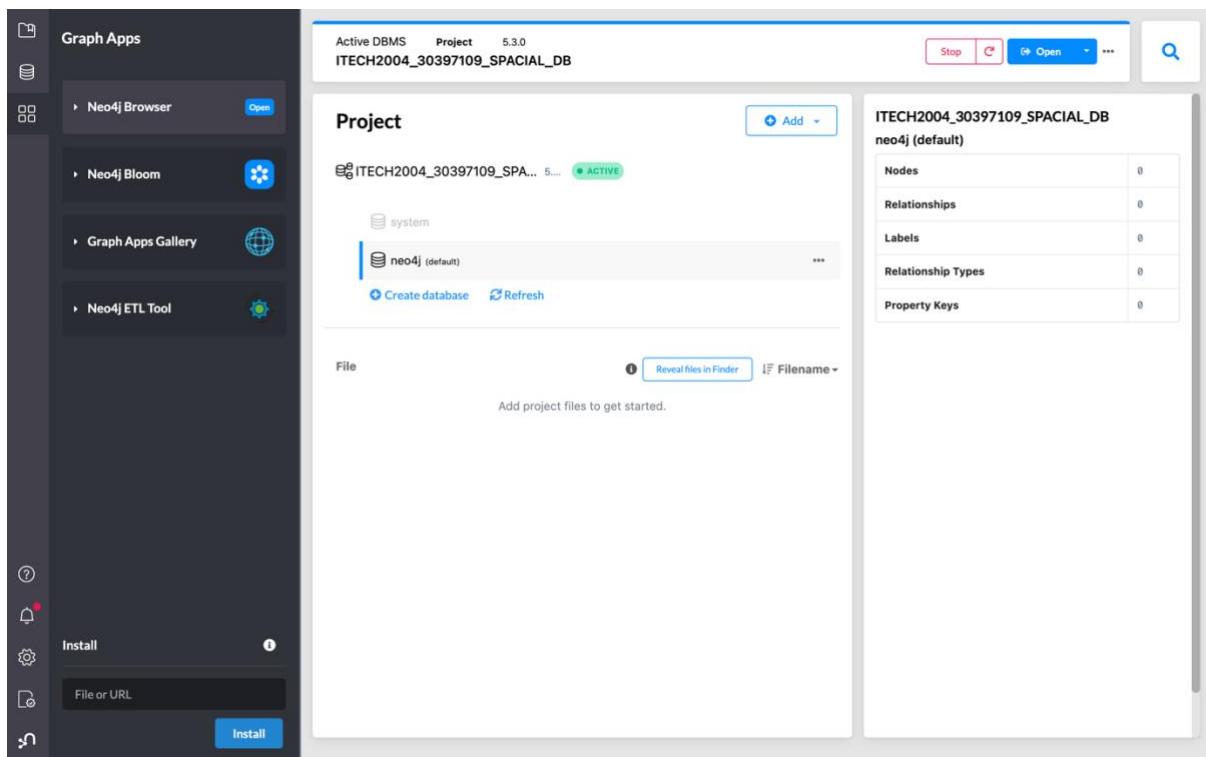


Figure 63 Database Creation Image (9)

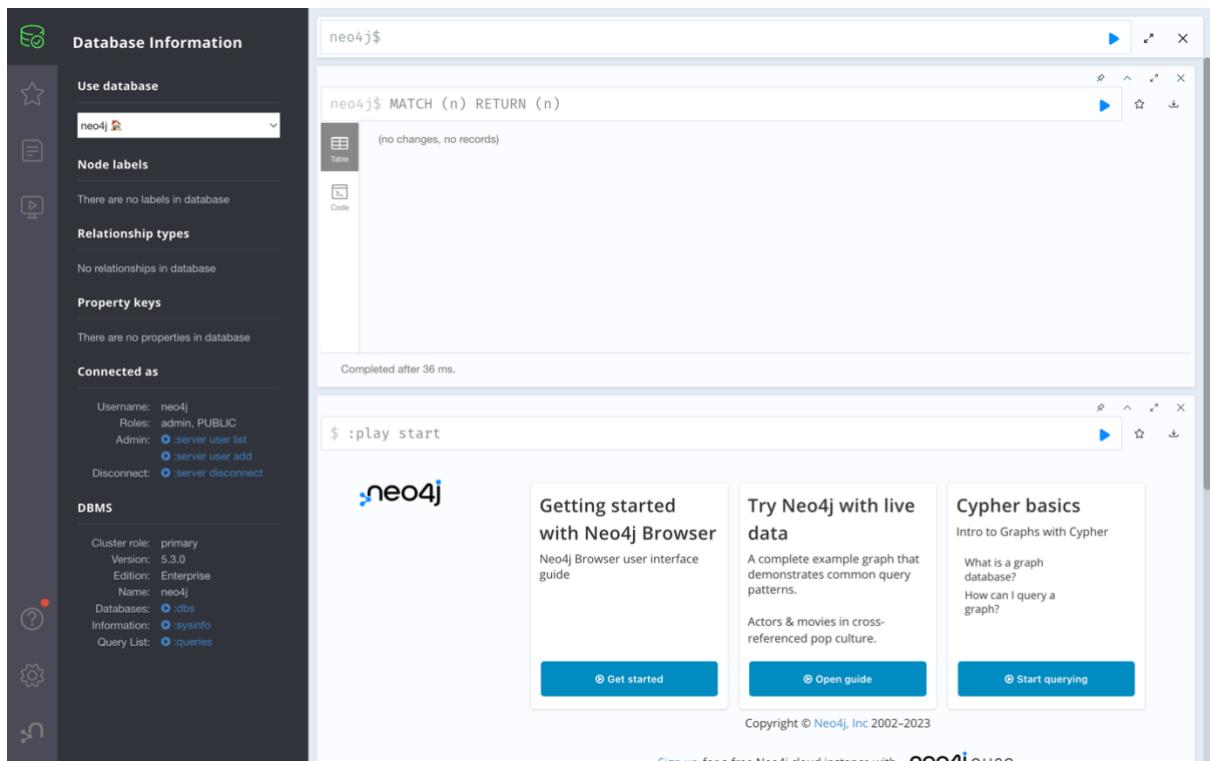


Figure 64 Screenshot to show it is connected to my database ITECH2004\_30397109\_SPACIAL\_DB

## 4.2 Inserting Nodes and Links

### 4.2.1 Inserting Person (Nodes)

```
CREATE (:Person {name: 'John Doe'}), (:Person {name: 'Bishow Budhamagar'}),  
(:Person {name: 'Michael Johnson'}), (:Person {name: 'Saurav Acharya'}), (:Person  
{name: 'David Brown'})
```

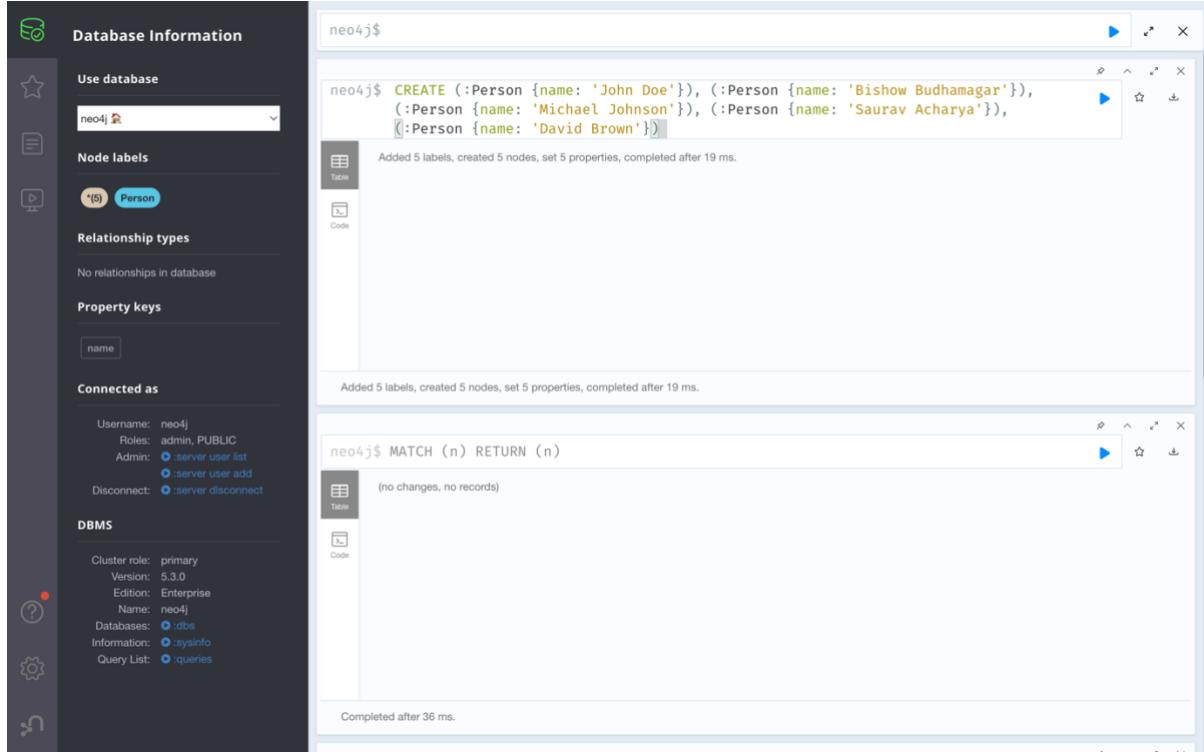


Figure 65 Inserting Person nodes

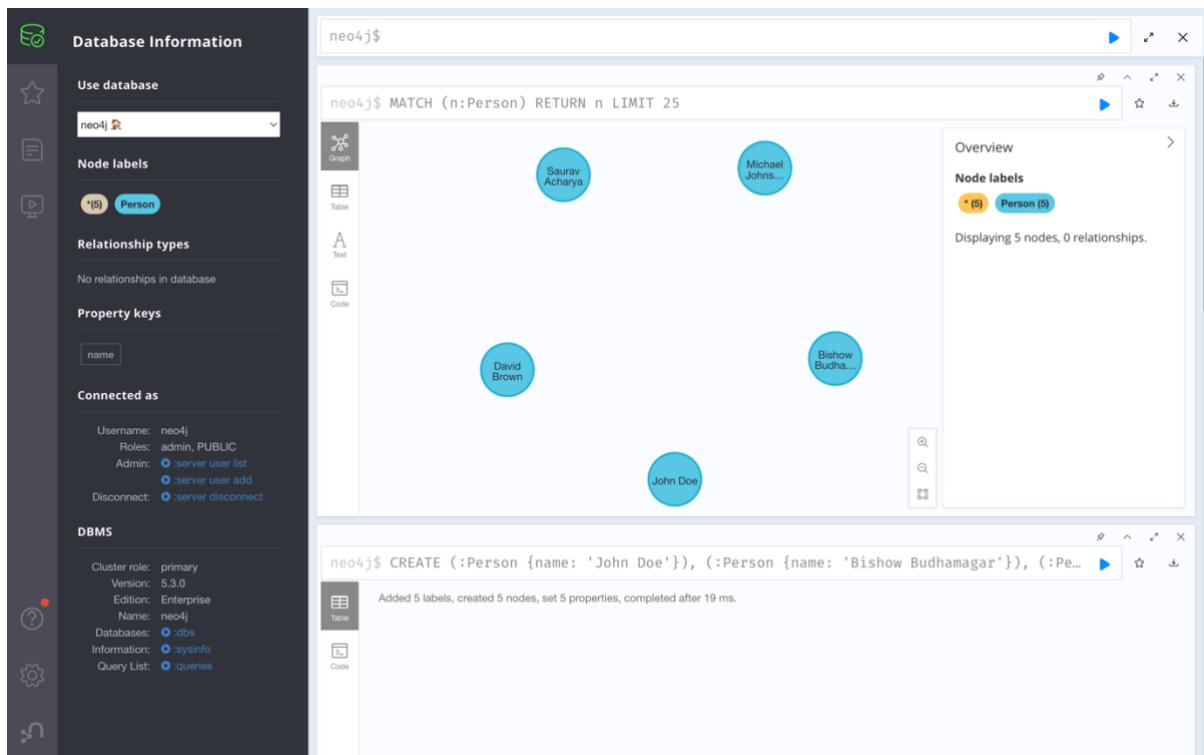


Figure 66 Graph representation of Person Nodes

#### 4.2.2 Inserting NewPerson (Nodes)

```
CREATE (:NewPerson {name: 'Emily Wilson'}), (:NewPerson {name: 'Benjamin Davis'}), (:NewPerson {name: 'Chloe Harris'}), (:NewPerson {name: 'James Smith'}), (:NewPerson {name: 'Sophia Jackson'})
```

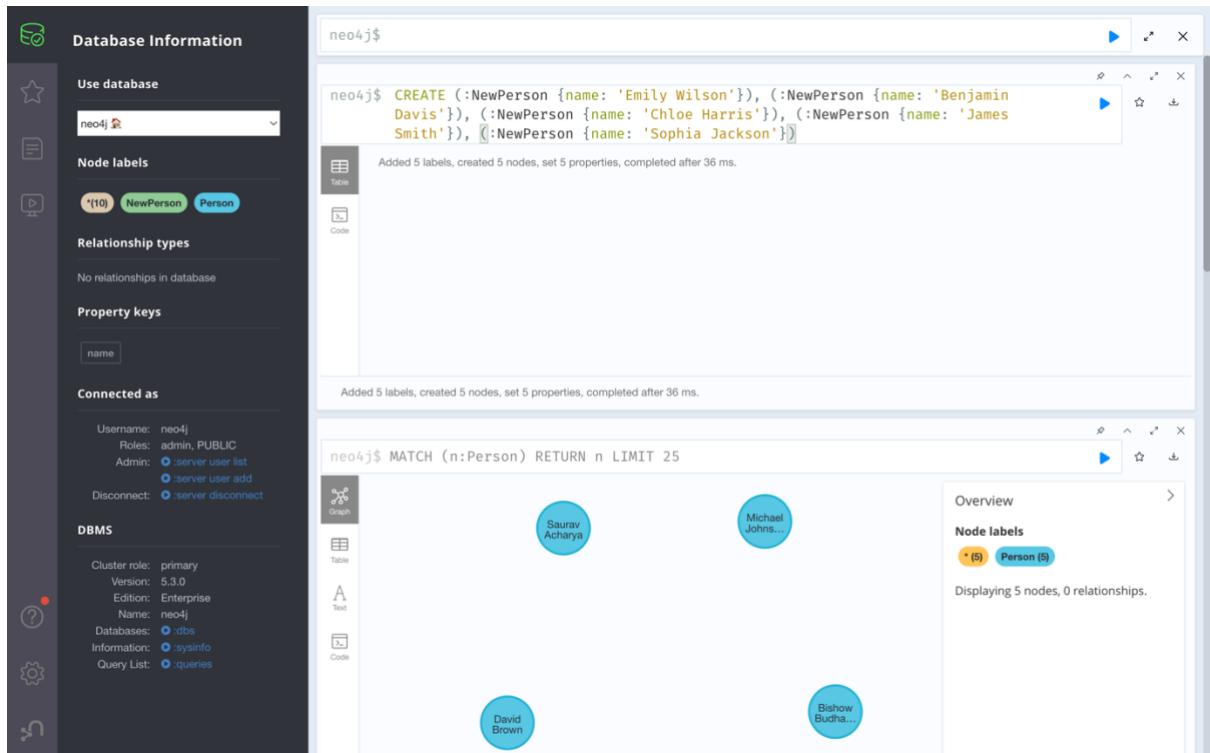


Figure 67 Inserting NewPerson nodes

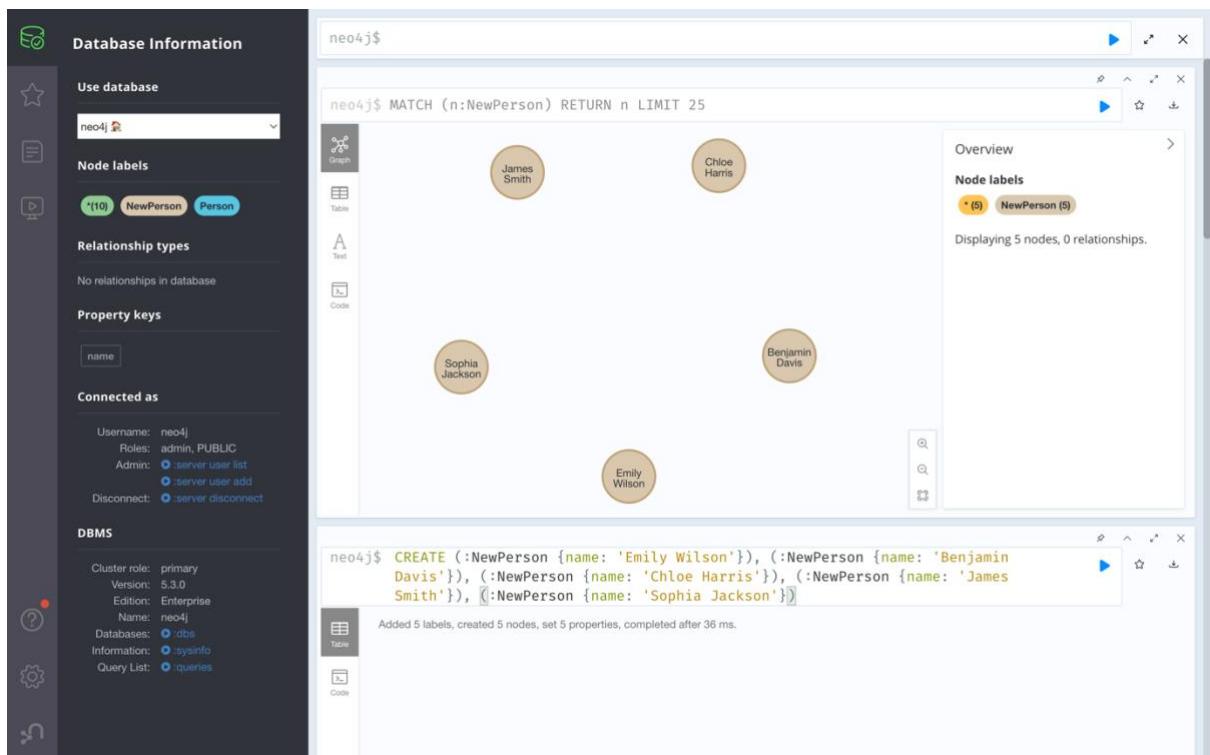


Figure 68 Graph representation of NewPerson Nodes

#### 4.2.3 Inserting links(relationships) between Person and NewPerson nodes

```

MATCH (John:Person {name: 'John Doe'})
MATCH (Bishow:Person {name: 'Bishow Budhamagar'})
MATCH (Michael:Person {name: 'Michael Johnson'})
MATCH (Saurav:Person {name: 'Saurav Acharya'})
MATCH (David:Person {name: 'David Brown'})
MATCH (Emily:NewPerson {name: 'Emily Wilson'})
MATCH (Benjamin:NewPerson {name: 'Benjamin Davis'})
MATCH (Chloe:NewPerson {name: 'Chloe Harris'})
MATCH (James:NewPerson {name: 'James Smith'})
MATCH (Sophia:NewPerson {name: 'Sophia Jackson'})
MERGE (John)-[:FRIEND]->(Bishow)
MERGE (John)-[:FRIEND]->(Michael)
MERGE (Bishow)-[:FRIEND]->(Saurav)
MERGE (Michael)-[:FRIEND]->(Saurav)
MERGE (Michael)-[:FRIEND]->(David)
MERGE (John)-[:FRIEND]->(Emily)
MERGE (Bishow)-[:FRIEND]->(Benjamin)
MERGE (Michael)-[:FRIEND]->(Chloe)
MERGE (Saurav)-[:FRIEND]->(James)
MERGE (David)-[:FRIEND]->(Sophia)

```

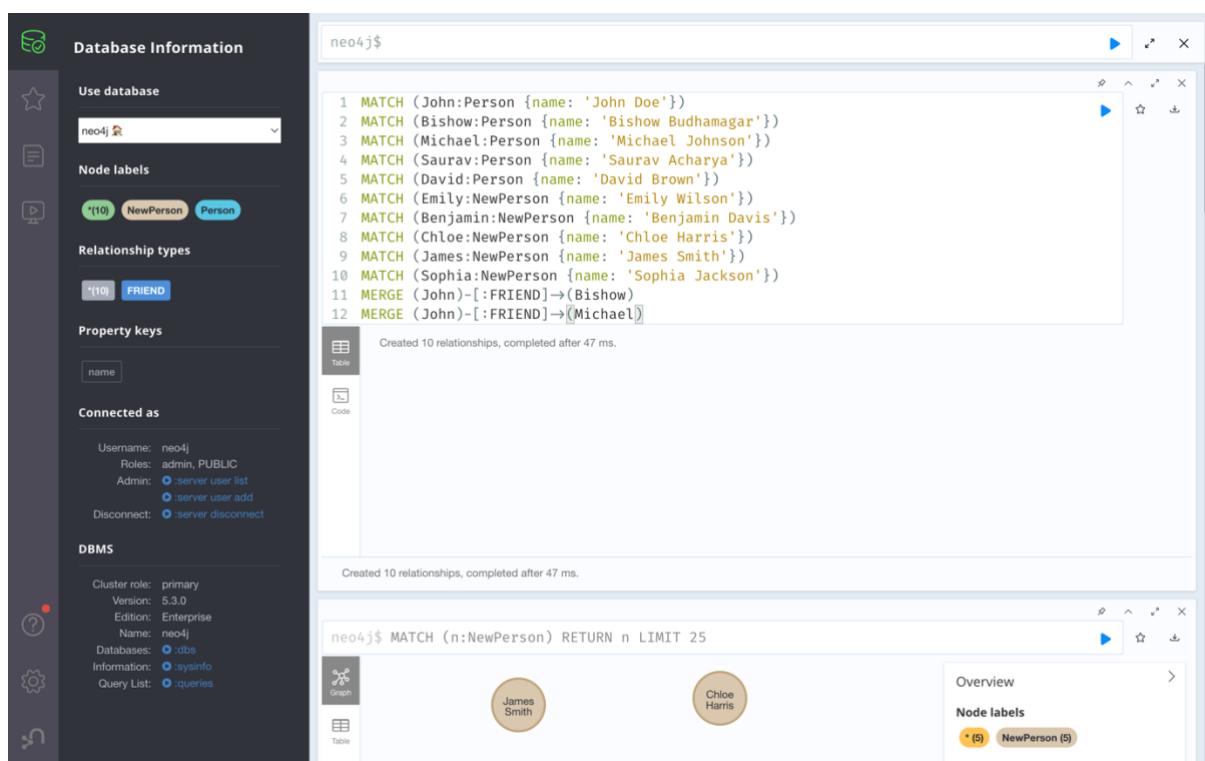


Figure 69 Inserting links(relationships) between Person and NewPerson nodes

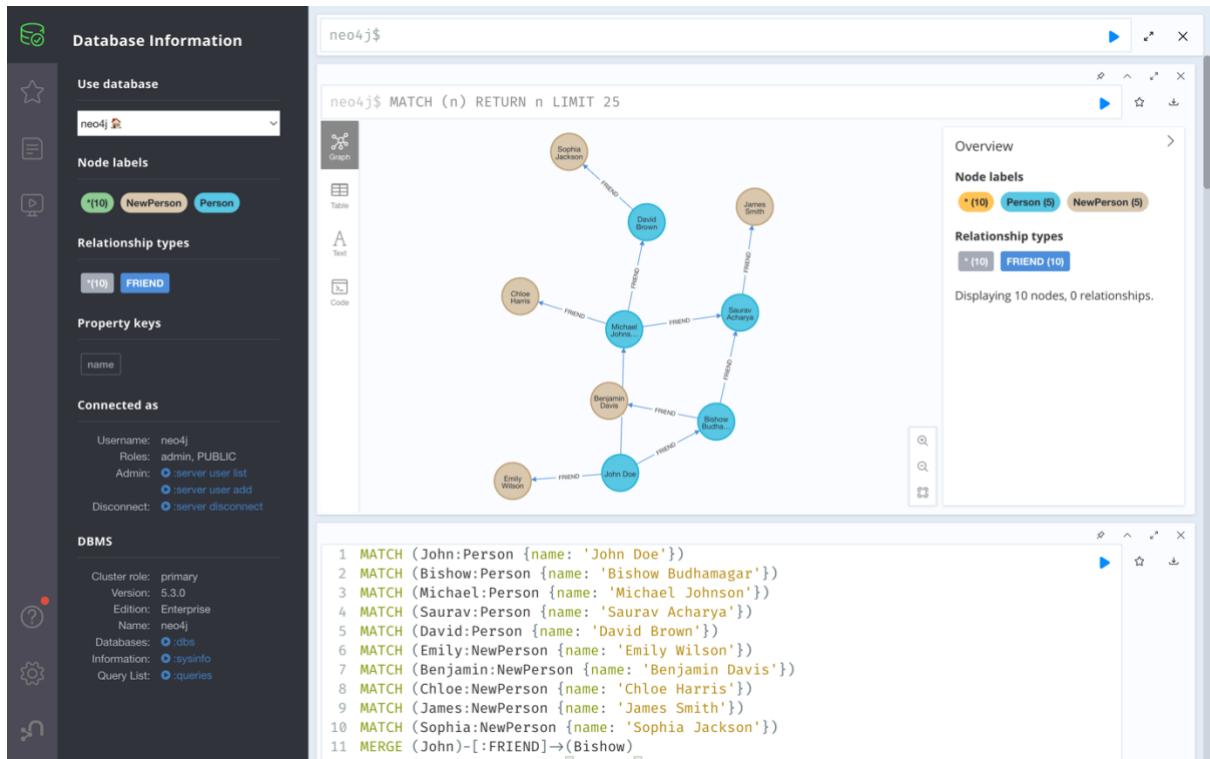


Figure 70 Graph representation of relationships between Person and NewPerson nodes

#### 4.2.4 Inserting Location Nodes

```
CREATE (:Location {name: 'Taronga Zoo Sydney'}), (:Location {name: 'Sydney Opera House'}), (:Location {name: 'South Cronulla Beach'}), (:Location {name: 'Sea Cliff Bridge'}), (:Location {name: 'Bondi Beach'})
```

The screenshot shows the Neo4j browser interface with two main panes. The left pane displays the 'Database Information' sidebar, which includes sections for 'Use database', 'Node labels', 'Relationship types', 'Property keys', 'Connected as', and 'DBMS'. The right pane shows the Neo4j shell and a graph visualization.

**Neo4j Shell:**

```
neo4j$ CREATE (:Location {name: 'Taronga Zoo Sydney'}), (:Location {name: 'Sydney Opera House'}), (:Location {name: 'South Cronulla Beach'}), (:Location {name: 'Sea Cliff Bridge'}), (:Location {name: 'Bondi Beach'})
```

Output: Added 5 labels, created 5 nodes, set 5 properties, completed after 10 ms.

**Graph View:**

```
neo4j$ MATCH (n) RETURN n LIMIT 25
```

Output: Displaying 10 nodes, 0 relationships.

The graph visualization shows ten nodes representing people: Sophie Jackson, David Brown, James Smith, Chloe Harris, Michael Johns, Sarah Adams, Benjamin Davis, and Anna White. They are interconnected by FRIEND relationships.

Figure 71 Inserting Location Nodes

The screenshot shows the Neo4j browser interface with two main panes. The left pane displays the 'Database Information' sidebar, which includes sections for 'Use database', 'Node labels', 'Relationship types', 'Property keys', 'Connected as', and 'DBMS'. The right pane shows the Neo4j shell and a graph visualization.

**Neo4j Shell:**

```
neo4j$ MATCH (n:Location) RETURN n LIMIT 25
```

Output: Displaying 5 nodes, 0 relationships.

**Graph View:**

```
neo4j$ CREATE (:Location {name: 'Taronga Zoo Sydney'}), (:Location {name: 'Sydney Opera House'}), (:Location {name: 'South Cronulla Beach'}), (:Location {name: 'Sea Cliff Bridge'}), (:Location {name: 'Bondi Beach'})
```

Output: Added 5 labels, created 5 nodes, set 5 properties, completed after 10 ms.

The graph visualization shows five nodes representing locations: Taronga Zoo Sydney, Sydney Opera House, South Cronulla Beach, Sea Cliff Bridge, and Bondi Beach.

Figure 72 Graph representation of Location Nodes

#### 4.2.5 Inserting links(relationships) between Person and Location Nodes

```

MATCH (John:Person {name: 'John Doe'})
MATCH (Bishow:Person {name: 'Bishow Budhamagar'})
MATCH (Michael:Person {name: 'Michael Johnson'})
MATCH (Saurav:Person {name: 'Saurav Acharya'})
MATCH (David:Person {name: 'David Brown'})
MATCH (T:Location {name: 'Taronga Zoo Sydney'})
MATCH (O:Location {name: 'Sydney Opera House'})
MATCH (C:Location {name: 'South Cronulla Beach'})
MATCH (S:Location {name: 'Sea Cliff Bridge'})
MATCH (B:Location {name: 'Bondi Beach'})
MERGE (John)-[:VISITED]->(T)
MERGE (Bishow)-[:VISITED]->(O)
MERGE (Michael)-[:VISITED]->(C)
MERGE (Saurav)-[:VISITED]->(S)
MERGE (David)-[:VISITED]->(B)

```

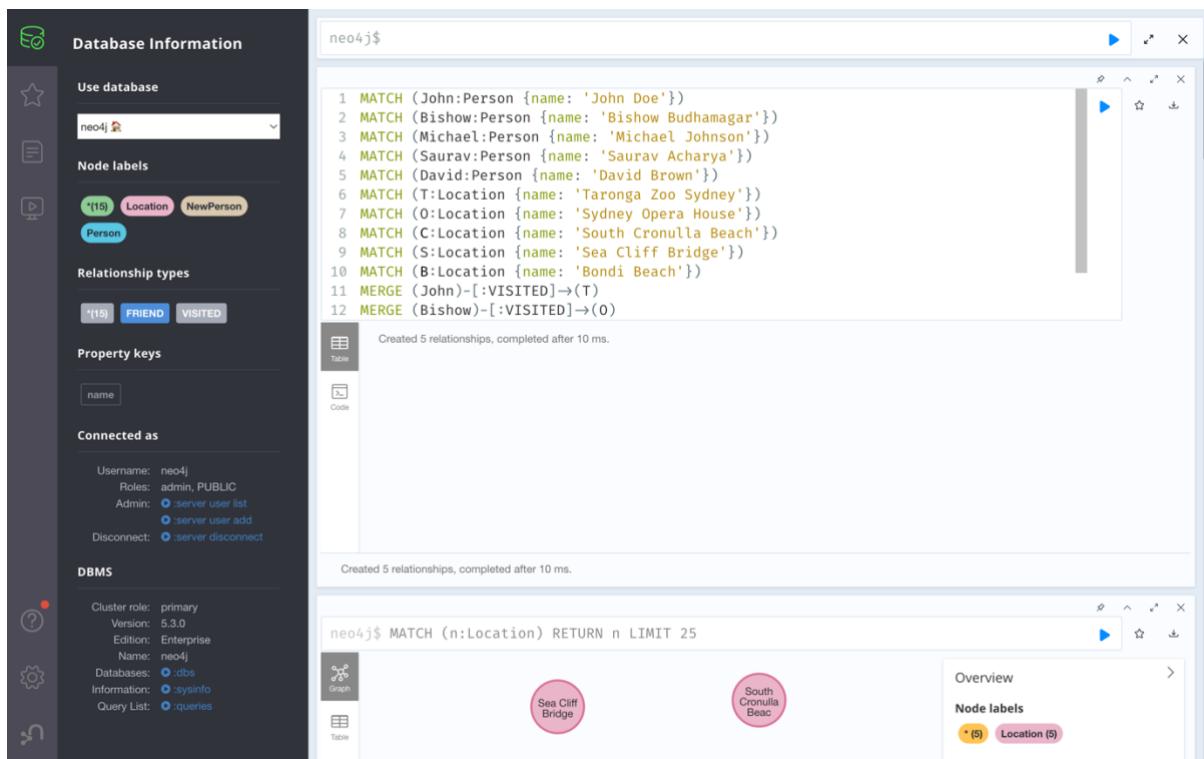


Figure 73 Inserting links(relationships) between Person and Location Nodes

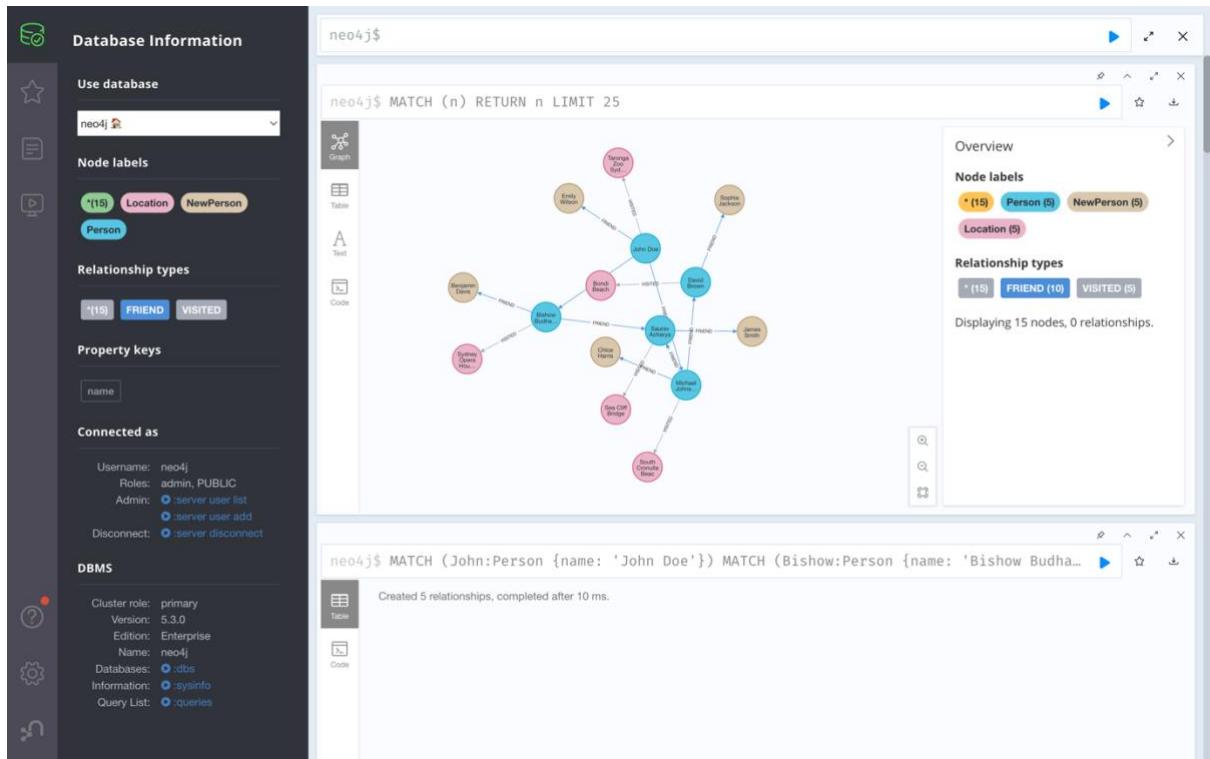


Figure 74 Graph representation of relationships between Person and Location Nodes

## 4.3 Retrieving all connected people for a specific person

### 4.3.1 For John Doe

```
MATCH (John:Person {name: 'John Doe'})-[:FRIEND]-(friend)  
RETURN friend
```

The screenshot shows the Neo4j interface. On the left, the 'Database Information' sidebar is open, showing the database is 'neo4j', node labels 'Person', relationship types 'FRIEND', and property keys 'name'. The main area shows the Neo4j browser with the following query:

```
1 MATCH (John:Person {name: 'John Doe'})-[:FRIEND]-(friend)  
2 RETURN friend
```

The results show three nodes: 'Bishow Budha...', 'Michael Johns...', and 'Emily Wilson', each with a blue circular background. The 'Overview' panel indicates 3 nodes and 0 relationships.

Below this, another query is shown:

```
neo4j$ MATCH (n) RETURN n LIMIT 25
```

This query displays a graph with several nodes (John Doe, Emily Wilson, Michael Johns, Bishow Budha, Benjamin Davis, Sophia Jackson, David Brown, James Smith) and their connections. The 'Overview' panel indicates 19 nodes, 10 FRIEND relationships, and 5 VISITED relationships.

Figure 75 Retrieving all the friends of John Doe

### 4.3.2 For Bishow Budhamagar

```
MATCH (Bishow:Person {name: 'Bishow Budhamagar'})-[:FRIEND]-(friend)  
RETURN friend
```

The screenshot shows the Neo4j interface. The 'Database Information' sidebar is identical to Figure 75. The main area shows the Neo4j browser with the following query:

```
1 MATCH (Bishow:Person {name: 'Bishow Budhamagar'})-[:FRIEND]-(friend)  
2 RETURN friend
```

The results show three nodes: 'John Doe', 'Saurav Acharya', and 'Benjamin Davis', each with a blue circular background. The 'Overview' panel indicates 3 nodes and 0 relationships.

Below this, another query is shown:

```
neo4j$ MATCH p=()-[r:FRIEND]->() RETURN p LIMIT 25
```

This query displays a graph with several nodes (John Doe, Saurav Acharya, Benjamin Davis, Sophia Jackson, David Brown, James Smith) and their connections. The 'Overview' panel indicates 10 nodes, 10 FRIEND relationships, and 0 VISITED relationships.

Figure 76 Retrieving all the friends of Bishow Budhamagar

#### 4.3.3 For Michael Johnson

```
MATCH (Michael:Person {name: 'Michael Johnson'})-[:FRIEND]-(friend)  
RETURN friend
```

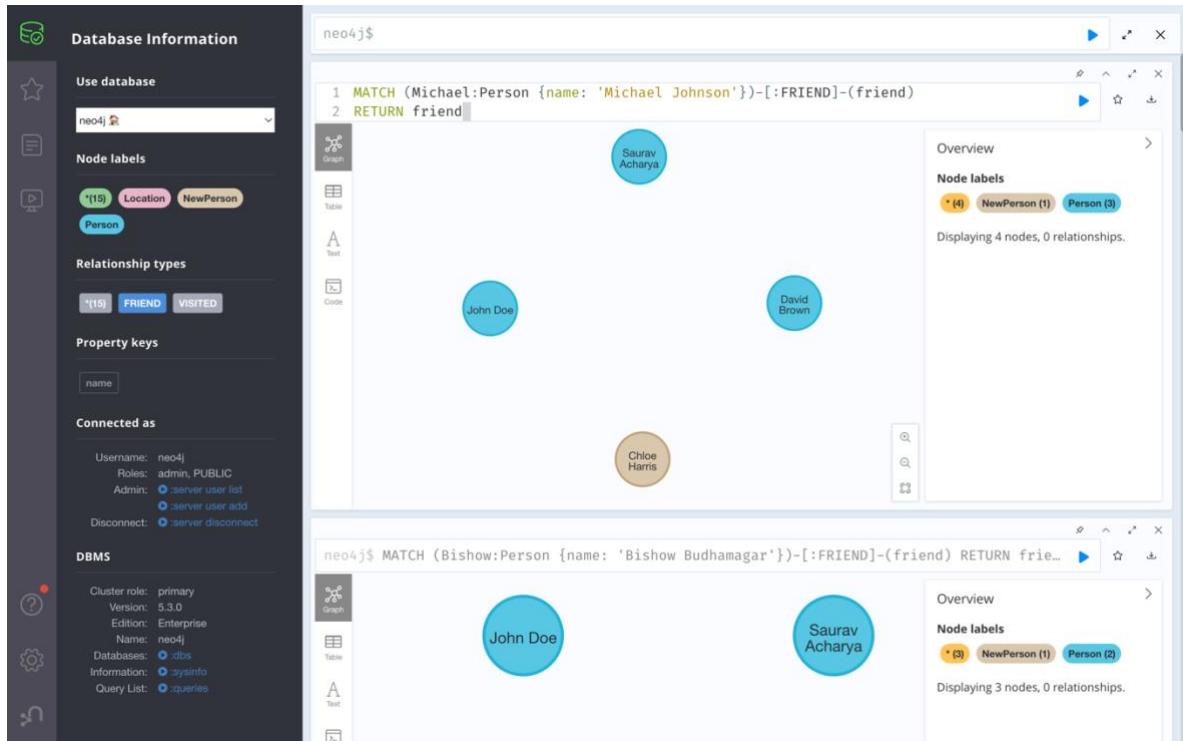


Figure 77 Retrieving all the friends of Michael Johnson

#### 4.3.4 For Saurav Acharya

```
MATCH (Saurav:Person {name: 'Saurav Acharya'})-[:FRIEND]-(friend)  
RETURN friend
```

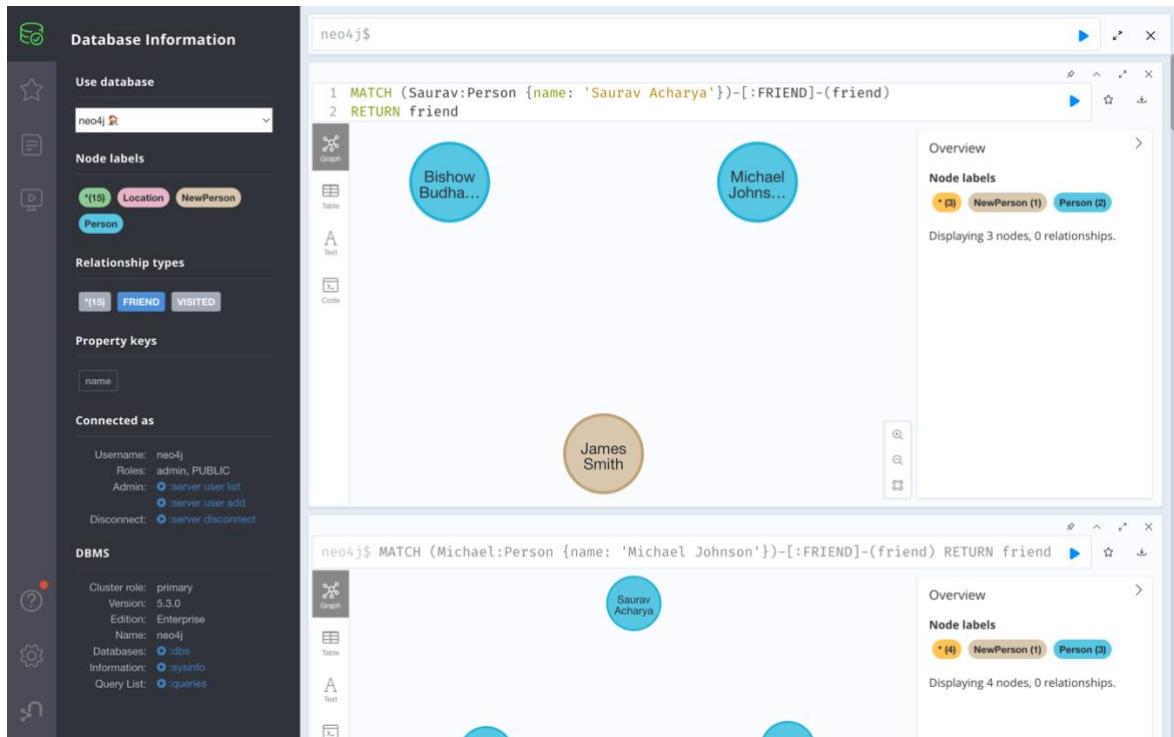


Figure 78 Retrieving all the friends of Saurav Acharya

#### 4.3.5 For David Brown

```
MATCH (David:Person {name: 'David Brown'})-[:FRIEND]-(friend)  
RETURN friend
```

The screenshot shows the Neo4j Browser interface with two separate query windows.

**Top Query:**

```
neo4j$  
1 MATCH (David:Person {name: 'David Brown'})-[:FRIEND]-(friend)  
2 RETURN friend
```

This query retrieves all friends of a person named 'David Brown'. The results are displayed in a graph view:

- A blue node labeled "Michael Johns..." is connected to a brown node labeled "Sophia Jackson" by a blue line.

**Bottom Query:**

```
neo4j$ MATCH (Saurav:Person {name: 'Saurav Acharya'})-[:FRIEND]-(friend) RETURN friend
```

This query retrieves all friends of a person named 'Saurav Acharya'. The results are displayed in a graph view:

- A blue node labeled "Bishow Budha..." is connected to a blue node labeled "Michael Johns..." by a blue line.

**Left Panel (DBMS Information):**

- Use database:** neo4j
- Node labels:** Location (1), NewPerson (1), Person (1)
- Relationship types:** FRIEND (1), VISITED (1)
- Property keys:** name
- Connected as:**
  - Username: neo4j
  - Roles: admin, PUBLIC
  - Admin: server user list, server user add
  - Disconnect: server disconnect
- DBMS:**
  - Cluster role: primary
  - Version: 5.3.0
  - Edition: Enterprise
  - Name: neo4j
  - Databases: :cbs, :sysinfo
  - Information: :sysinfo
  - Query List: :queries

Figure 79 Retrieving all the friends of David Brown

## 5. Critical Reflection

This task provides valuable practical experience in building and utilising various kinds of databases. The significance of data modelling, which includes the creation of tables, linkages, and data structures, is emphasised. Assuring data integrity and consistency across several database systems is a regular challenge. Data synchronisation between various systems can be difficult.

### 5.1 Proposed extensions to the system – Spatial

#### **Things Learned:**

PostGIS spatial databases offer insights into the usage of spatial reference systems (SRID), spatial queries, and processing of geographic data.

#### **Difficulties and Challenges:**

It can be difficult and requires a thorough understanding of geographic principles to handle geospatial data, particularly when working with complicated spatial queries.

### 5.2 NoSQL

#### **Things Learned:**

The document database (MongoDB) is an example of how NoSQL databases may be flexible, especially when dealing with semi-structured data like JSON.

#### **Difficulties and Challenges:**

It can be challenging to select the ideal NoSQL database structure and build documents that meet the requirements of the application.

### 5.3 Graph

#### **Things Learned:**

The graph database (Neo4j) serves as an example of how graph databases are suitable for relationship modelling and executing sophisticated graph-based queries.

#### **Difficulties and Challenges:**

Learning several database query languages (SQL, MQL, Cypher) and becoming accustomed to the distinctive querying strategies used by each database can be challenging.

### 5.4 How Could I Improve:

I can take into account the following to improve:

**Examine the Extra Features:** Examine each database system's more complex features in greater detail. Learn about performance tuning, optimisation, and indexing.

**Practise Query Optimisation:** Work on SQL, MQL, and Cypher query optimisation. Data retrieval will be more effective as a result.

**Improved Documentation:** Make sure my documentation is detailed and clear. Justify your choice of a certain geographic reference system or NoSQL database, for example.

**Peer Review:** Take into account reviewing my strategy or talking to your peers about it. I can find better answers and get fresh insights through collaborative learning. Read broadly: To better my understanding, I will research scholarly articles and resources on geographical databases, NoSQL databases, and graph databases.

## 6. Conclusion:

ITECH2004 Assignment 2, which covers a variety of database systems, including PostgreSQL, PostGIS, MongoDB, and Neo4j, offers useful hands-on experience. By recommending system extensions, it encourages innovative thinking. The complexity of spatial data, the adaptability of NoSQL databases, and relationship modelling in graph databases are among the key takeaways. Data consistency, geographical searches, and adjusting to different query languages are challenges.