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

1. Postgres Database (PostgreSQL)

1.1 Tables Setup and Data

1.1.1 Postgres database creation

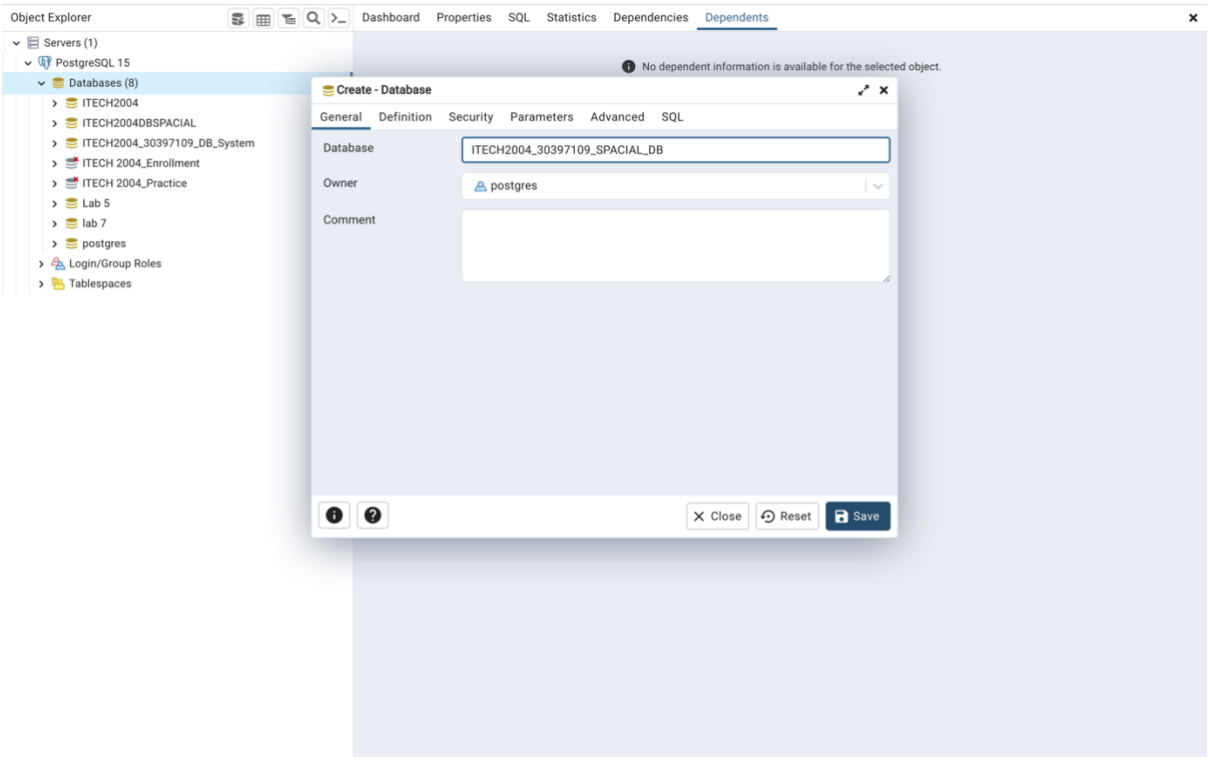

Figure 1 Postgres database creation (1)

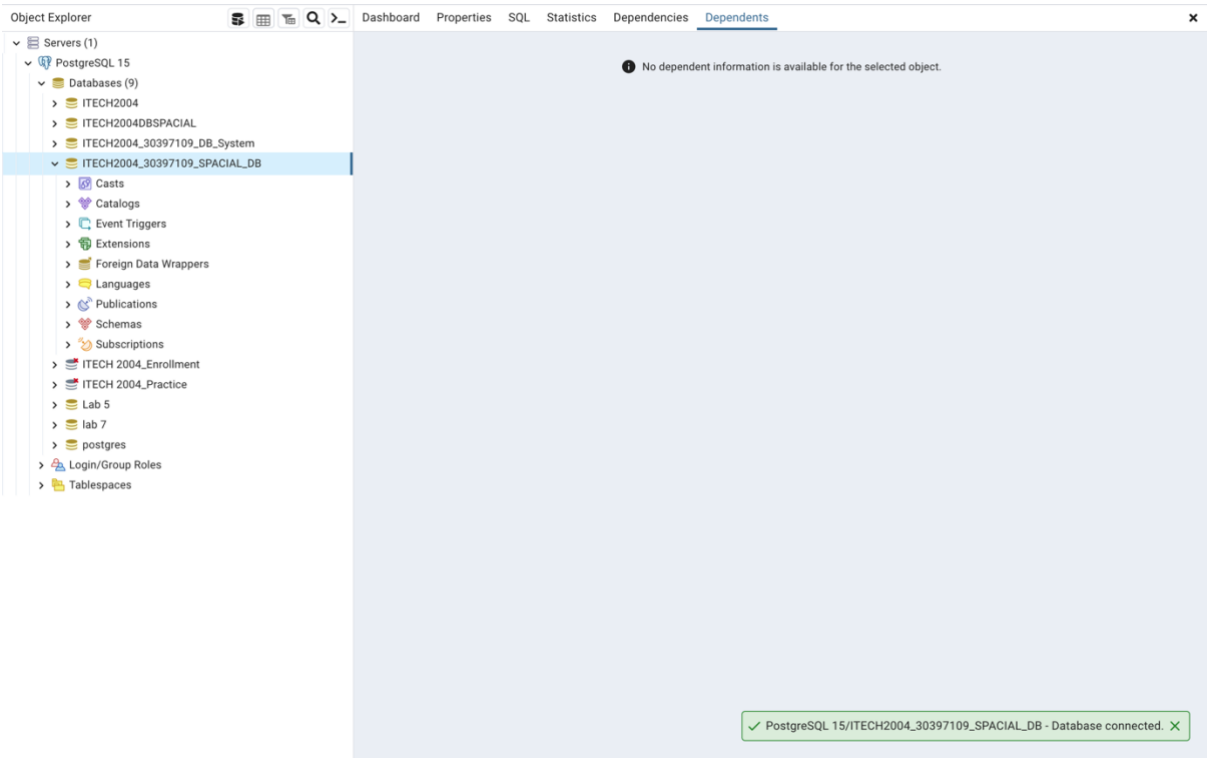

Figure 2 Postgres database creation (2)

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;
```

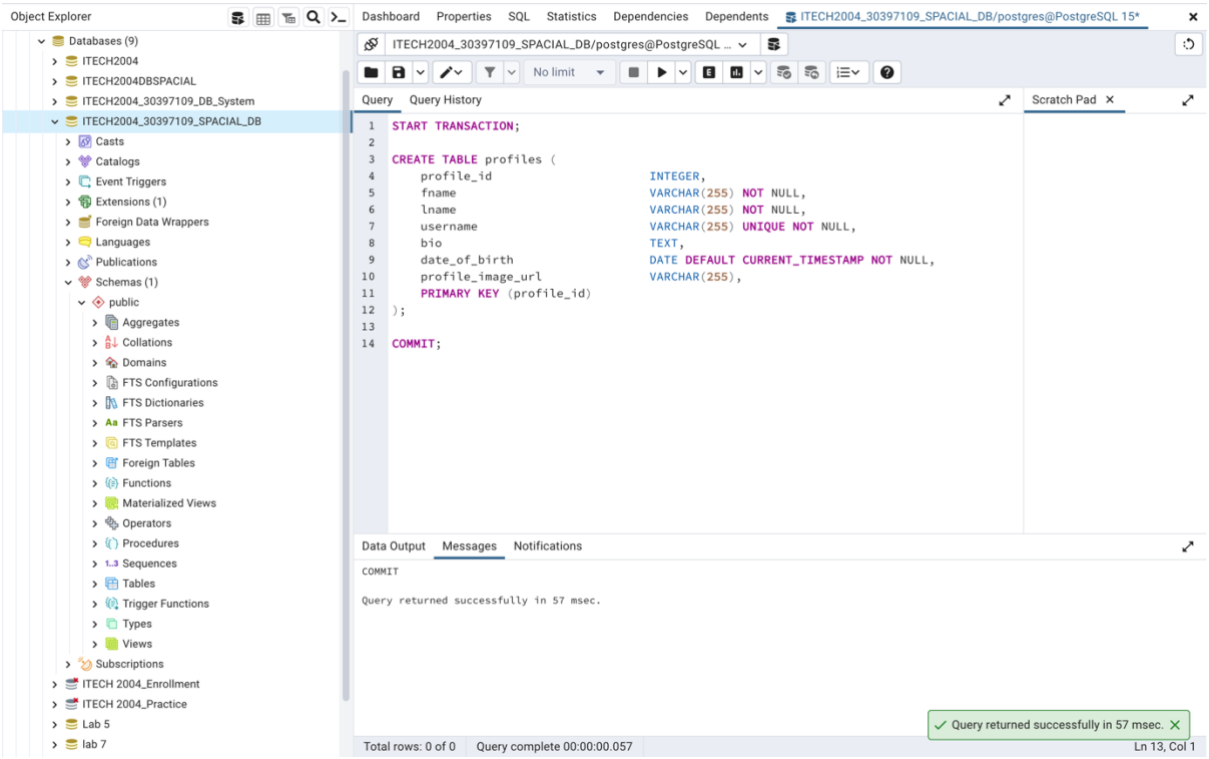

Figure 3 Profiles table creation

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;

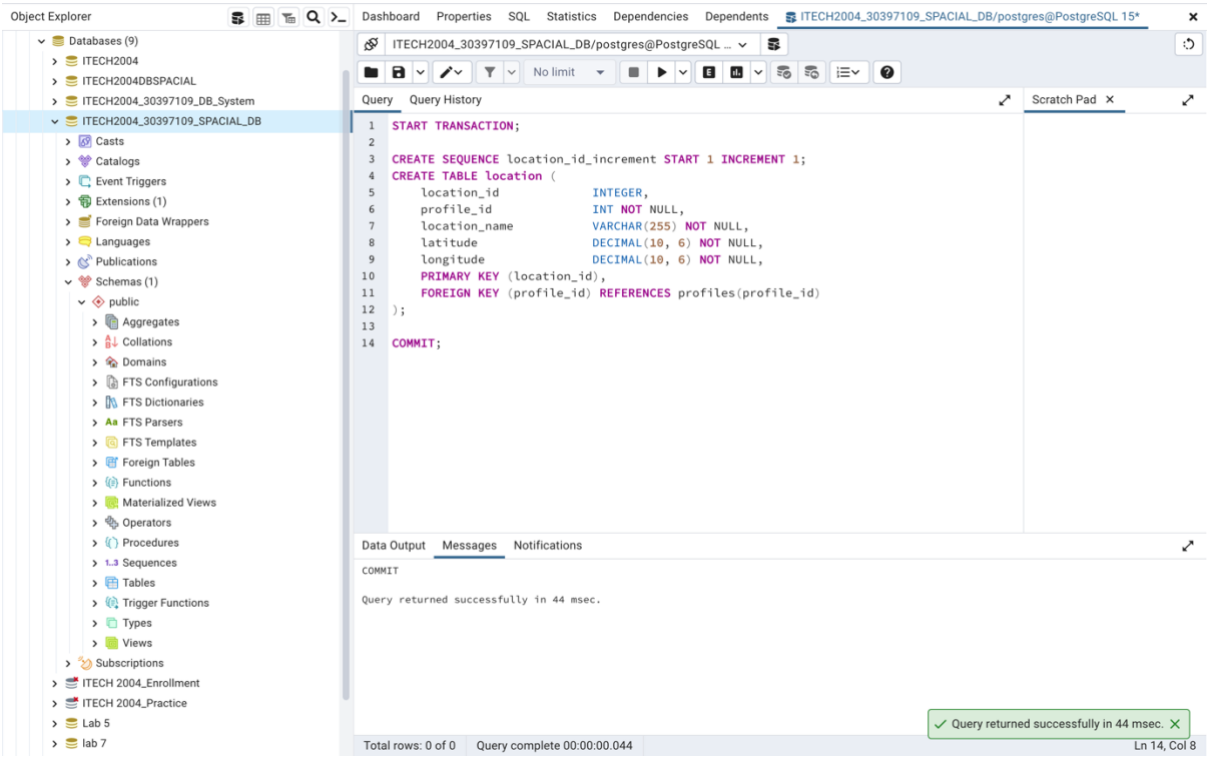

Figure 4 Location table creation

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;
```

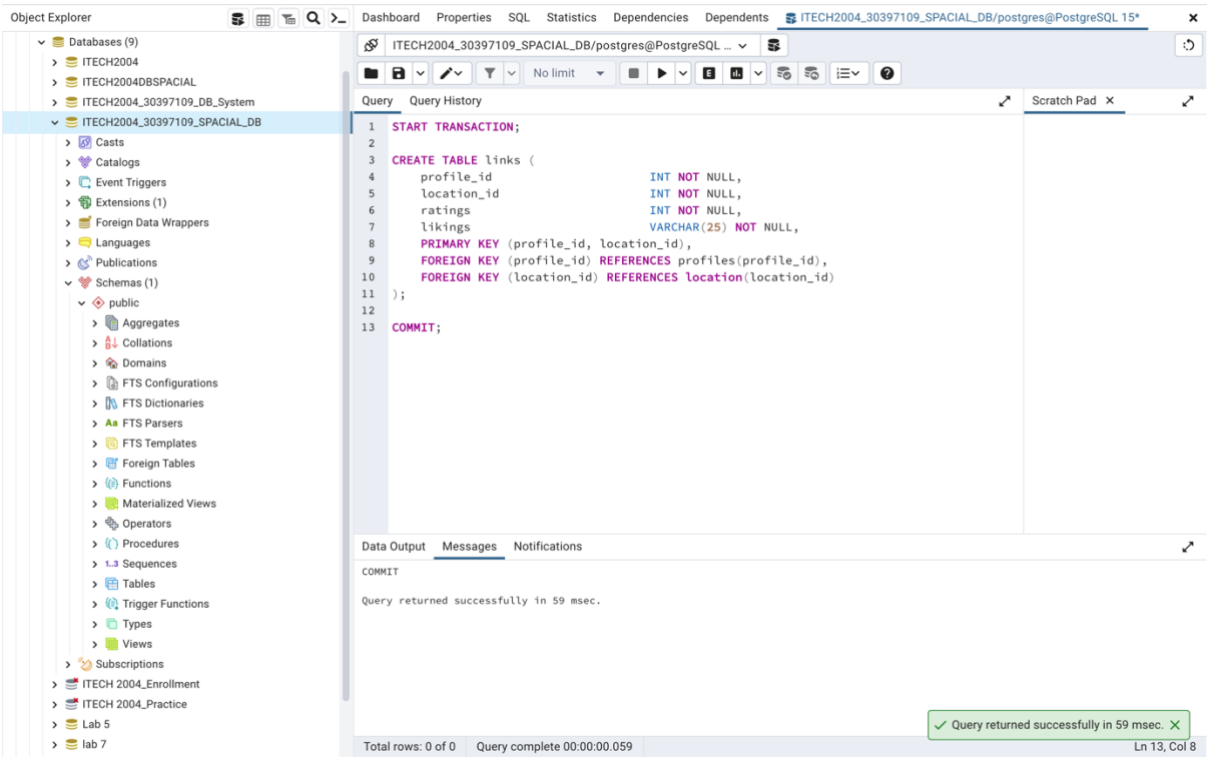

Figure 5 Links table creation

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;
```

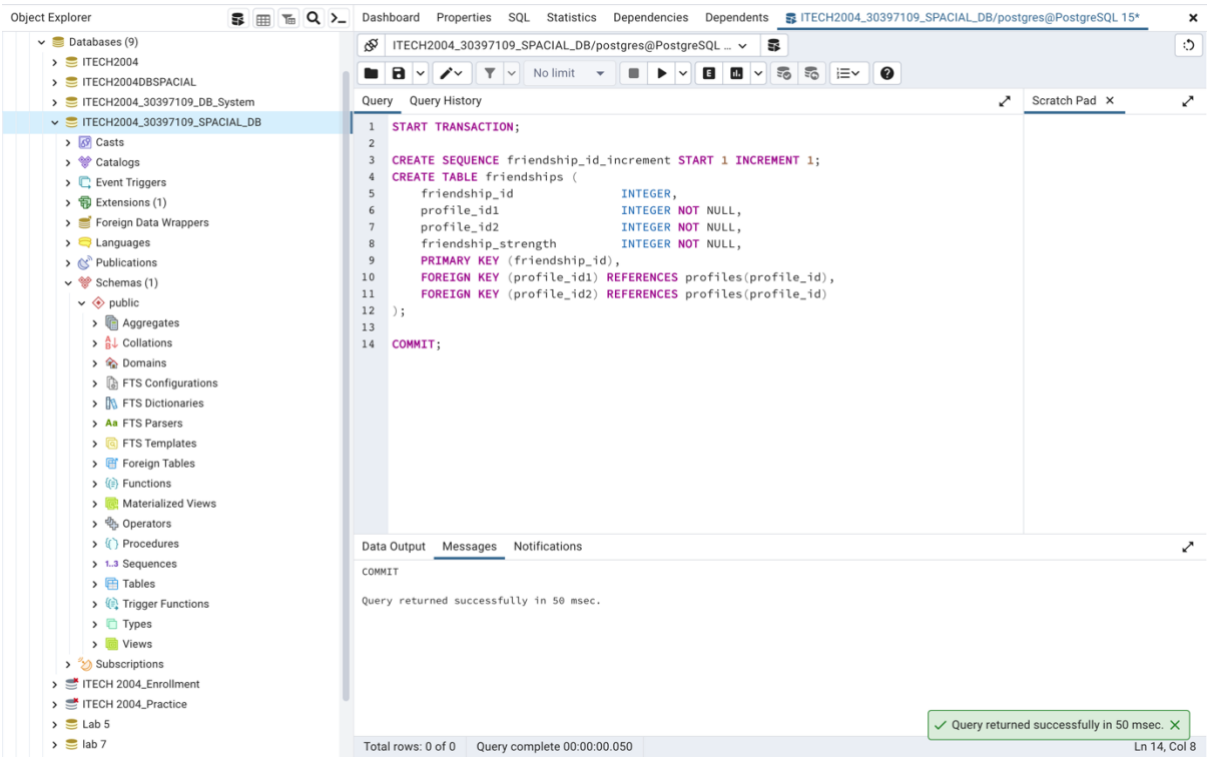

Figure 6 Friendships table creation

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;
```

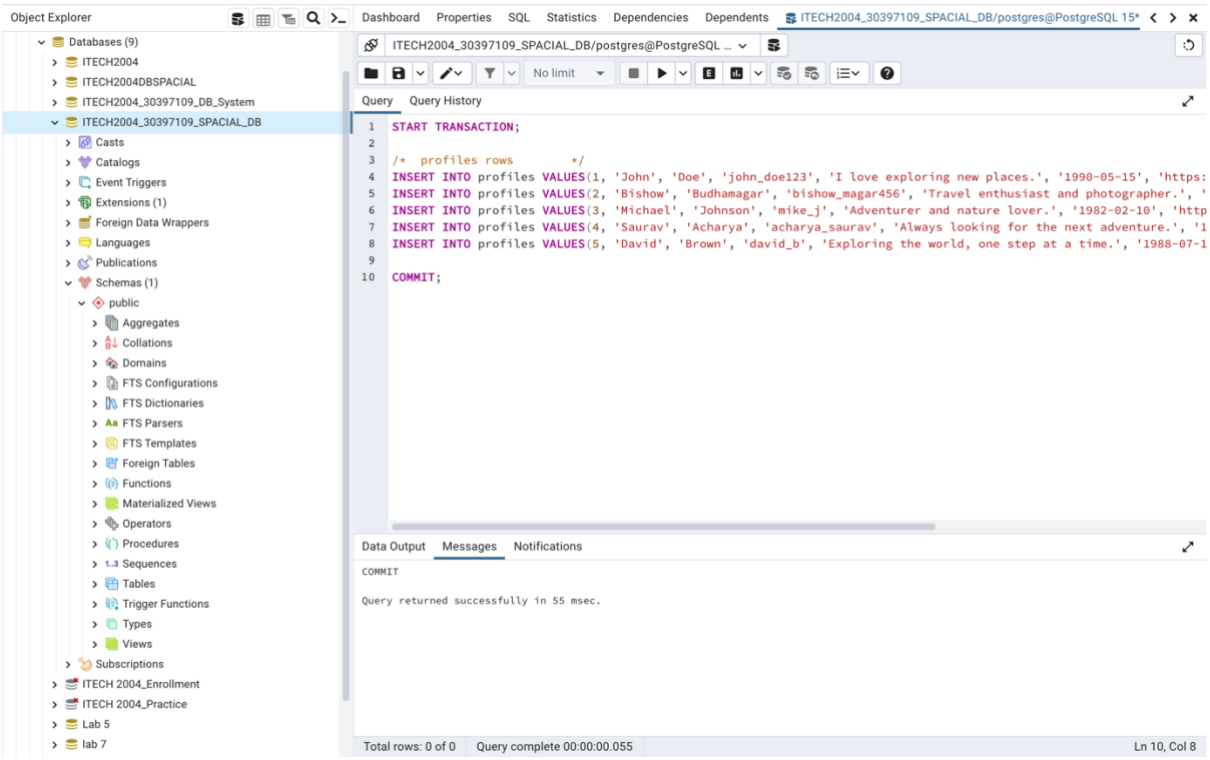

Figure 7 Profiles data insertion

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;
```

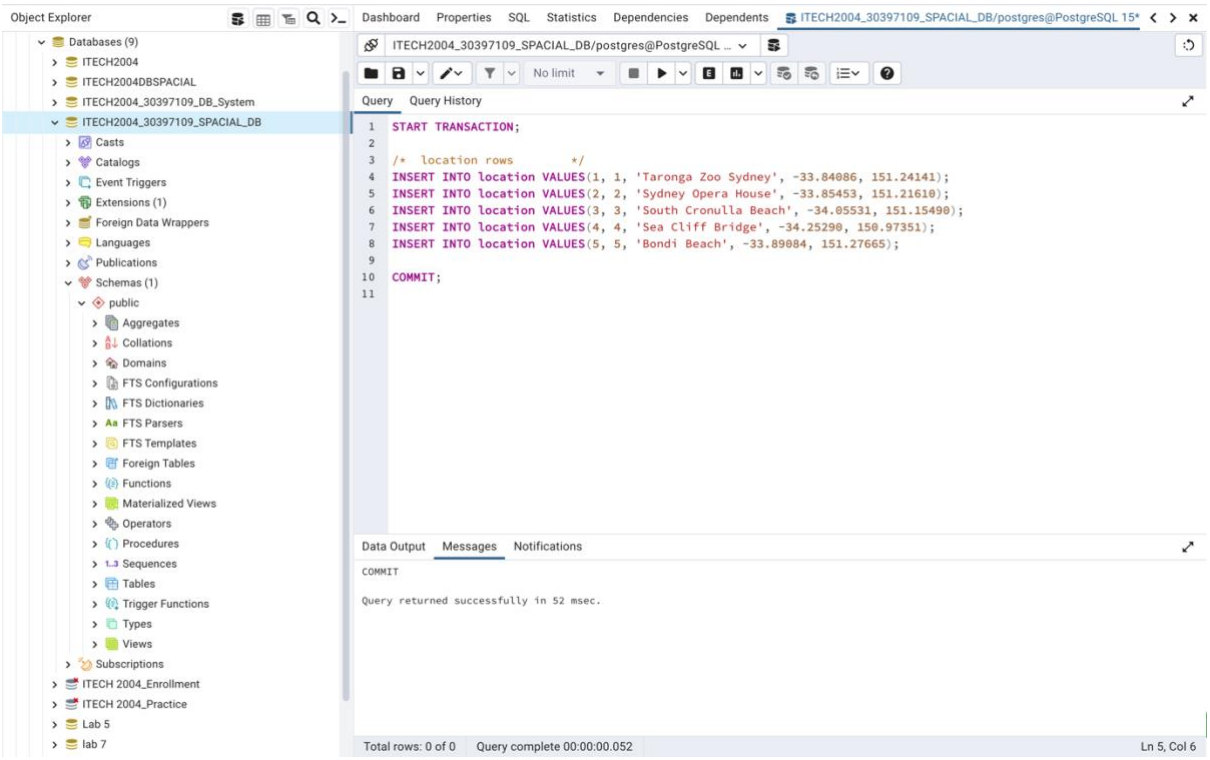

Figure 8 Location data insertion

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;
```

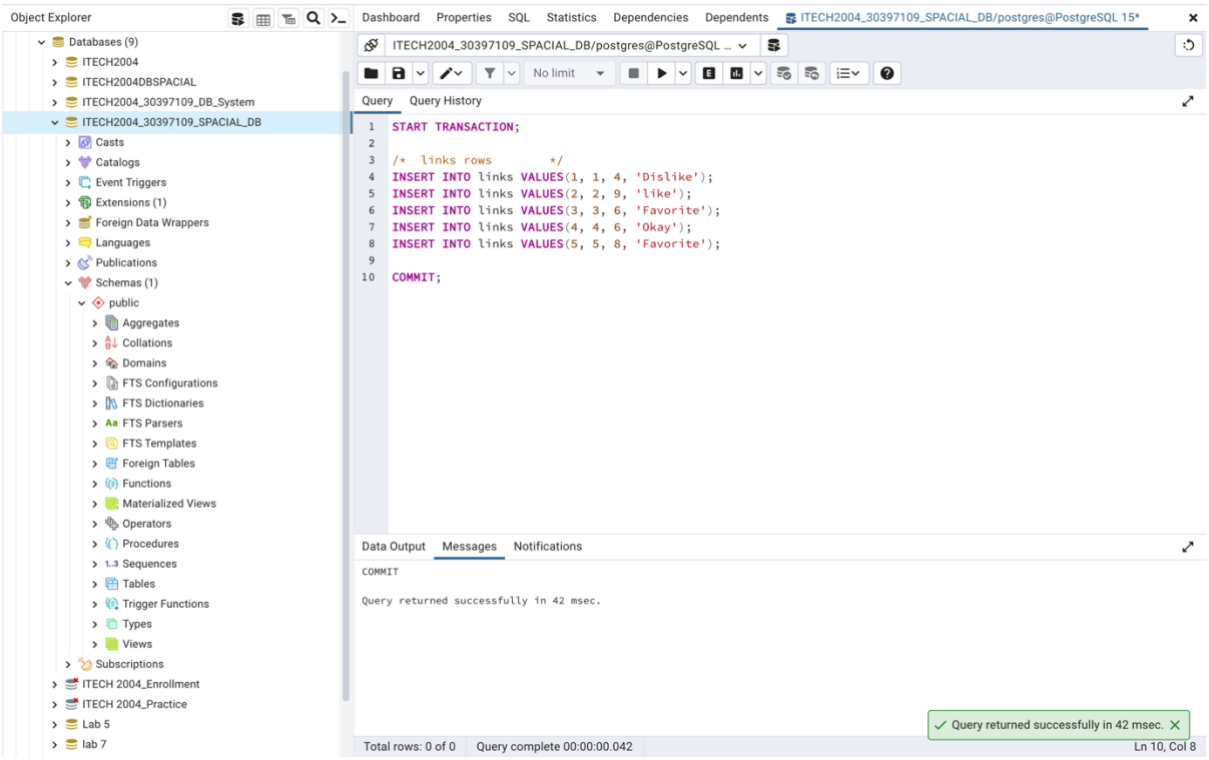

Figure 9 Links data insertion

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;
```

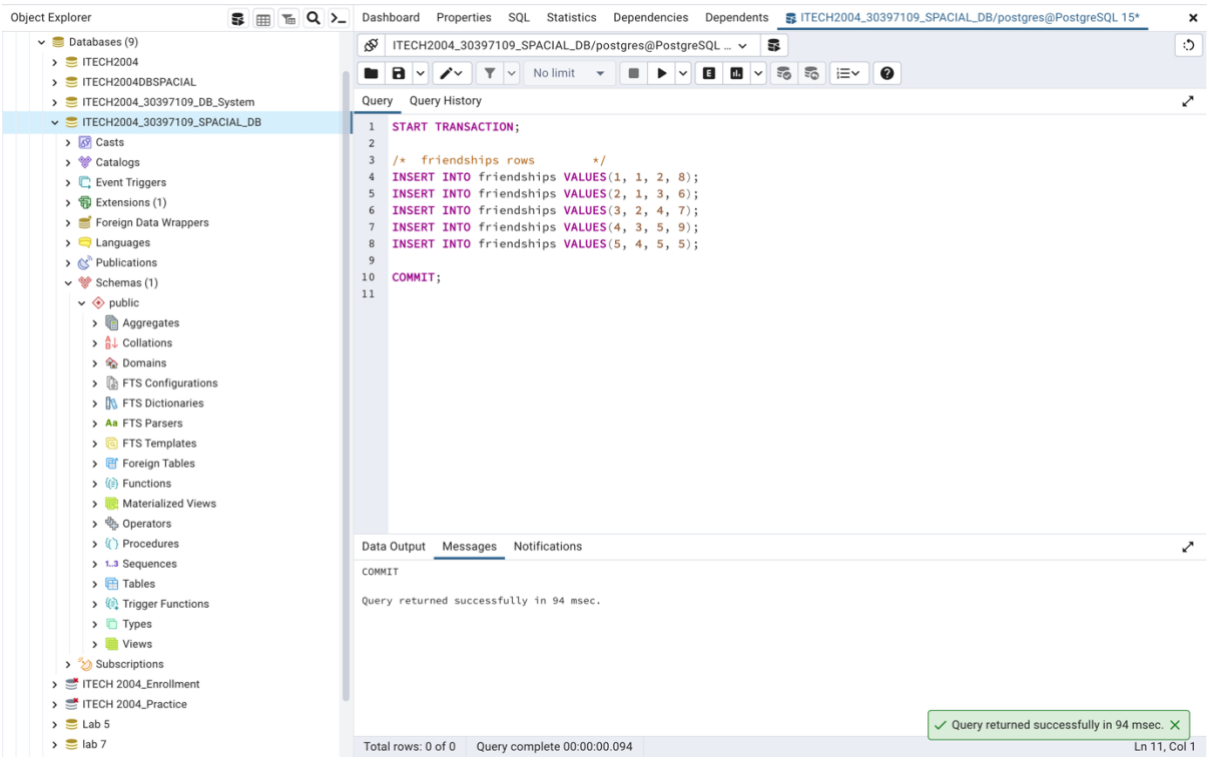

Figure 10 Friendships data insertion

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';

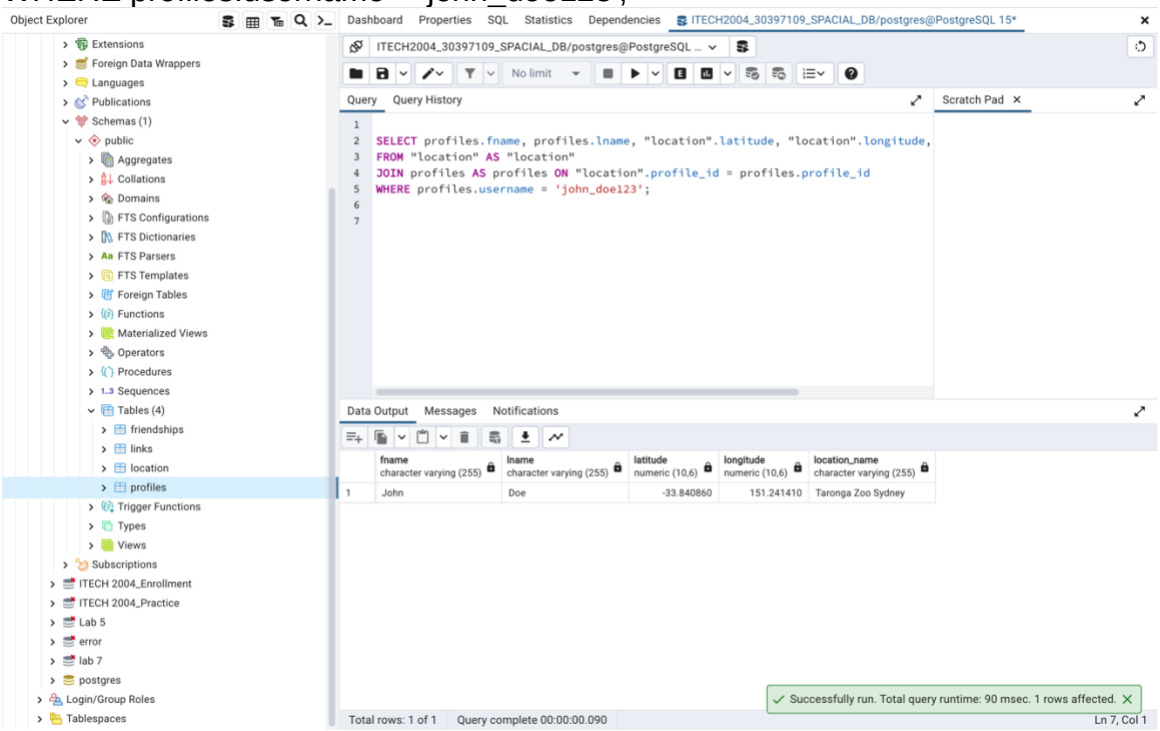

Figure 11 Retrieving coordinates for John Doe

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';
```

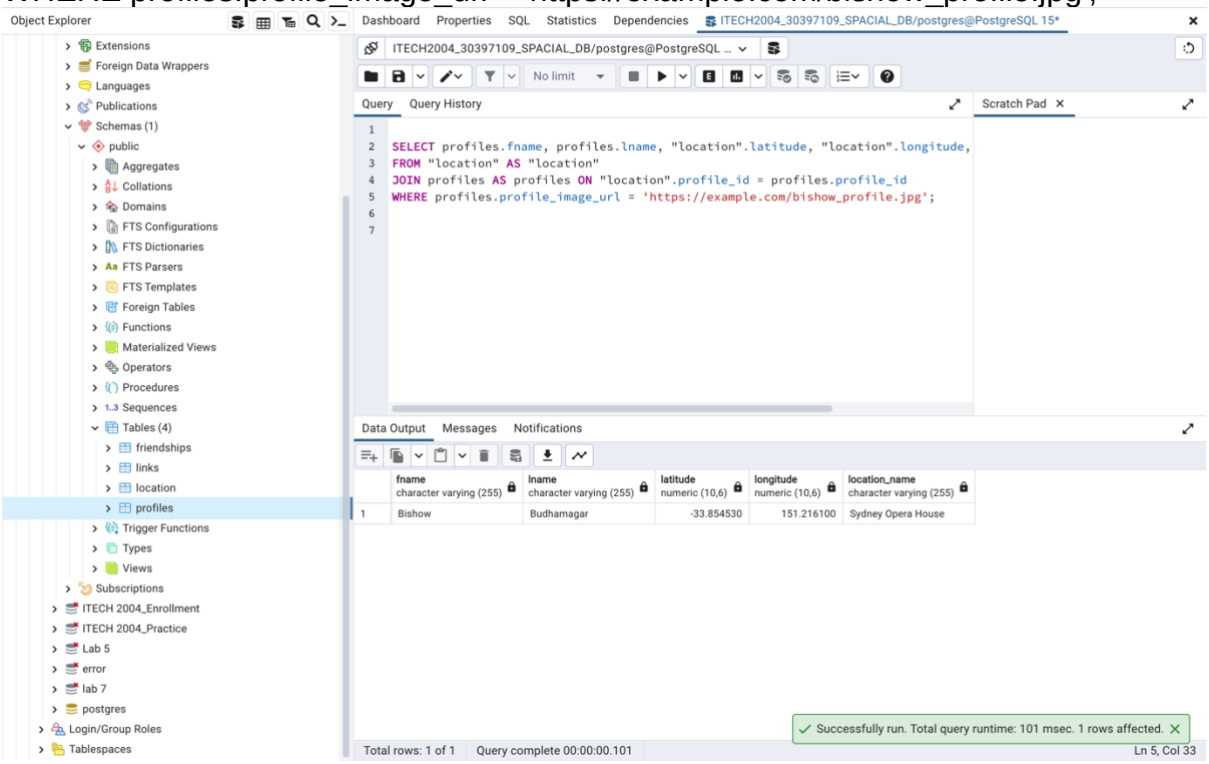

Figure 12 Retrieving coordinates for Bishow Budhamagar

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';
```

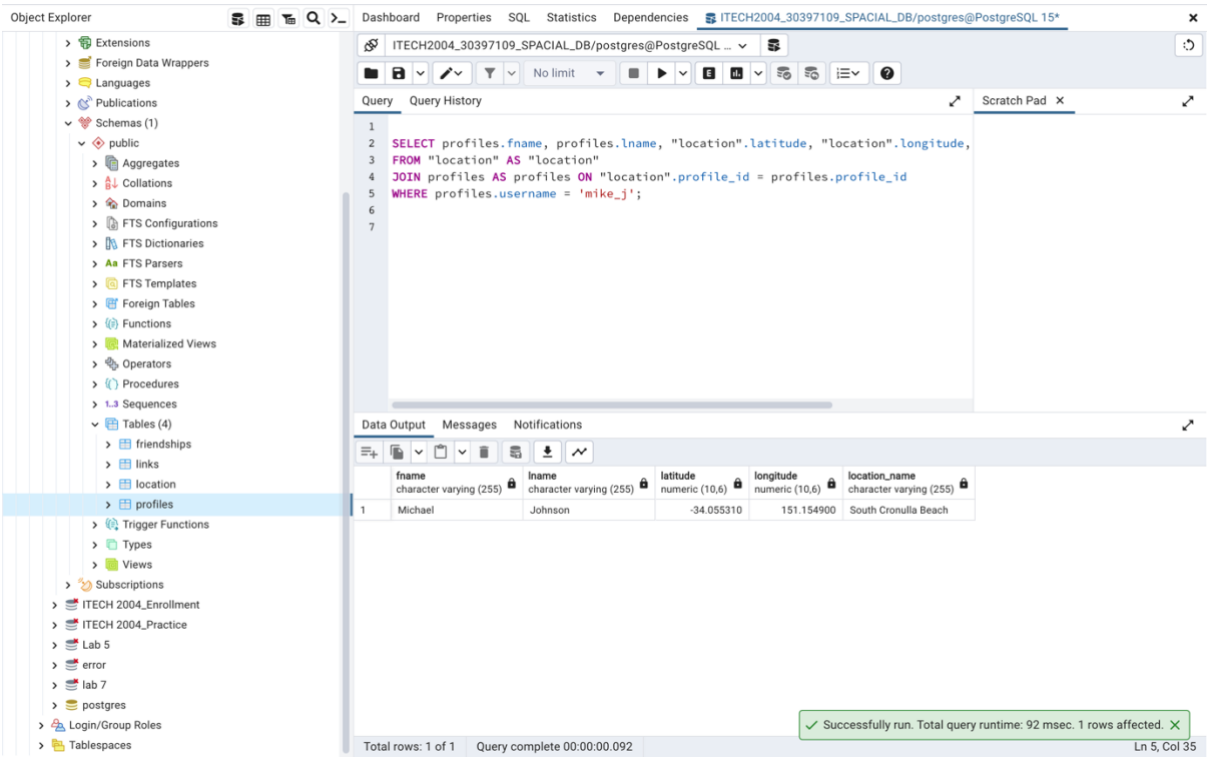

Figure 13 Retrieving coordinates for Michael Johnson

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.';
```

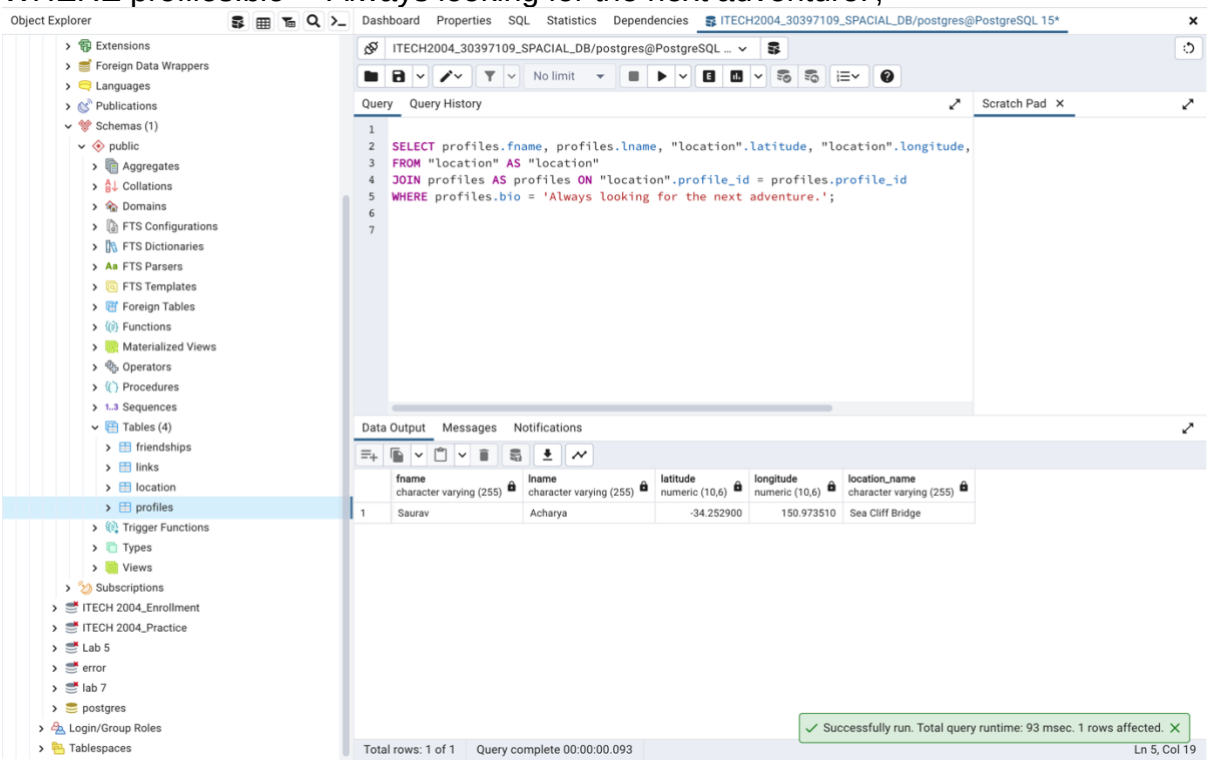

Figure 14 Retrieving coordinates for Saurav Acharya

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';
```

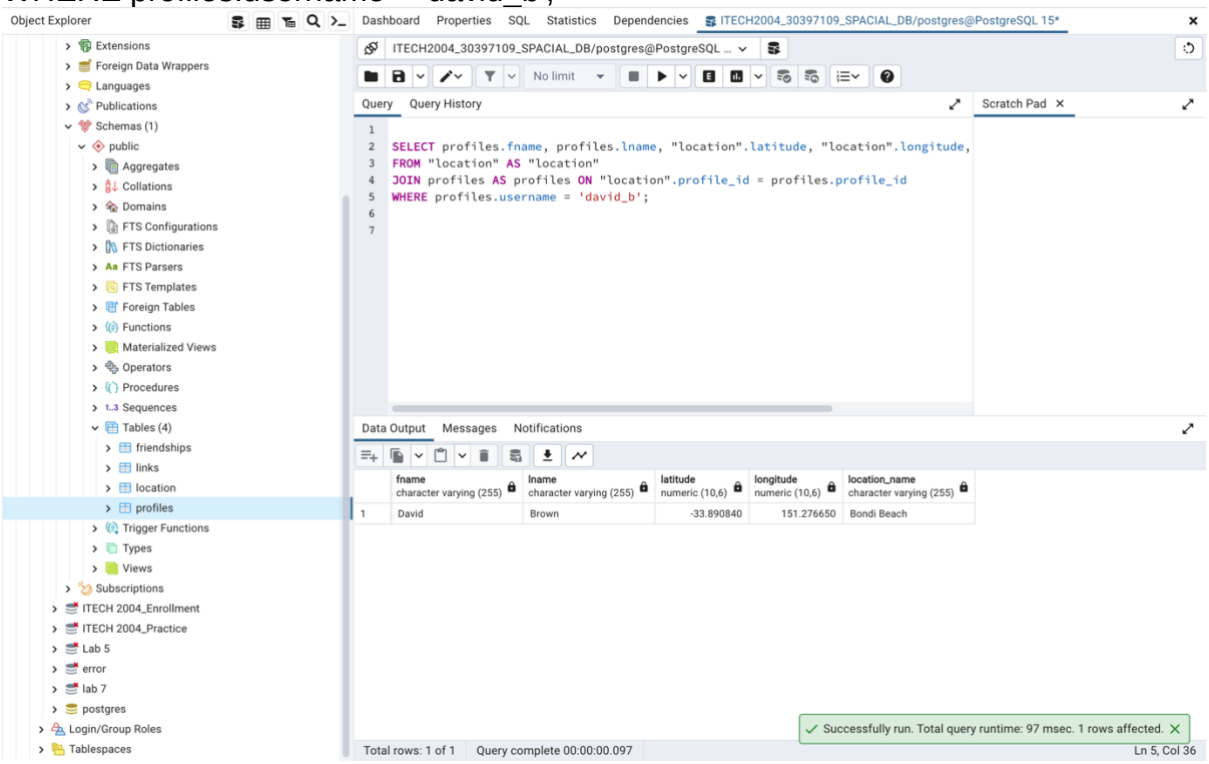

Figure 15 Retrieving coordinates for David Brown

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;
```

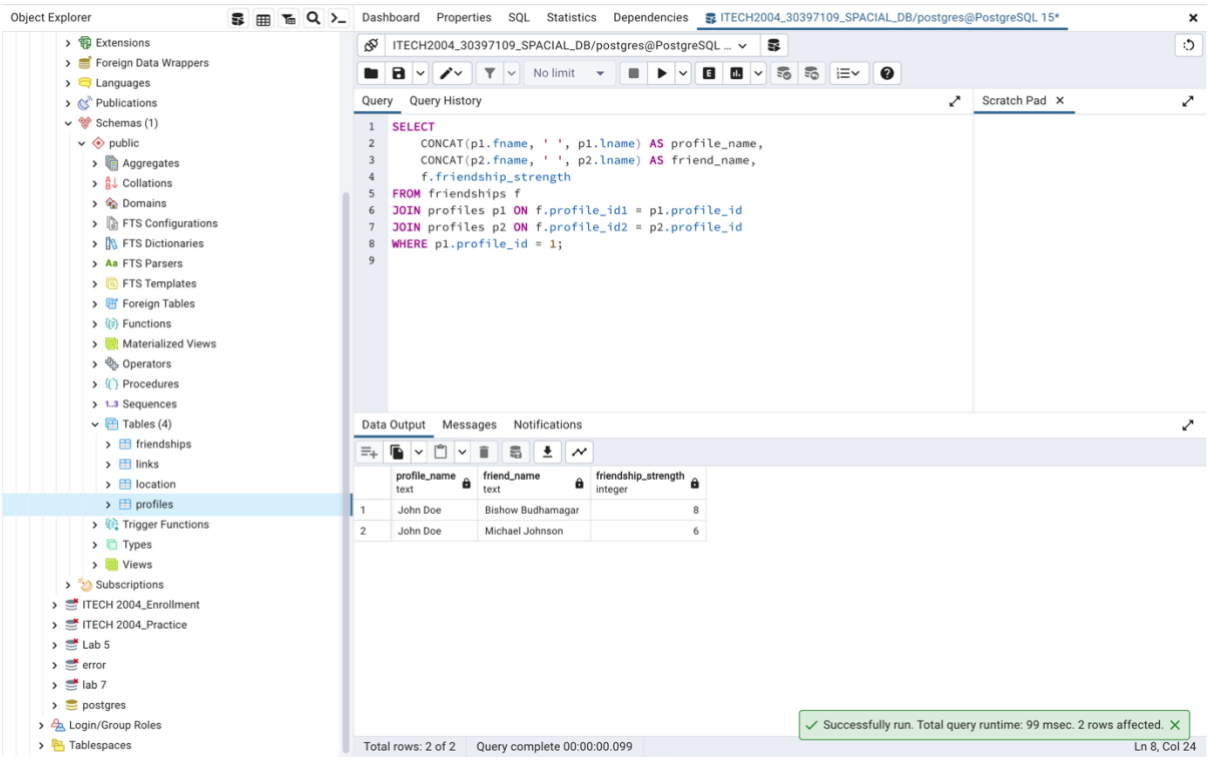

Figure 16 Retrieving friendship/social media links for John Doe

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;
```

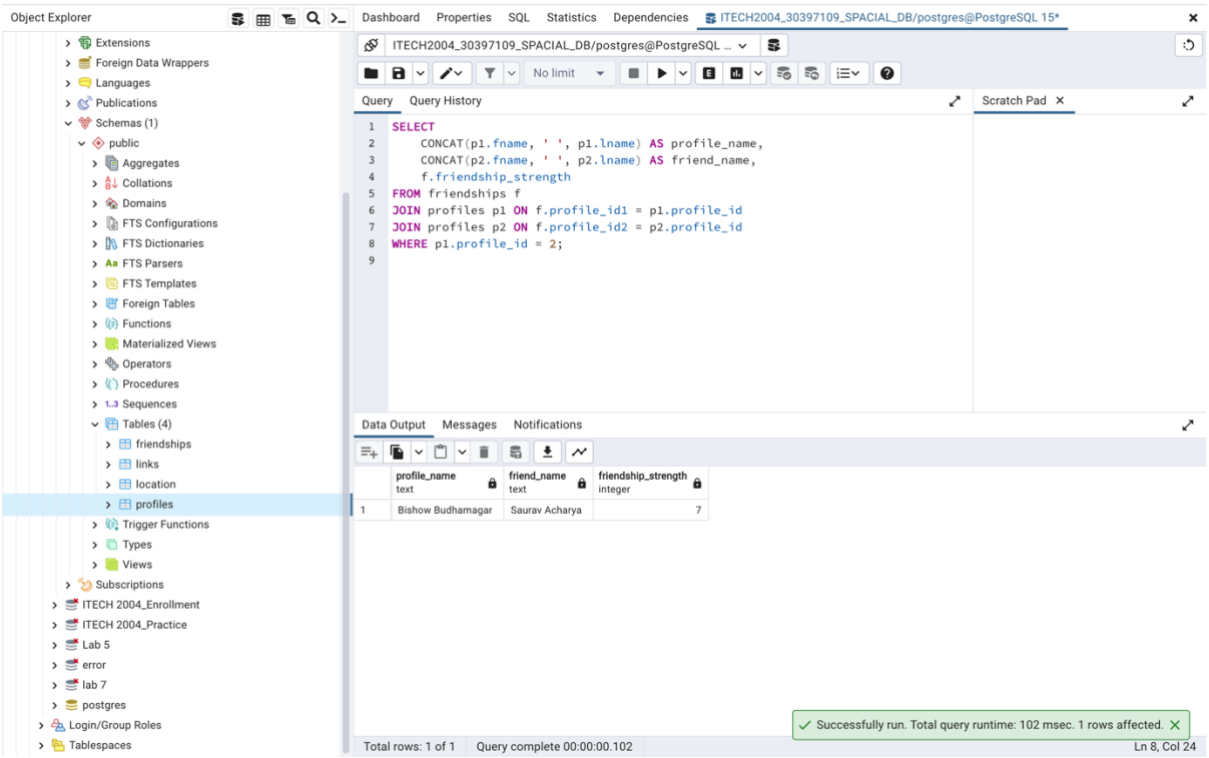

Figure 17 Retrieving friendship/social media links for Bishow Budhamagar

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;
```

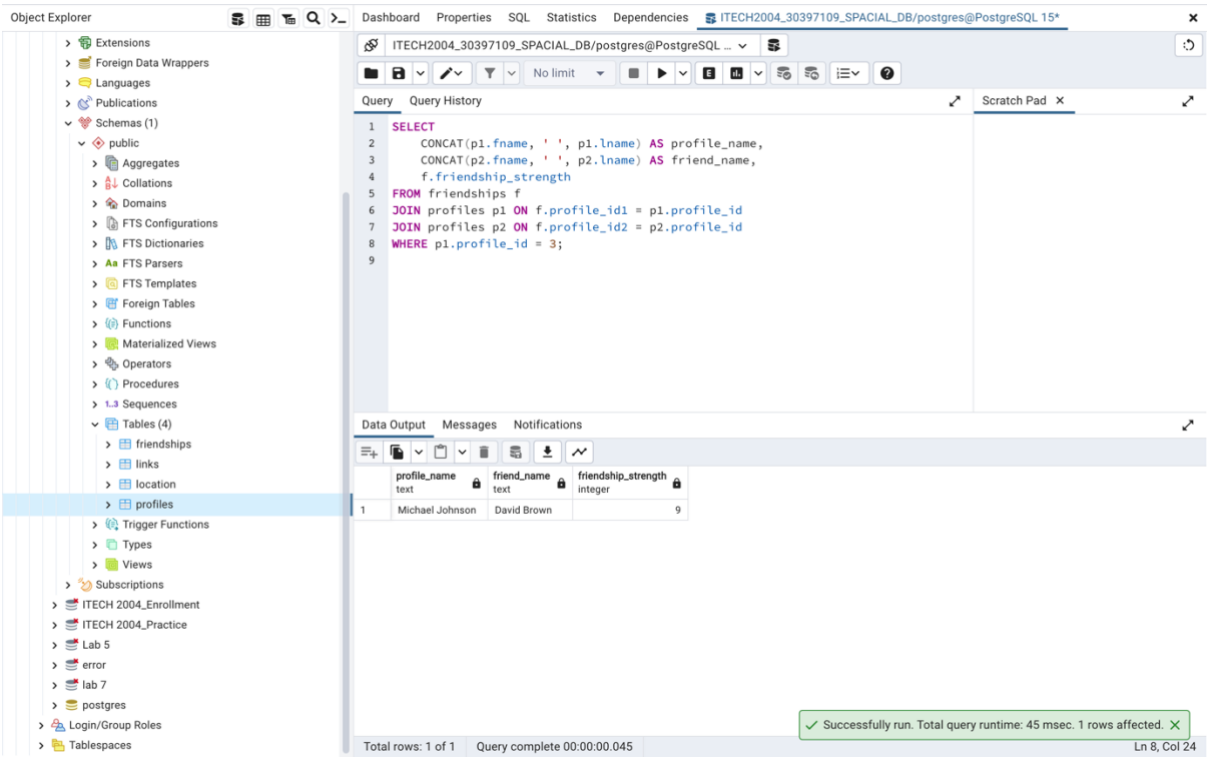

Figure 18 Retrieving friendship/social media links for Michael Johnson

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;
```

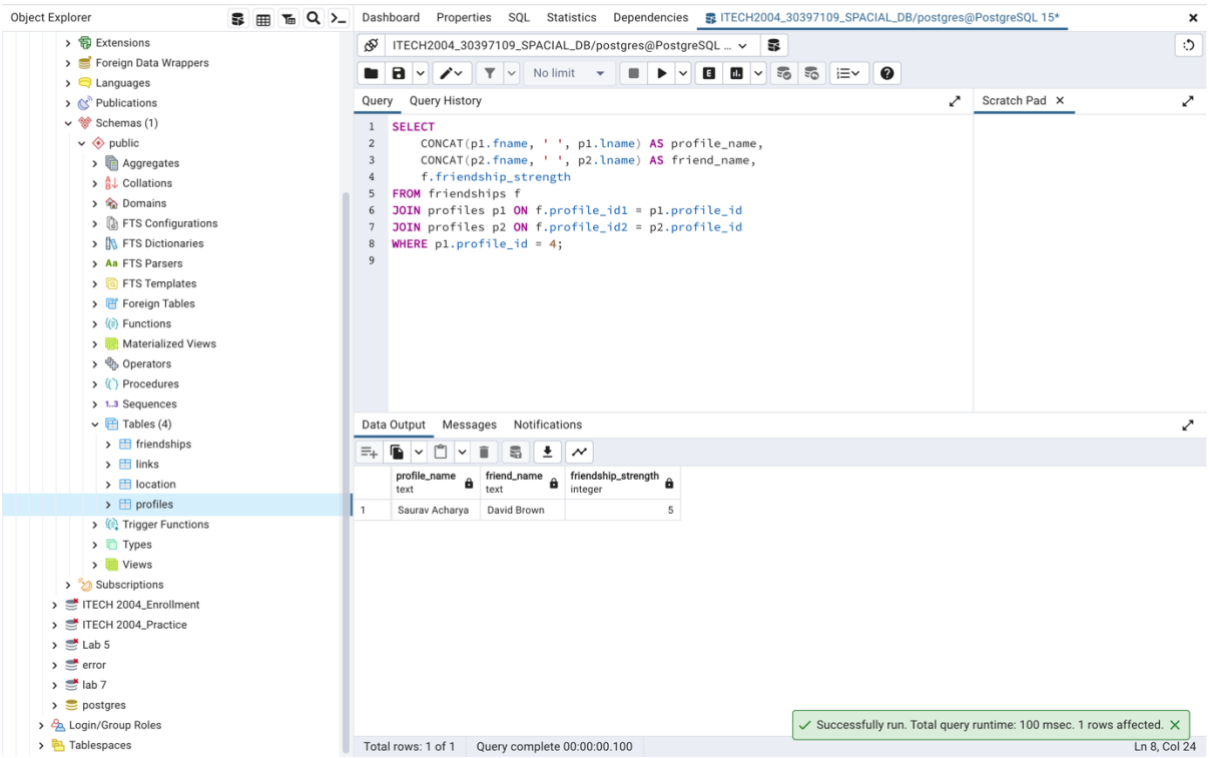

Figure 19 Retrieving friendship/social media links for Saurav Acharya

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;
```

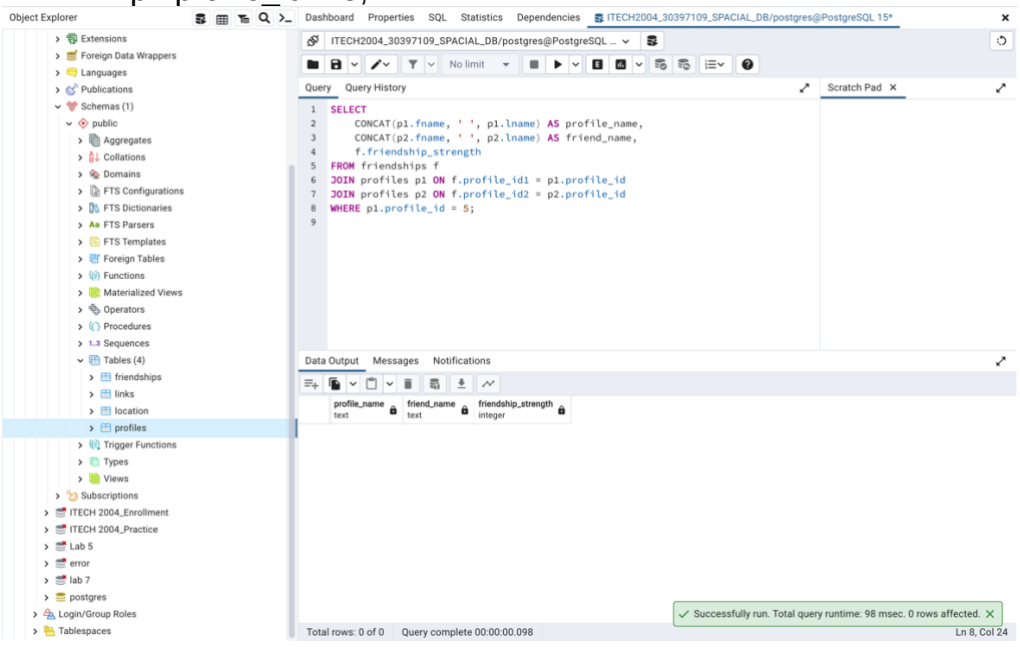

Figure 20 Retrieving friendship/social media links for David Brown

2. PostGIS Spatial Database

2.1 Tables Setup and Data

Extension creation

/* Creating extension for Spatial Database Assignment */
CREATE extension postgis;

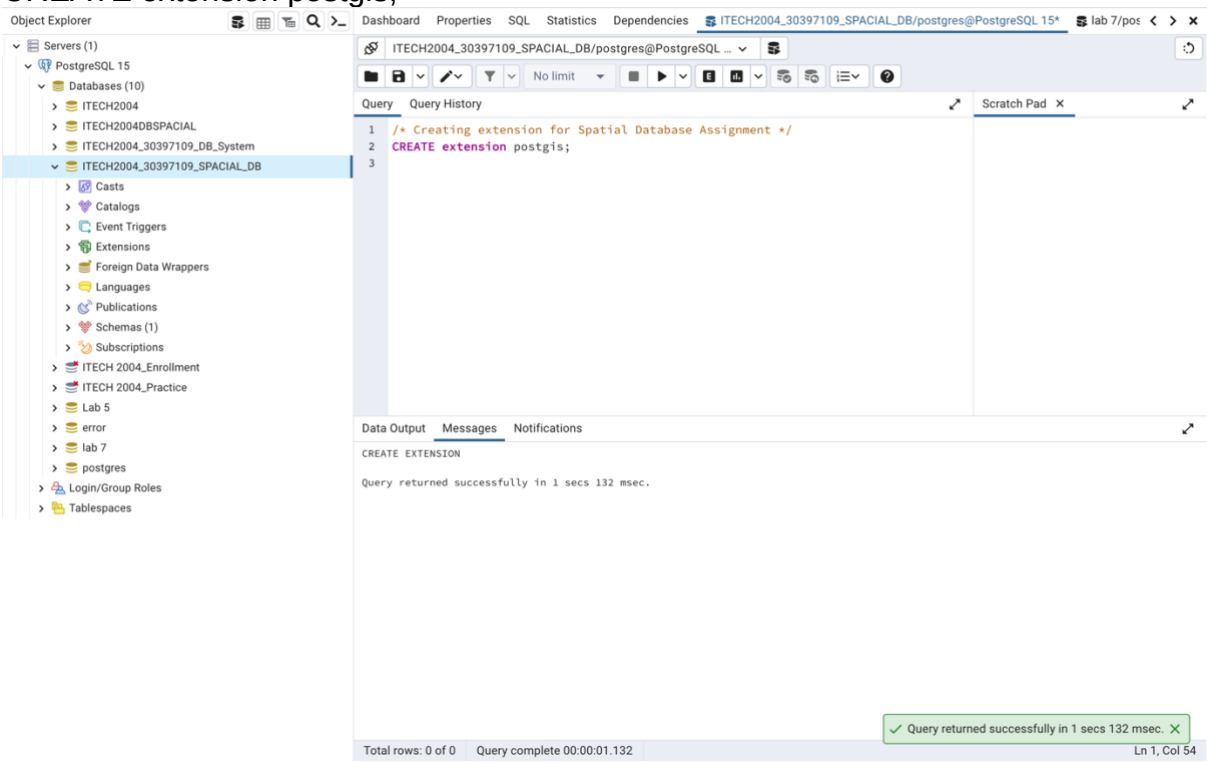

Figure 21 Creating PostGIS Extension (1)

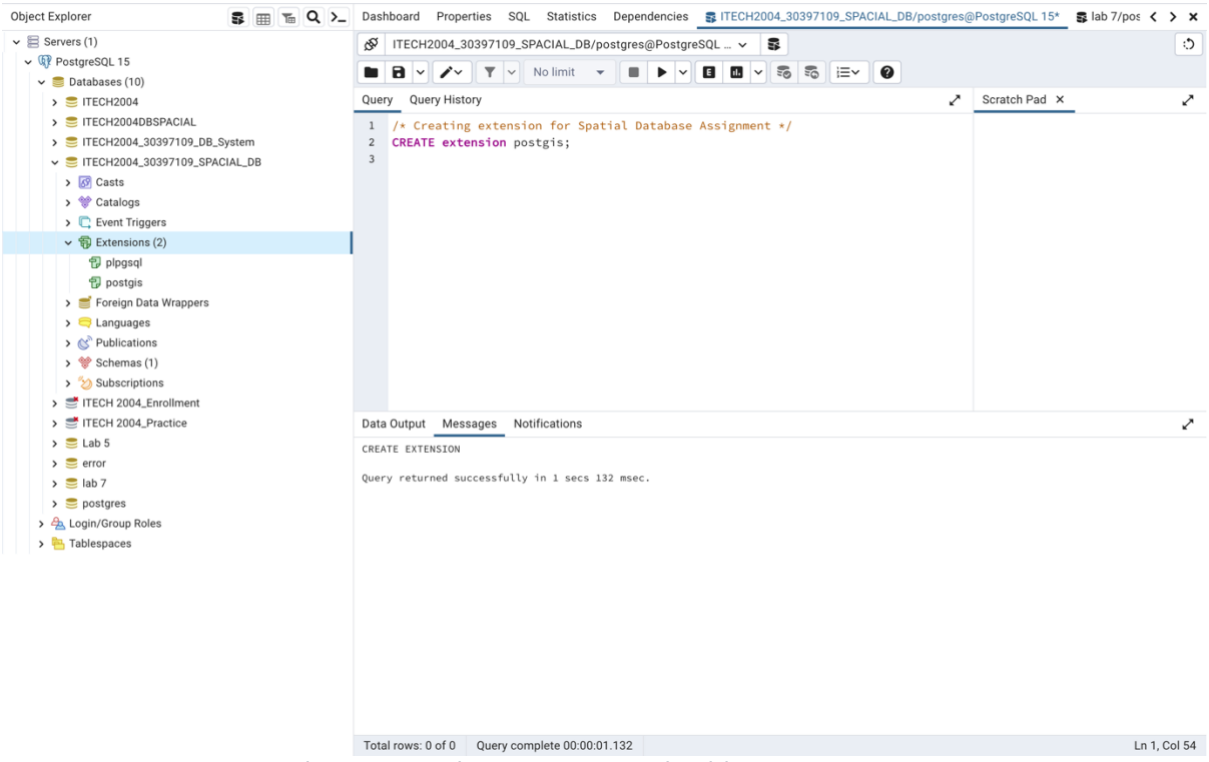

Figure 22 Creating PostGIS Extension (2)

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;
```

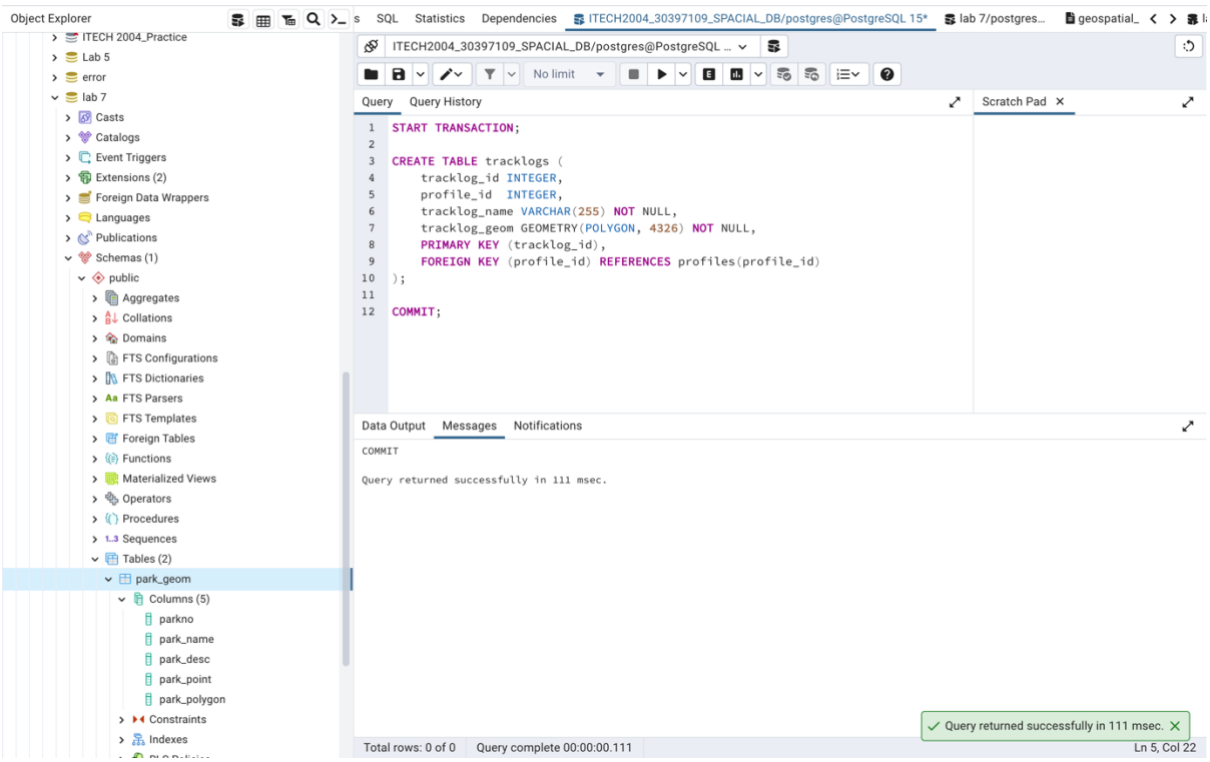

Figure 23 Spatial Tables(tracklogs) creation

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;

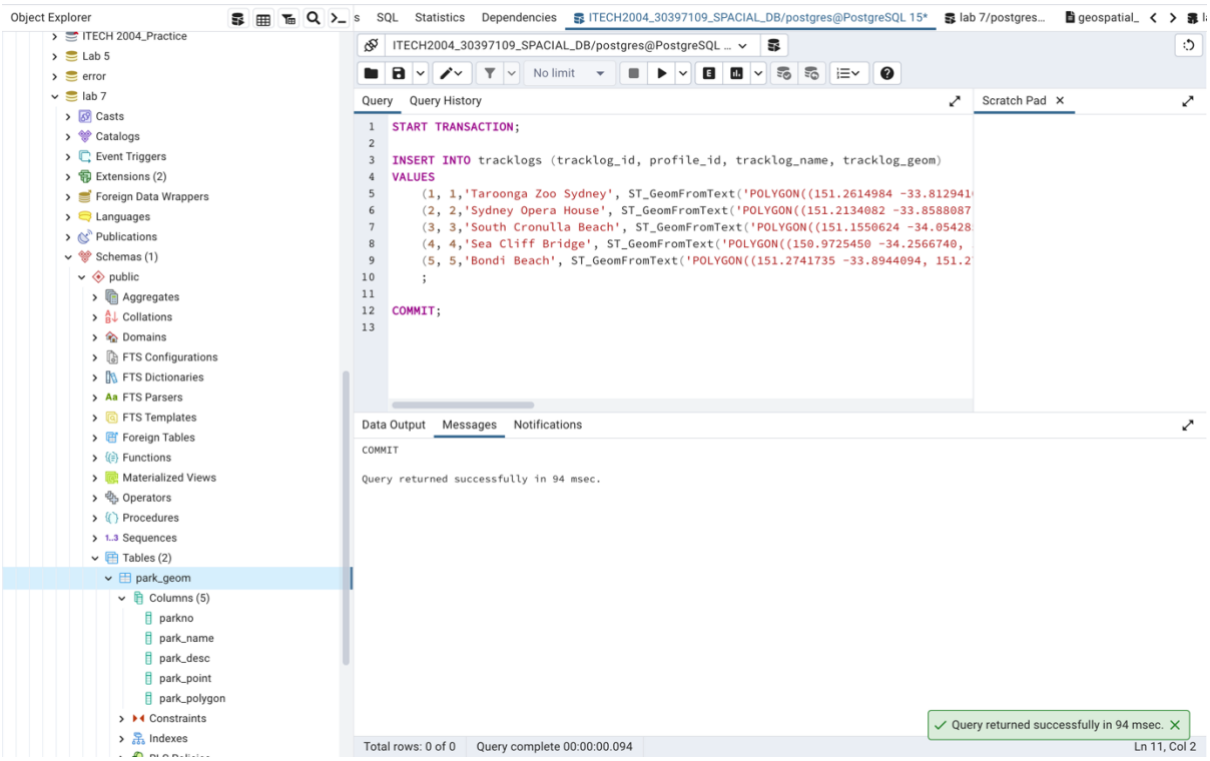

Figure 24 Spatial Data insertion

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);
```

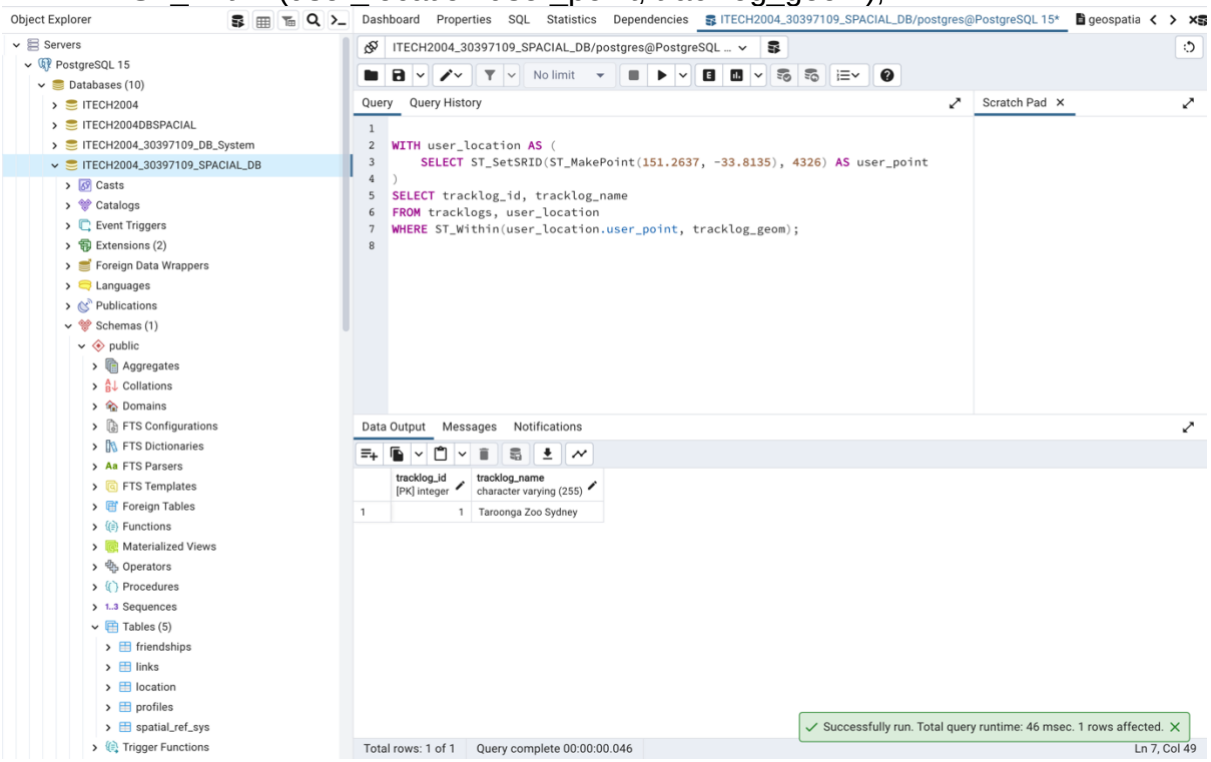

Figure 25 Retrieving a Tracklog within Taroonga Zoo Sydney

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);
```

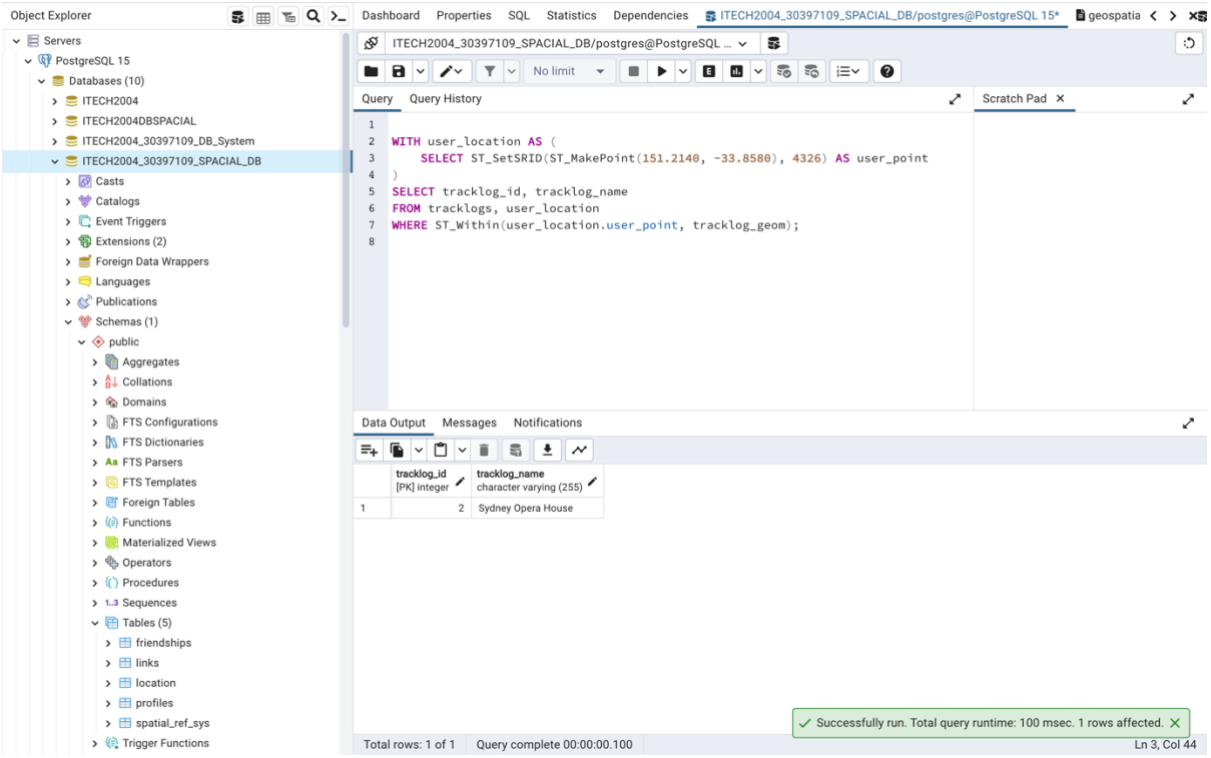

Figure 26 Retrieving a Tracklog within Sydney Opera House

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);
```

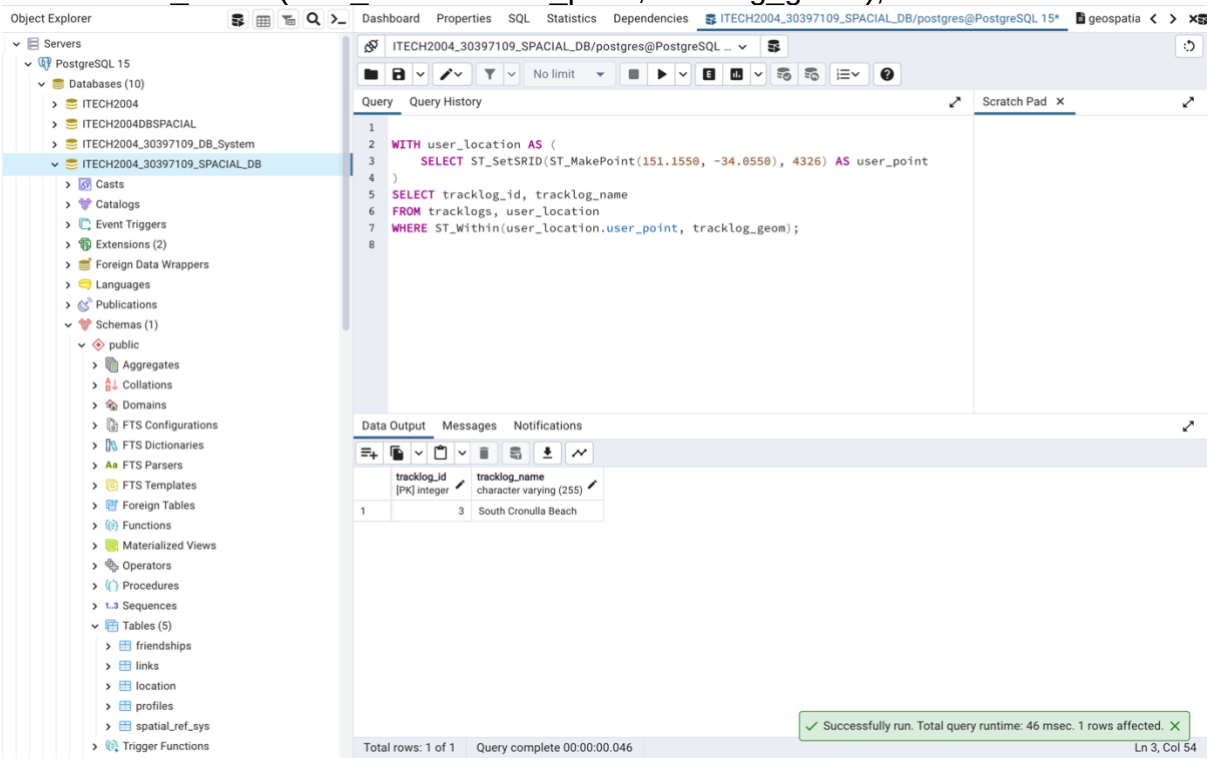

Figure 27 Retrieving a Tracklog within South Cronulla Beach

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);
```

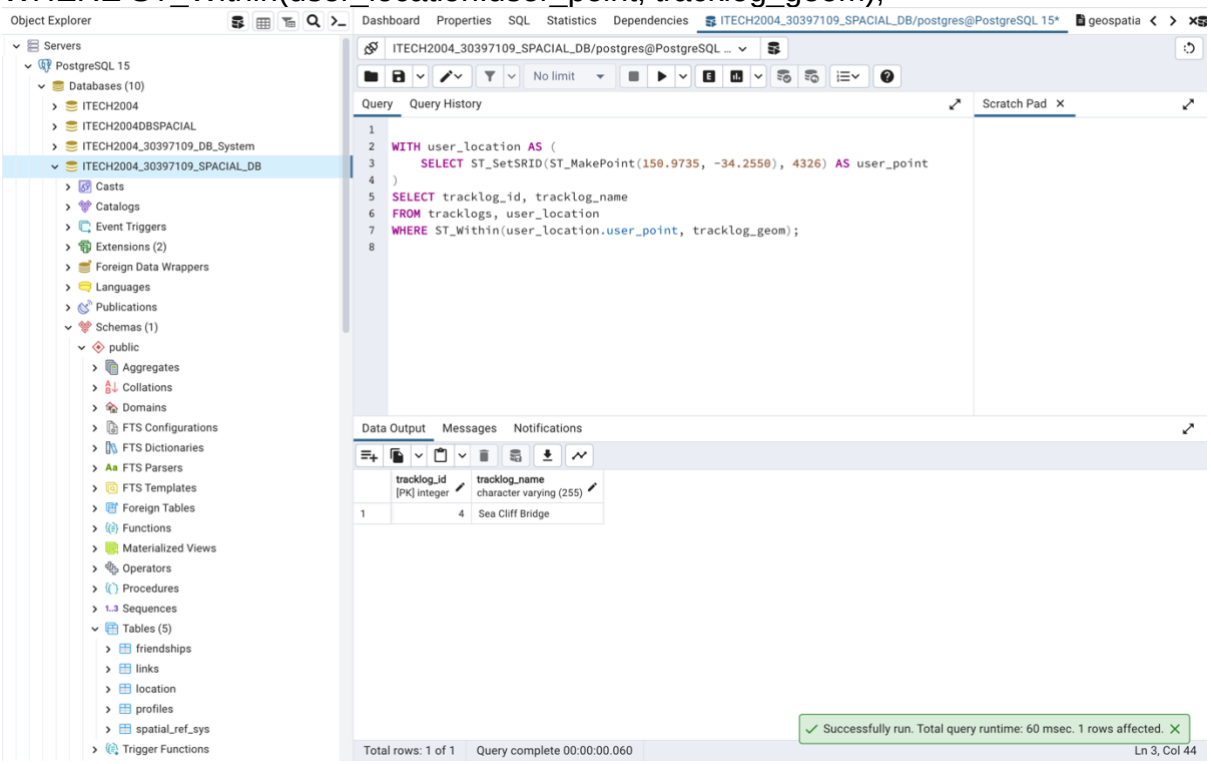

Figure 28 Retrieving a Tracklog within Sea Cliff Bridge

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);
```

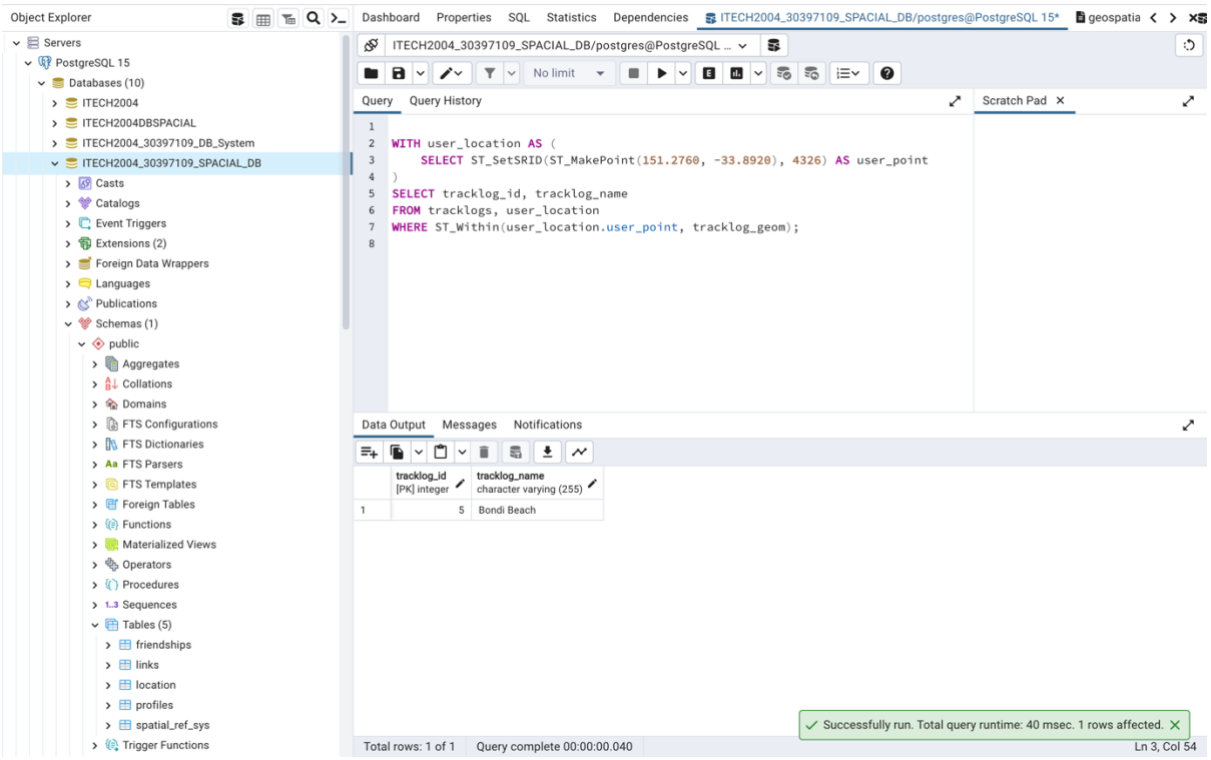

Figure 29 Retrieving a Tracklog within Bondi Beach

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;
```

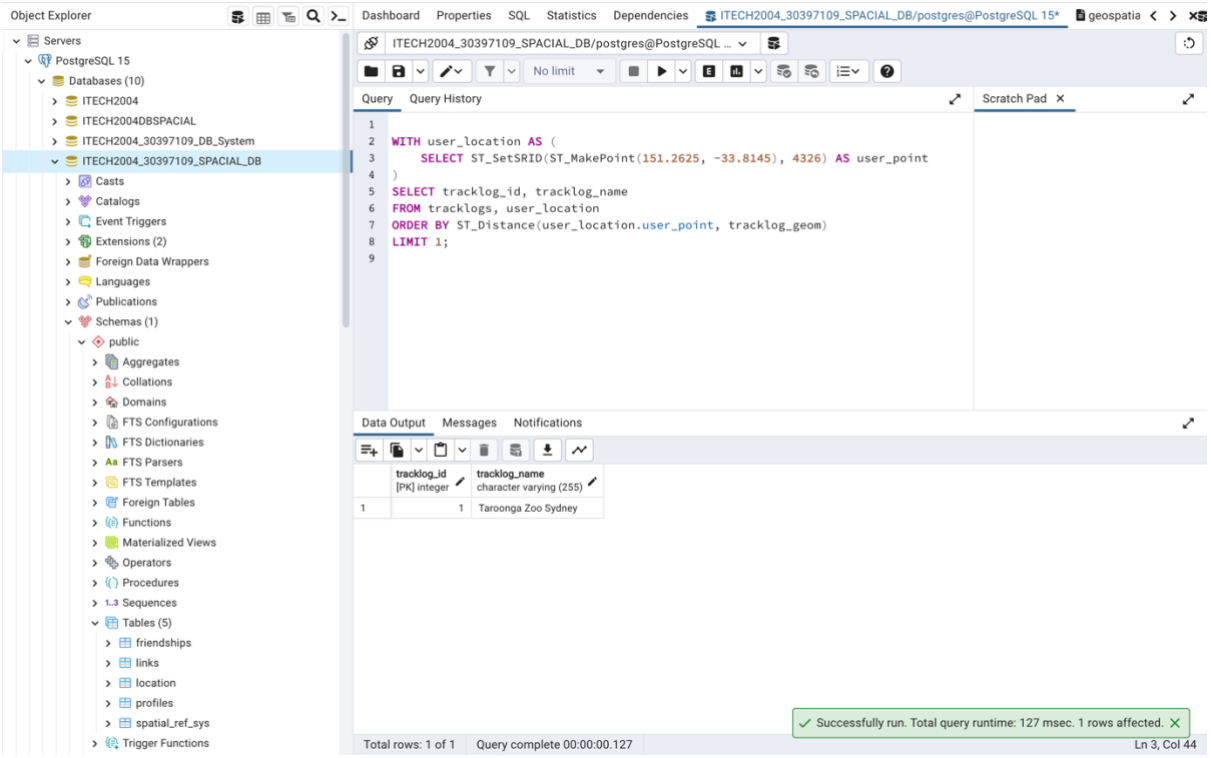

Figure 30 Retrieving a Tracklog nearest to Taroonga Zoo Sydney

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;
```

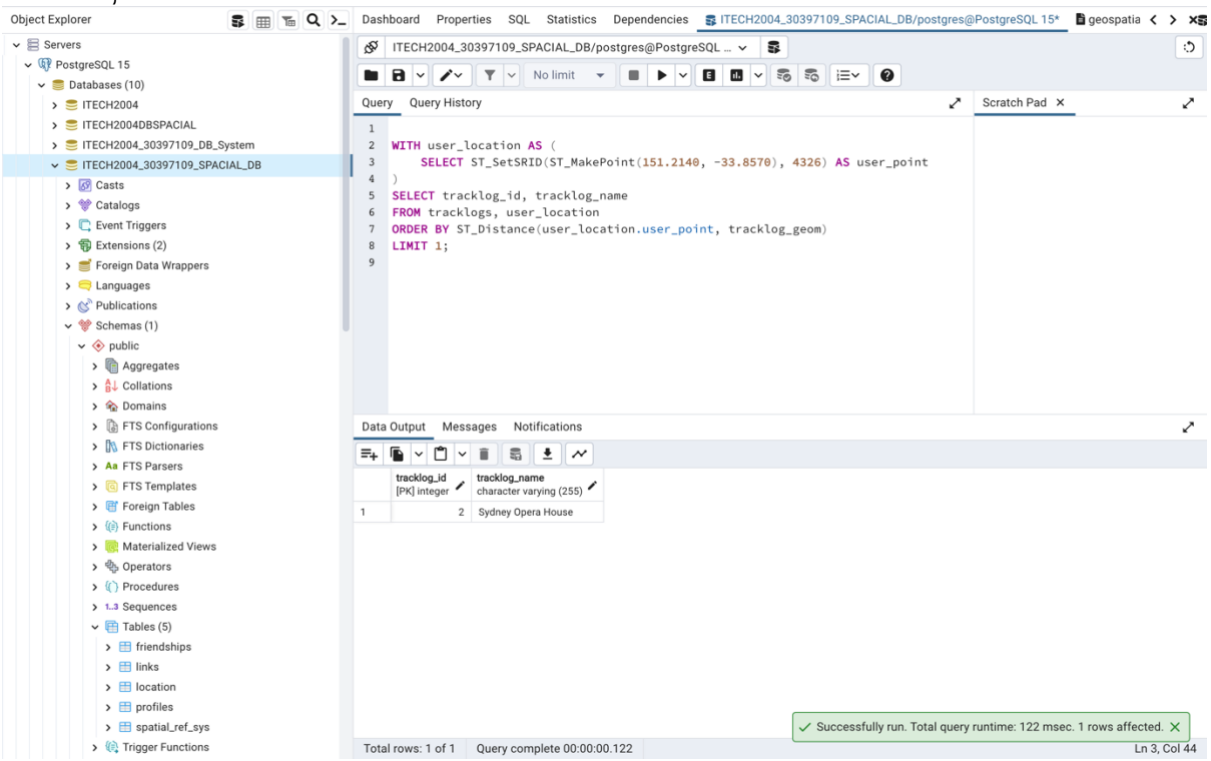

Figure 31 Retrieving a Tracklog nearest to Sydney Opera House

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;
```

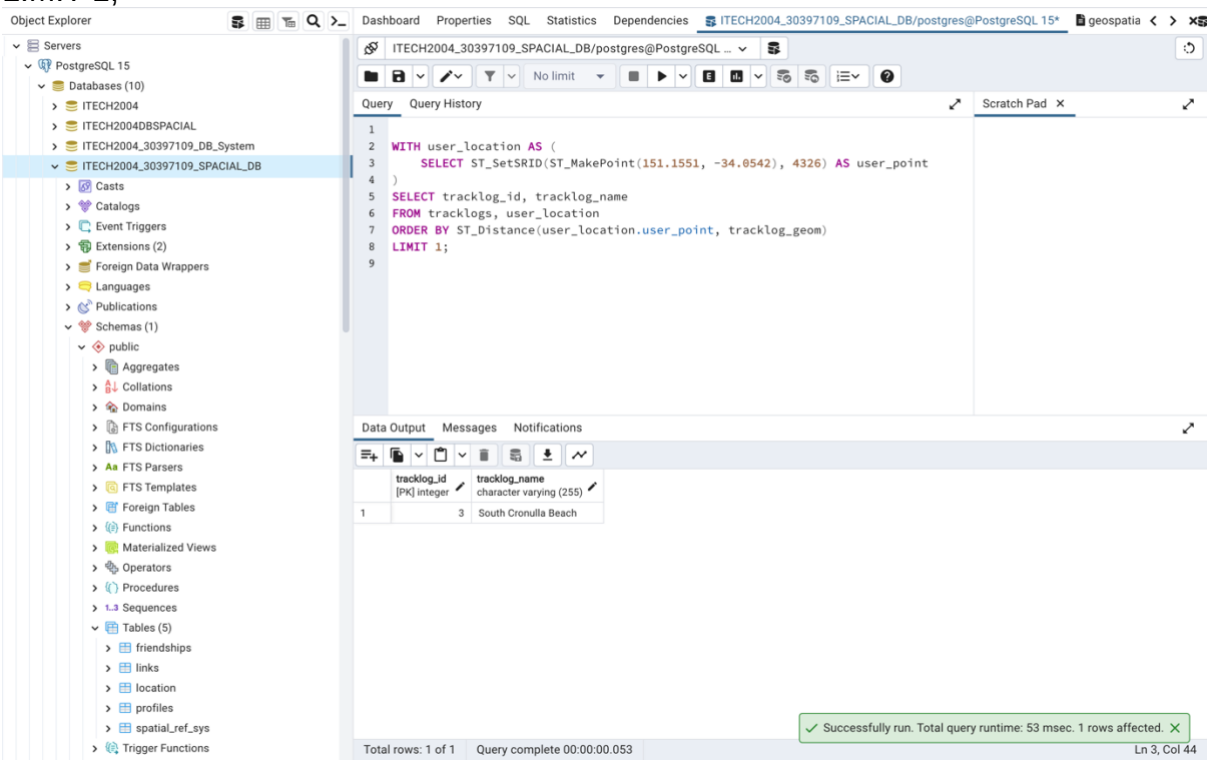

Figure 32 Retrieving a Tracklog nearest to South Cronulla Beach

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;
```

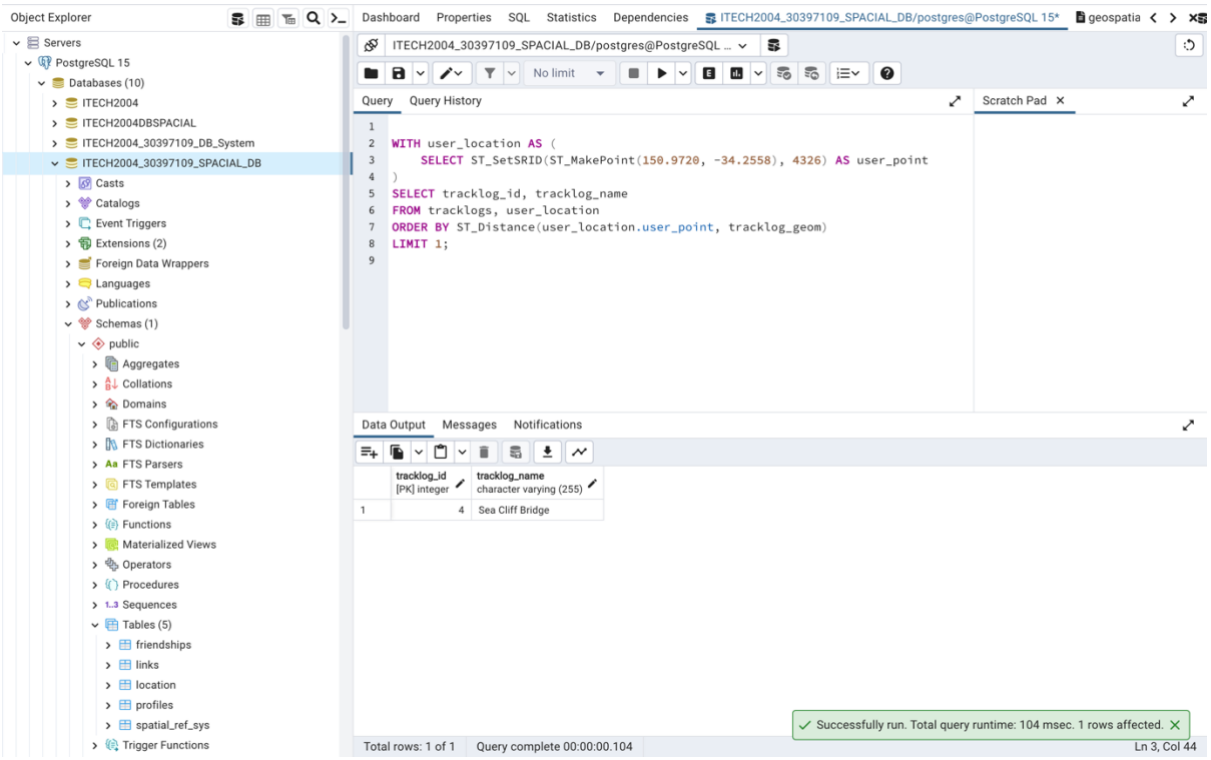

Figure 33 Retrieving a Tracklog nearest to Sea Cliff Bridge

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;
```

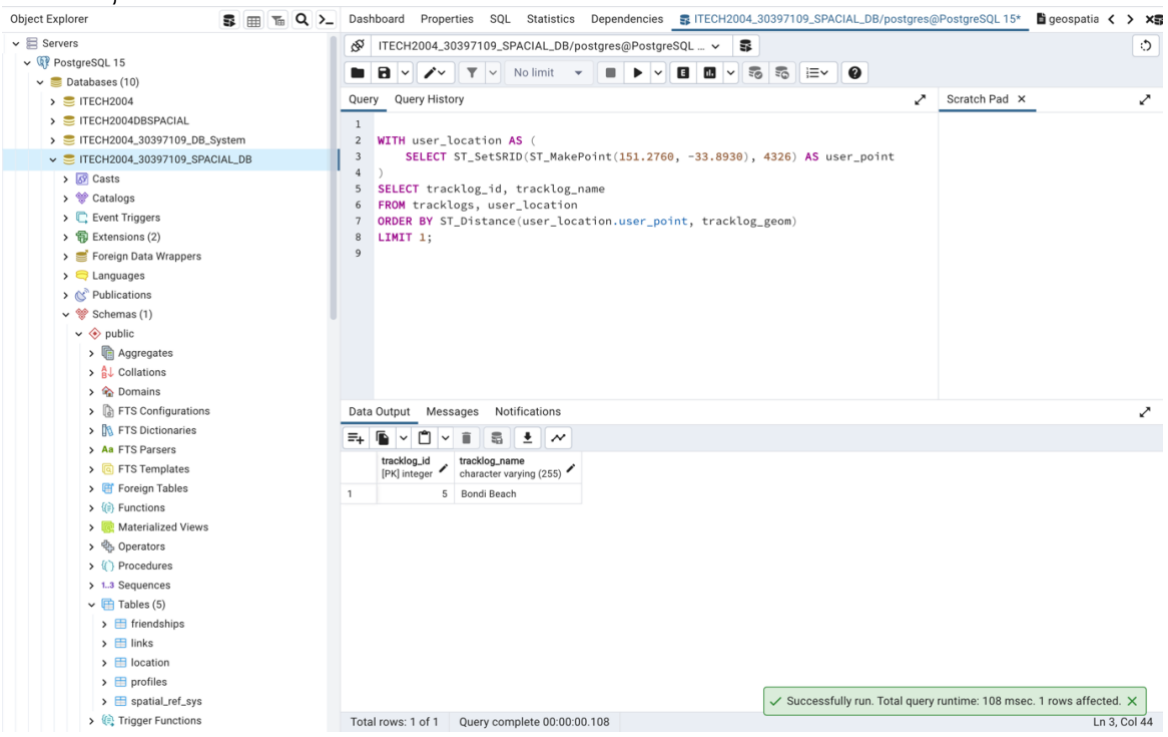

Figure 34 Retrieving a Tracklog nearest to Bondi Beach

3. Document NoSQL database (MongoDB)

3.1 Database Setup, Collection, and Document

3.1.1 Database Setup and Collection

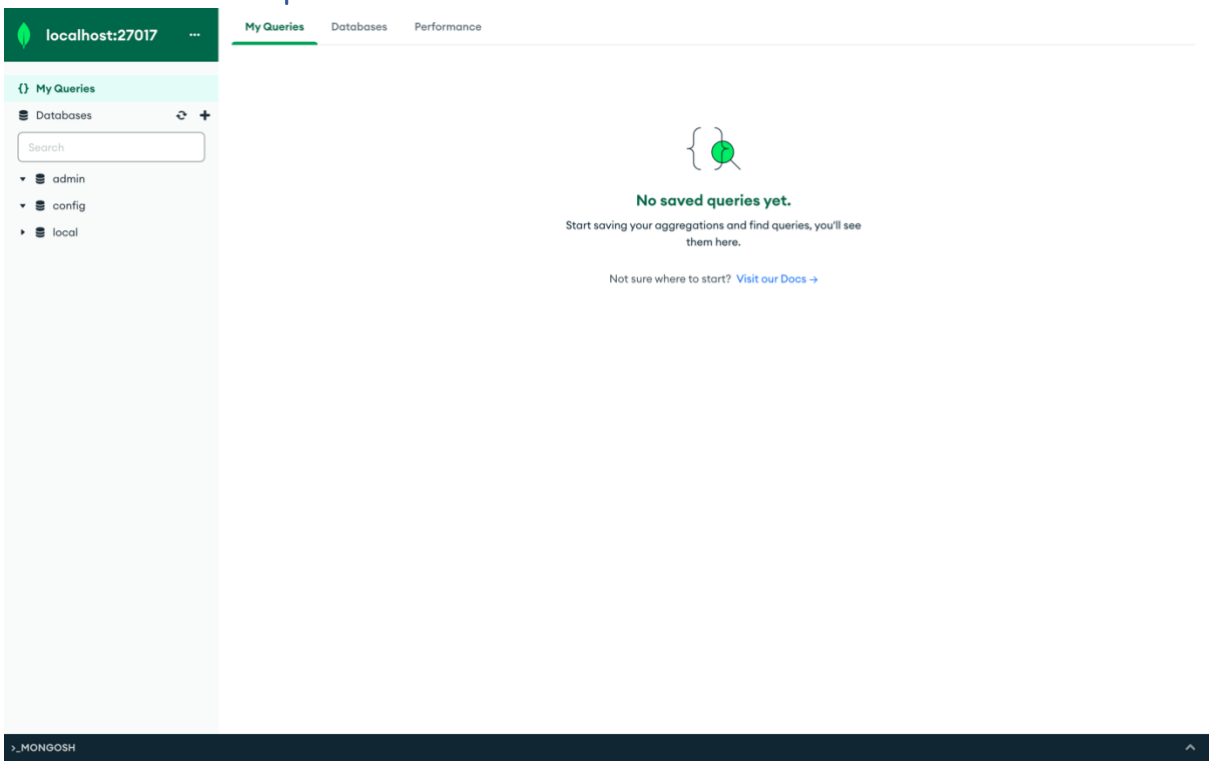

Figure 35 Database Setup and Collection (1)

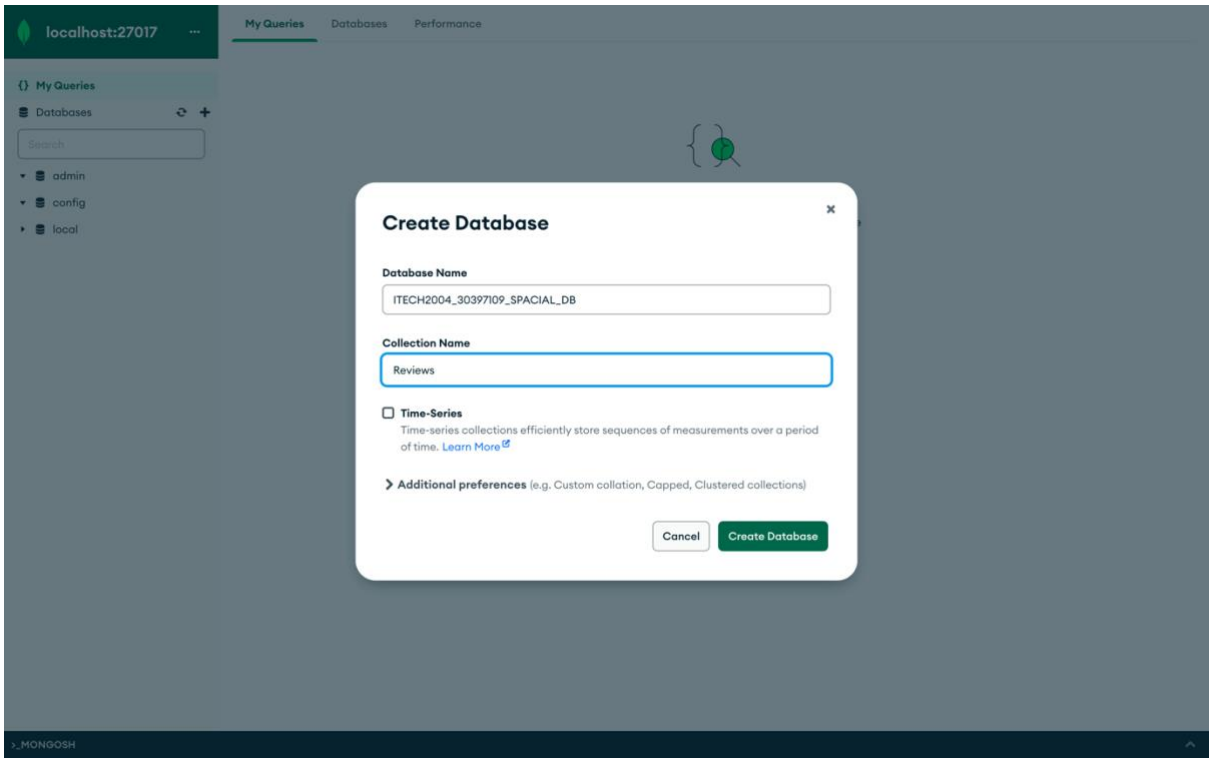

Figure 36 Database Setup and Collection (2)

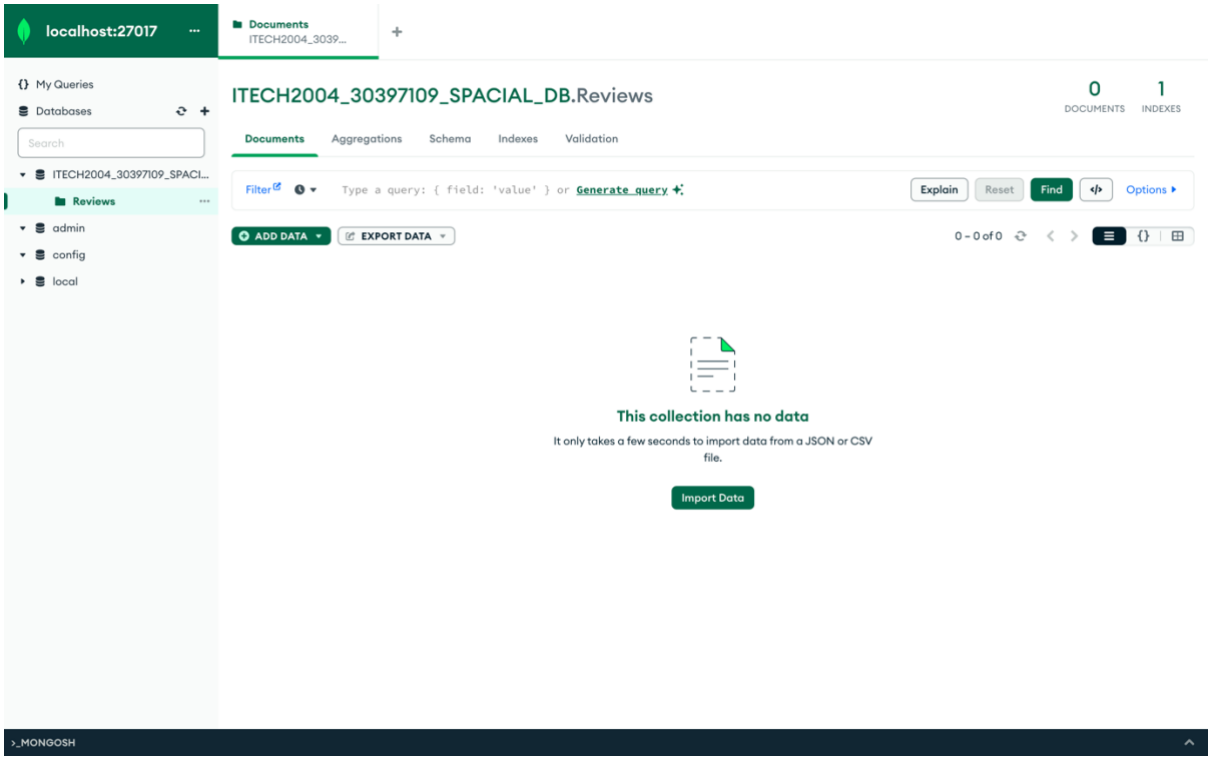

Figure 37 Database Setup and Collection (3)

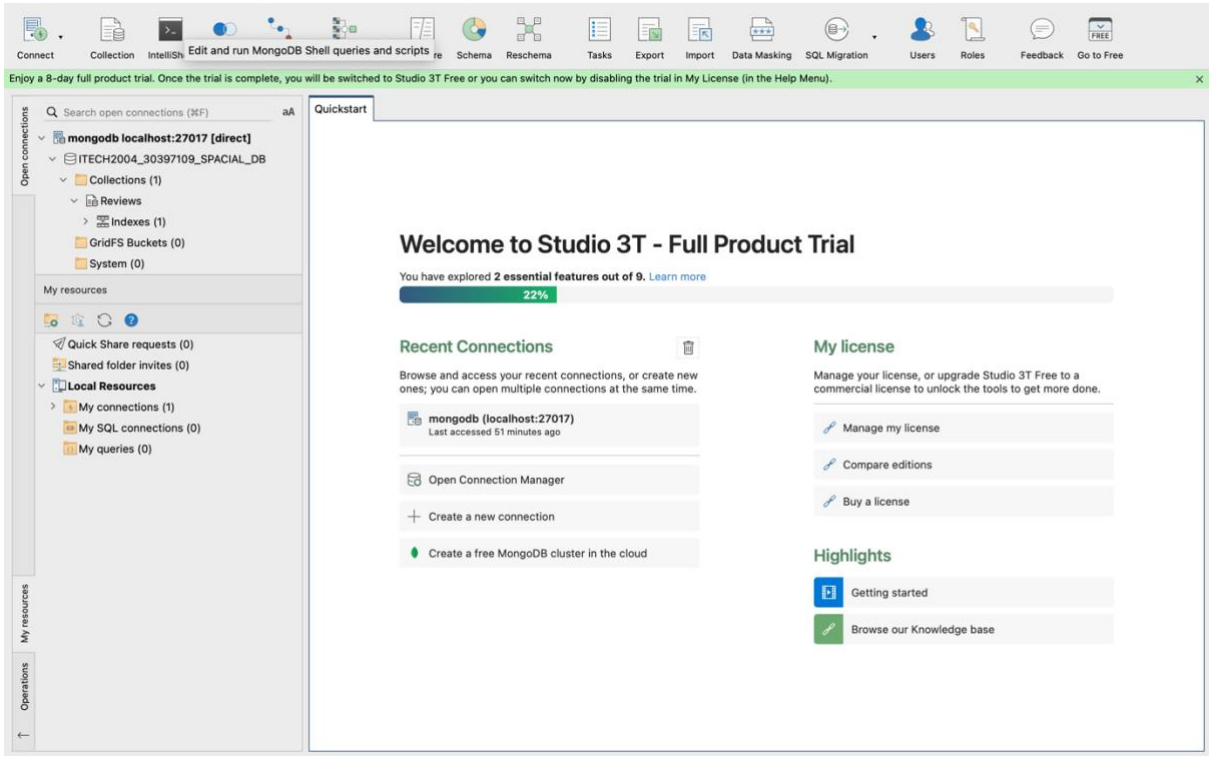

Figure 38 Database Setup and Collection (4)

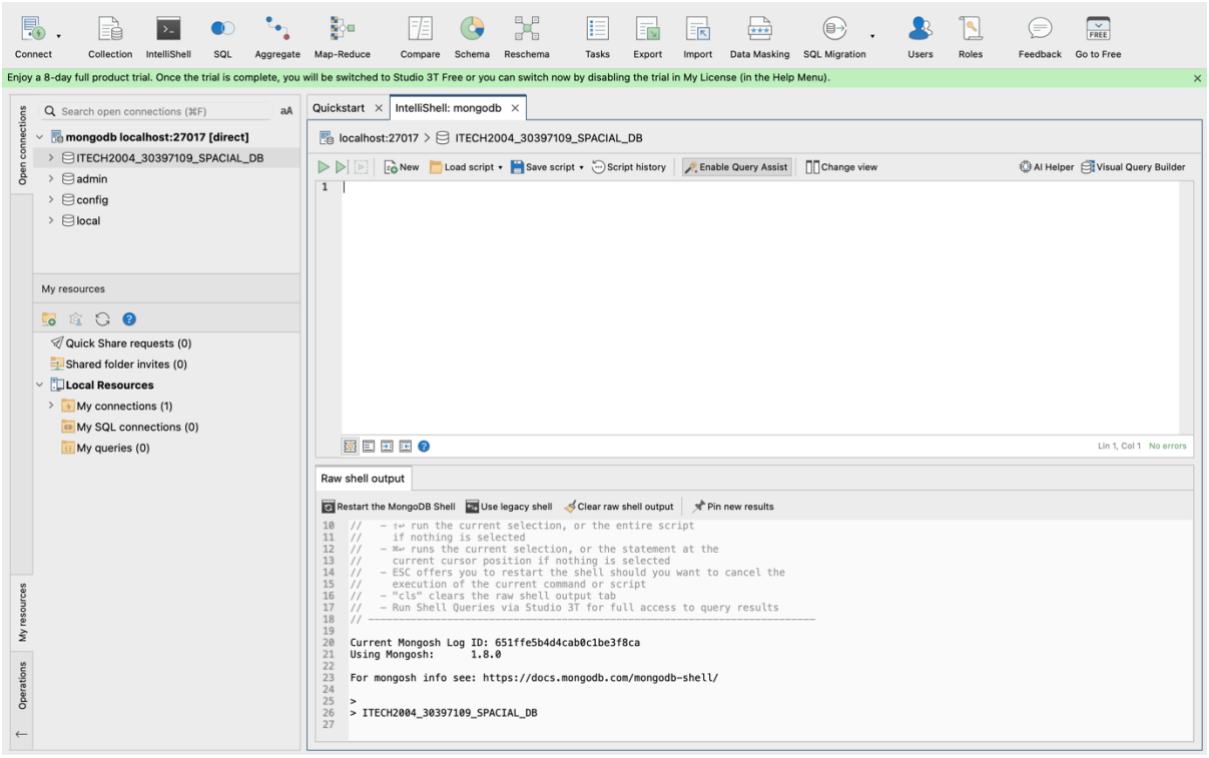

Figure 39 Database Setup and Collection (5)

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
```

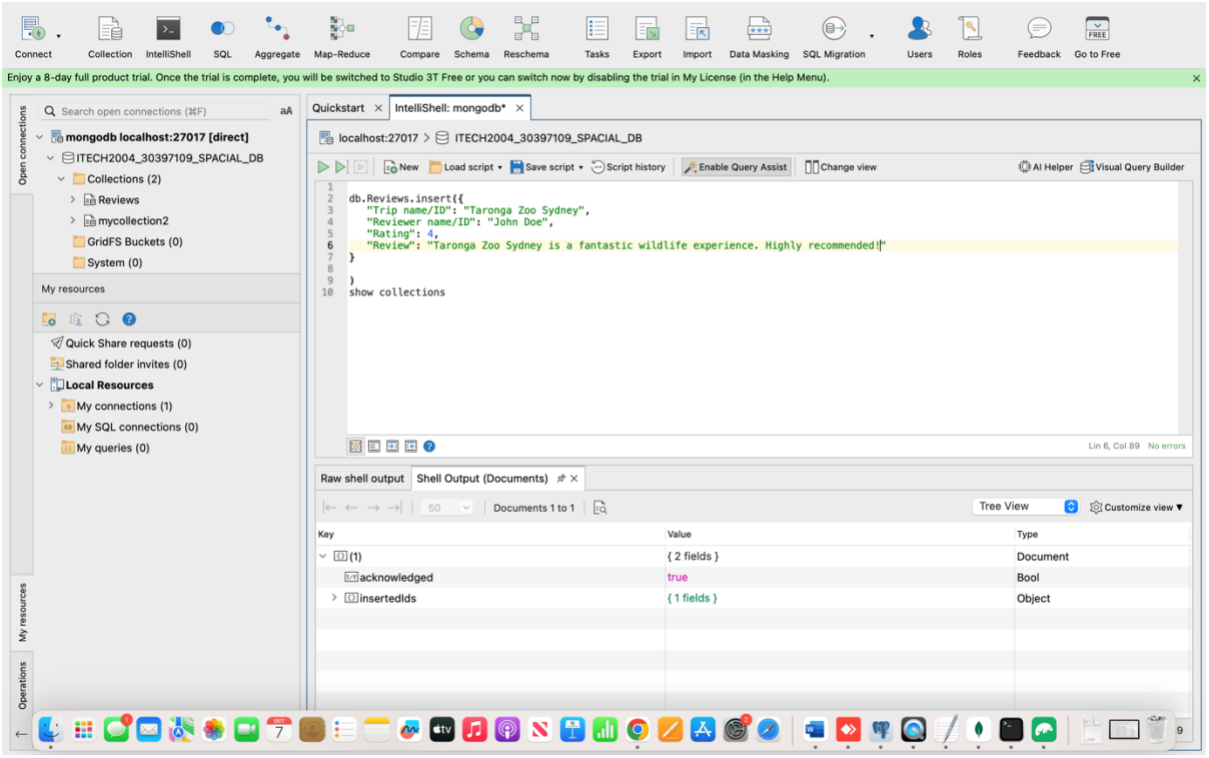

Figure 40 Documents (Reviews) of John Doe

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

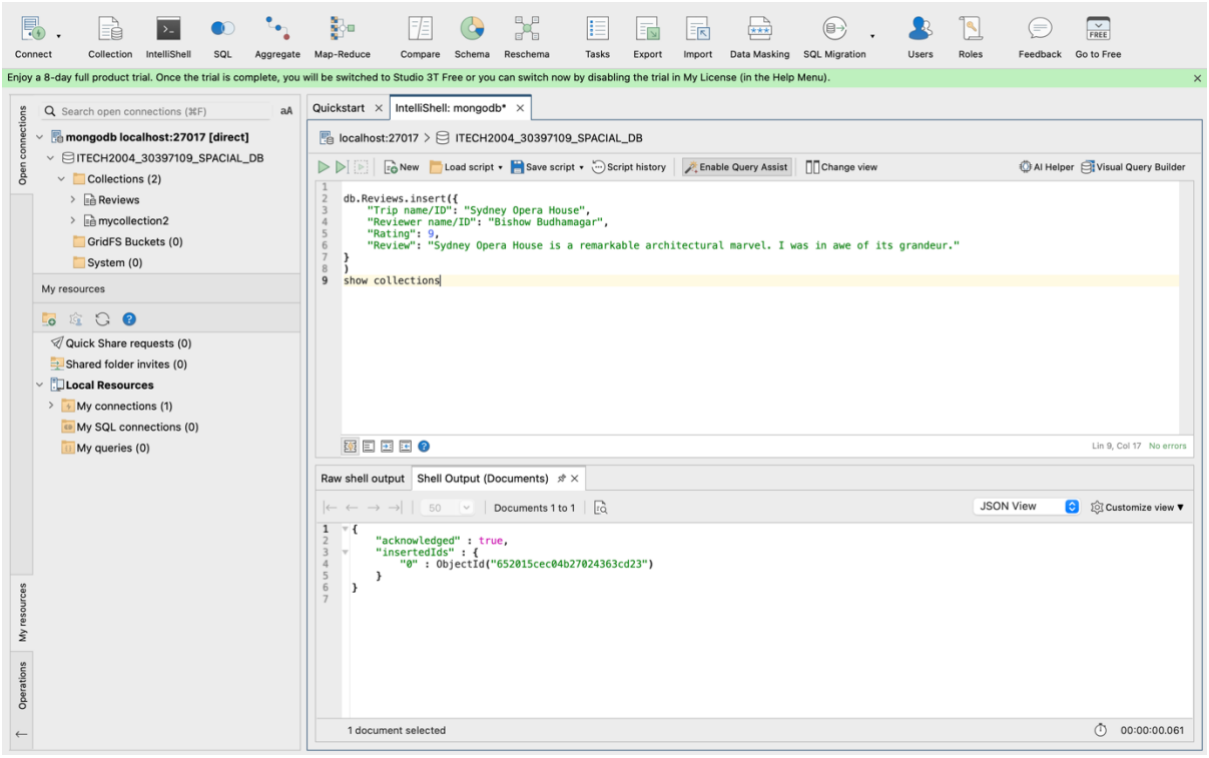

Figure 41 Documents (Reviews) of Bishow Budhamagar

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

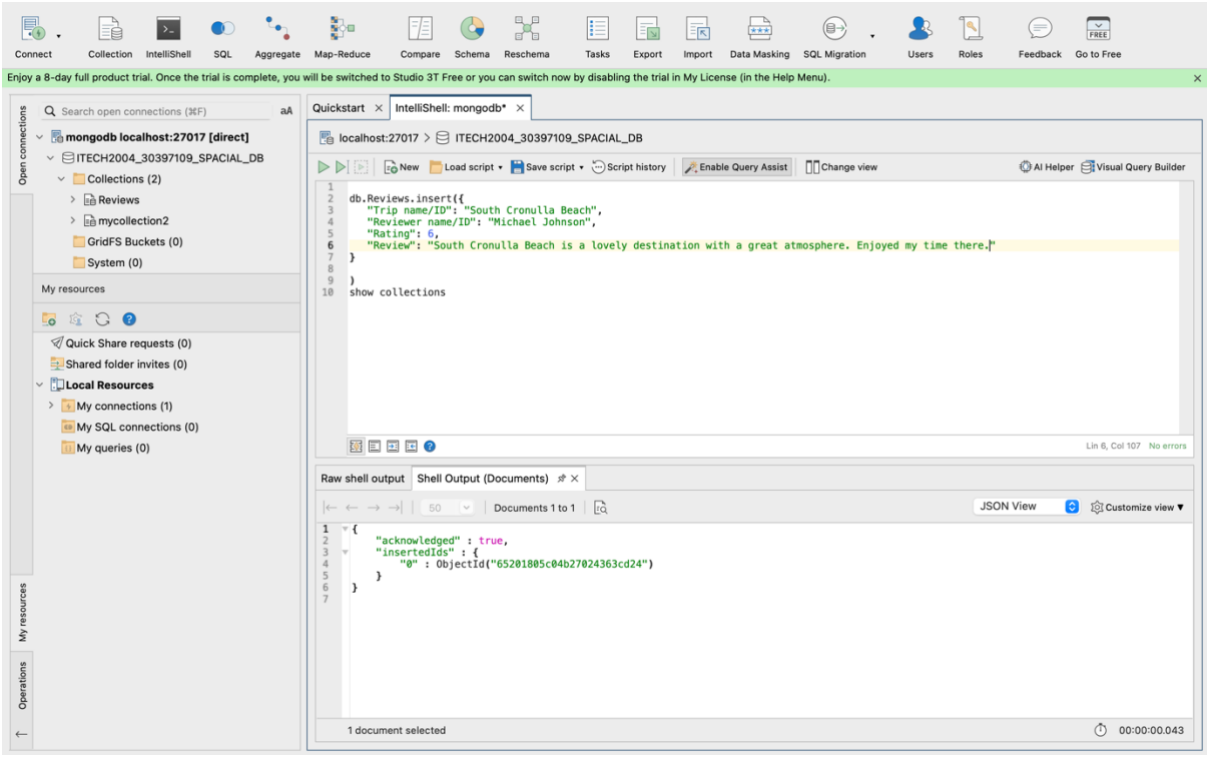

Figure 42 Documents (Reviews) of Michael Johnson

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

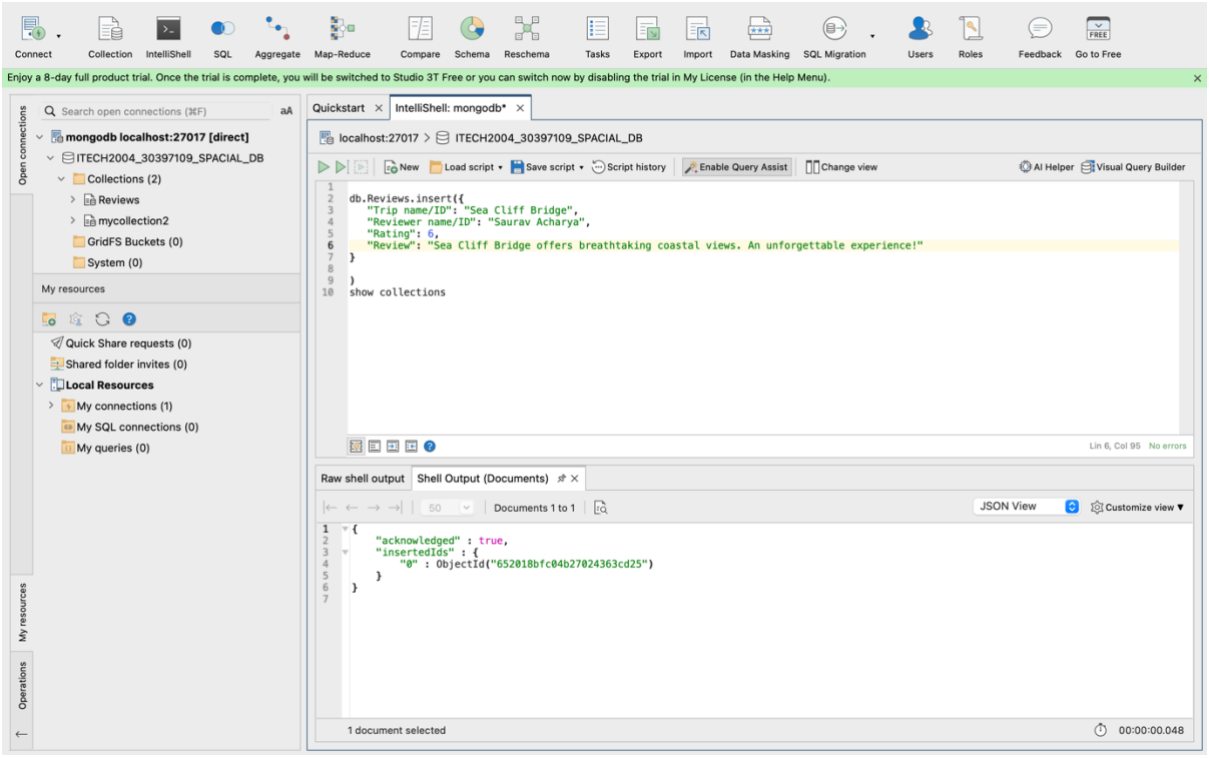

Figure 43 Documents (Reviews) of Saurav Acharya

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

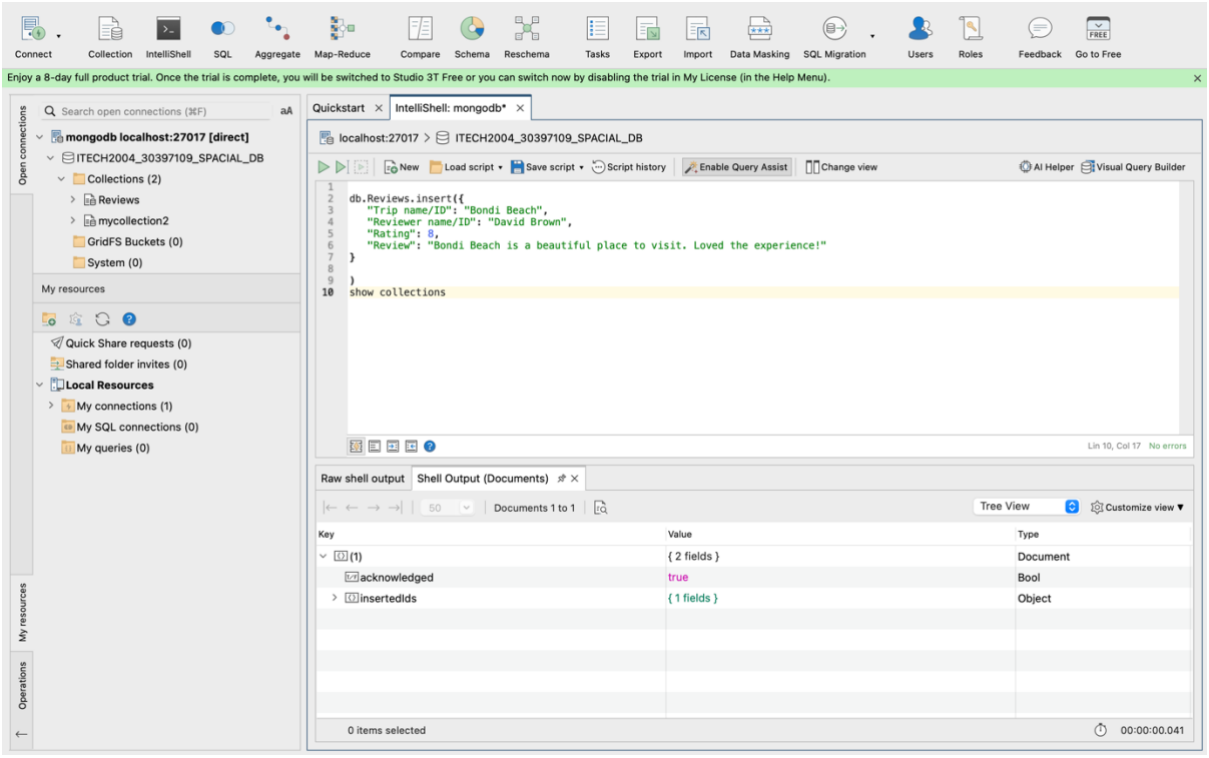

Figure 44 Documents (Reviews) of David Brown

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
```

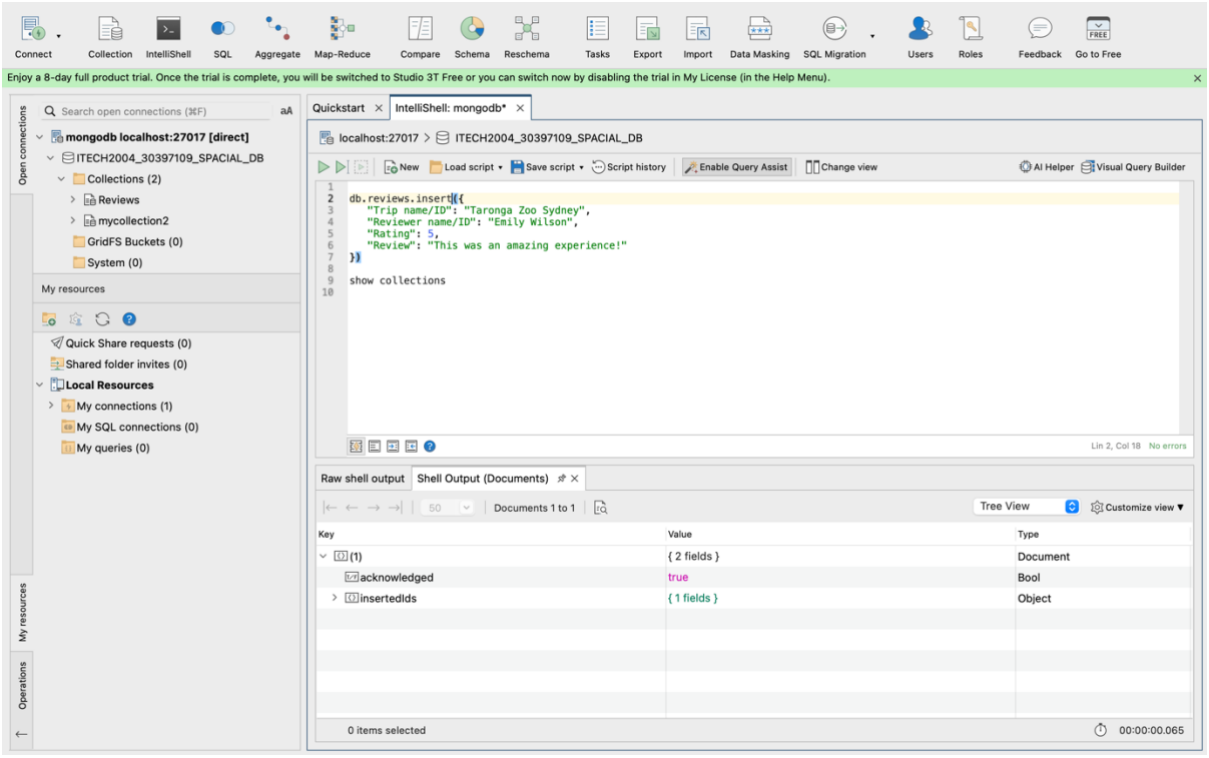

Figure 45 Inserting new review into database of Emily Wilson

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

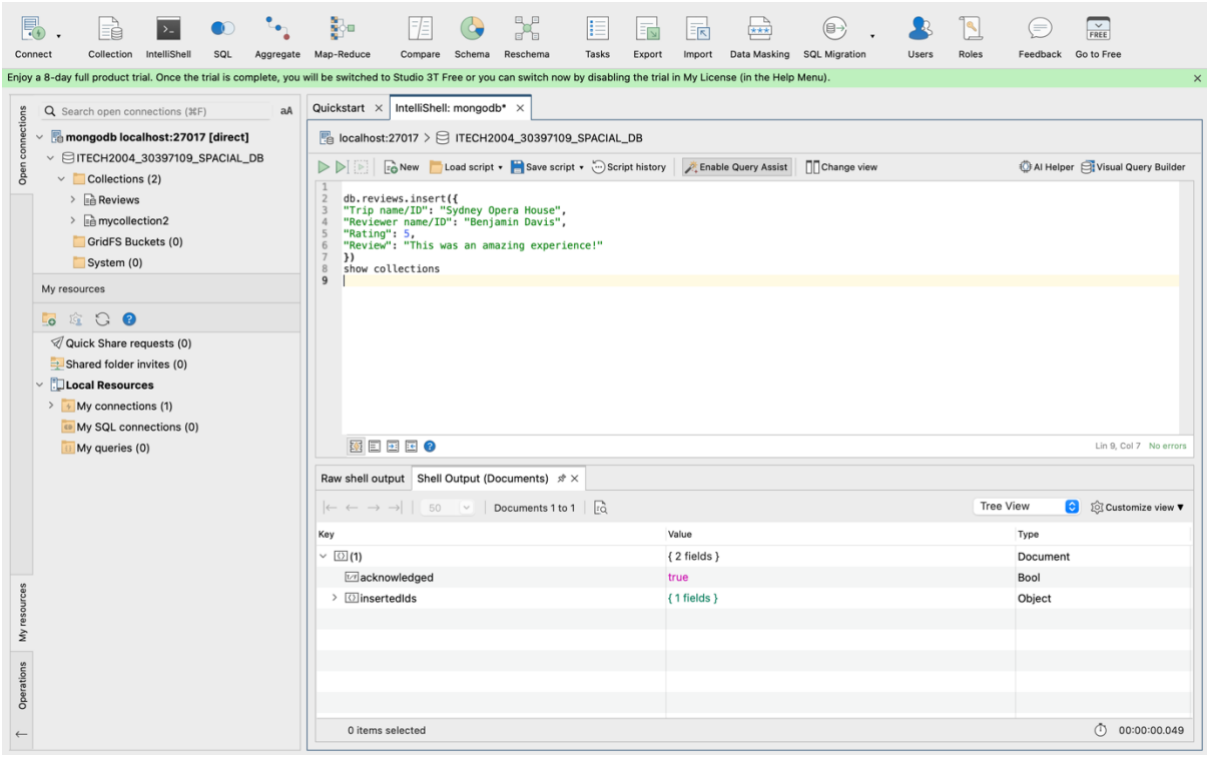

Figure 46 Inserting new review into database of Benjamin Davis

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
```

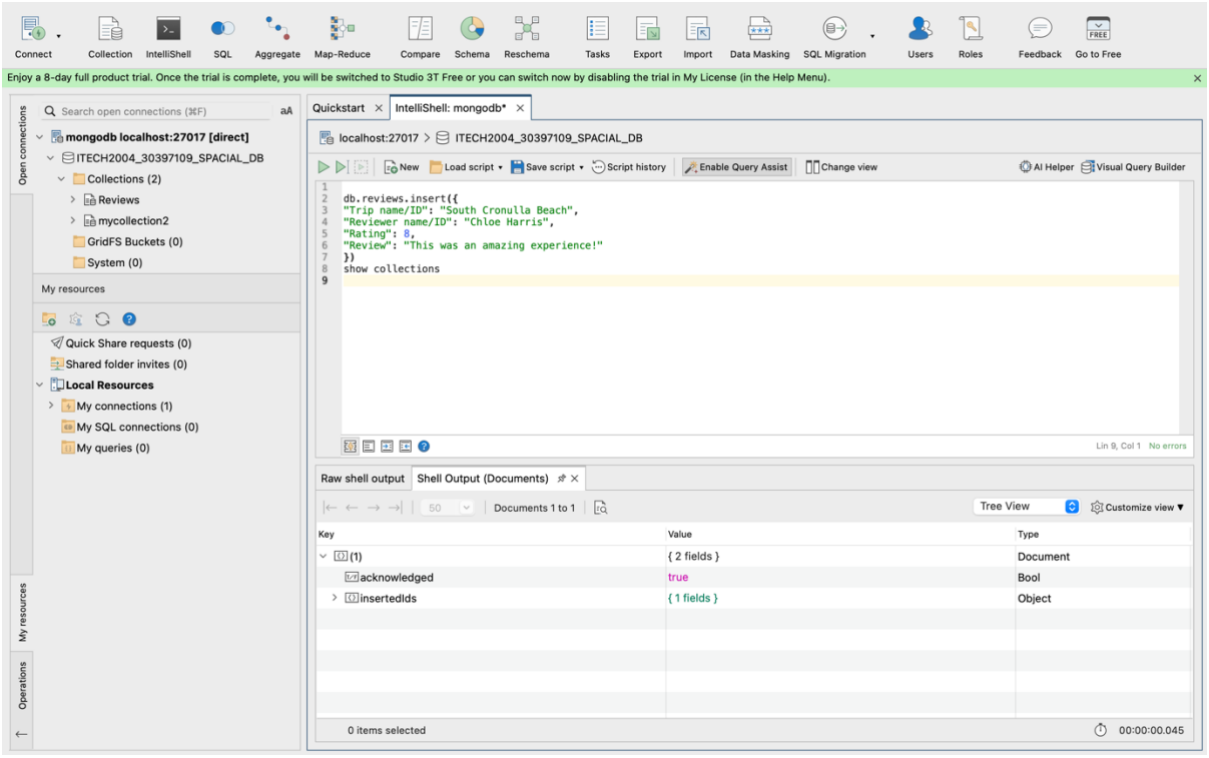

Figure 47 Inserting new review into database of Chloe Harris

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
```

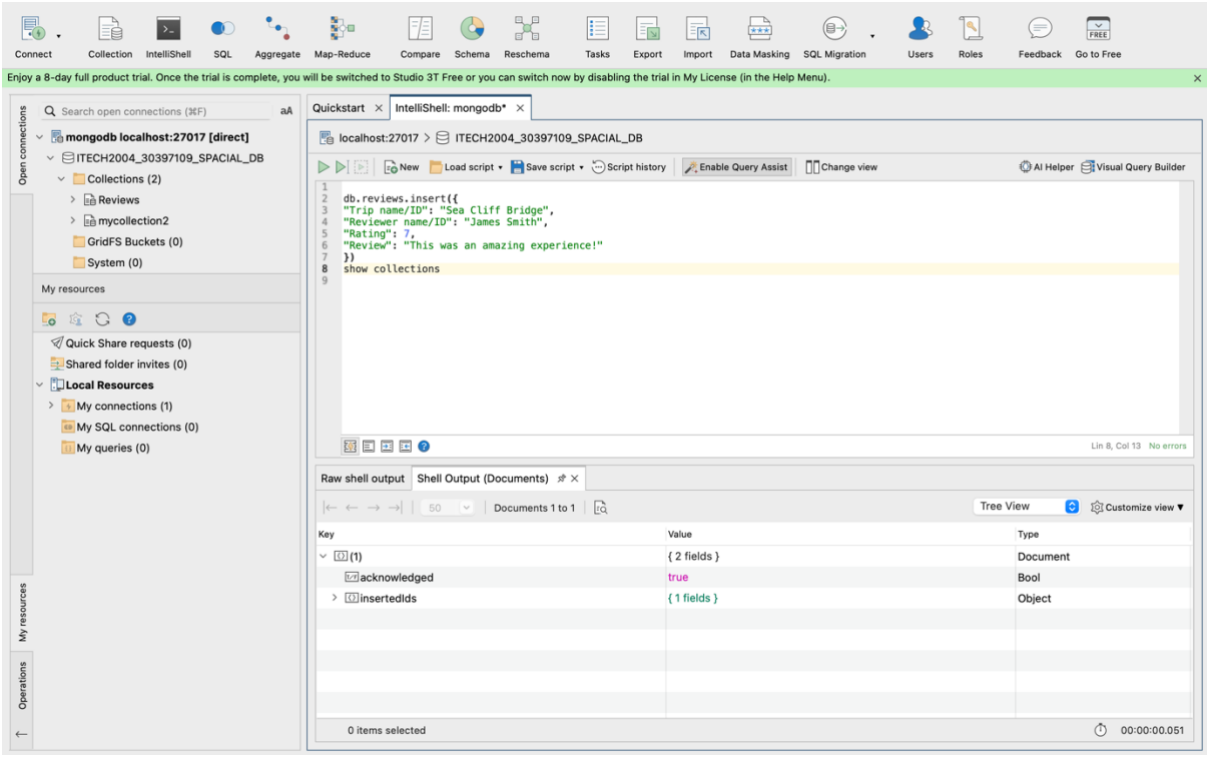

Figure 48 Inserting new review into database of James Smith

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

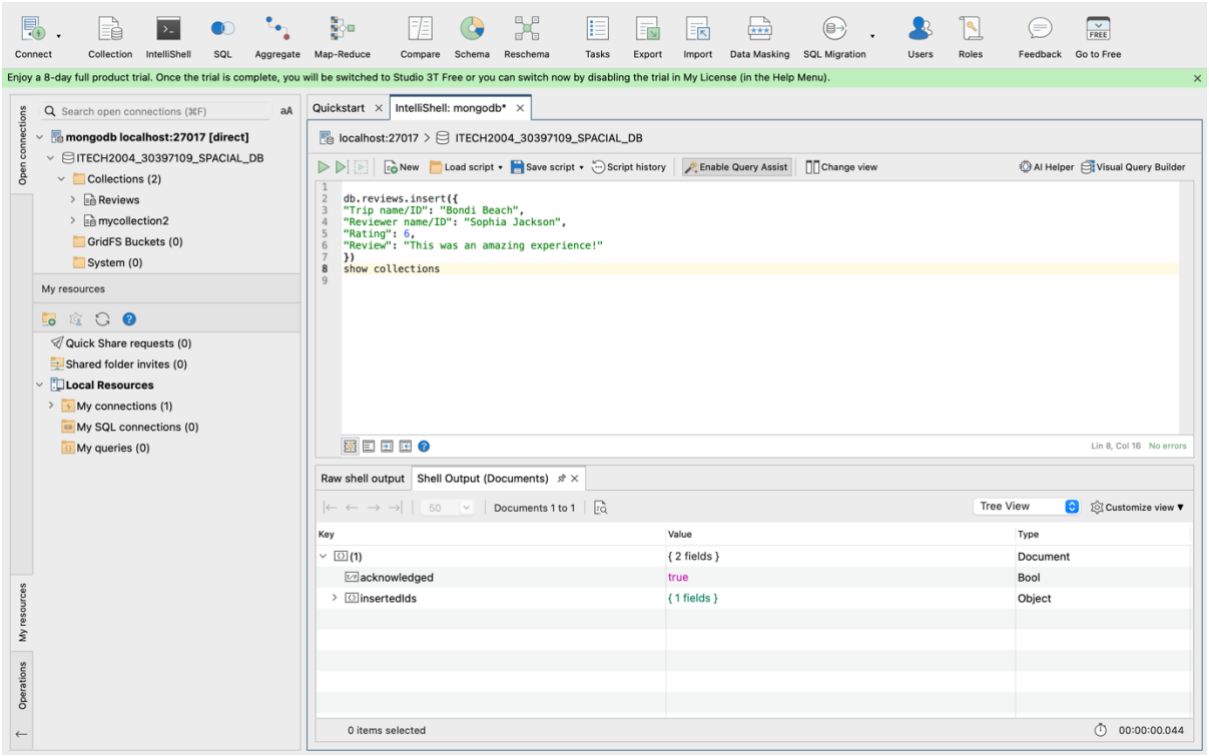

Figure 49 Inserting new review into database of Sophia Jackson

3.3 Retrieving highest rated review

For Taronga Zoo Sydney

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

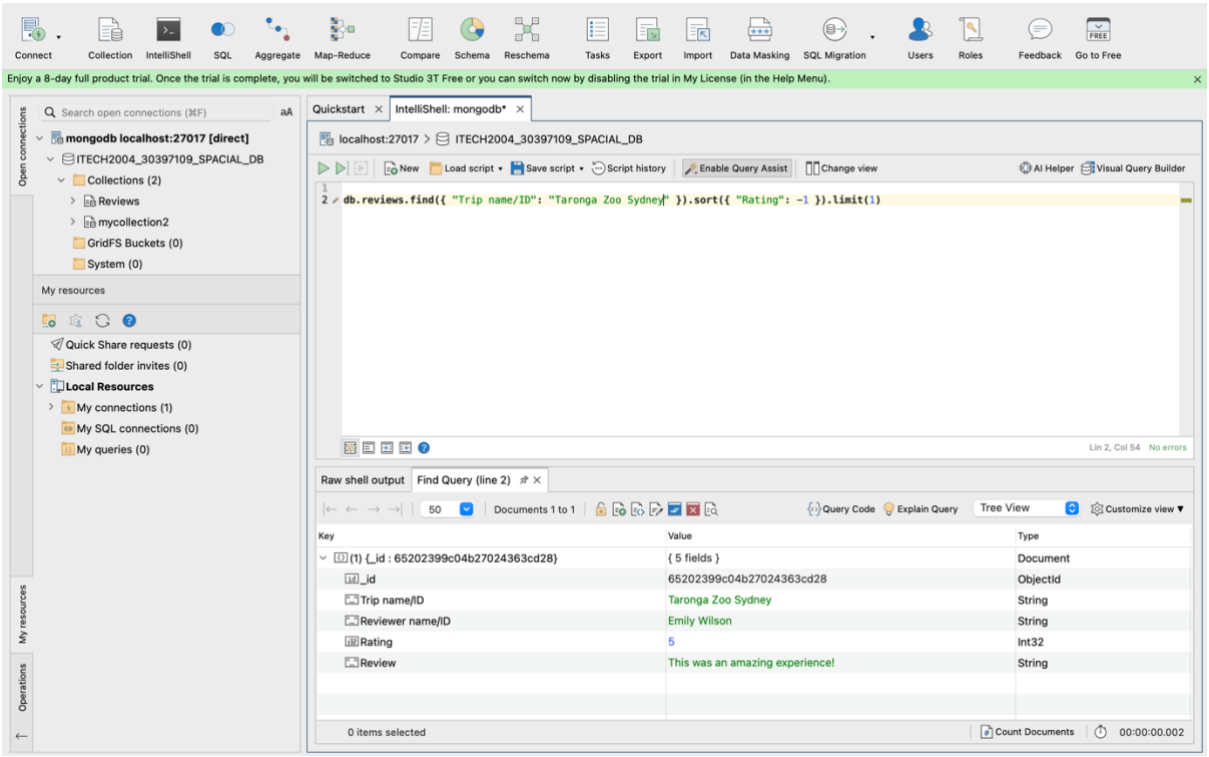

Figure 50 Retrieving highest rated review for Taronga Zoo Sydney

For Sydney Opera House

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

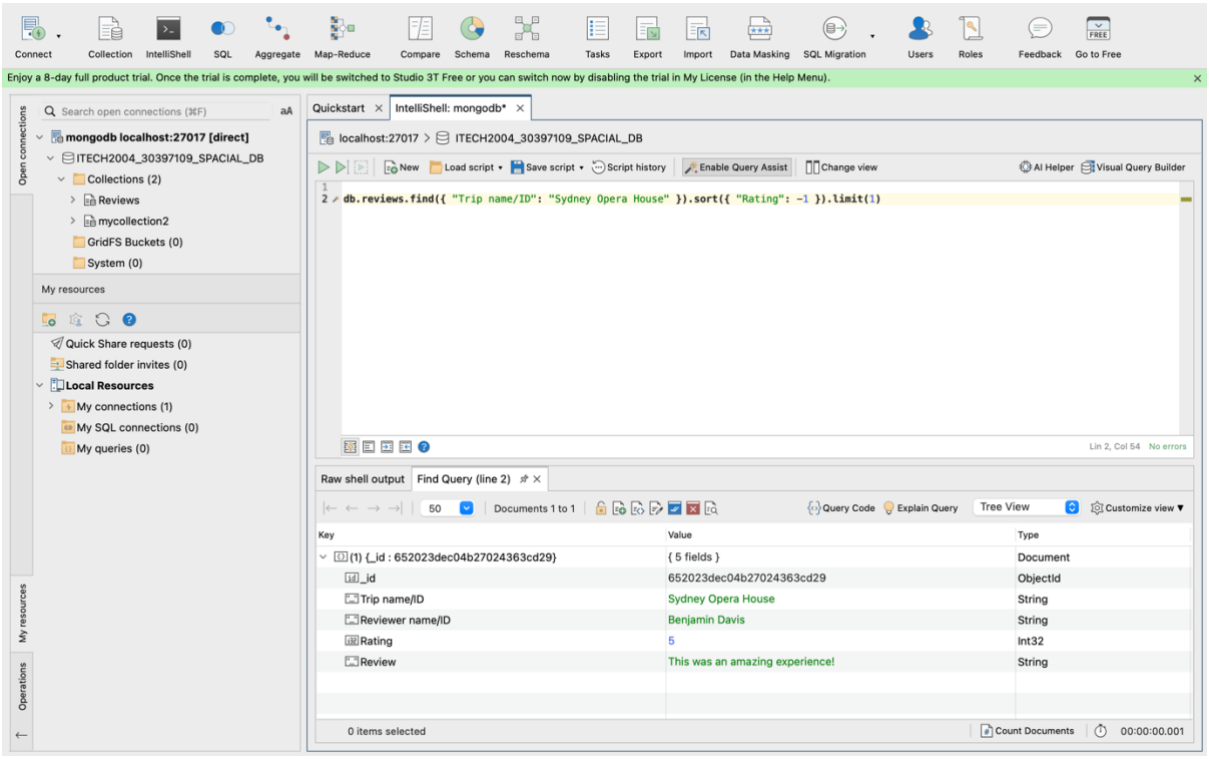

Figure 51Retrieving highest rated review for Sydney Opera House

For South Cronulla Beach

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

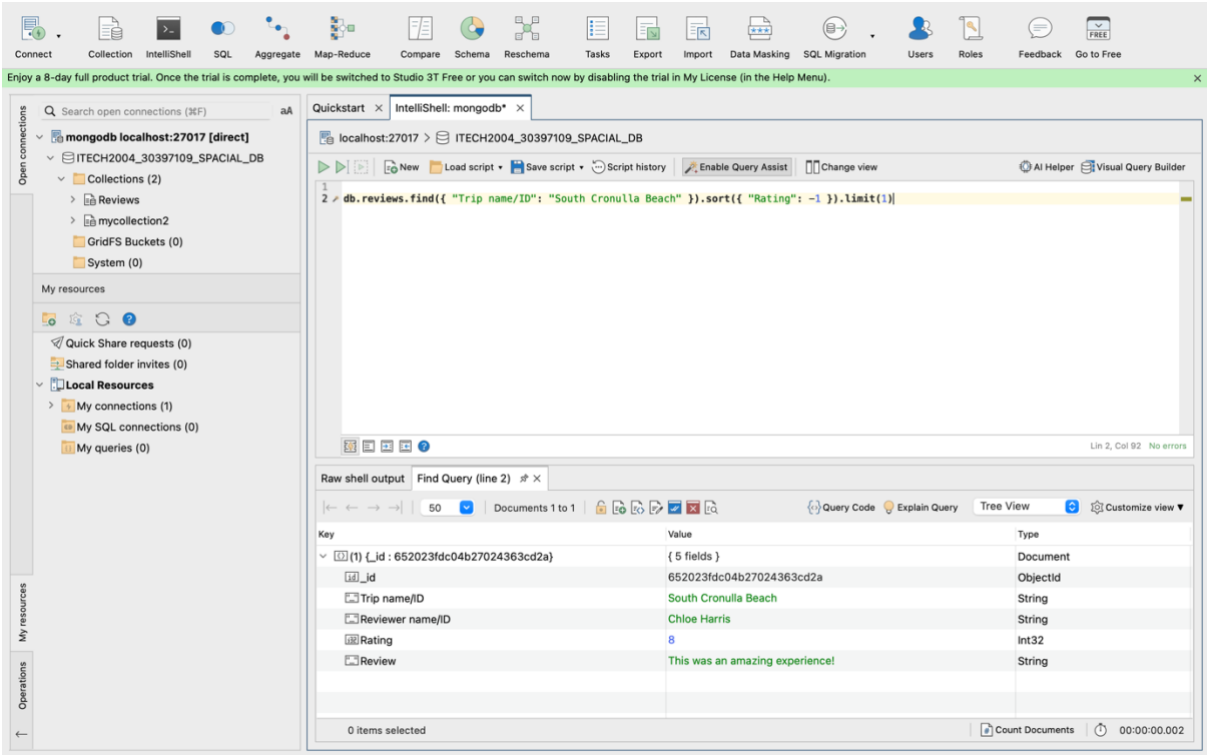

Figure 52 Retrieving highest rated review for South Cronulla Beach

For Sea Cliff Bridge

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

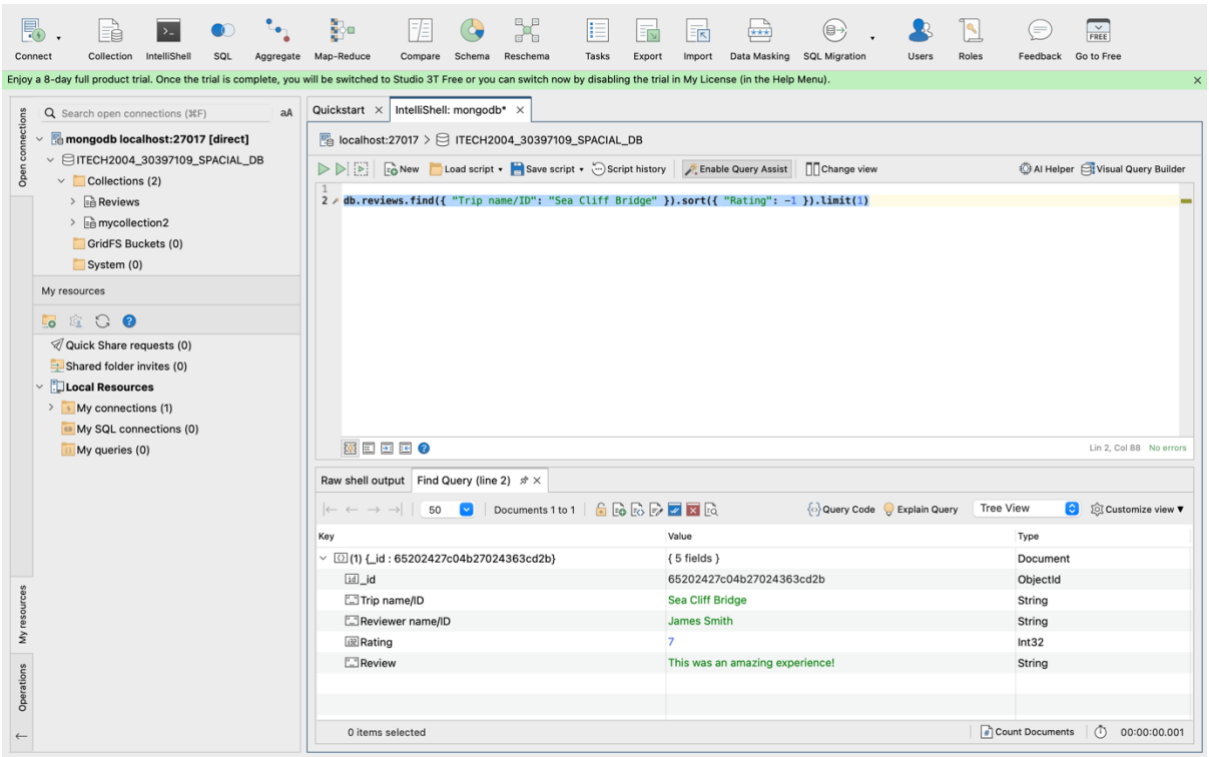

Figure 53 Retrieving highest rated review for Sea Cliff Bridge

For Bondi Beach

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

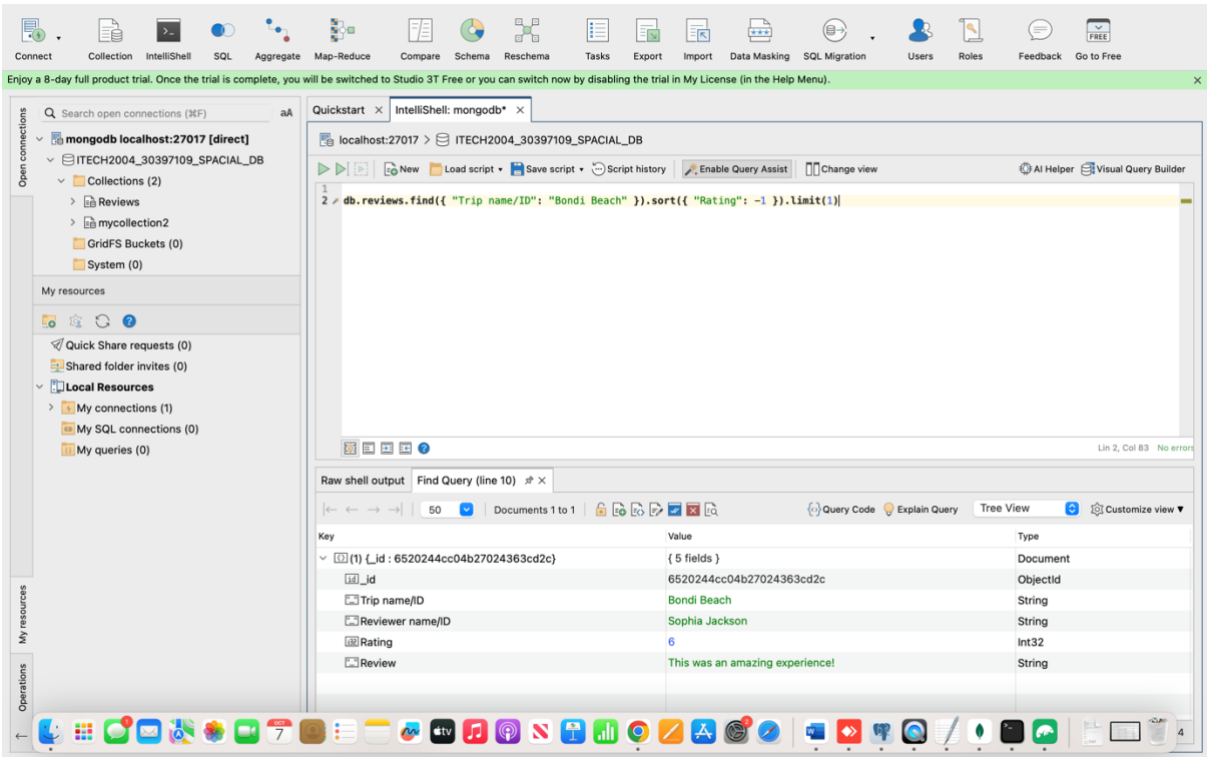

Figure 54 Retrieving highest rated review for Bondi Beach

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

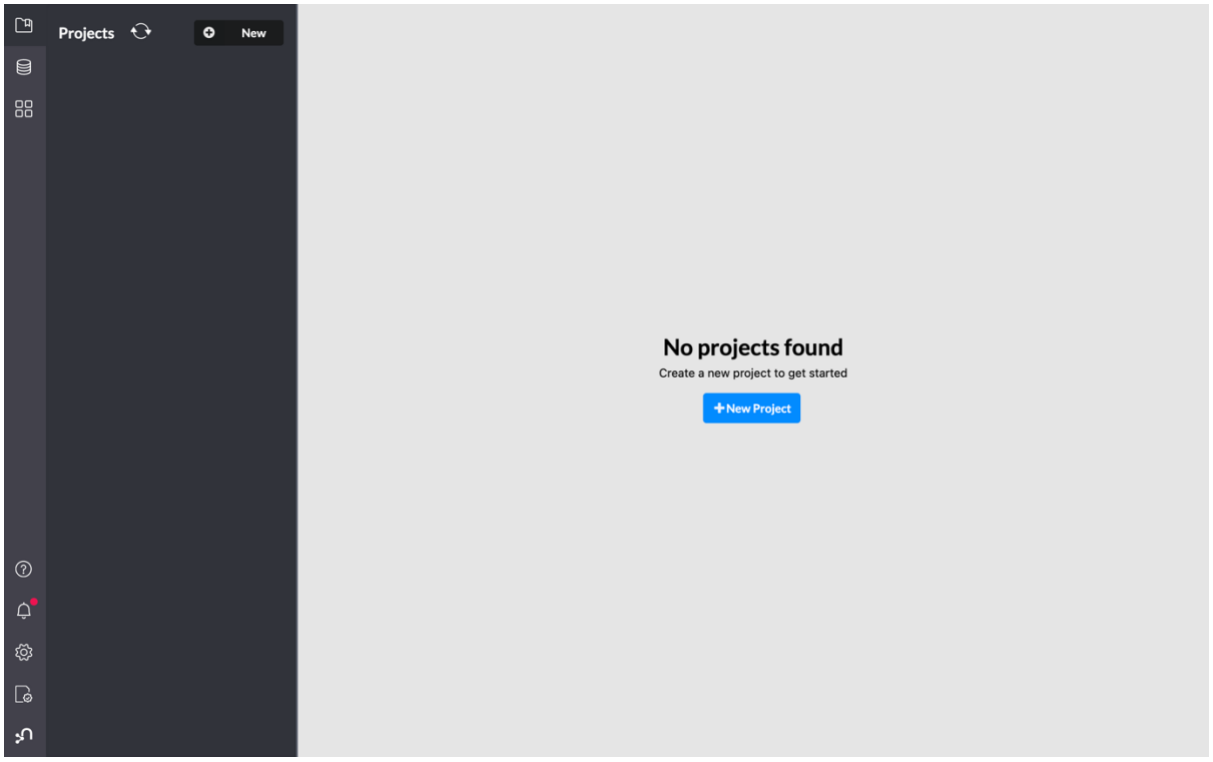

Figure 55 Database Creation Image (1)

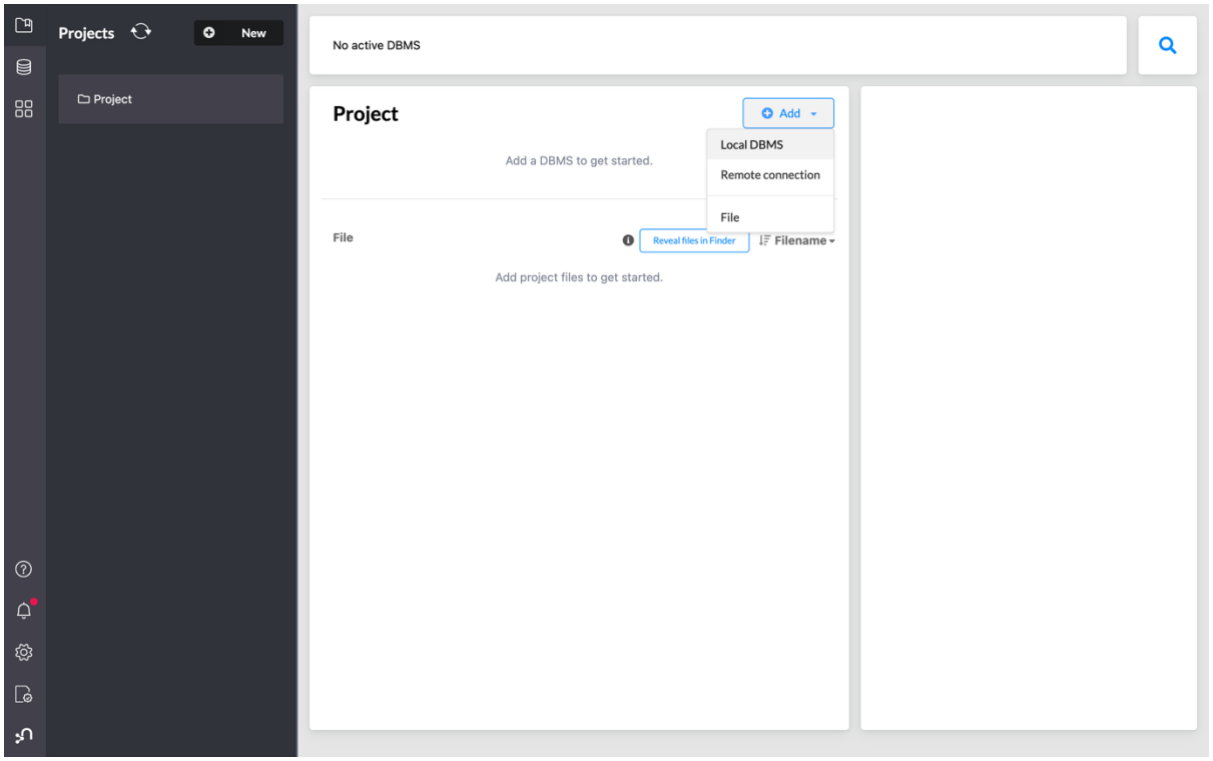

Figure 56 Database Creation Image (2)

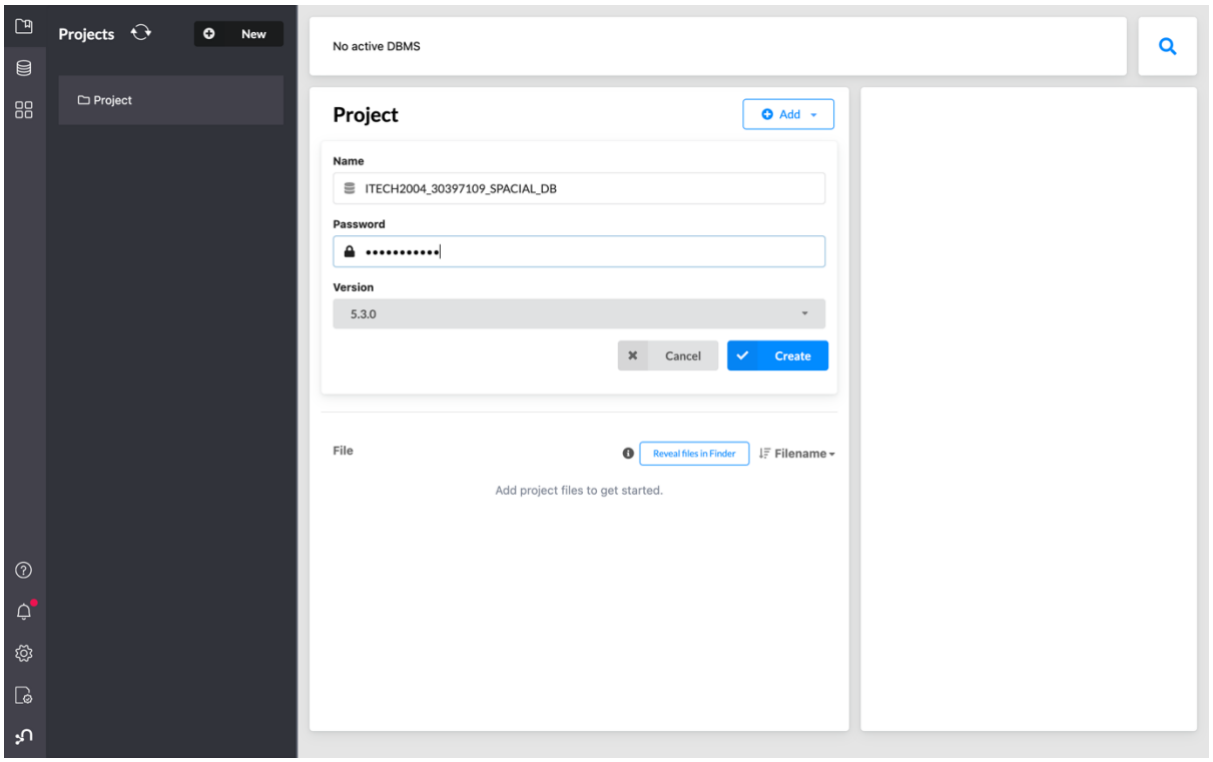

Figure 57 Database Creation Image (3)

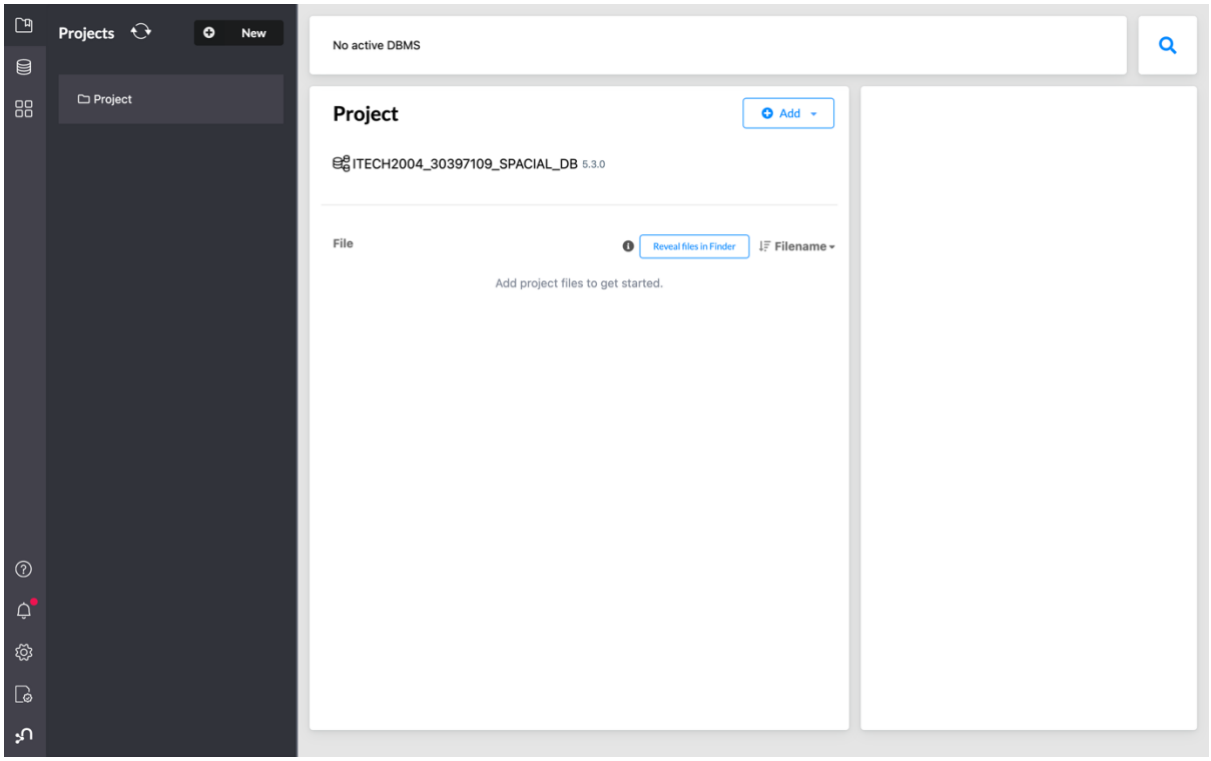

Figure 58 Database Creation Image (4)

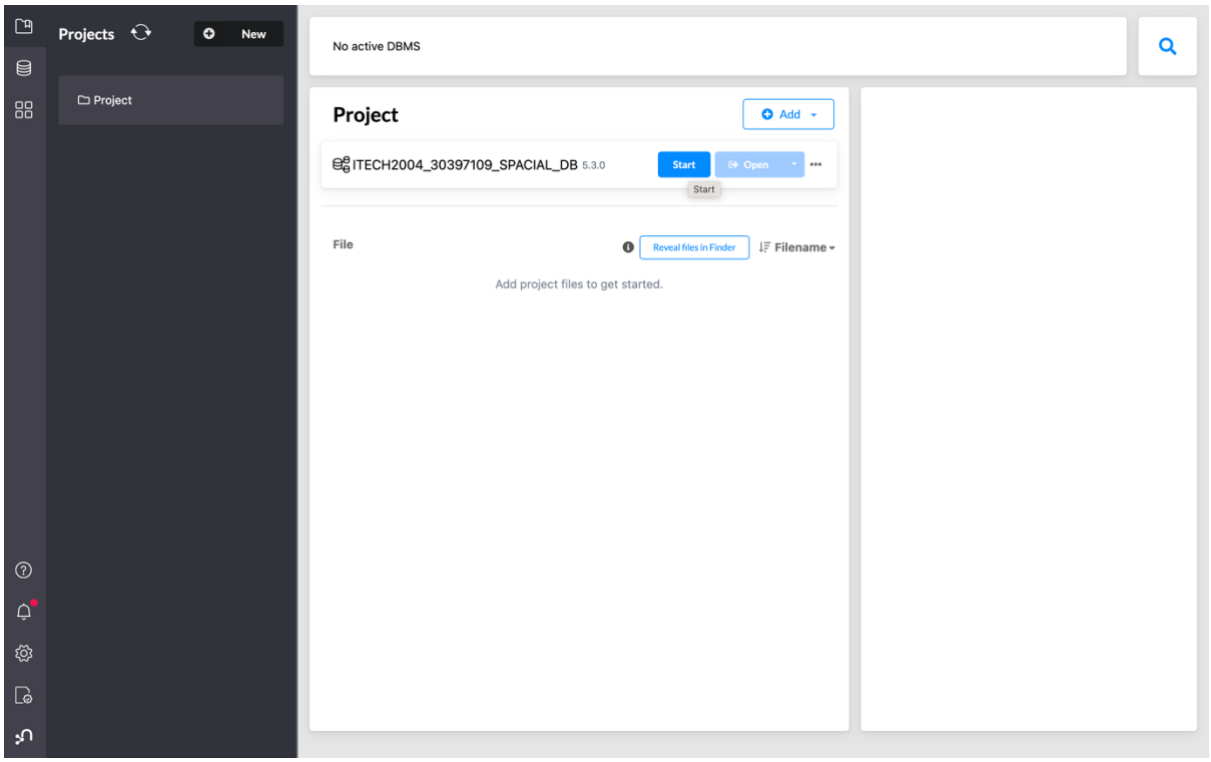

Figure 59 Database Creation Image (5)

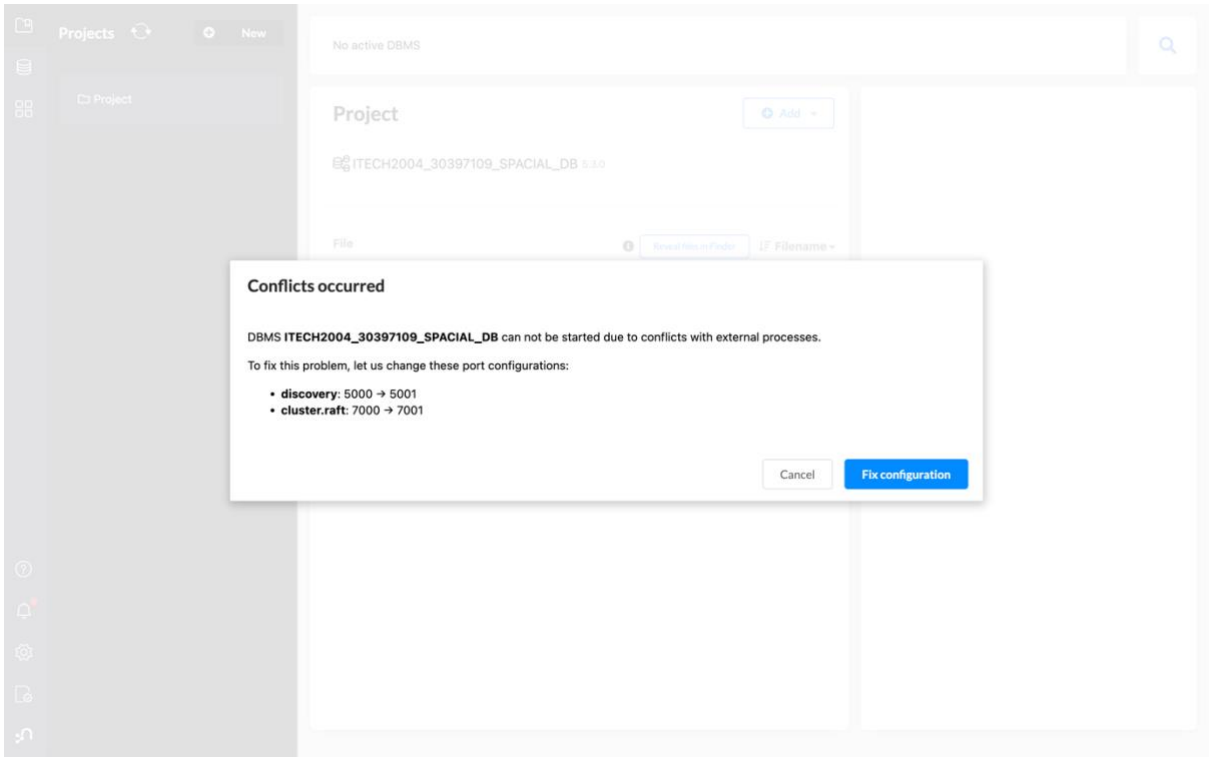

Figure 60 Database Creation Image (6)

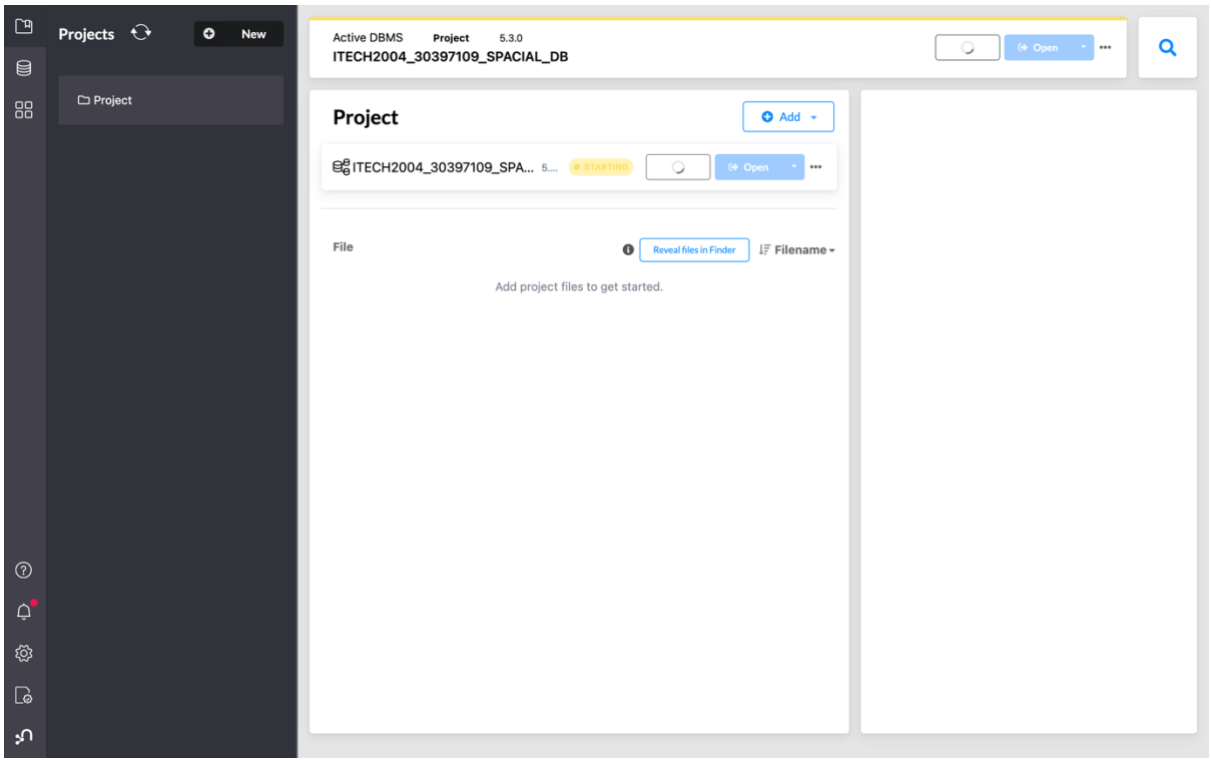

Figure 61 Database Creation Image (7)

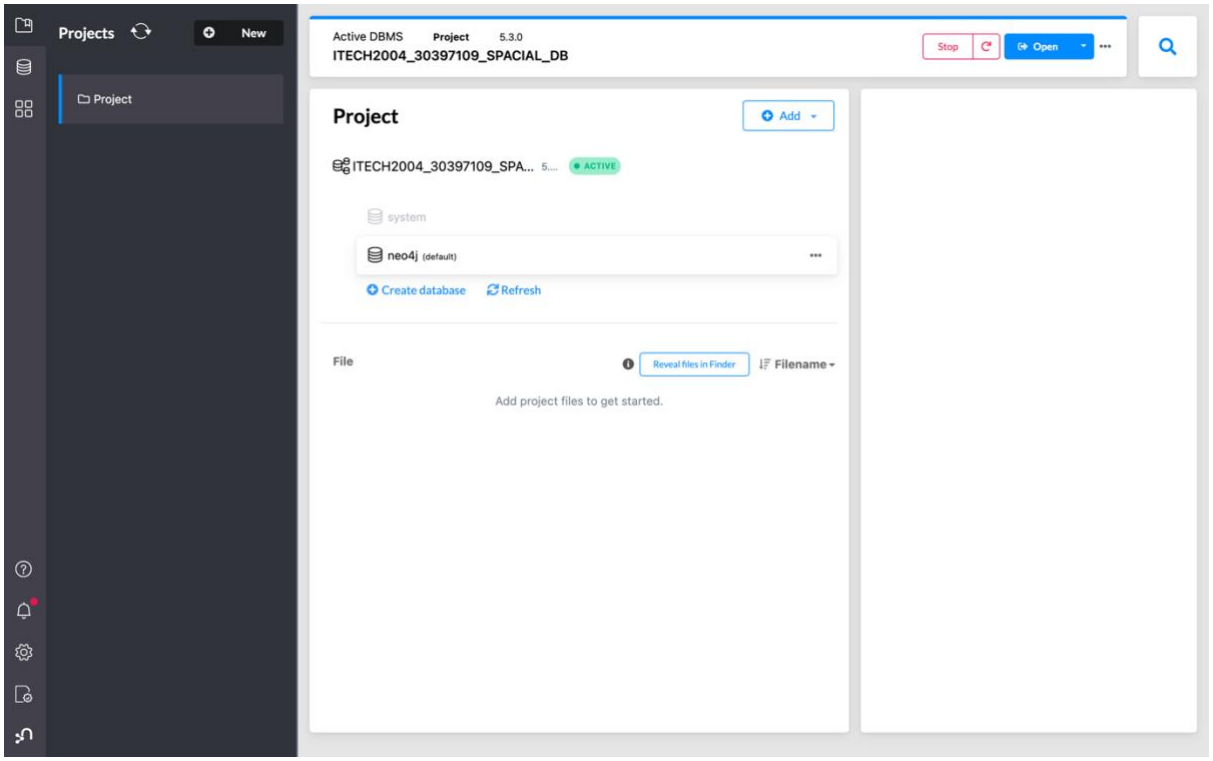

Figure 62 Database Creation Image (8)

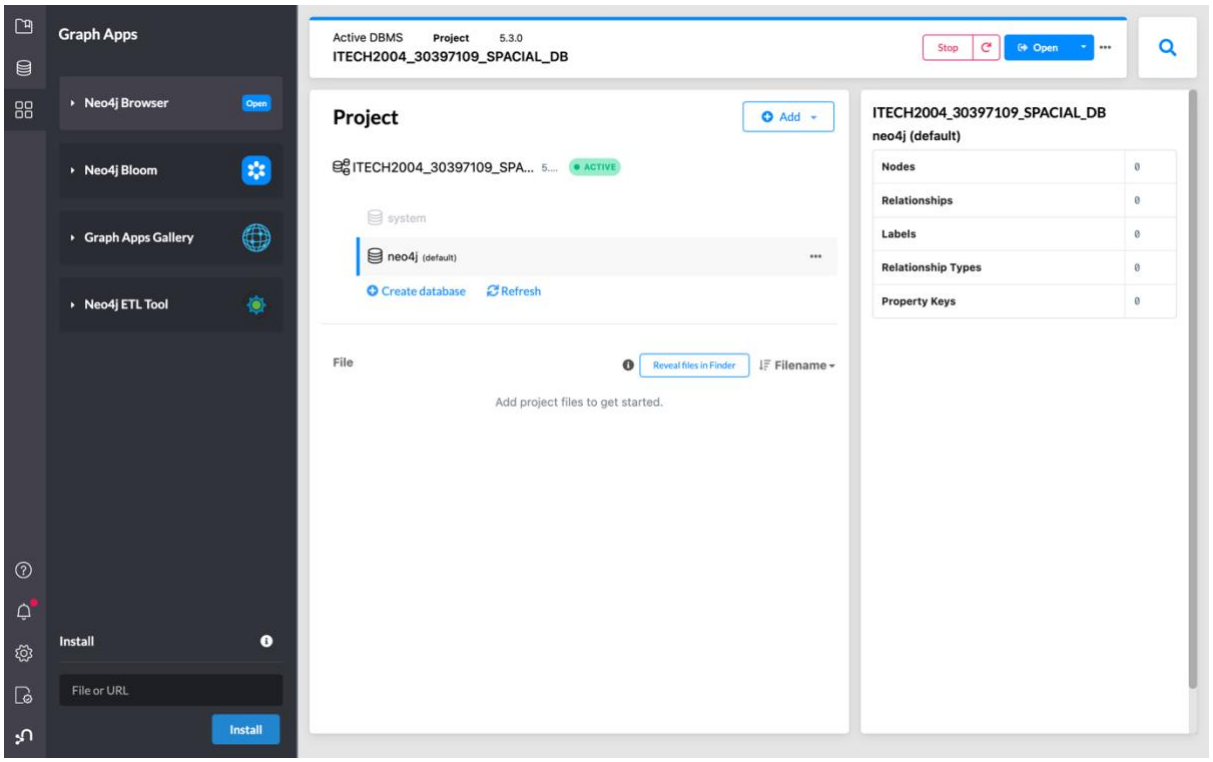

Figure 63 Database Creation Image (9)

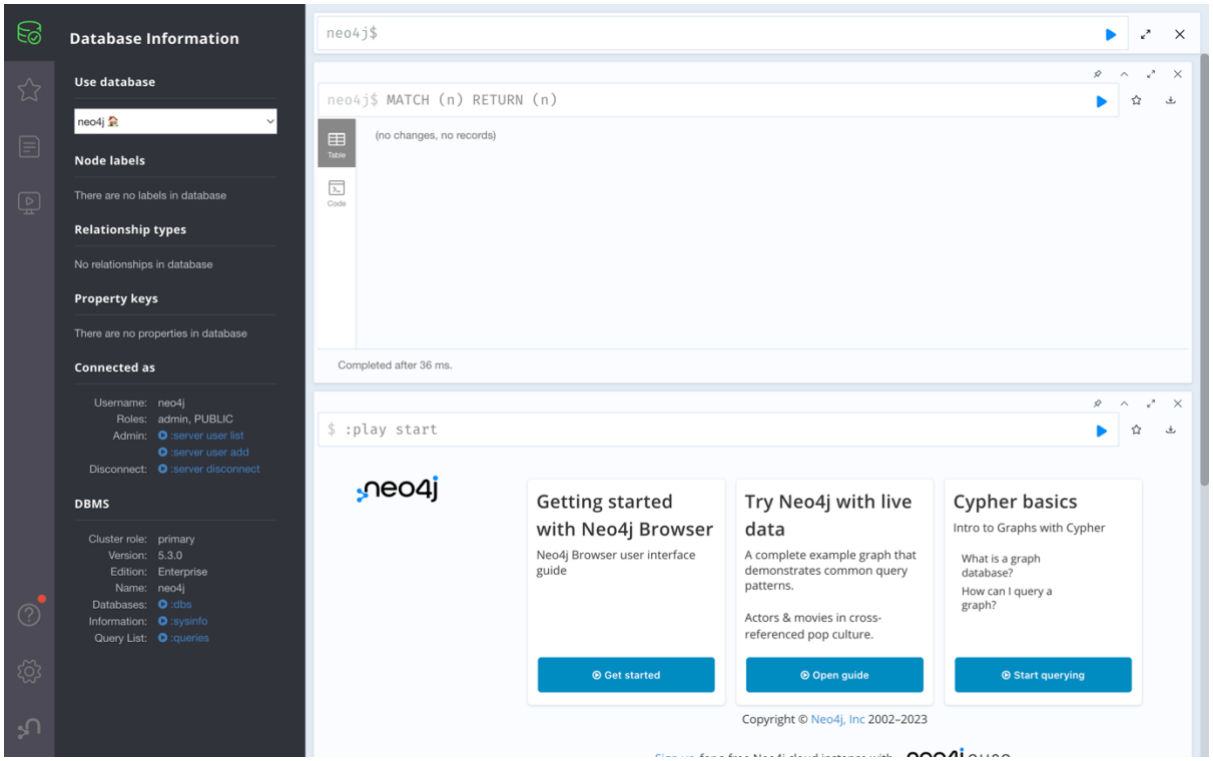

Figure 64 Screenshot to show it is connected to my database ITECH2004_30397109_SPACIAL_DB

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'})
```

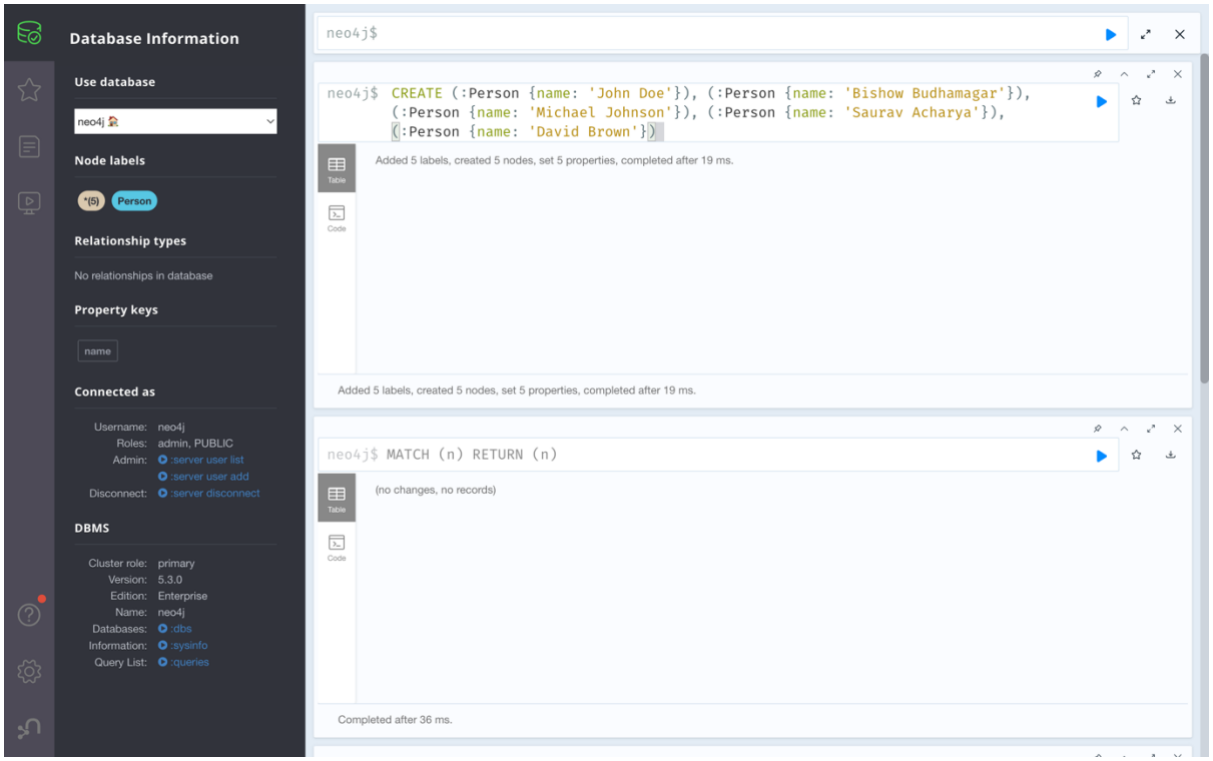

Figure 65 Inserting Person nodes

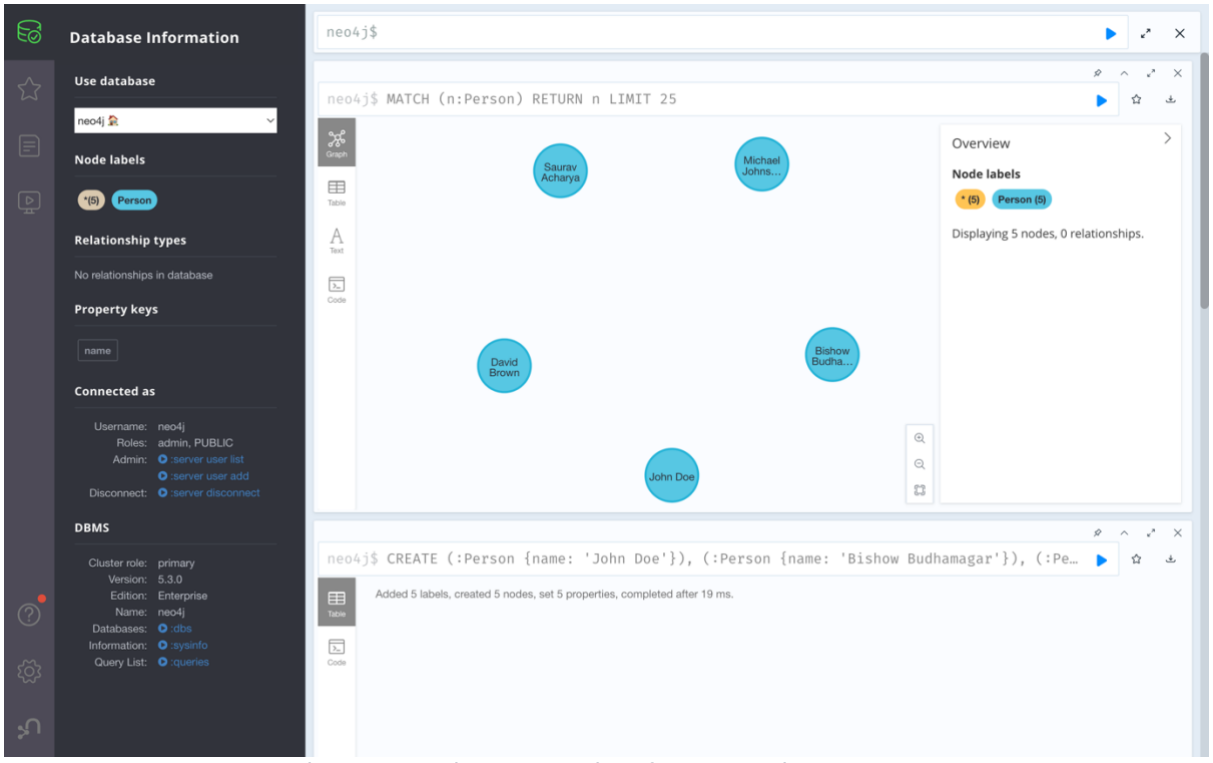

Figure 66 Graph representation of Person Nodes

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'})
```

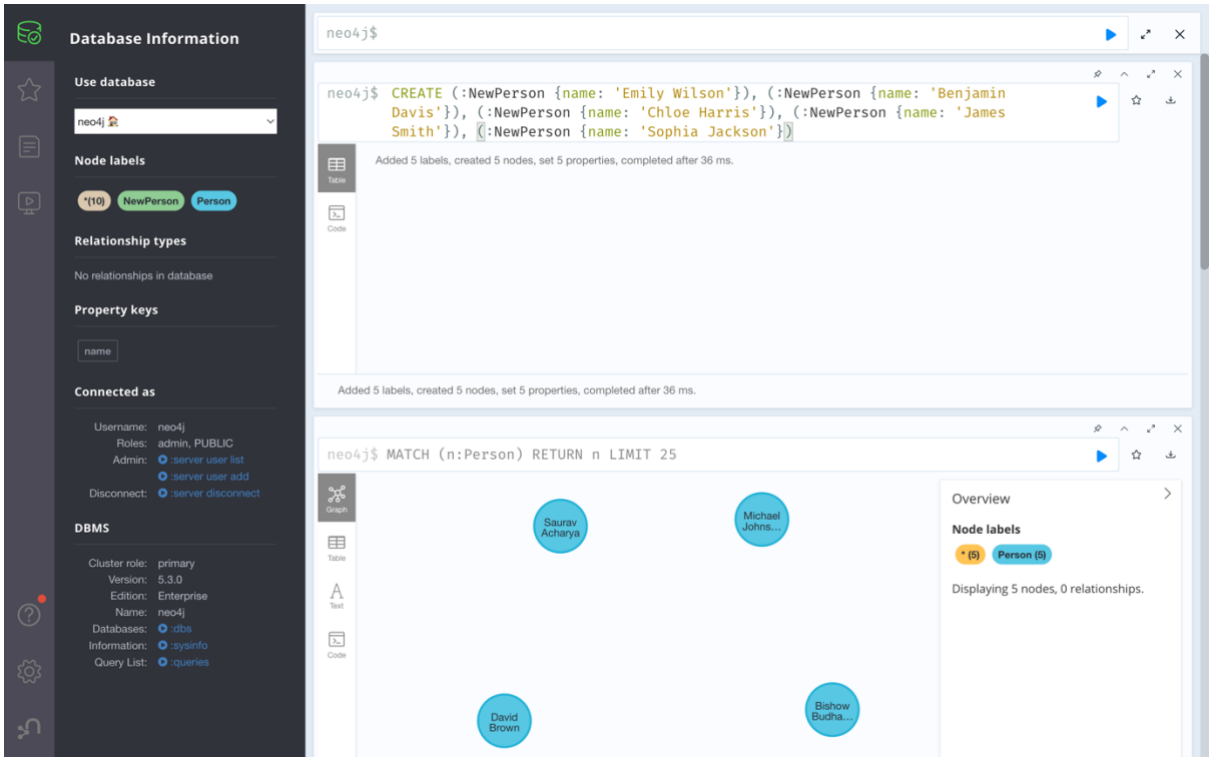

Figure 67 Inserting NewPerson nodes

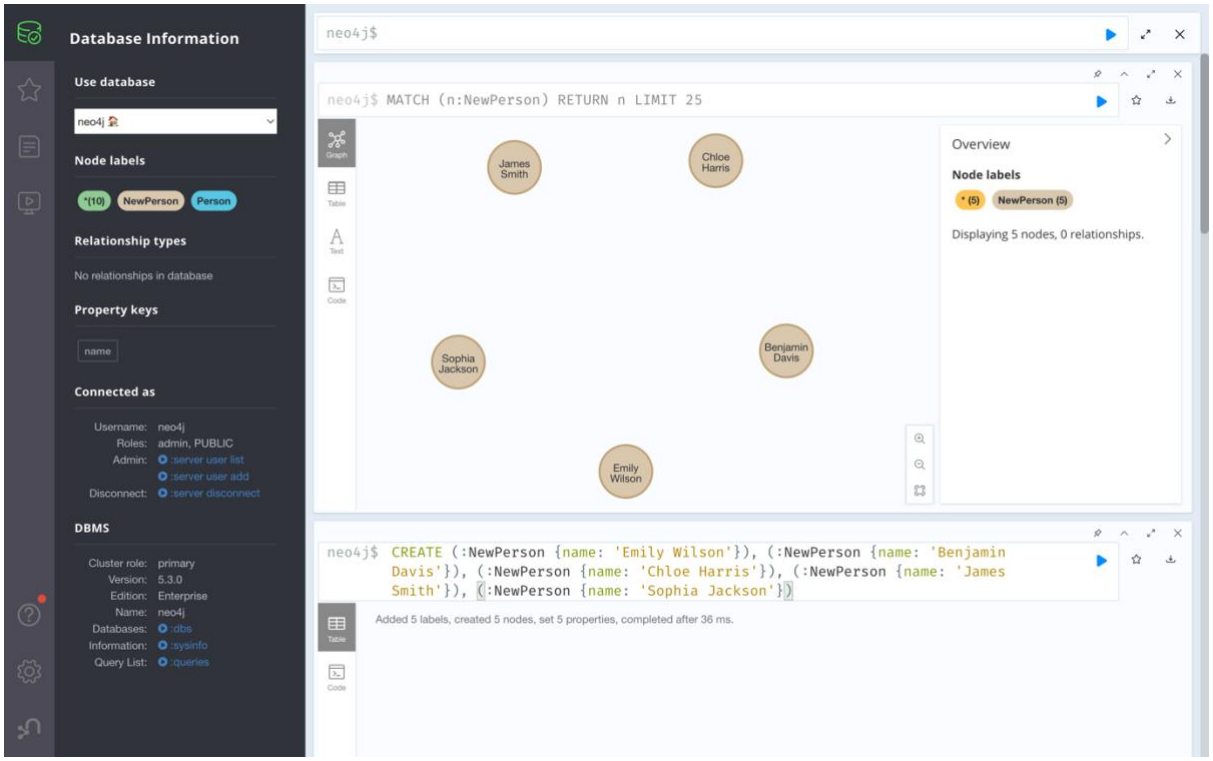

Figure 68 Graph representation of NewPerson Nodes

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)
```

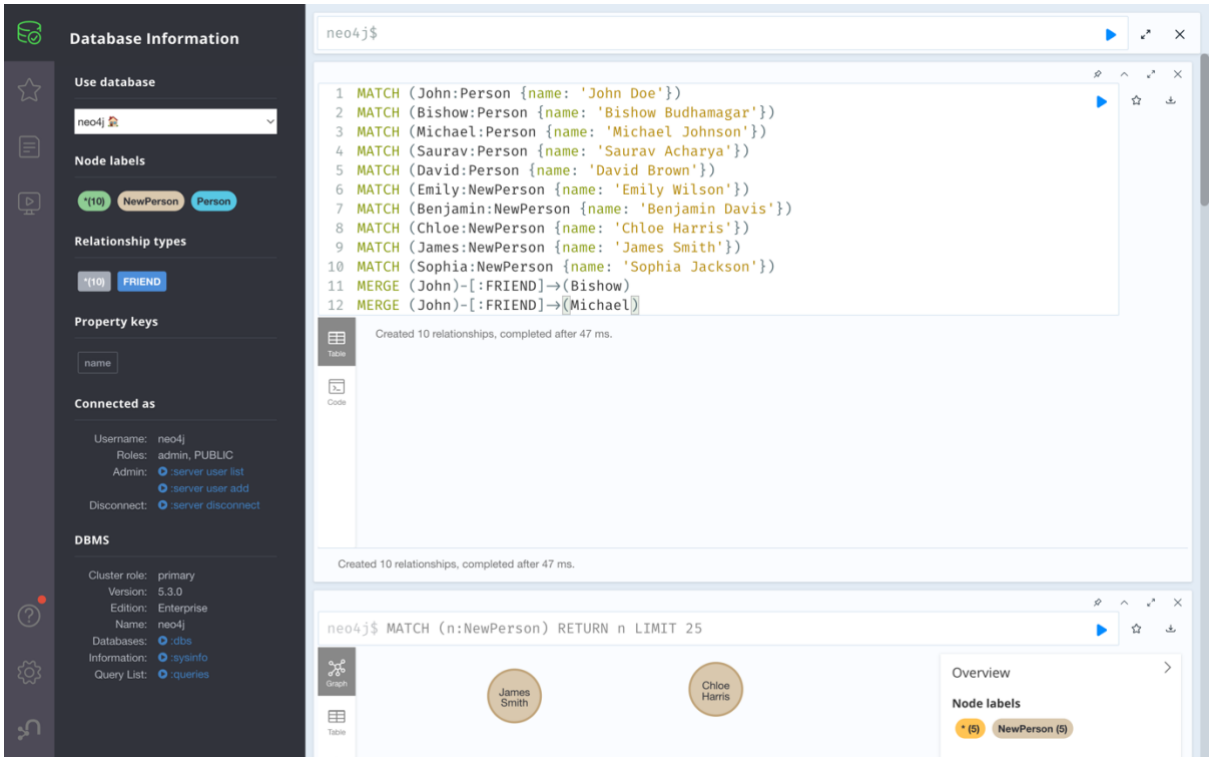

Figure 69 Inserting links(relationships) between Person and NewPerson nodes

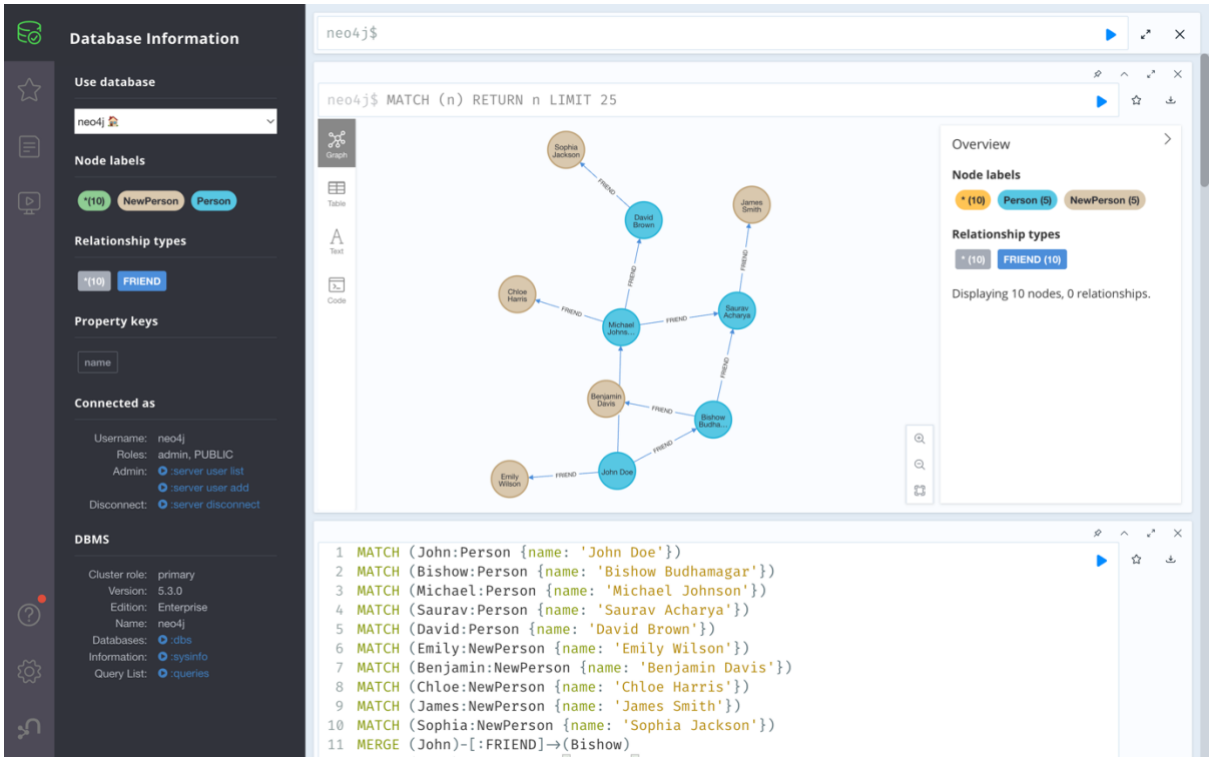

Figure 70 Graph representation of relationships between Person and NewPerson nodes

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'})
```

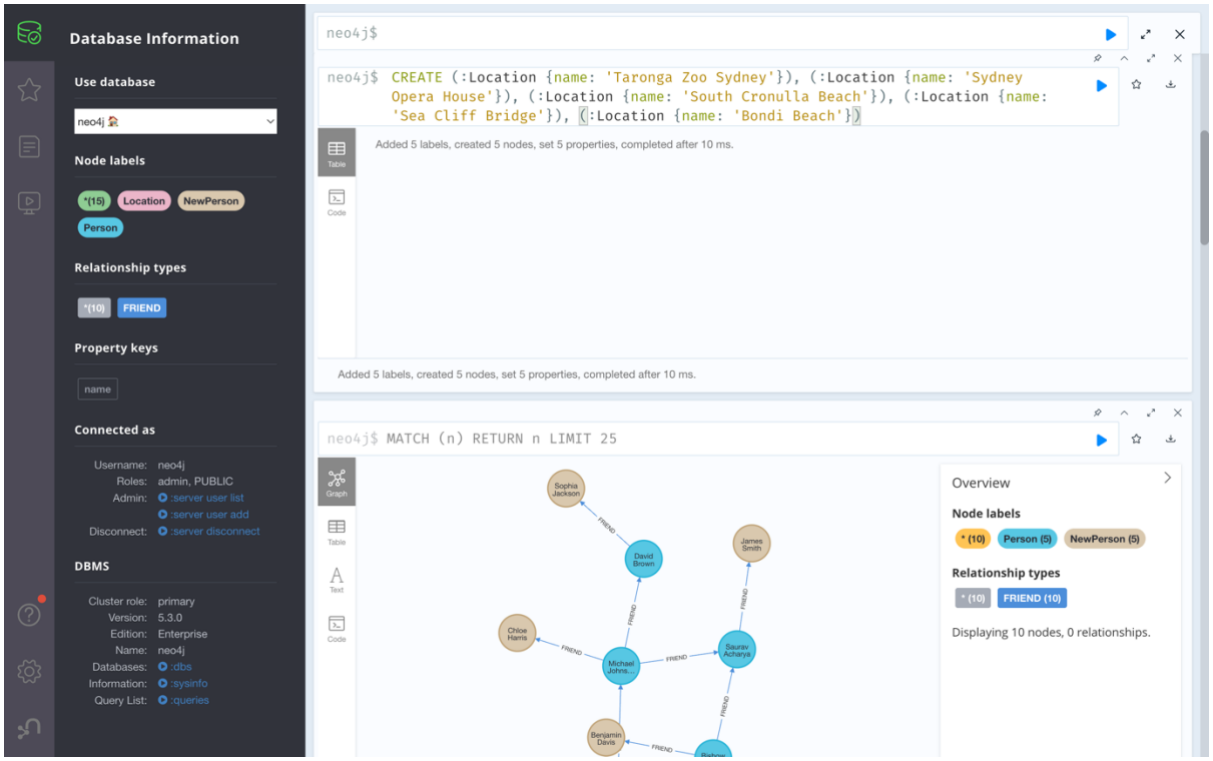

Figure 71 Inserting Location Nodes

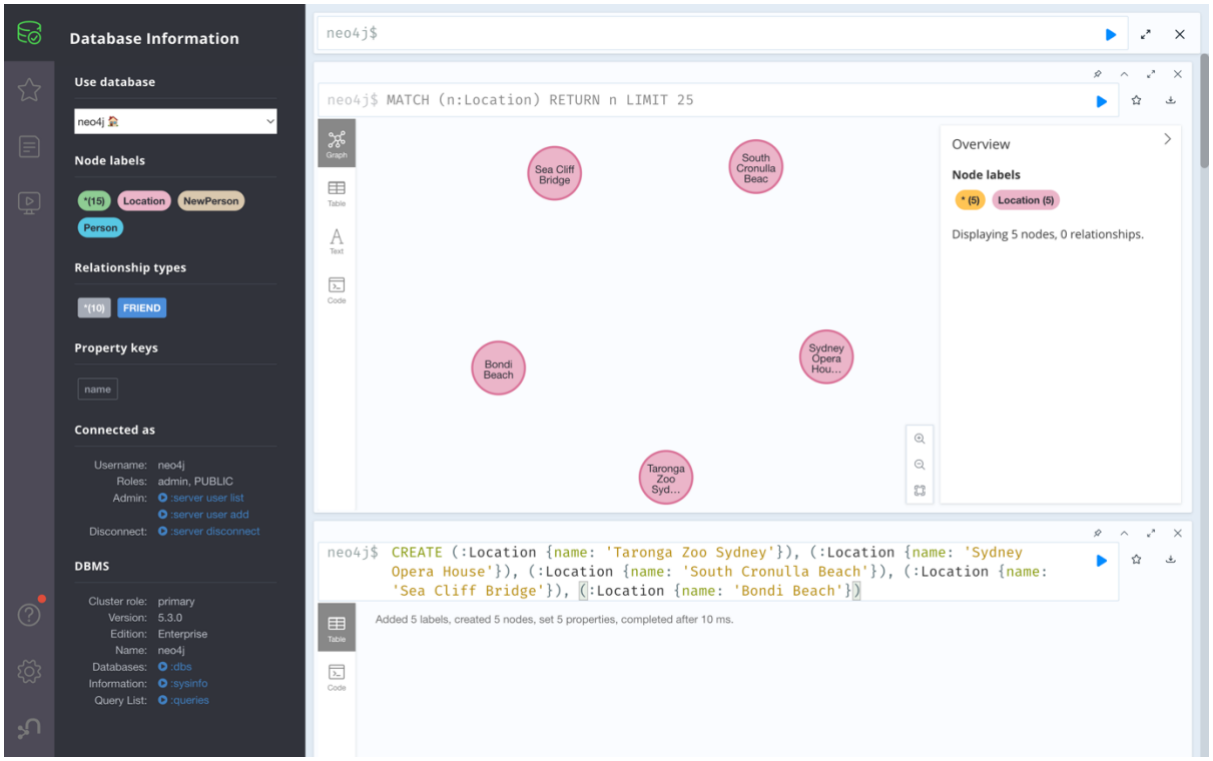

Figure 72 Graph representation of Location Nodes

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)
```

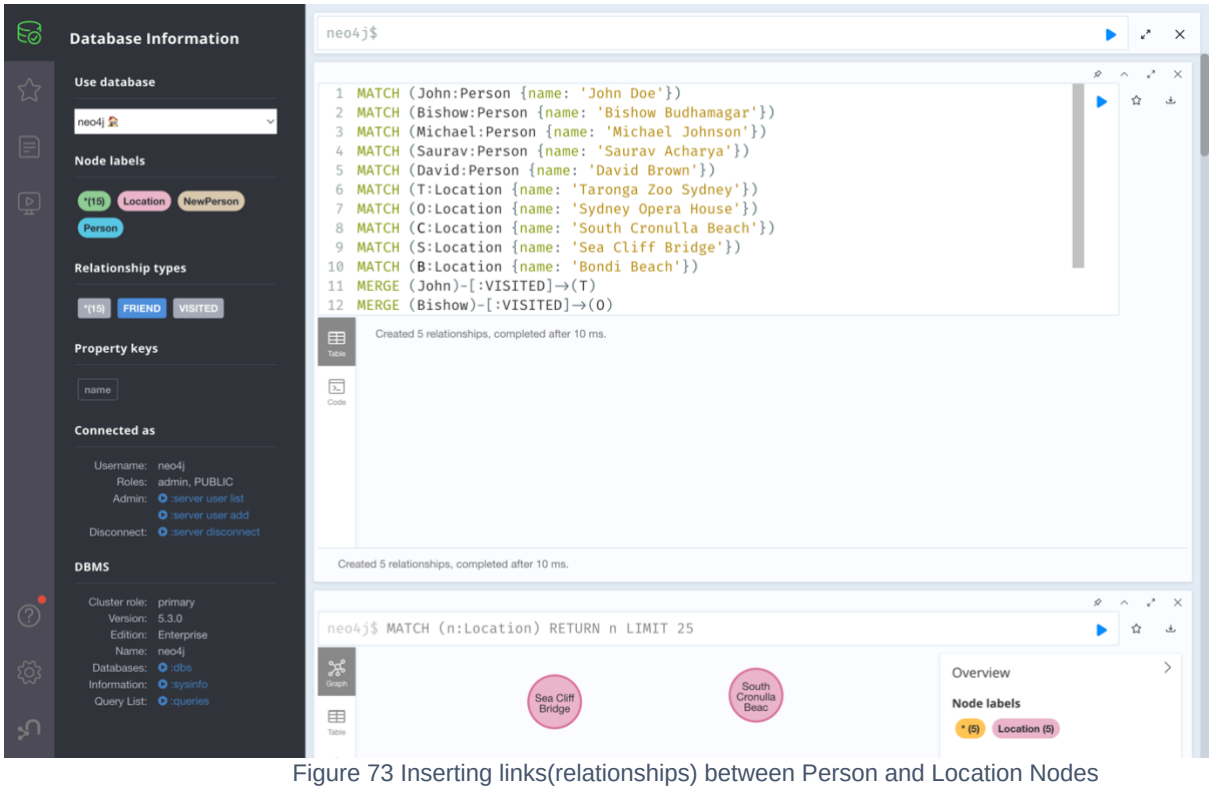

Figure 73 Inserting links(relationships) between Person and Location Nodes

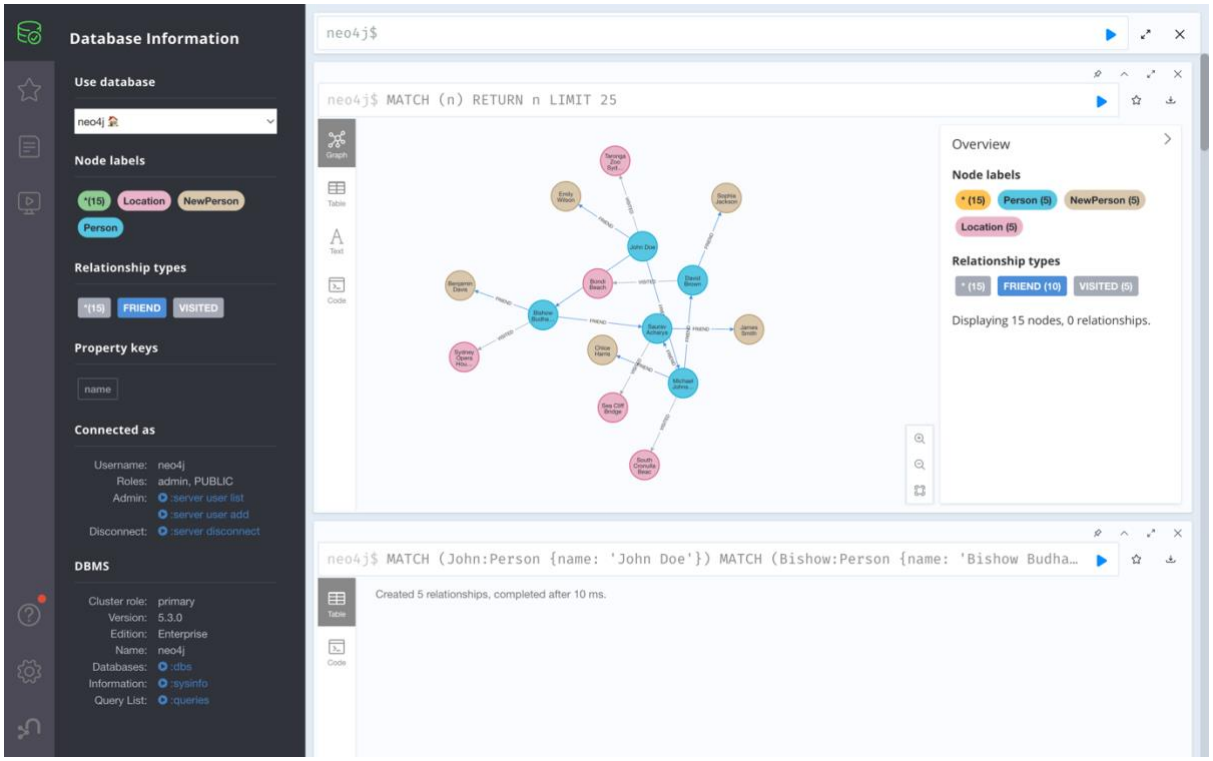

Figure 74 Graph representation of relationships between Person and Location Nodes

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
```

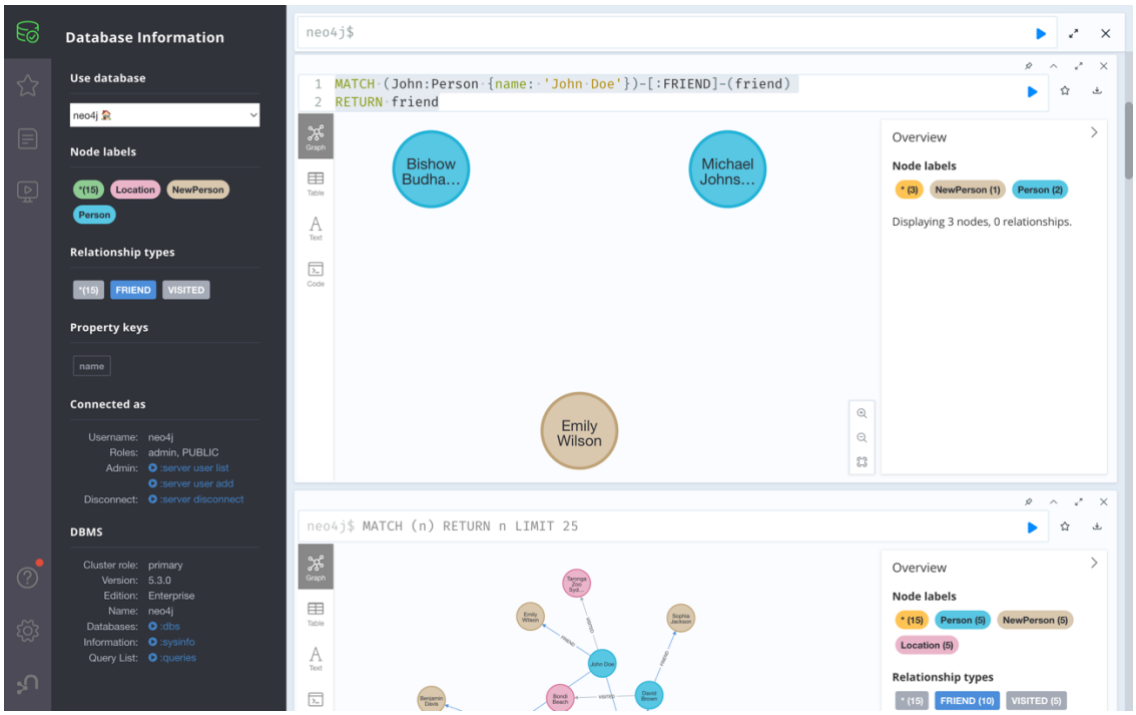

Figure 75 Retrieving all the friends of John Doe

4.3.2 For Bishow Budhamagar

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

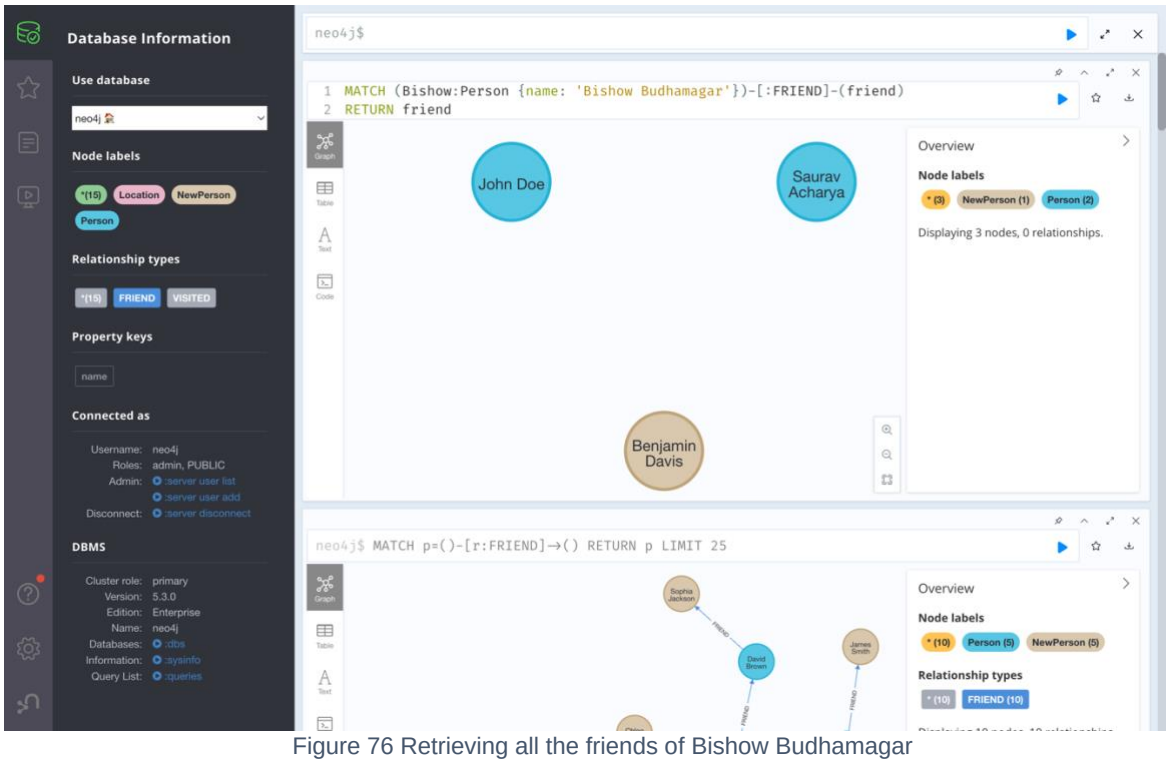

Figure 76 Retrieving all the friends of Bishow Budhamagar

4.3.3 For Michael Johnson

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

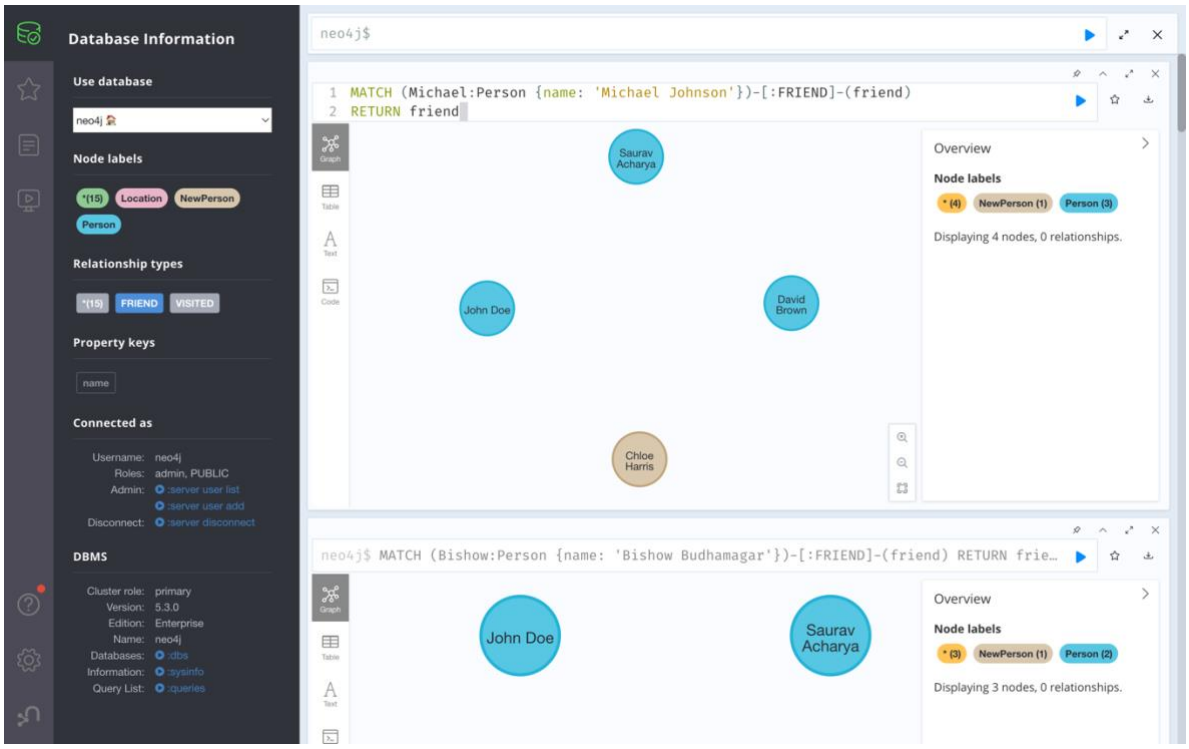

Figure 77 Retrieving all the friends of Michael Johnson

4.3.4 For Saurav Acharya

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

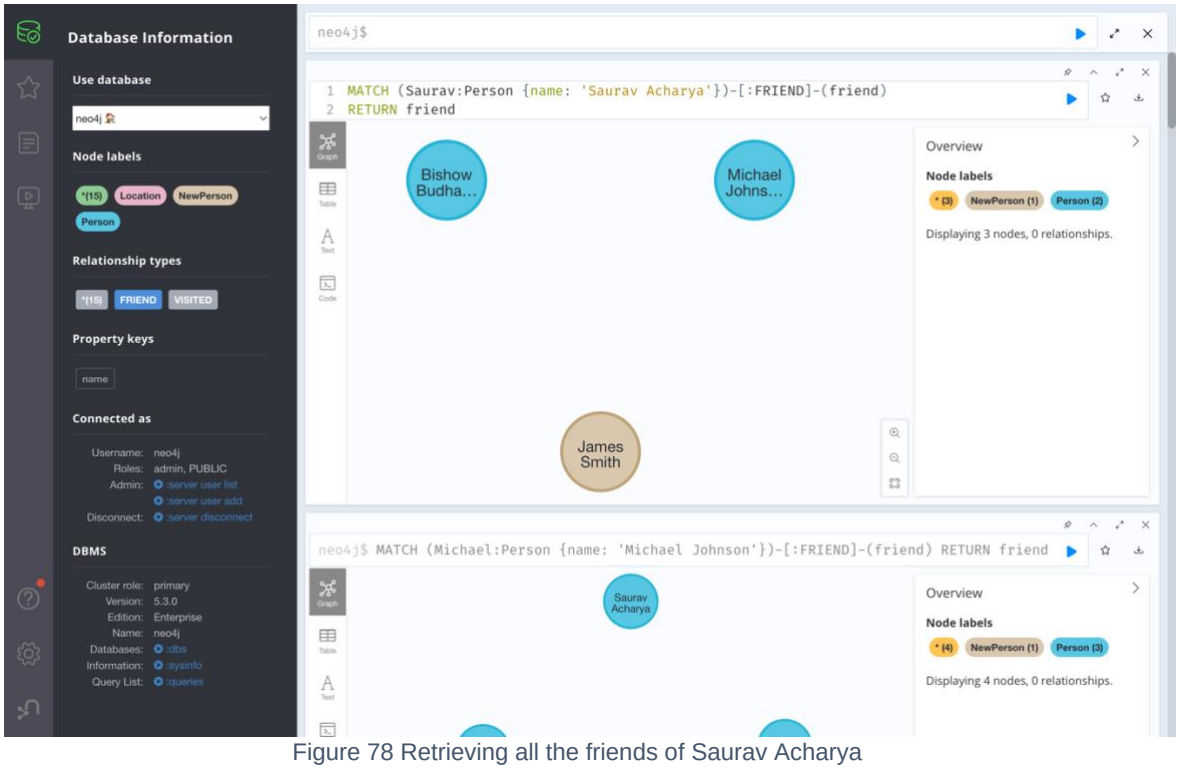

Figure 78 Retrieving all the friends of Saurav Acharya

4.3.5 For David Brown

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

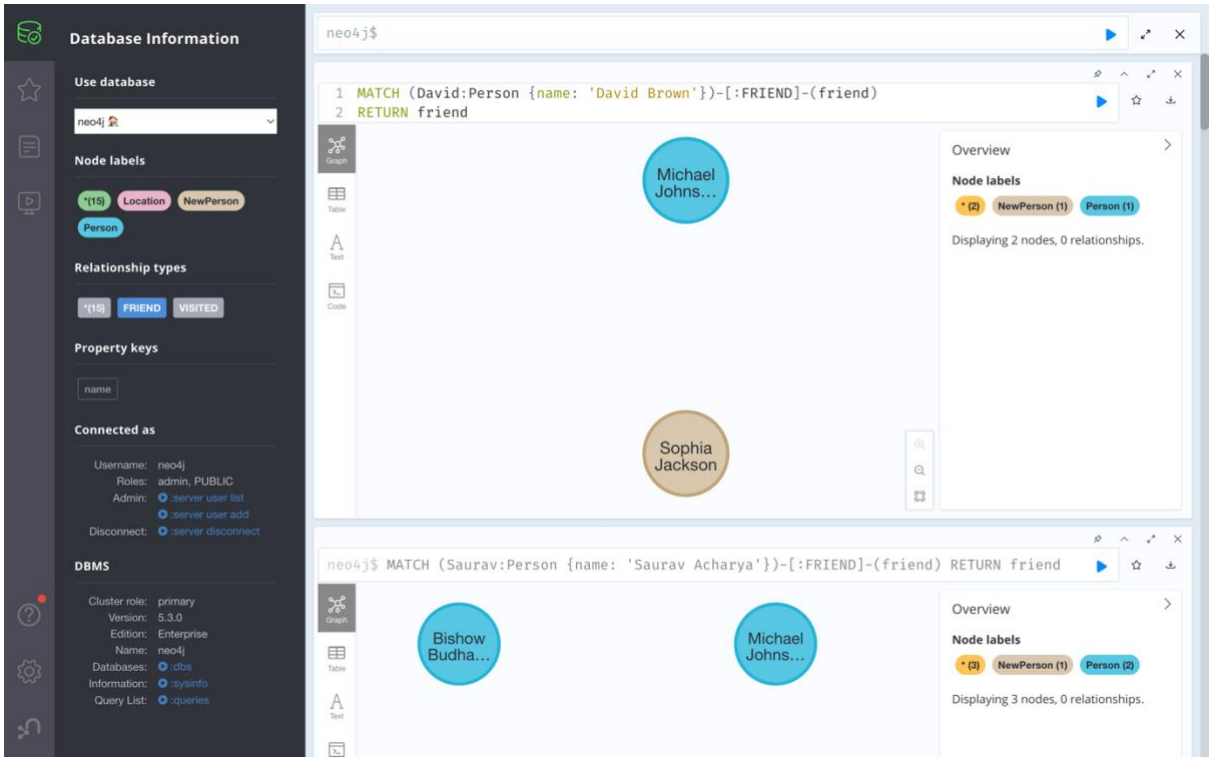

Figure 79 Retrieving all the friends of David Brown

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.