ALLIANCE UNIVERSITY

Private University Estd. in Karnataka State by Act No. 34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

Project Report
Master of Computer Application
Semester – II
Database Design and Implementation

# Neo4j Movie Ticket Booking System

By
ADITI PAITANDY
2411022250019
Department of Computer Application
Alliance University
Chandapura - Anekal Main Road, Anekal
Bengaluru - 562106
APRIL 2025

# Neo4j Movie Ticket Booking System

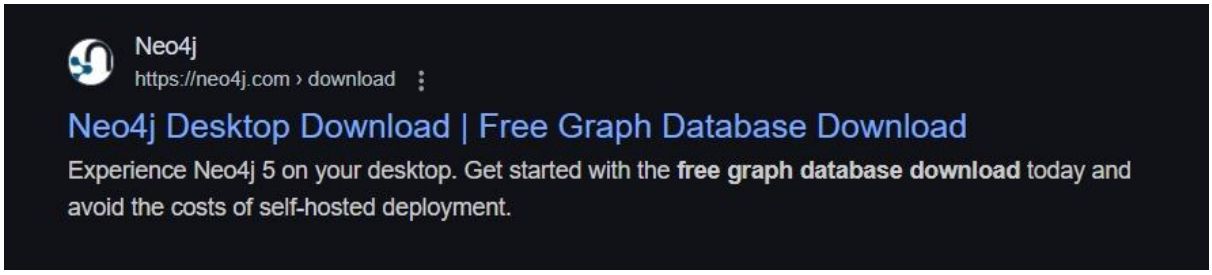
## Why Neo4j?

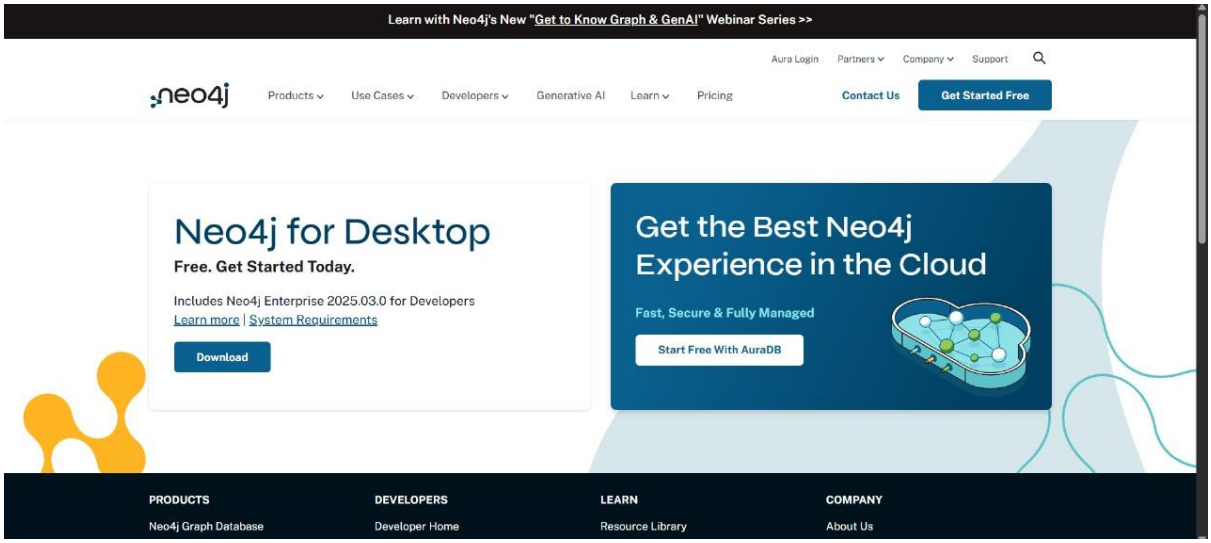
- Neo4j is a graph database ideal for highly connected data.
- Movie ticket booking involves users, theaters, movies, showtimes, and relationships.
- Graph databases make it easier to traverse relationships (e.g., "Which user booked which movie in which theater?")

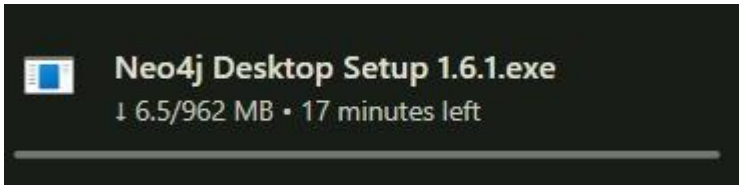
## Project Setup

### Step 1: Installation

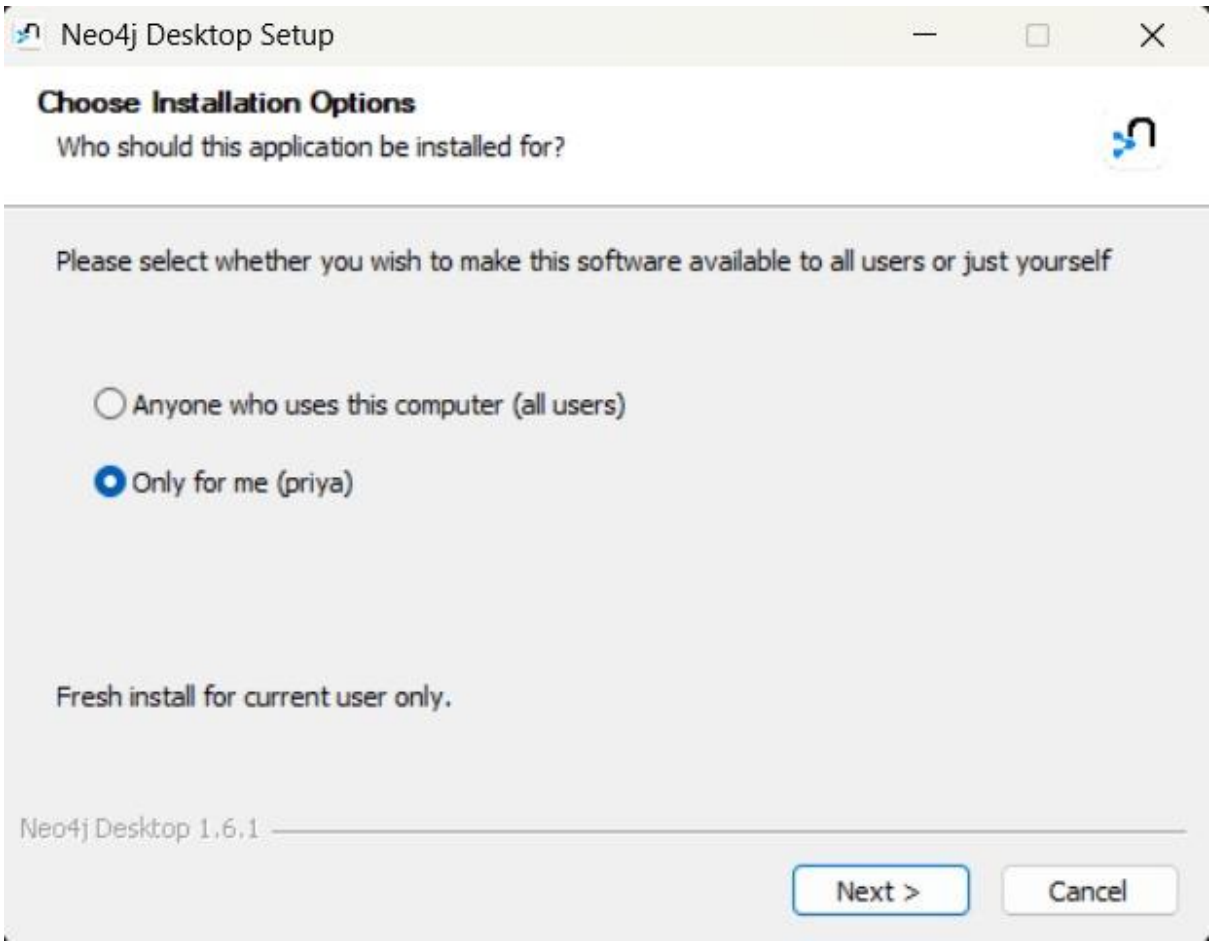
- Download Neo4j Desktop from neo4j.com

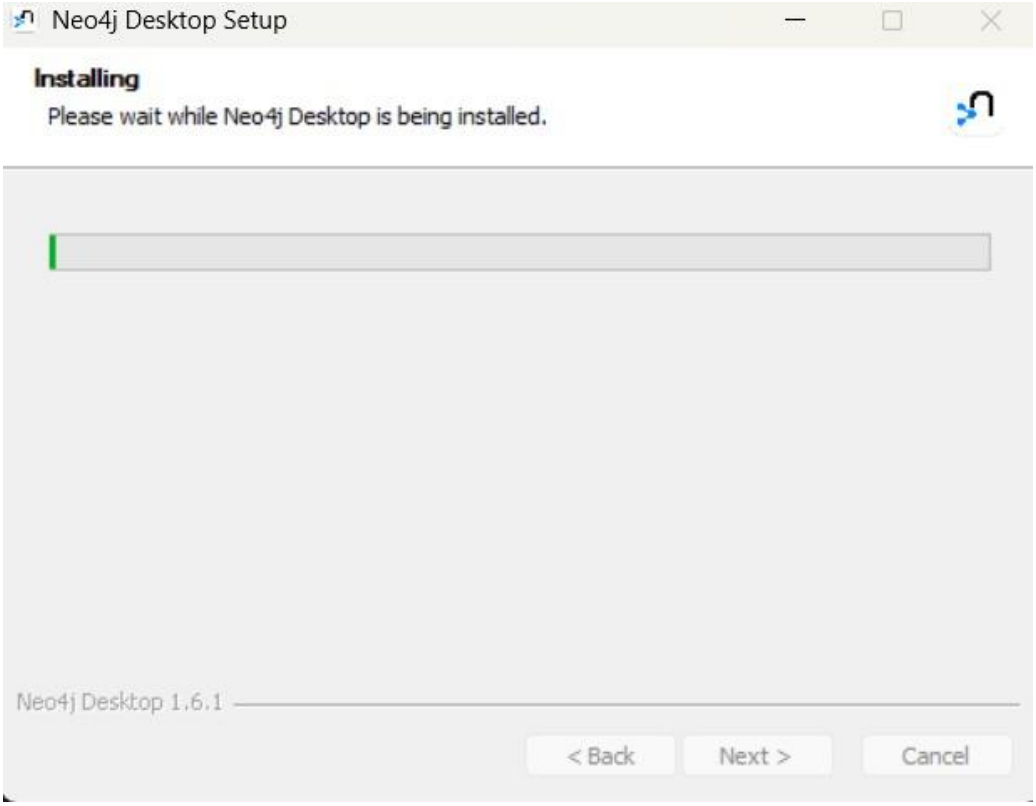


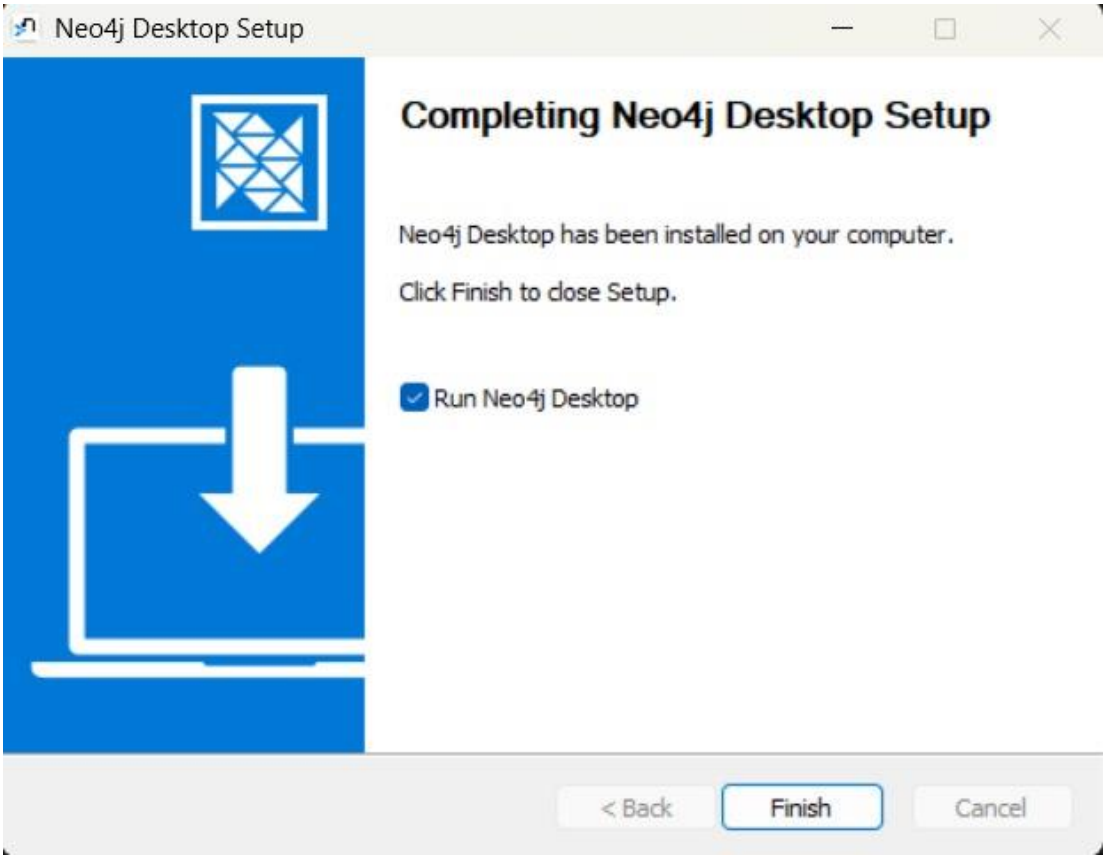





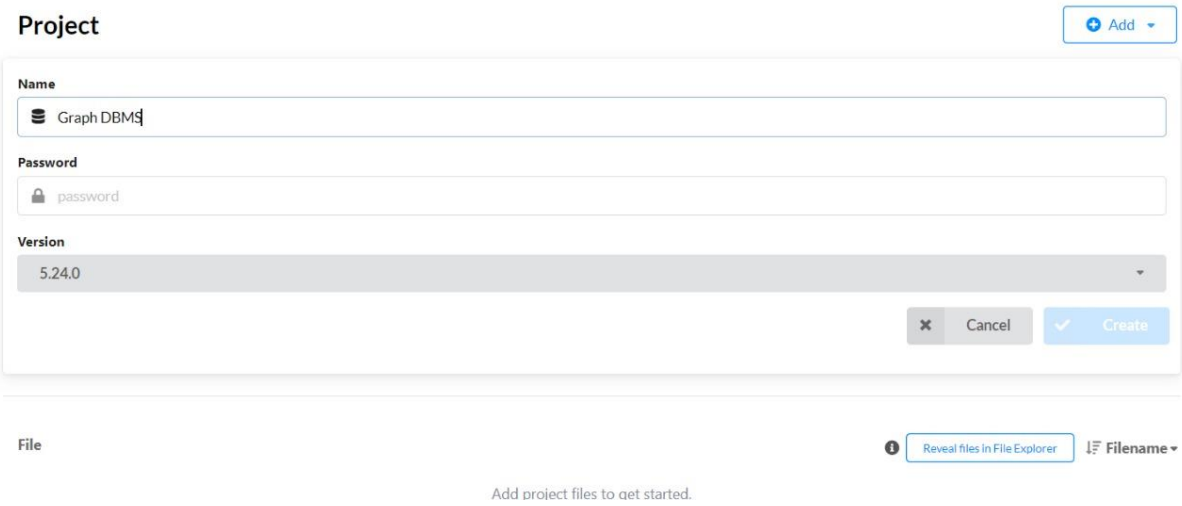

- Install Neo4j Desktop



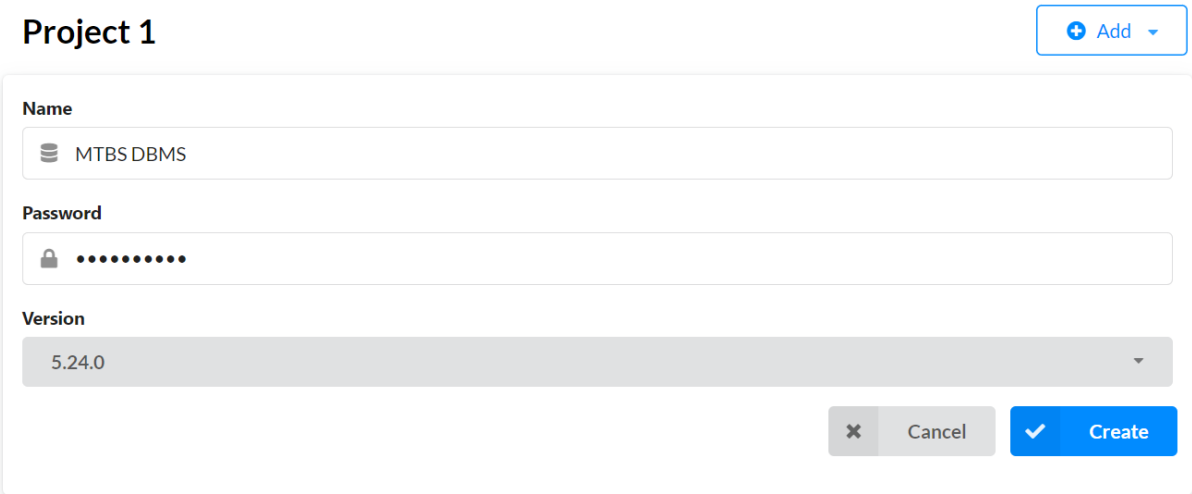

Neo4j Desktop Setup
Installing
Please wait while Neo4j Desktop is being installed.
Neo4j Desktop 1.6.1
< Back
Next >
Cancel


Neo4j Desktop Setup
Completing Neo4j Desktop Setup
Neo4j Desktop has been installed on your computer.
Click Finish to close Setup.
Run Neo4j Desktop
< Back
Finish
Cancel

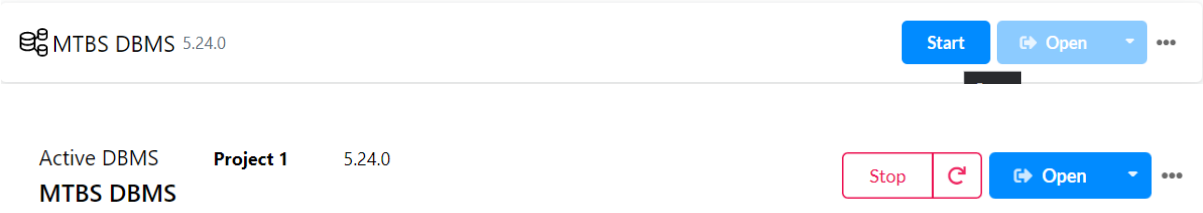
- Create a new project and database




- Set username/password (default: neo4j/neo4j)




- Start the database



## Step 2: Database Schema Design

We will be having  5 types of nodes and their relationships:

**Nodes (5 types):**

1. **User**
   Attributes:
   id, name, email, phone, registered_date
2. **Movie**
   Attributes:
   id, title, genre, rating, language
3. **Theater**
   Attributes:
   id, name, location, capacity, opened_date
4. **Show**
   Attributes:
   id, time, date, price, screen
5. **Booking**
   Attributes:
   id, ticket_count, total_amount, status, booking_time

**Relationships:**

- **(User) -[:BOOKED]-> (Booking)**
  A user makes a booking.
- **(Booking) -[:FOR_SHOW]-> (Show)**
  Each booking is for a particular show.
- **(Show) -[:PLAYS]-> (Movie)**
  The show plays a specific movie.
- **(Show) -[:IN_THEATER]-> (Theater)**
  The show is conducted in a specific theater.
- **(Booking) -[:BOOKED_BY]-> (User)**
  Extra link back from booking to user (redundancy for flexibility).
- **(User) -[:WATCHED]-> (Movie)**
  To show user has watched a movie (optional but adds connection).

**Step 3: Sample Data**

- CREATE OPERATION AND SAMPLE DATA :

1. USER Table (Label: User)

CREATE

(:User {id: 1, name: 'Aditi', email: 'aditi@example.com', phone: '9991110001', registered_date: '2024-06-01'}),

(:User {id: 2, name: 'Ravi', email: 'ravi@example.com', phone: '9991110002', registered_date: '2024-06-02'}),

(:User {id: 3, name: 'Priya', email: 'priya@example.com', phone: '9991110003', registered_date: '2024-06-03'}),

(:User {id: 4, name: 'Arjun', email: 'arjun@example.com', phone: '9991110004', registered_date: '2024-06-04'}),

(:User {id: 5, name: 'Neha', email: 'neha@example.com', phone: '9991110005', registered_date: '2024-06-05'}),

(:User {id: 6, name: 'Kiran', email: 'kiran@example.com', phone: '9991110006', registered_date: '2024-06-06'}),

(:User {id: 7, name: 'Rahul', email: 'rahul@example.com', phone: '9991110007', registered_date: '2024-06-07'}),

(:User {id: 8, name: 'Meena', email: 'meena@example.com', phone: '9991110008', registered_date: '2024-06-08'}),

(:User {id: 9, name: 'Aarav', email: 'aarav@example.com', phone: '9991110009', registered_date: '2024-06-09'}),

(:User {id: 10, name: 'Sneha', email: 'sneha@example.com', phone: '9991110010', registered_date: '2024-06-10'});

2. MOVIE Table (Label: Movie)

CREATE

(:Movie {id: 1, title: 'Inception', genre: 'Sci-Fi', rating: 8.8, language: 'English'}),

(:Movie {id: 2, title: '3 Idiots', genre: 'Comedy/Drama', rating: 8.4, language: 'Hindi'}),

(:Movie {id: 3, title: 'Interstellar', genre: 'Sci-Fi', rating: 8.6, language: 'English'}),

(:Movie {id: 4, title: 'Dangal', genre: 'Biography', rating: 8.3, language: 'Hindi'}),

(:Movie {id: 5, title: 'The Dark Knight', genre: 'Action', rating: 9.0, language: 'English'}),

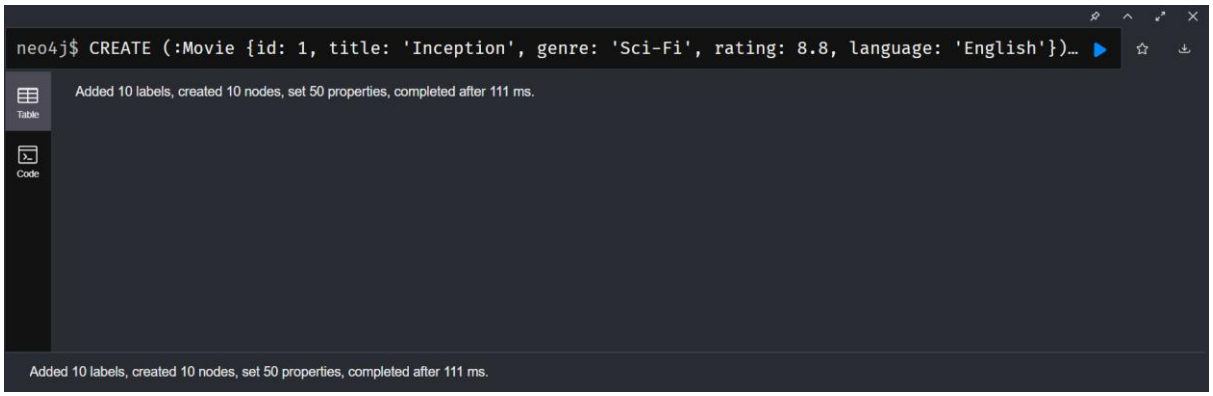
(:Movie {id: 6, title: 'Parasite', genre: 'Thriller', rating: 8.6, language: 'Korean'}),

(:Movie {id: 7, title: 'Drishyam', genre: 'Thriller', rating: 8.2, language: 'Hindi'}),

(:Movie {id: 8, title: 'Avatar', genre: 'Fantasy', rating: 7.8, language: 'English'}),

(:Movie {id: 9, title: 'Jawan', genre: 'Action', rating: 7.0, language: 'Hindi'}),

(:Movie {id: 10, title: 'The Pursuit of Happyness', genre: 'Drama', rating: 8.0, language: 'English'});



3. THEATER Table (Label: Theater)

CREATE

(:Theater {id: 1, name: 'PVR Phoenix', location: 'Bangalore', capacity: 300, opened_date: '2015-06-01'}),

(:Theater {id: 2, name: 'INOX Garuda', location: 'Bangalore', capacity: 250, opened_date: '2016-08-15'}),

(:Theater {id: 3, name: 'Cinepolis Nexus', location: 'Mumbai', capacity: 400, opened_date: '2017-03-20'}),

(:Theater {id: 4, name: 'Carnival Kolkata', location: 'Kolkata', capacity: 350, opened_date: '2014-11-11'}),

(:Theater {id: 5, name: 'INOX South City', location: 'Kolkata', capacity: 320, opened_date: '2018-09-30'}),

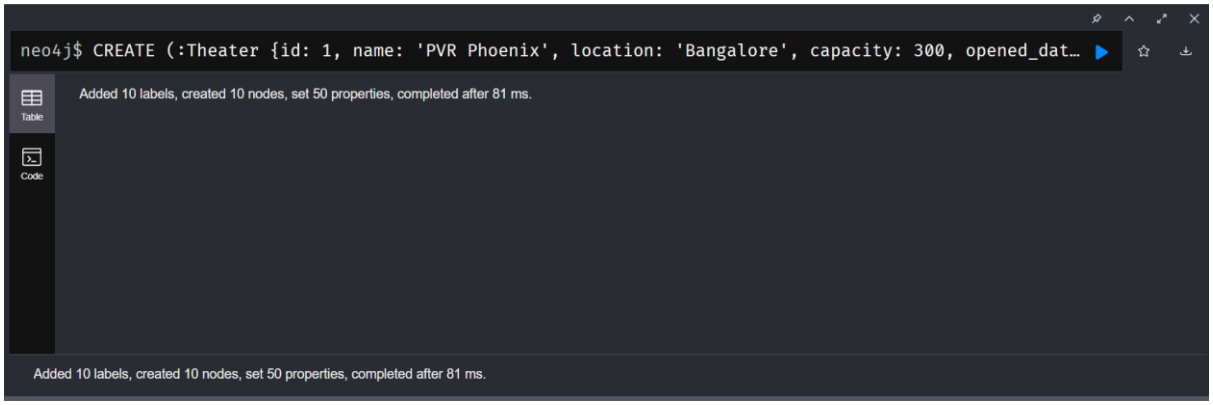
(:Theater {id: 6, name: 'Miraj Cinemas', location: 'Delhi', capacity: 280, opened_date: '2019-01-05'}),

(:Theater {id: 7, name: 'PVR Elante', location: 'Chandigarh', capacity: 300, opened_date: '2016-07-12'}),

(:Theater {id: 8, name: 'PVR Lulu', location: 'Kochi', capacity: 270, opened_date: '2017-10-21'}),

(:Theater {id: 9, name: 'INOX Quest', location: 'Kolkata', capacity: 260, opened_date: '2013-12-12'}),

(:Theater {id: 10, name: 'PVR Ambience', location: 'Gurgaon', capacity: 350, opened_date: '2020-02-01'});



4. SHOW Table (Label: Show)

```
CREATE
(:Show {id: 1, time: '10:00', date: '2024-06-15', price: 200, screen: 'Screen 1'}),
(:Show {id: 2, time: '13:00', date: '2024-06-15', price: 250, screen: 'Screen 2'}),
(:Show {id: 3, time: '16:00', date: '2024-06-15', price: 200, screen: 'Screen 3'}),
(:Show {id: 4, time: '19:00', date: '2024-06-15', price: 300, screen: 'Screen 1'}),
(:Show {id: 5, time: '22:00', date: '2024-06-15', price: 350, screen: 'Screen 2'}),
(:Show {id: 6, time: '10:00', date: '2024-06-16', price: 200, screen: 'Screen 1'}),
(:Show {id: 7, time: '13:00', date: '2024-06-16', price: 250, screen: 'Screen 2'}),
(:Show {id: 8, time: '16:00', date: '2024-06-16', price: 300, screen: 'Screen 3'}),
(:Show {id: 9, time: '19:00', date: '2024-06-16', price: 350, screen: 'Screen 1'}),
(:Show {id: 10, time: '22:00', date: '2024-06-16', price: 400, screen: 'Screen 2'});
```

5. BOOKING Table (Label: Booking)

CREATE

(:Booking {id: 1, ticket_count: 2, total_amount: 400, status: 'Confirmed', booking_time: '2024-06-10T10:30:00'}),

(:Booking {id: 2, ticket_count: 3, total_amount: 750, status: 'Confirmed', booking_time: '2024-06-10T11:00:00'}),

(:Booking {id: 3, ticket_count: 1, total_amount: 250, status: 'Confirmed', booking_time: '2024-06-10T11:30:00'}),

(:Booking {id: 4, ticket_count: 4, total_amount: 1200, status: 'Confirmed', booking_time: '2024-06-10T12:00:00'}),

(:Booking {id: 5, ticket_count: 2, total_amount: 600, status: 'Cancelled', booking_time: '2024-06-10T12:30:00'}),

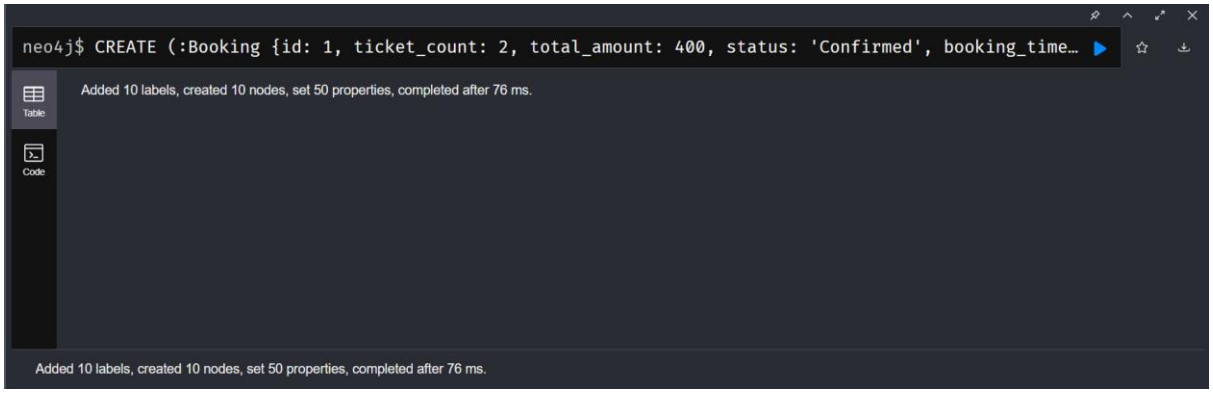
(:Booking {id: 6, ticket_count: 1, total_amount: 200, status: 'Confirmed', booking_time: '2024-06-10T13:00:00'}),

(:Booking {id: 7, ticket_count: 3, total_amount: 900, status: 'Confirmed', booking_time: '2024-06-10T13:30:00'}),

(:Booking {id: 8, ticket_count: 2, total_amount: 700, status: 'Confirmed', booking_time: '2024-06-10T14:00:00'}),

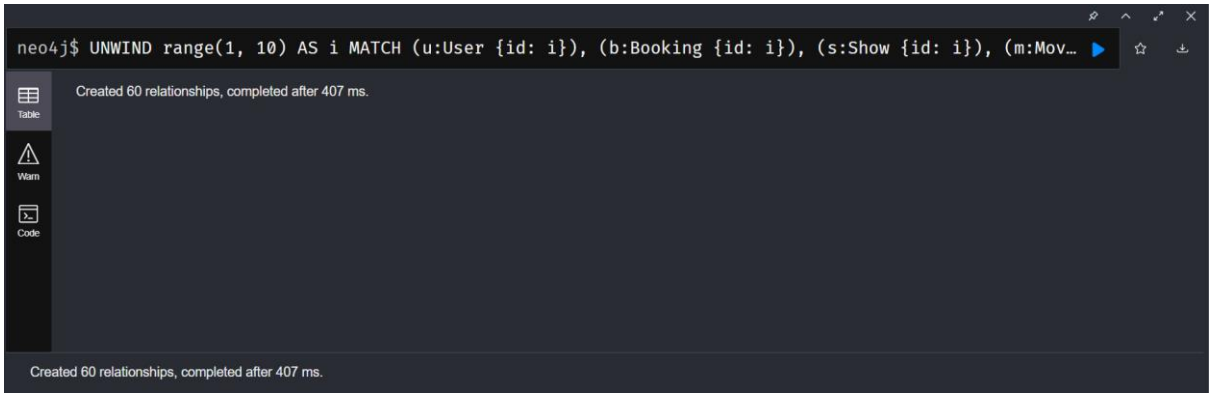
(:Booking {id: 9, ticket_count: 1, total_amount: 350, status: 'Cancelled', booking_time: '2024-06-10T14:30:00'}),

(:Booking {id: 10, ticket_count: 2, total_amount: 800, status: 'Confirmed', booking_time: '2024-06-10T15:00:00'});



## CREATE RELATIONSHIP BETWEEN TABLES

```
UNWIND range(1, 10) AS i
MATCH (u:User {id: i}), (b:Booking {id: i}), (s:Show {id: i}), (m:Movie {id: i}), (t:Theater {id: i})
CREATE
  (u)-[:BOOKED]->(b),
  (b)-[:FOR_SHOW]->(s),
  (s)-[:PLAYS]->(m),
  (s)-[:IN_THEATER]->(t),
  (b)-[:BOOKED_BY]->(u),
  (u)-[:WATCHED]->(m);
```



## READ Operations (R)

1. View All Users and Their Booked Movies:

MATCH (u:User)-[:BOOKED]->(:Booking)-[:FOR_SHOW]->(:Show)-[:PLAYS]->(m:Movie)

RETURN u.name AS UserName, m.title AS MovieTitle;

```
neo4j$ MATCH (u:User)-[:BOOKED]→(:Booking)-[:FOR_SHOW]→(:Show)-[:PLAYS]→(m:Movie) RETURN u.name A…
```

| | UserName | MovieTitle |
|---|---|---|
| 1 | "Aditi" | "Inception" |
| 2 | "Ravi" | "3 Idiots" |
| 3 | "Priya" | "Interstellar" |
| 4 | "Arjun" | "Dangal" |
| 5 | "Neha" | "The Dark Knight" |
| 6 | "Kiran" | "Parasite" |

Started streaming 10 records after 27 ms and completed after 37 ms.

2. See Which Movie is Played in Which Theater:

MATCH (s:Show)-[:PLAYS]->(m:Movie), (s)-[:IN_THEATER]->(t:Theater)

RETURN m.title AS MovieTitle, t.name AS TheaterName;

```
neo4j$ MATCH (s:Show)-[:PLAYS]→(m:Movie), (s)-[:IN_THEATER]→(t:Theater) RETURN m.title AS MovieTit…
```

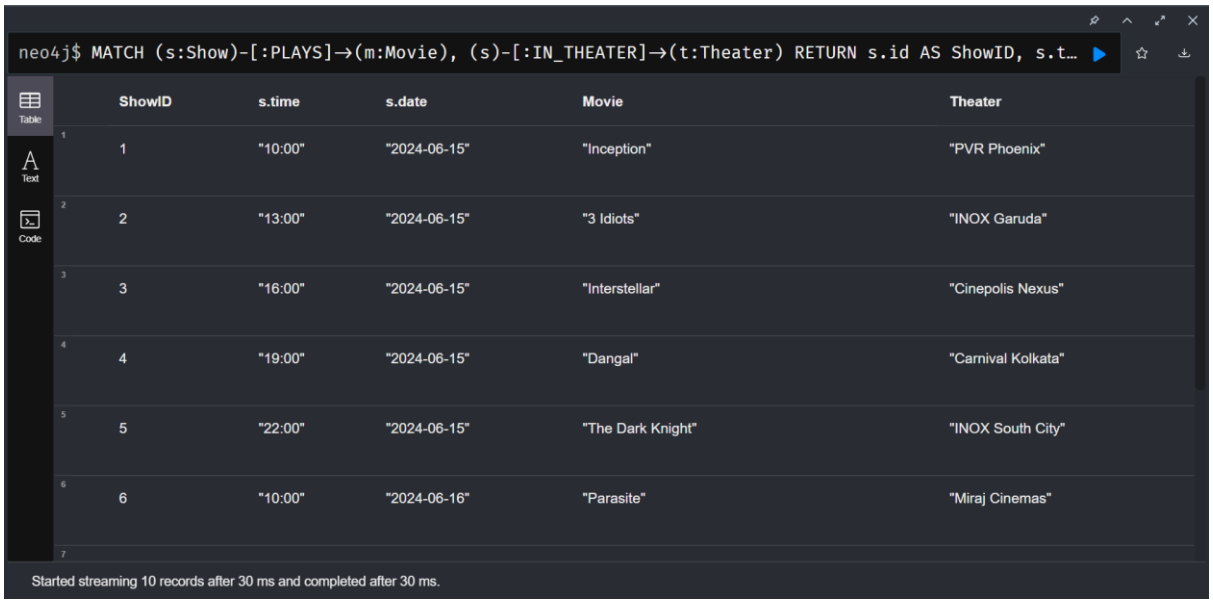
| | MovieTitle | TheaterName |
|---|---|---|
| 1 | "Inception" | "PVR Phoenix" |
| 2 | "3 Idiots" | "INOX Garuda" |
| 3 | "Interstellar" | "Cinepolis Nexus" |
| 4 | "Dangal" | "Carnival Kolkata" |
| 5 | "The Dark Knight" | "INOX South City" |
| 6 | "Parasite" | "Miraj Cinemas" |

Started streaming 10 records after 45 ms and completed after 54 ms.

3. Get All Shows and Their Time, Date, Theater, and Movie:

MATCH (s:Show)-[:PLAYS]->(m:Movie), (s)-[:IN_THEATER]->(t:Theater)

RETURN s.id AS ShowID, s.time, s.date, m.title AS Movie, t.name AS Theater;




| | ShowID | s.time | s.date | Movie | Theater |
|---|---|---|---|---|---|
| 1 | 1 | "10:00" | "2024-06-15" | "Inception" | "PVR Phoenix" |
| 2 | 2 | "13:00" | "2024-06-15" | "3 Idiots" | "INOX Garuda" |
| 3 | 3 | "16:00" | "2024-06-15" | "Interstellar" | "Cinepolis Nexus" |
| 4 | 4 | "19:00" | "2024-06-15" | "Dangal" | "Carnival Kolkata" |
| 5 | 5 | "22:00" | "2024-06-15" | "The Dark Knight" | "INOX South City" |
| 6 | 6 | "10:00" | "2024-06-16" | "Parasite" | "Miraj Cinemas" |
| 7 | | | | | |

## UPDATE operations (U):

### 1. Update a User's Email:

MATCH (u:User {id: 1})

SET u.email = "newemail1@example.com"

RETURN u;


### 2. Update Booking Ticket Count and Recalculate Amount:

MATCH (b:Booking {id: 1})

SET b.ticket_count = 4,

b.total_amount = 4 * 200

RETURN b;

## DELETE operations (D):

**1. Delete a Specific User and Their Relationships:**

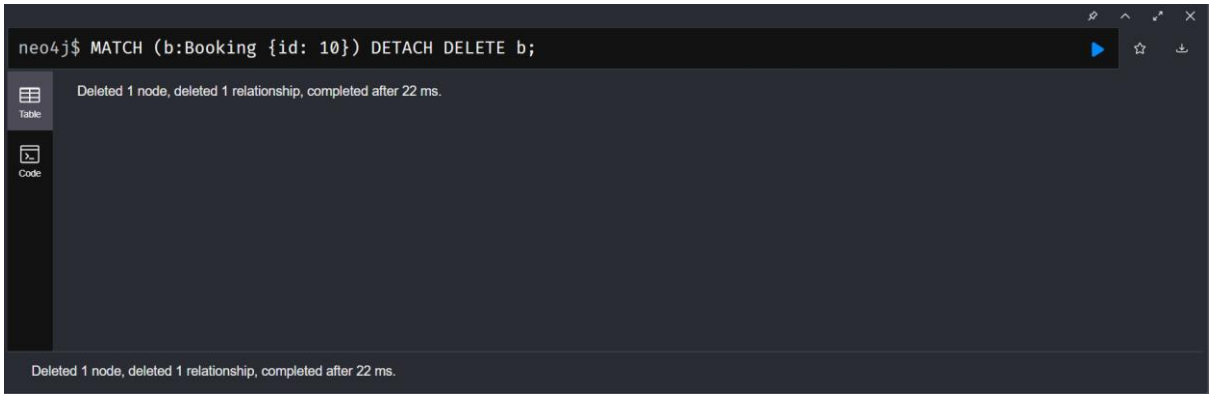
MATCH (u:User {id: 10})

DETACH DELETE u;


**2. Delete a Booking Record:**

MATCH (b:Booking {id: 10})

DETACH DELETE b;



# FINAL GRAPH REPRESENTATION