**Defining the Database**

Let's create a simple database containing people and cities, and the relationships between them.

**Create People and Cities Nodes**

CREATE (alice:Person {name: 'Alice', age: 30}),

(bob:Person {name: 'Bob', age: 25}),

(charlie:Person {name: 'Charlie', age: 35}),

(New York: City {name: 'New York'}),

(london:City {name: 'London'}),

(Paris:City {name: 'Paris'})

**Creating relationships between people and cities**

MATCH (alice:Person {name: 'Alice'}), (newyork:City {name: 'New York'})

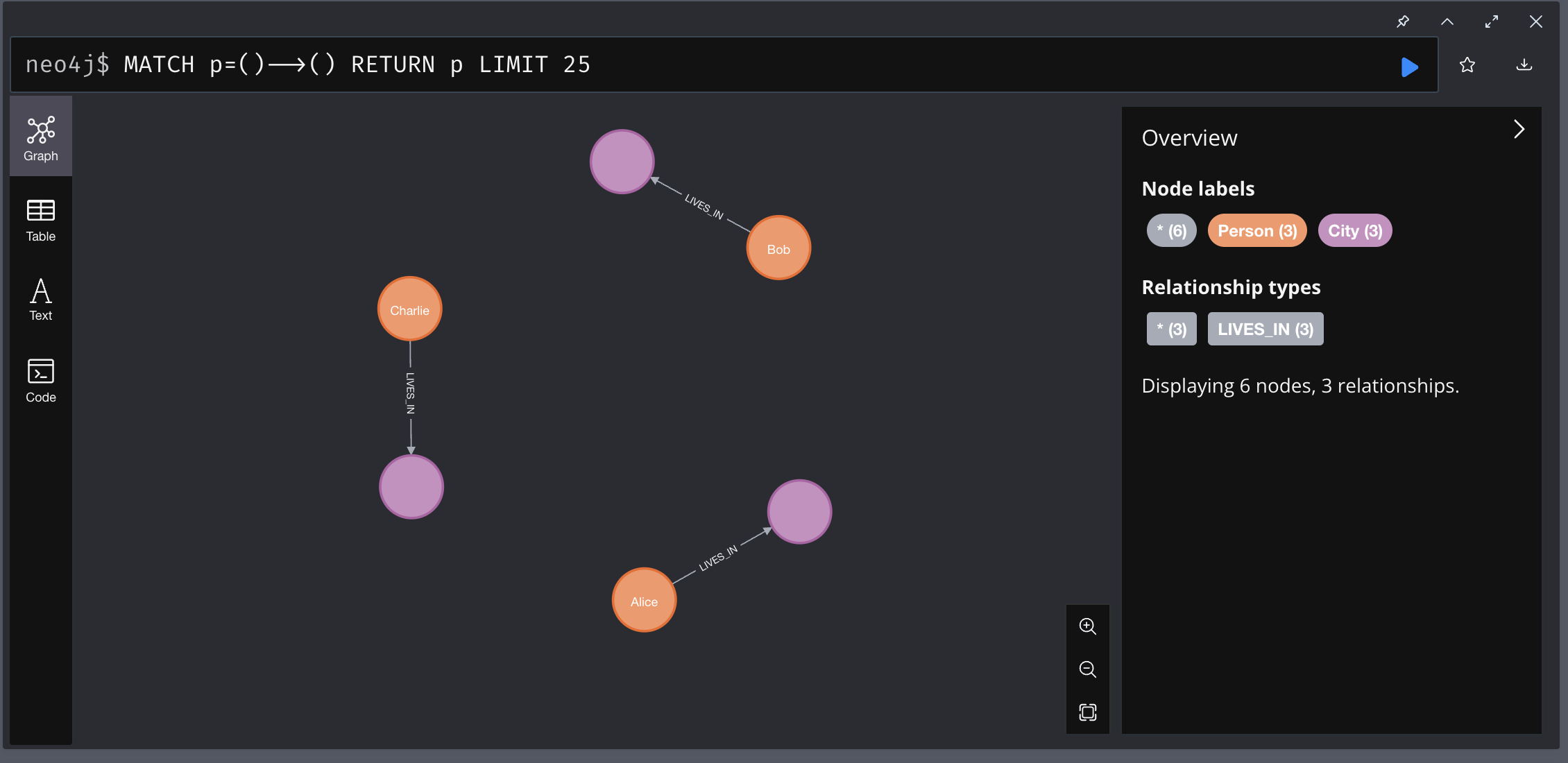
CREATE (alice)-[:LIVES\_IN]->(newyork)

MATCH (bob:Person {name: 'Bob'}), (london:City {name: 'London'})

CREATE (bob)-[:LIVES\_IN]->(london)

MATCH (charlie:Person {name: 'Charlie'}), (paris:City {name: 'Paris'})

CREATE (charlie)-[:LIVES\_IN]->(paris)



**Neo4j Exercises**

1. **Exercise 1**: Add more people: Create nodes for two new people, Dave and Eve, with their respective ages. You can add more people.

Query:

CREATE (dave:Person {name: 'Dave', age: 28}),

(eve:Person {name: 'Eve', age: 22});

Result:

MATCH (p:Person) RETURN p;

Text::

p │

╞═══════════════════════════════════╡

│(:Person {name: "Alice",age: 30}) │

├───────────────────────────────────┤

│(:Person {name: "Bob",age: 25}) │

├───────────────────────────────────┤

│(:Person {name: "Charlie",age: 35})│

├───────────────────────────────────┤

│(:Person {name: "Dave",age: 28}) │

├───────────────────────────────────┤

│(:Person {name: "Eve",age: 22}) │

Image:

A screenshot of a computer

Description automatically generated

1. **Exercise 2**: Relate people to cities: Relate Dave to New York City (LIVES\_IN). You can add more relationships.

Query:

MATCH (dave:Person {name: 'Dave'}), (newyork:City {name: 'New York'})

CREATE (dave)-[:LIVES\_IN]->(newyork);

MATCH (eve:Person {name: 'Eve'}), (paris:City {name: 'Paris'})

CREATE (eve)-[:LIVES\_IN]->(paris);

Result:

MATCH (p:Person)-[r:LIVES\_IN]->(c:City) RETURN p, r, c;

Text:

╒═══════════════════════════════════╤═══════════╤══════════════════════════╕

│p │r │c │

╞═══════════════════════════════════╪═══════════╪══════════════════════════╡

│(:Person {name: "Alice",age: 30}) │[:LIVES\_IN]│(:City {name: "New York"})│

├───────────────────────────────────┼───────────┼──────────────────────────┤

│(:Person {name: "Bob",age: 25}) │[:LIVES\_IN]│(:City {name: "London"}) │

├───────────────────────────────────┼───────────┼──────────────────────────┤

│(:Person {name: "Charlie",age: 35})│[:LIVES\_IN]│(:City {name: "Paris"}) │

├───────────────────────────────────┼───────────┼──────────────────────────┤

│(:Person {name: "Dave",age: 28}) │[:LIVES\_IN]│(:City {name: "New York"})│

├───────────────────────────────────┼───────────┼──────────────────────────┤

│(:Person {name: "Eve",age: 22}) │[:LIVES\_IN]│(:City {name: "Paris"}) │

└───────────────────────────────────┴───────────┴──────────────────────────┘

Image:

A screenshot of a computer

Description automatically generated

1. **Exercise 3**: Query all people: Perform a query to get all nodes of type Person.

Query:

MATCH (p:Person) RETURN p;

Result:

Text:

╒═══════════════════════════════════╕

│p │

╞═══════════════════════════════════╡

│(:Person {name: "Alice",age: 30}) │

├───────────────────────────────────┤

│(:Person {name: "Bob",age: 25}) │

├───────────────────────────────────┤

│(:Person {name: "Charlie",age: 35})│

├───────────────────────────────────┤

│(:Person {name: "Dave",age: 28}) │

├───────────────────────────────────┤

│(:Person {name: "Eve",age: 22}) │

└───────────────────────────────────┘

Image:

A screenshot of a computer

Description automatically generated

1. **Exercise 4**: Query residence relationships: Perform a query to obtain all LIVES\_IN relationships between people and cities.

Query:

MATCH (p:Person)-[r:LIVES\_IN]->(c:City)

RETURN p.name AS Person, c.name AS City, r;

Result:

Text:

╒═════════╤══════════╤═══════════╕

│Person │City │r │

╞═════════╪══════════╪═══════════╡

│"Alice" │"New York"│[:LIVES\_IN]│

├─────────┼──────────┼───────────┤

│"Bob" │"London" │[:LIVES\_IN]│

├─────────┼──────────┼───────────┤

│"Charlie"│"Paris" │[:LIVES\_IN]│

├─────────┼──────────┼───────────┤

│"Dave" │"New York"│[:LIVES\_IN]│

├─────────┼──────────┼───────────┤

│"Eve" │"Paris" │[:LIVES\_IN]│

└─────────┴──────────┴───────────┘

1. **Exercise 5**: Consult people over 30 years of age: Make a consultation to obtain all people over 30 years of age.

Query:

MATCH (p:Person)

WHERE p.age > 30

RETURN p.name AS Name, p.age AS Age;

Result:

Text:

╒═════════╤═══╕

│Name │Age│

╞═════════╪═══╡

│"Charlie"│35 │

└─────────┴───┘

1. **Exercise 6**: Update Eve's age: Update Eve's age to 23.

Query:

MATCH (eve:Person {name: 'Eve'})

SET eve.age = 23

RETURN eve.name AS Name, eve.age AS UpdatedAge;

Result:

Text:

╒═════╤══════════╕

│Name │UpdatedAge│

╞═════╪══════════╡

│"Eve"│23 │

└─────┴──────────┘

1. **Exercise 7**: Delete Bob's Residence Relationship: Delete the residence relationship between Bob and the City of London.

Query:

MATCH (bob:Person {name: 'Bob'})-[r:LIVES\_IN]->(london:City {name: 'London'})

DELETE r;

MATCH (bob:Person {name: 'Bob'})-[r:LIVES\_IN]->(c:City)

RETURN bob.name AS Person, c.name AS City, r;

Result:

Text:

(no changes, no records)

Image

A screenshot of a computer

Description automatically generated

1. **Exercise 8**: Delete Dave's node: Delete Dave's node.

Query:

MATCH (dave:Person {name: 'Dave'})

DETACH DELETE dave;

Result:

A screenshot of a computer

Description automatically generated

1. **Exercise 9**: Create a Company Node and Relate It to Alice: Create a node for a company called TechCorp and relate Alice to this company. Add more business nodes and match them to people.

Query:

CREATE (techcorp:Company {name: 'TechCorp'})

WITH techcorp

MATCH (alice:Person {name: 'Alice'})

CREATE (alice)-[:WORKS\_AT]->(techcorp);

CREATE (innovatech:Company {name: 'Innovatech'})

WITH innovatech

MATCH (eve:Person {name: 'Eve'})

CREATE (eve)-[:WORKS\_AT]->(innovatech);

CREATE (softsolutions:Company {name: 'SoftSolutions'})

WITH softsolutions

MATCH (charlie:Person {name: 'Charlie'})

CREATE (charlie)-[:WORKS\_AT]->(softsolutions);

Result:

MATCH (p:Person)-[r:WORKS\_AT]->(c:Company)

RETURN p.name AS Person, c.name AS Company, r;

Text:

╒═════════╤═══════════════╤═══════════╕

│Person │Company │r │

╞═════════╪═══════════════╪═══════════╡

│"Alice" │"TechCorp" │[:WORKS\_AT]│

├─────────┼───────────────┼───────────┤

│"Charlie"│"SoftSolutions"│[:WORKS\_AT]│

├─────────┼───────────────┼───────────┤

│"Eve" │"Innovatech" │[:WORKS\_AT]│

└─────────┴───────────────┴───────────┘

1. **Exercise 10**: Consult people and the companies where they work: Consult to obtain the names of the people and companies where they work.

Query:

MATCH (p:Person)-[:WORKS\_AT]->(c:Company)

RETURN p.name AS Person, c.name AS Company;

Result:

╒═════════╤═══════════════╕

│Person │Company │

╞═════════╪═══════════════╡

│"Alice" │"TechCorp" │

├─────────┼───────────────┤

│"Charlie"│"SoftSolutions"│

├─────────┼───────────────┤

│"Eve" │"Innovatech" │

└─────────┴───────────────┘

Delivery: Word file with the query and the result below each exercise. The result must be in the form of TEXT, optionally with the graph image. The image of the graph alone is not enough, but it helps.