01) The dataset involves **complex relationships between various entities** where the **interconnections are critical**, These types of **relationship-driven queries** are difficult and computationally expensive with traditional databases so **graph databases like Neo4j** make the appropriate choice. Some of **the advantages** of **Neo4J** are Fully flexible with labels, index free adjacency, efficient.

04) Traditional databases often struggle with **deep relationship traversal, Real time pattern matching , Evolving schema structures.**

How ever Neo4j is optimized for these scenarios because Relationships are first-class citizens, The schema is flexible and it provides a powerful declarative query language which is known as **cypher.**

Example(Querying the data)

For the example Neo4j has a built in movie database which is also known as Movie graph. Below is the overview of the movie graph.

Nodes

- (:Person) — represents actors, directors, etc.
- (:Movie) — represents movies

Relationships

- (:Person)-[:ACTED_IN]->(:Movie)
- (:Person)-[:DIRECTED]->(:Movie)
    - (:Movie)-[:REVIEWED_BY]->(:Person)
- (:Person)-[:FOLLOWS]->(:Person)

Query the Data(Find all movies an actor appeared in)

```
MATCH (p:Person {name: "Tom Hanks"})-[:ACTED_IN]->(m:Movie)

 RETURN m.title, m.released
```

Query the Data (Find all actors in a specific movie)

```
MATCH (m:Movie {title: "The Matrix"})<-[:ACTED_IN]-(actor:Person)

RETURN actor.name
```

05)

SQL	**No-SQL**
This database is relational	No-SQL is non-relational
Uses a structured query language	Use dynamic schemas for unstructured data
Vertically scalable	Horizontally scalable
Table based	Document, Key-Value graph, Wide Column Store