README.md

Supervision 3 Questions

1. To insert a movie into the document database, you would need to insert an object into the `movies` dictionary whose key is the movie ID and whose values contain informaton about the movie, the writers, the producers, the actors and so on. For each of the people involved the object in the `people` dictionary whose key is the person's ID would need to be edited to indicate that they wrote, produced, or acted in this new movie.

   To insert a movie into the relational database, you would have to run a query to insert a record into the `movies`, and then run subsequent queries to insert records into `acted_in`, `wrote`, `produced` and so in to associate the movie with people. These latter queries will likely need nested `SELECT` queries to determine the relevant person IDs

2. In the document database you would need to use a path-finding algorithm such as Djikstra's algorithm in order to find the shortest path between a given actor and Kevin Bacon. This is already implemented in neo4j with the `shortestPath` function.

3. The document database does contain all of the relevant information about the relationships between movies and people. Movies have `producers`, `writers`, and `actors` lists, and people have `wrote`, `produced`, and `acted_in` lists. Code could be written to automate the process of looking through these lists and construct a relationship model.

4. ii. Relational algebra describes sets with no repetition whereas SQL relations can contain duplicates. Relational algebra does not allow for `NULL` values whereas these are an intrinsic property of SQL. Relational algebra operates on relations, attributes, and tuples, which are analogous to the tables, columns, and rows of SQL. Relational algebra is also similar to SQL in the sense that both of their structures are self-similar. For example, a subsection of a table/relation is still a table/relation, and likewise for rows/tuples.

   iii. The equation is not always true. A counterexample is below:

   S:

a	b
1	v
v	3
v	v

   R:

a	c
v	4
1	4

   Left-hand side:

a	b	c
v	3	4
v	v	4

Right-hand side:

a	b	c
v	v	4

5. i. OLTP is the management of day-to-day transactional operations on a database. It works on current data and is optimised for updates, with low data redundancy.

   ii. OLAP is the technology which allows for complex analysis of large databases. The data is usually historical. OLAP is optimised for reads, with high data redundancy.

   iii. Schemas for OLAP systems usually have much more redundant data than those for OLTP systems as this makes for faster read operations as opposed to update operations. The schema for an OLTP system is usually quite closely linked to the entity-relationship model of the data, whereas the schema for OLAP systems is often a multi-dimensional star schema.

   iv. NoSQL databases provide high read and right throughput for large objects, but are typically not designed to work with the atomic-level data which is required for OLAP. Most NoSQL databases do not have optimised aggregation functions which is required for data analysis.

   v. If a bank wanted to keep track of customers' electronic purchases/transfers, they would require an OLTP system to handle this. They might need some graph functionality in order to cluster a customer's purchases together for the purposes of fraud detection. If they also wanted to store historical transaction data to analyse aggregate data from previous years. In such a case they might use a Neo4j database for the OLTP day-to-day operations, and every so often it would be backed up to a SQL database for OLAP.

6. ii. a. $\pi_{gender}(RockStar) \subseteq \{male, female\}$

   b. $\rho_{starname=name}(RockManager) \subseteq \sigma_{name}(RockStar)$

   c. $\sigma_{a1 \neq a2}(\rho_{a1=address}(\pi_{name,address}(RockStar)) \bowtie \rho_{a2=address}(\pi_{name,address}(RockStar))) = \emptyset$