Midterm Review

DSC 100

Generated on May 8, 2025

Questions

1. **Definitions:** Provide formal definitions.

   (a) Define a *Data Model*.

   (b) Define *First Normal Form (1NF)*.

2. **SQL Basics:** Write SQL queries.

   (a) Retrieve the names and genres of all movies.

   (b) Find all movies released after the year 2000.

3. **Joins:** Use appropriate JOINs.

   (a) List all genres of movies that "Marlon Brando" has acted in.

   (b) Show each employee's name alongside their manager's name using a self-join.

(c) Retrieve all movies and their actors, including movies with no actors.

4. **Aggregation and Grouping:**

(a) Count the total number of movies in the `Movie` table.

(b) Compute total revenue by genre for movies after 2008.

(c) List genres with total profit (revenue - budget) in descending order.

5. **Set Operations:**

(a) Retrieve actors who have acted in at least one 'Crime' or 'Horror' movie.

(b) Retrieve actors who have acted in both 'Crime' and 'Horror' movies.

(c) Retrieve actors in 'Action' movies but not in 'Drama'.

6. **Nested Queries:**

(a) For each actor, return the genre of the movie they acted in using a subquery.

(b) Find actors in any 'Sci-Fi' movie using EXISTS.

(c) List movies where all actors earn more than 100,000.

7. **Relational Algebra:** For each of the following, write a relational algebra expression:

(a) Select movies released after 2010.

(b) Project names and genres from `Movie`.

(c) Find intersection of actors in 'Crime' and 'Horror' movies.

8. **Relational Calculus:**

(a) Write a tuple relational calculus expression to retrieve movie names rated above 8.

(b) Write a domain relational calculus expression to retrieve actor names in 'Sci-Fi' genre.

Answer Key

1. Definitions:
   (a) Data Model: An abstraction comprising structure, constraints, and manipulation operations.
   (b) 1NF: A relation where all attributes are atomic (no nested or repeating groups).
2. SQL Basics:
   (a) `SELECT name, genre FROM Movie;`
   (b) `SELECT * FROM Movie WHERE year > 2000;`
3. Joins:
   (a) `SELECT DISTINCT m.genre FROM Movie m JOIN ActedIN a ON m.name = a.moviename WHERE a.actorname = 'Marlon Brando';`
   (b) `SELECT e1.Name AS Employee, e2.Name AS Manager FROM Employees e1 JOIN Employees e2 ON e1.ManagerID = e2.ID;`
   (c) `SELECT m.name, a.actorname FROM Movie m LEFT JOIN ActedIN a ON m.name = a.moviename;`
4. Aggregation:
   (a) `SELECT COUNT(*) FROM Movie;`
   (b) `SELECT genre, SUM(revenue) AS TotalRevenue FROM Movie WHERE year > 2008 GROUP BY genre;`
   (c) `SELECT genre, SUM(revenue - budget) AS TotalProfit FROM Movie GROUP BY genre ORDER BY TotalProfit DESC;`
5. Set Operations:
   (a) `SELECT DISTINCT actorname FROM Movie m JOIN ActedIN a ON m.name=a.moviename WHERE m.genre IN ('Crime','Horror');`
   (b) `SELECT actorname FROM ActedIN WHERE moviename IN (SELECT name FROM Movie WHERE genre='Crime') INTERSECT SELECT actorname FROM ActedIN WHERE moviename IN (SELECT name FROM Movie WHERE genre='Horror');`
   (c) `SELECT DISTINCT actorname FROM Movie m JOIN ActedIN a ON m.name=a.moviename WHERE m.genre='Action' EXCEPT SELECT DISTINCT actorname FROM Movie m JOIN ActedIN a ON m.name=a.moviename WHERE m.genre='Drama';`
6. Nested Queries:
   (a) `SELECT a.actorname, (SELECT m.genre FROM Movie m WHERE m.name=a.moviename) AS genre FROM ActedIN a;`
   (b) `SELECT DISTINCT a.actorname FROM ActedIN a WHERE EXISTS (SELECT 1 FROM Movie m WHERE m.name=a.moviename AND m.genre='Sci-Fi');`
   (c) `SELECT m.name FROM Movie m WHERE 100000 < ALL (SELECT a.salary FROM ActedIN a WHERE a.moviename = m.name);`

7. Relational Algebra:

   (a) $\sigma_{year>2010}(\text{Movie})$

   (b) $\pi_{name,genre}(\text{Movie})$

   (c) $\pi_{actorname}(\sigma_{genre='Crime'}(\text{Movie}) \bowtie \text{ActedIN}) \cap \pi_{actorname}(\sigma_{genre='Horror'}(\text{Movie}) \bowtie \text{ActedIN})$

8. Relational Calculus:

   (a) $\{m.name \mid m \in Movie \wedge m.rating > 8\}$

   (b) $\{a \mid \exists m(m \in Movie \wedge m.genre =' Sci - Fi' \wedge (m.name, a) \in ActedIN)\}$