Solution Assignment 03

Students (sid, sname, age, grade)

Courses (cid, cname, credits)

Enrollments (sid, cid, semester, grade)

1. Find the names of students enrolled in 'Database Systems'

Answer: πsname ((σcname = ’Database Systems’ (c) ⋈ E ⋈ s))

2. Find student IDs of students with a grade above 90.

Answer: πsid (σ grade>90 (Enrollment))

3. Find the names of students who have never taken a course with less than 3 credits.

Answer: πsname (π sid (s)−π sid (σ credits<3 (c)⋈E))

4. Find the IDs of students who have taken all courses.

Answer: π sid,cid (E)÷π cid (c)

5. Find the names of students who have taken at least two courses.

Answer: πsname (σE1.sid=E2.sid ∧ E1.cid≠E2.cid (E×ρE2(E))⋈s)

6. Find the names of students who have taken all courses in the Spring semester

Answer: πsname ((πsid,cid (σsemester=’Spring’ (E)) ÷ πcid (c)) ⋈s)

7. Find the student IDs of students whose grade is higher than some student named 'Aliyan'

Answer:

πs1.sid (σs1.grade>s2.grade[ρs1(E)×(σsname=’Aliyan’(s)⋈E)])

8. Find the student IDs of students whose grade is higher than every student named 'huzaifa’.

Answer:

πsid(E)−πs1.sid(σs1.grade≤s2.grade[ρs1(E)×(σsname=′huzaifa′(s)⋈E)])

9. Find the names of students who have enrolled in courses worth exactly 4 credits

Answer:

πsname ((σcredits=4 (c)⋈E) ⋈s)

10. Find the student IDs of students who have taken the course with the most credits

Answer:

πsid(E)⋈(πcid(c)−πc1.cid(σc1.credits<c2.credits(ρc1(c)×c)))

Relational Algebra and SQL Queries Explained

1. π_title (σ_inColor = 0 ∧ year > 1970 (Movie))

SQL Query:

SELECT title FROM Movie WHERE inColor = 0 AND year > 1970;

Explanation:

Filters the Movie relation to get all movies where inColor = 0 (black-and-white) and year > 1970. Projects the title.

2. π_title (σ_(studioName = 'MGM' ∧ year > 1970) ∨ length < 90 (Movie))

SQL Query:

SELECT title FROM Movie WHERE (studioName = 'MGM' AND year > 1970) OR length < 90;

Explanation:

Filters movies where either the studioName is 'MGM' and year > 1970, or length is less than 90 minutes. Projects the title.

3. π_title, length (σ_studioName = 'Disney' ∧ year = 1990 (Movie))

SQL Query:

SELECT title, length FROM Movie WHERE studioName = 'Disney' AND year = 1990;

Explanation:

Filters movies produced by 'Disney' in 1990. Projects the title and length.

4. **π_title, length/60 (σ_studioName = 'Disney' ∧ year = 1990 (Movie))**

SQL Query:

SELECT title, length / 60 AS length_in_hours FROM Movie WHERE studioName = 'Disney' AND year = 1990;

Explanation:

Similar to previous query but converts the length from minutes to hours.

5. **π_producerC (Movie ⋈ StarsIn ⋈ σ_starName = 'Harrison Ford' (StarsIn))**

SQL Query:

SELECT DISTINCT M.producerC FROM Movie M JOIN StarsIn S ON M.title = S.movieTitle AND M.year = S.movieYear WHERE S.starName = 'Harrison Ford';

Explanation:

Joins the Movie and StarsIn relations based on movie title and year. Filters where starName is 'Harrison Ford' and projects producerC.

6. **π_Star.name, Exec.name (σ_Star.address = Exec.address (MovieStar ⋈ MovieExec))**

SQL Query:

SELECT S.name AS movie_star, E.name AS movie_producer FROM MovieStar S, MovieExec E WHERE S.address = E.address;

Explanation:

Joins MovieStar and MovieExec relations based on matching addresses. Projects the names of movie stars and producers.

7. **π_producerC (σ_title ≠ 'Star Wars' (Movie))**

SQL Query:

SELECT DISTINCT producerC FROM Movie WHERE title != 'Star Wars';

Explanation:

Filters the Movie relation to exclude the movie titled 'Star Wars' and projects producerC.

8. **π_producerC (σ_studioName = 'Disney' (Movie)) ∩ π_producerC (σ_studioName = 'MGM' (Movie))**

SQL Query:

SELECT DISTINCT M1.producerC FROM Movie M1 WHERE M1.studioName = 'Disney' AND EXISTS (SELECT * FROM Movie M2 WHERE M2.studioName = 'MGM' AND M1.producerC = M2.producerC);

Explanation:

Finds the intersection of producers who have made movies for both 'Disney' and 'MGM'.

9. **π_title (Movie) - π_title (Movie ⋈ MovieExec)**

SQL Query:

SELECT title FROM Movie WHERE producerC NOT IN (SELECT name FROM MovieExec);

Explanation:

Retrieves titles of movies with no matching producer in the MovieExec table.

10. **S1=ρtitle1,year1,name1(StarsIn)**

    S2=ρtitle2,year2,name2(StarsIn)

    (S1) ⋈name1=name2 AND (title1 != title2 or year1!=year2)(S2)

SQL Query:

SELECT S1.title1, S1.year1, S1.name1, S2.title2, S2.year2, S2.name2 FROM StarsIn S1, StarsIn S2

WHERE S1.name1 = S2.name2 AND (S1.title1 != S2.title2 OR S1.year1 != S2.year2);

Explanation:

Retrieve the pairs of movies (with different titles or years) that share the same actor/actress. The result excludes the case where an actor appears in the same movie in the same year, as those are not considered "pairs of different movies."