SQL SERVER – TEST 4

Topic – Joins and Sub Query

Questions – 20

Date – 7/7/2024

CREATE TABLE Authors

(

Id INT PRIMARY KEY, -- The author’s unique ID within the database.

AuthorName VARCHAR(50), -- The author’s name.

BirthYear INT, -- The year when that author was born.

DeathYear INT -- The year when that author died (null if author alive)

)

GO

-- Sample Date:

INSERT INTO Authors VALUES

(1,'Marcella Cole',1983,NULL),

(2,'Lisa Mullins',1891,1950),

(3,'Dennis Stokes',1935,1994),

(4,'Randolph Vasquez',1957,2004),

(5,'Daniel Branson',1965,1990)

GO

CREATE TABLE Books

(

Id INT PRIMARY KEY, -- The ID of a given book.

AuthorId INT, -- The ID of the author who wrote that book.

Title VARCHAR(50), -- The book’s title.

PublishYear INT, -- The year when the book was published.

PublishingHouse VARCHAR(50), -- The name of the publishing house that printed

Rating NUMERIC -- The average rating for the book.

)

GO

-- SAMPLE DATA

INSERT INTO Books VALUES

(1,NULL,'Soulless girl',2008,'Golden Albatros',4.3),

(2,NULL,'Weak Heart',1980,'Diarmud Inc.',3.8),

(3,4,'Faith Of Light',1995,'White Cloud Press',4.3),

(4,NULL,'Memory Of Hope',2000,'Rutis Enterprises',2.7),

(5,6,'Warrior Of Wind',2005,'Maverick',4.6)

GO

CREATE TABLE Adaptation

(

BookId INT, -- The ID of the adapted book.

Type VARCHAR(20), -- The type of adaptation (e.g. movie, game, play, musical).

Title VARCHAR(50), -- The name of this adaptation.

ReleaseYear INT, -- The year when the adaptation was created.

Rating NUMERIC -- The average rating for the adaptation.

)

GO

-- Sample Data

INSERT INTO Adaptation VALUES

(1,'Movie','Gone With The Wolves: The Beginning',2008,3),

(3,'Movie','Companions Of Tomorrow',2001,4.2),

(5,'Game','Homeless Warrior',2008,4),

(2,'Movie','Blacksmith With Silver',2014,4.3),

(4,'Play','Patrons And Bearers',2004,3.2)

CREATE TABLE BooksReview

(

BookId INT , -- The ID of a reviewed book.

Review VARCHAR(100), -- The summary of the review.

Author VARCHAR(50) -- The name of the review's author.

)

GO

-- Sample Data

INSERT INTO BooksReview VALUES

(1,'An incredible book','Sylvia Jones'),

(1,'Great, although it has some flaws','Jessica Parker'),

(2,'Dennis Stokes takes the reader for a ride full of emotions','Thomas Green'),

(3,'Incredible craftsmanship of the author','Martin Freeman'),

(4,'Not the best book by this author','Jude Falth'),

(5,'Claudia Johnson at her best!','Joe Marqiz'),

(6,'I cannot recall more captivating plot','Alexander Durham')

GO

SELECT \* FROM dbo.Authors

SELECT \* FROM dbo.Books

SELECT \* FROM dbo.Adaptation

SELECT \* FROM dbo.BooksReview

Q1. Show the name of each author together with the title of the book they wrote and the year in which that book was published.

Q2. Show the name of each author together with the title of the book they wrote and the year in which that book was published. Show only books published after 2005.

Q3. Exercise: For each book, show its title, adaptation title, adaptation year, and publication year.

Include only books with a rating lower than the rating of their corresponding adaptation. Additionally, show only those books for which an adaptation was released within four years of the book’s publication.

Rename the title column from the book table to book\_title and the title column from the adaptation table to adaptation\_title.

Q4. Show the title of each book together with the title of its adaptation and the date of the release. Show all books, regardless of whether they had adaptations.

Q5. Show all books with their movie adaptations. Select each book's title, the name of its publishing house, the title of its adaptation, and the type of the adaptation. Keep the books with no adaptations in the result.

Q6. Show the title of the book, the corresponding review, and the name of the review's author. Consider all books, even those that weren't reviewed.

Q7. Display the title of each book along with the name of its author. Show all books, even those without an author. Show all authors, even those who haven't published a book yet. Use a FULL JOIN.

/\* Dataset 2 \*/

CREATE TABLE dbo.departments

(

id int, -- The unique ID of the department.

name varchar(20) -- The department name, i.e. where a particular type of product is sold.

)

-- Sample Data

Insert Into dbo.departments VALUES

(1,'fruits'),

(2,'vegetables'),

(3,'seafood'),

(4,'deli'),

(5,'bakery'),

(6,'meat'),

(7,'dairy')

GO

CREATE TABLE product

(

id INT, -- The ID of a given product.

name VARCHAR(20), -- The product’s name.

department\_id INT, -- The ID of the department where the product is located.

shelf\_id INT, -- The ID of the shelf of that department where the product is located.

producer\_id INT, -- The ID of the company that manufactures this product.

price NUMERIC -- The product’s price.

)

GO

-- Sample Data

INSERT INTO product VALUES

(1,'Apple',1,1,NULL,0.5),

(2,'Avocado',1,1,7,1),

(3,'Banana',1,1,7,0.5),

(4,'Grapefruit',NULL,1,1,0.5),

(5,'Grapes',1,1,4,2)

GO

CREATE TABLE nutrition\_data

(

product\_id INT, -- The ID of a product.

calories INT, -- The calorific value of that product.

fat NUMERIC, -- The amount of fat in that product.

carbohydrate NUMERIC, -- The amount of carbohydrates in that product.

protein NUMERIC -- The amount of protein in that product.

)

GO

-- Sample Data

INSERT INTO nutrition\_data VALUES

(1,130,0,5,1),

(2,50,4.5,3,1),

(3,110,0,30,1),

(4,60,0,15,1),

(NULL,90,0,23,0)

GO

CREATE TABLE producer

(

id INT, -- The ID of a given food producer.

name VARCHAR(50) -- The name of the producer.

)

GO

-- Sample Data

INSERT INTO producer VALUES

(1,'BeHealthy'),

(2,'HealthyFood Inc.'),

(3,'SupremeFoods'),

(4,'Foodie'),

(5,'Gusto'),

(6,'Baker n Sons'),

(7,'GoodFoods'),

(8,'Tasty n Healthy')

GO

CREATE TABLE sales\_history

(

date DATE, -- The date of sale.

product\_id INT, -- The ID of the product sold.

amount MONEY -- The amount of that product sold on a particular day.

)

GO

-- Sample Data

INSERT INTO sales\_history VALUES

('2015-01-14',1,14),

('2015-01-14',1,13),

('2015-01-15',2,2),

('2015-01-16',2,6),

('2015-01-17',3,8)

GO

SELECT \* FROM departments

SELECT \* FROM sales\_history

SELECT \* FROM producer

SELECT \* FROM nutrition\_data

SELECT \* FROM product

Q8. List all products that have fewer than 150 calories. For each product, show its name (rename the column product) and the name of the department where it can be found (name the column department).

Q9. For each product, display the:

Name of the company that produced it (name the column producer\_name).

Name of the department where the product is located (name it department\_name).

Product name (name it product\_name).

Total number of carbohydrates in the product.

Q10. For each product, show its name, price, producer name, and department name.

Alias the columns as product\_name, product\_price, producer\_name, and department\_name, respectively. Include all the products, even those without a producer or department. Also, include the producers and departments without a product.

/\* Dataset 3 \*/

CREATE TABLE workshop\_workers

(

id INT, -- The worker’s ID.

name VARCHAR(25), --The worker’s first and last name.

specialization VARCHAR(25), -- The worker's specialization.

master\_id INT, -- The ID of the worker's supervisor.

experience INT, -- The worker's years of experience.

project\_id INT -- The ID of the project to which the worker is currently assigned.

)

GO

-- Sample Data

INSERT INTO workshop\_workers VALUES

(1,'Mathew Conn','woodworking',NULL,20,1),

(2,'Kate Brown','woodworking',1,4,1),

(3,'John Doe','incrusting',5,3,1),

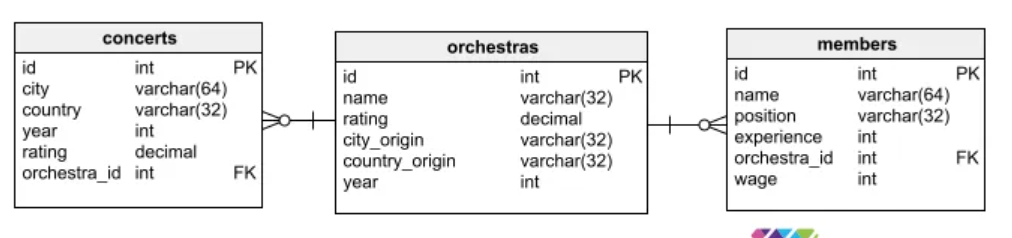
(4,'John Kowalsky','watchmaking',7,2,3),

(5,'Suzan Gregowitch','incrusting',NULL,15,4)

GO

Q11. Show all workers names together with the names of their direct supervisors. Rename the columns apprentice\_name and master\_name, respectively. Consider only workers who have a supervisor (i.e. a master).

/\* Subquery – Question Set \*/



/\* SUBQUERY \*/

CREATE TABLE concerts

(

id int primary key,

city varchar(50),

country varchar(50),

year int,

rating decimal,

orchestra\_id int

)

go

CREATE TABLE orchestras

(

id int primary key,

name varchar(50),

city\_origin varchar(50),

country\_origin varchar(50),

year int,

rating decimal

)

go

CREATE TABLE members

(

id int primary key,

name varchar(50),

position varchar(50),

experience int,

weg int,

orchestra\_id int

)

go

* The orchestras table stores all orchestras. The columns are id, name, rating, city\_origin, country\_origin, and year in which the orchestra was founded.
* The concerts table contains all concerts played by the orchestras. The columns are id, city, country, year, rating, and orchestra\_id (references the orchestras table).
* The members table stores the members of (i.e. musicians playing in) each orchestra. The columns are id, name, position (i.e. the instrument played), wage, experience, and orchestra\_id (references the orchestras table).

Q12. Select the names of all orchestras that have the same city of origin as any city in which any orchestra performed in 2013.

Q13. Select the names and positions (i.e. instrument played) of all orchestra members that have above 10 years of experience and do not belong to orchestras with a rating below 8.0.

Q14. Show the name and position of orchestra members who earn more than the average wage of all violinists.

Q15. Show the names of orchestras that were created after the 'Chamber Orchestra' and have a rating greater than 7.5.



Q16. Write a query to display name (first name, last name) for those employees who gets more salary than employee who is having id 103.

Q17. Write a query to display EmployeeId, EmployeeName (FirstName and LastName) for all those employees who are earning more than the average salary.

Q18. Write a query to display EmployeeId, EmployeeName (FirstName and LastName) and salary of those employees who reports to ‘**Payam’.**

Q19. Write a query to display employeeId, EmployeeName (FirstName and LastName), salary and SalaryStatus column with value HIGH or LOW based on salary. If salary of employee if greater than average salary then HIGH else LOW as values in SalaryStatus column.

Q20. Find the second highest salary and employee Id holding second highest salary.