Ayhan Demir

1. Write a query to find the tracks which have longer duration than the average. Your query should include TrackId, Name, Milliseconds.

SELECT TrackId, Name, Milliseconds from tracks WHERE Milliseconds > (SELECT AVG(Milliseconds) from tracks);

Multiple-Row Subqueries:

1. Write a query that returns the customers whose sales representatives are Jane Peacock and Margaret Park. Your query should include CustomerID, First Name and Last Name of the customers.

SELECT CustomerId, FirstName, LastName, SupportRepId FROM customers WHERE SupportRepId IN (SELECT EmployeeId FROM employees WHERE FirstName IN ('Jane', 'Margaret'));

1. Write the same query above this time using JOIN and add columns employee’s first name and last name.

SELECT c.CustomerId, c.FirstName, c.LastName, c.SupportRepId, e.FirstName, e.LastName FROM customers c JOIN employees e ON e.EmployeeId = c.SupportRepId WHERE e.FirstName = 'Jane' or e.FirstName = 'Margaret' ;

DDL (CREATE, ALTER, DELETE) and DML (SELECT, INSERT, UPDATE, DELETE) Statements

1. Create two tables inside the chinook database.

First table will be used to track the available courses inside the organization.

And the second table will be used to track employee’s registration status for each course (employees can be registered different courses on different times

( You can think about logic as follows; there are courses like SQL, HTML and etc. with different ID’s and prices and employees can join different courses.

There are start and end dates for each course and also the score that employees get.

E.g. John registered for the SQL course, Jane registered for the HTML course etc.

You can give NULL to dates and scores for now, just try to build the structure)

Name of the table: courses

Columns:

* CourseId (Primary Key)
* CourseName (Cannot be null)
* CoursePrice

Name of the table: courseRegistrations

Columns:

* RegistrationID PRIMARY KEY,
* CourseId NOT NULL,
* EmployeeId NOT NULL,
* StartDate DATE, can be NULL,
* EndDate DATE , can be NULL,
* Score int , can be NULL,

(You should assign EmployeeId and CourseId as foreign keys so that you can join courseRegistrations table with courses and employees table)

CREATE TABLE courses ( CourseId INT PRIMARY KEY, CourseName TEXT NOT NULL, CoursePrice)

CREATE TABLE courseRegistrations ( RegistrationID INT PRIMARY KEY, CourseId INT Not Null, EmployeeId INT NOT NULL, StartDate date, EndDate date, Score INT)

1. Insert at least 5 rows into the courses and courseRegistration tables.

Be careful with the foreign keys (You can give NULL to dates and scores for now, just try to build the structure)

INSERT INTO courseRegistrations(RegistrationID, CourseId, EmployeeId, StartDate, EndDate, Score)

VALUES ( '7110', '1', '238', NULL,NULL, NULL);

( '7111', '2', '290', NULL,NULL, NULL),

( ‘7129’, ‘3’, ‘284’, NULL,NULL, NULL),

(‘7140’, ‘4’, ‘277’, NULL,NULL, NULL),

(‘7101’, ‘5’, ‘210’, NULL,NULL, NULL),

(‘7190’, ‘6’, ‘222’, NULL,NULL, NULL);

INSERT OR IGNORE INTO courses (CourseId, CourseName, CoursePrice)

VALUES ('238', 'HTML', '800');

( '290', 'SQL', '1050'),

('277', 'Operating System', '1050'),

('284', 'E-Commerce', '950'),

('210', 'IT Fundamental', '700'),

('222', 'Computer Hardware', '750');

1. Suddenly the last registered employee changed his mind and doesn’t want to participate in the course you inserted. So delete the last record from courseRegistration table

delete from courseRegistrations where EmployeeId = 1

1. The courses have started for the top 2 registered employees. Update the StartDate as you wish

UPDATE courseRegistrations

SET StartDate = '2020-11-10'

WHERE EmployeeId IN('5', '2');

1. The courses have finished for the top 2 registered employees. Update the EndDate and the score as you wish

update courseRegistrations

set EndDate = '2021-04-10',

Score = '79'

where EmployeeId IN ('5', '2');

1. Suddenly you realize you forgot to add PassScore for courses. Add it now to courses table, it can’t be NULL and give default value 50 for each course

ALTER TABLE courses  
ADD COLUMN PassScore int DEFAULT 50;

JOINS:

1. There are 3503 tracks on the database.

Some of these tracks have no selling record which means they haven’t been sold even one time.

Find these tracks (You should get 1519 distinct tracks).

select TrackId, name

from tracks

where TrackId not in (

select i.TrackId

from invoice\_items i

inner join tracks t

on i.TrackId = t.TrackId);

And also find the tracks that have been sold one or more times.(You should get 1984 distinct tracks)

select DISTINCT i.TrackId, t.Name

from invoice\_items i

inner join tracks t

on i.TrackId = t.TrackId ;