## Introduction

This exercise requires you to know the following aspects of SQL:

|  |  |
| --- | --- |
| CREATE TABLE | Concatenation |
| SQL Data Types | Formatting dates and numbers |
| INSERT INTO | Column Aliases |
| SELECT | Simple JOIN statements |
| WHERE clause | Complex JOIN statements |
| LIKE and wildcards |  |

## Exercise 1 – Northwind Queries (35 marks: 5 for each question)

* 1. Write a query that lists all Customers in either Paris or London. Include Customer ID, Company Name and all address fields.

1. SELECT CustomerID AS 'Customer ID', CompanyName AS 'Company Name',
2. Address, City, Region, PostalCode, Country
3. FROM Customers
4. WHERE City = 'Paris' OR City = 'London'
   1. List all products stored in bottles.
5. SELECT ProductName
6. FROM Products
7. WHERE QuantityPerUnit LIKE '%bottles'
   1. Repeat question above but add in the Supplier Name and Country.
8. SELECT ProductName, CompanyName, Country
9. FROM Products
10. INNER JOIN Suppliers
11. ON Products.SupplierID = Suppliers.SupplierID
12. WHERE QuantityPerUnit LIKE '%bottles'
    1. Write an SQL Statement that shows how many products there are in each category. Include Category Name in result set and list the highest number first.
13. SELECT CategoryName, COUNT(\*) AS 'Products per Category'
14. FROM Products
15. INNER JOIN Categories
16. ON Products.CategoryID = Categories.CategoryID
17. GROUP BY CategoryName
18. ORDER BY 'Products per Category' DESC
    1. List all UK employees using concatenation to join their title of courtesy, first name and last name together. Also include their city of residence.

SELECT TitleOfCourtesy + ' ' + FirstName + ' ' + LastName AS 'Name', City

FROM Employees

WHERE Country = 'UK'

* 1. Count how many Orders have a Freight amount greater than 100.00 and either USA or UK as Ship Country.

1. SELECT ShipCountry AS 'Ship Country', COUNT(\*) AS 'Freight greater than 100'
2. FROM Orders
3. WHERE Freight > 100
4. AND (ShipCountry = 'USA' OR ShipCountry = 'UK')
5. GROUP BY ShipCountry
   1. Write an SQL Statement to identify the Order Number of the Order with the highest amount of discount applied to that order.

SELECT TOP 2(OrderID) AS 'Order Number',

(Discount \* Quantity \* UnitPrice) AS 'Total Discount'

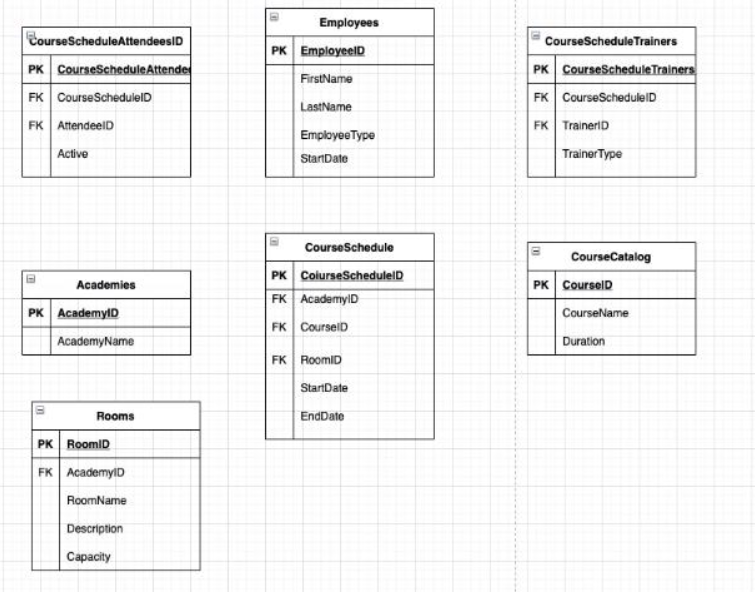
FROM [Order Details]

ORDER BY 'Total Discount' DESC

## Exercise 2 – Create Database Schema (20 marks)

2.1 Design and build a suitable set of fully normalised tables to store the following information:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Course Name** | **Start Date** | **End Date** | **Academy Name** | **Room Name** | **Trainer** | **Spartan** |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Adam Smith |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | John Williams |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Nick Willis |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Jenny Jones |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Katie Prince |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Peter Brown |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Mo Khan |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Juan Carlos |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Jan Miller |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Kyle Carpenter |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Alvarao Carlos |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Margaret Baker |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Oti Mwase |



You should have 7 tables in total.

CREATE DATABASE SpartaAcademy

CREATE TABLE Employees

(

    EmployeeID INT IDENTITY PRIMARY KEY,

    FirstName VARCHAR(20),

    LastName VARCHAR(20),

    EmployeeType CHAR(1),

    StartDate DATE

);

CREATE TABLE CourseCatalog

(

    CourseID INT IDENTITY PRIMARY KEY,

    CourseName VARCHAR(25),

    Duration INT

);

CREATE TABLE Academies

(

    AcademyID INT IDENTITY PRIMARY KEY,

    AcademyName VARCHAR(20),

);

CREATE TABLE Rooms

(

 RoomID INT IDENTITY PRIMARY KEY,

 AcademyID INT,

**CONSTRAINT** FK\_AcademyID\_Academies **FOREIGN KEY** (AcademyID) **REFERENCES** Academies(AcademyID),

 RoomName VARCHAR(15),

 Description VARCHAR(MAX),

 Capacity INT

)

CREATE TABLE CourseSchedule

(

 CourseScheduleID INT IDENTITY PRIMARY KEY,

 AcademyID INT,

**CONSTRAINT** FK\_AcademyID **FOREIGN KEY** (AcademyID) **REFERENCES** Academies(AcademyID),

 CourseID INT,

**CONSTRAINT** FK\_CourseID\_CourseCatalog **FOREIGN KEY** (CourseID) **REFERENCES** CourseCatalog(CourseID),

 RoomID INT,

**CONSTRAINT** FK\_RoomID\_Rooms **FOREIGN KEY** (RoomID) **REFERENCES** Rooms(RoomID),

 StartDate DATE,

 EndDate DATE

)

CREATE TABLE CourseScheduleAttendees

(

 CourseScheduleAttendeesID INT IDENTITY PRIMARY KEY,

 CourseScheduleID INT,

**CONSTRAINT** FK\_CourseScheduleID **FOREIGN KEY** (CourseScheduleID) **REFERENCES** CourseSchedule(CourseScheduleID),

 AttendeeID INT,

**CONSTRAINT** FK\_AttendeeID **FOREIGN KEY** (AttendeeID) **REFERENCES** Employees(EmployeeID),

 Active BINARY

)

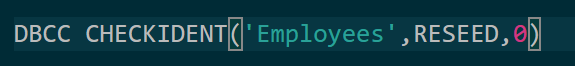
Create SQL files containing the SQL statements for the following:

* 1. One that contains all statements required to create and re-create this database. DDL only.

NOTE: You will be provided with the DML file which contains all of the INSERT statements required to populate the tables as per the example given. (20 marks)   
**IMPORTANT: Some tables will need the IDENTITY keyword to match this DML exactly.**

* 1. Add more sample data to include all current trainers and at least one TA (Training Assistant) and Spartans on your current course. (Included in above marks)

1. INSERT INTO Employees
2. (
3. FirstName, LastName, EmployeeType, StartDate
4. )
5. VALUES
6. (
7. 'Myles', 'Langston', 'S', '2021-05-10'
8. )
9. SELECT \* FROM Employees



To reset PKs toi start at 1

## Exercise 3 – Write SELECT statements for all of the following: (20 marks)

3.1 Produce a report similar to the above table (see 2.1) from all 7 tables using one SQL statement (use FORMAT for the dates). (10 marks)

SELECT cc.CourseName AS 'Course Name', cs.StartDate AS 'Start Date',

cs.EndDate AS 'End Date', a.AcademyName AS 'Academy Name',

r.RoomName AS 'Room Name',

t.FirstName + ' ' + t.LastName AS 'Trainer',

e.FirstName + ' ' + e.LastName AS 'Spartan'

FROM CourseSchedule cs

INNER JOIN CourseCatalog cc ON cs.CourseID = cc.CourseID

INNER JOIN Academies a ON cs.AcademyID = a.AcademyID

INNER JOIN Rooms r ON cs.RoomID = r.RoomID

INNER JOIN CourseScheduleAttendees csa ON cs.CourseScheduleID = csa.CourseScheduleID

INNER JOIN Employees e ON csa.AttendeeID = e.EmployeeID

INNER JOIN CourseScheduleTrainers cst ON cs.CourseScheduleID = cst.TrainerID

INNER JOIN Employees t ON cst.TrainerID = t.EmployeeID

3.2 Repeat 3.1 above but replace the Spartan Name column with Spartan Initials. (5 marks)

SELECT cc.CourseName AS 'Course Name', cs.StartDate AS 'Start Date',

cs.EndDate AS 'End Date', a.AcademyName AS 'Academy Name',

r.RoomName AS 'Room Name',

t.FirstName + ' ' + t.LastName AS 'Trainer',

LEFT(e.FirstName, 1) + '.' + LEFT(e.LastName, 1) AS 'Spartan Initials'

FROM CourseSchedule cs

INNER JOIN CourseCatalog cc ON cs.CourseID = cc.CourseID

INNER JOIN Academies a ON cs.AcademyID = a.AcademyID

INNER JOIN Rooms r ON cs.RoomID = r.RoomID

INNER JOIN CourseScheduleAttendees csa ON cs.CourseScheduleID = csa.CourseScheduleID

INNER JOIN Employees e ON csa.AttendeeID = e.EmployeeID

INNER JOIN CourseScheduleTrainers cst ON cs.CourseScheduleID = cst.TrainerID

INNER JOIN Employees t ON cst.TrainerID = t.EmployeeID

3.3 Add a new column after End Date headed “Check Date” which uses the DATEADD function to add 8 weeks to the Start Date column for BA Test courses and 12 weeks for any others. (5 marks)

SELECT cc.CourseName AS 'Course Name', cs.StartDate AS 'Start Date',

cs.EndDate AS 'End Date',

CASE

    WHEN CourseName Like '%Ba%'

    THEN DATEADD(mm,2,cs.StartDate)

    ELSE DATEADD(mm,3,cs.StartDate)

END AS 'Check Date', a.AcademyName AS 'Academy Name',

r.RoomName AS 'Room Name',

t.FirstName + ' ' + t.LastName AS 'Trainer',

LEFT(e.FirstName, 1) + '.' + Left(e.LastName, 1) AS 'Spartan Initials'

FROM CourseSchedule cs

INNER JOIN CourseCatalog cc ON cs.CourseID = cc.CourseID

INNER JOIN Academies a ON cs.AcademyID = a.AcademyID

INNER JOIN Rooms r ON cs.RoomID = r.RoomID

INNER JOIN CourseScheduleAttendees csa ON cs.CourseScheduleID = csa.CourseScheduleID

INNER JOIN Employees e ON csa.AttendeeID = e.EmployeeID

INNER JOIN CourseScheduleTrainers cst ON cs.CourseScheduleID = cst.TrainerID

INNER JOIN Employees t ON cst.TrainerID = t.EmployeeID

## Exercise 4 – Add Constraints (10 marks)

4.1 Add Primary Keys and Foreign Keys to the Sparta Academy database, where needed. (If not already included). (8 marks)

CREATE TABLE CourseScheduleAttendees

(

 CourseScheduleAttendeesID INT IDENTITY PRIMARY KEY,

 CourseScheduleID INT,

**CONSTRAINT** FK\_CourseScheduleID **FOREIGN KEY** (CourseScheduleID) **REFERENCES** CourseSchedule(CourseScheduleID),

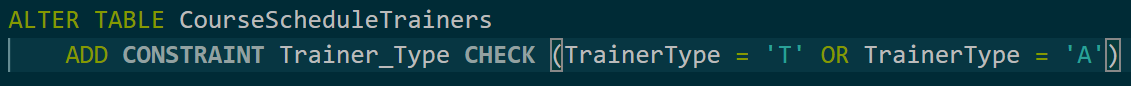
 AttendeeID INT,

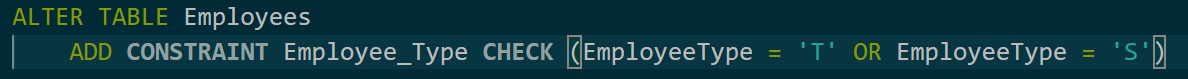
**CONSTRAINT** FK\_AttendeeID **FOREIGN KEY** (AttendeeID) **REFERENCES** Employees(EmployeeID),

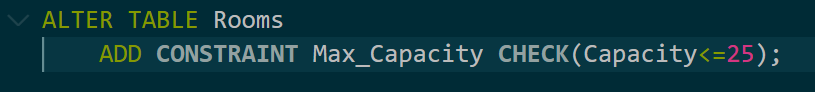
 Active BINARY

)

* 1. Add constraints for other tables including Trainer Type ('T' Trainer or 'A' Training Assistant) and Employee Type ('T' Trainer, 'S' Spartan) and set a maximum capacity for Rooms to 25. (2 marks)







## Standards (10 marks)

Remember to apply all the following standards:

* Use consistent capitalisation and indentation of SQL Statements
* Use concise and consistent table alias names
* Use column aliases to ensure tidy column headings (spaces and consistent capitalisation)
* Concatenate any closely related columns e.g. First Name and Last Name or Address and City etc
* Put comments throughout

