BAX-421 - Data Management
Homework 3

Mehul Rangwala

Fall 2022

Assignment Weight: 7.5% of your grade

Due: Monday, November 21, 2022 11:59 PM

Instructions

1. Please follow the instructions available on the recording on the homework submission guidelines.
2. Write the queries for two databases in MySQL:
   - ENTERTAINMENT AGENCY database
   - ACCOUNTS PAYABLE database
3. You should have already installed these databases from prior weeks. Continue to work with it.
4. Please reuse the ER diagrams for the Entertainment Agency and Accounts Payable databases from prior week assignments.

ENTERTAINMENT AGENCY DATABASE (MYSQL)

Question 1 *(2 points)*

For each customer, display the CustomerID, name of the customer (First name and Last name separated by a space), StyleName (style of music each customer prefers), and the total number of preferences for each customer.

Question 2 *(2 points)*

For each customer, display the CustomerID, name of the customer (First name and Last name separated by a space), StyleName (style of music each customer prefers), the total number of preferences for each customer, and a running total of the number of styles selected for all the customers. Display the results sorted by Customer Name.

Question 3 *(2 points)*

Display the Customer City, Customer, Number of Preferences of Music Styles, and a running total of preferences for each city overall.

Question 4 *(2 points)*

Assign a row number for each customer. Display their CustomerID, their combined (First and Last) name, and their state. Return the customers in alphabetical order.

Question 5 *(2 points)*

Assign a number for each customer within each city in each state. Display their Customer ID, their combined name (First and Last), their city, and their state. Return the customers in alphabetical order.

Question 6 *(2 points)*

Show a list of all engagements. Display the start date for each engagement, the name of the customer, and the entertainer. Number the entertainers overall and number the engagements within each start date.

Question 7 *(2 points)*

Rank all the entertainers based on the number of engagements booked for each. Arrange the entertainers into three groups (buckets). Remember to include any entertainers who haven't been booked for any engagements.

Question 8 *(2 points)*

Rank all the agents based on the total dollars associated with the engagements that they've booked. Make sure to include any agents that haven't booked any acts.

Question 9 *(2 points)*

Display a list of all of the engagements our entertainers are booked for. Display the entertainer's stage name, the customer's (combined) name, and the start date for each engagements, as well as the total number of engagements booked for each entertainer.

Question 10 *(2 points)*

Display a list of all of the Entertainers and their members. Number each member within a group.

ACCOUNTS PAYABLE DATABASE (MYSQL)

Question 1 *(5 points)*

Write a query to display the vendor id, balance due, total balance due for all vendors in the Invoices table, and the total balance due for each vendor in the Invoices table. The total balance due for each vendor should contain a cumulative total by balance due. This query should return 11 rows.

Question 2 *(5 points)*

Modify the query to Question 1 above so it includes a column that calculates the average balance due for each vendor in the Invoices table. This column should contain a cumulative average by balance due.

Question 3 *(10 points)*

Write a query to calculate a 3-day moving average of the invoice totals. Display the invoice date, invoice total, and the three-day moving average of the invoice totals sorted by invoice date.

SQL SERVER BONUS QUERY 1 *(20 points)*

Consider a table with two columns. The first, `ID`, is an incrementing integer, the primary key of the table. The second column contains data values. Some data values may be `NULL`. Here is the script that creates the temp table with data values. The script is also available under the Files on Canvas.

```
-- Create the table
CREATE TABLE #TableValues(ID INT, Data INT);

-- Populate the table
INSERT INTO #TableValues(ID, Data)
VALUES(1,100),(2,100),(3,NULL),
(4,NULL),(5,600),(6,NULL),
(7,500),(8,1000),(9,1300),
(10,1200),(11,NULL);

-- Display the results
SELECT * FROM #TableValues;
```

The exercise is to use window functions to replace each `NULL` value with the previous non-`NULL` value.

SQL SERVER BONUS QUERY 2 *(20 points)*

The script below generates a table containing data for customer subscriptions to an online magazine. The data are randomly generated so everyone will have different data in the temporary table. Use this data and window functions to calculate the number of active subscriptions at the end of each month. There may be cancellation dates past the latest subscription date, and you can ignore those rows. The script to generate this table is also available under Files on Canvas.

```
-- Create the temp table
CREATE TABLE #Registrations(ID INT NOT NULL IDENTITY PRIMARY KEY,
DateJoined DATE NOT NULL, DateLeft DATE NULL);

-- Variables
DECLARE @Rows INT = 10000,
@Years INT = 5,
@StartDate DATE = '2011-01-01'

-- Insert 10,000 rows with five years of possible dates
INSERT INTO #Registrations (DateJoined)
SELECT TOP(@Rows) DATEADD(DAY,CAST(RAND(CHECKSUM(NEWID())) * @Years *
365 as INT) ,@StartDate)
FROM sys.objects a
```

```
CROSS JOIN sys.objects b
CROSS JOIN sys.objects c;

-- Give cancellation dates to 75% of the subscribers
UPDATE TOP(75) PERCENT #Registrations
SET DateLeft = DATEADD(DAY,CAST(RAND(CHECKSUM(NEWID())) * @Years * 365
as INT),DateJoined)
```

Your final result set should include four columns: `MonthEnding` containing the end date of the month, `Number Subscribed`, `Number Unsubscribed`, and `Active Subscriptions`.

SQL SERVER BONUS QUERY 3 *(20 points)*

In this exercise you have a table showing clock in and clock out times for employees. The script below generates a temporary table containing the clock in and clock out data for each employee. The script to generate this table is also available under Files on Canvas.

```
-- Create the table
DROP TABLE IF EXISTS #TimeCards;

CREATE TABLE #TimeCards(
TimeStampID INT NOT NULL IDENTITY PRIMARY KEY,
EmployeeID INT NOT NULL,
ClockDateTime DATETIME2(0) NOT NULL,
EventType VARCHAR(5) NOT NULL);

-- Populate the table
INSERT INTO #TimeCards(EmployeeID,
ClockDateTime, EventType)
VALUES
(1,'2021-01-02 08:00','ENTER'),
(2,'2021-01-02 08:03','ENTER'),
(2,'2021-01-02 12:00','EXIT'),
(2,'2021-01-02 12:34','ENTER'),
(3,'2021-01-02 16:30','ENTER'),
(2,'2021-01-02 16:00','EXIT'),
(1,'2021-01-02 16:07','EXIT'),
(3,'2021-01-03 01:00','EXIT'),
(2,'2021-01-03 08:10','ENTER'),
(1,'2021-01-03 08:15','ENTER'),
(2,'2021-01-03 12:17','EXIT'),
(3,'2021-01-03 16:00','ENTER'),
(1,'2021-01-03 15:59','EXIT'),
(3,'2021-01-04 01:00','EXIT');
```

Write a query which uses window functions to calculate how long each employee worked on each shift. The time worked on each shift in your result set should include hours:minutes:seconds. For example, if an employee worked for 7 hours 15 minutes, then it should display `07:15:00`. The final result set should contain three columns: `EmployeeID`, `WorkDate`, and `TimeWorked`.

SQL SERVER BONUS QUERY 4 *(20 points)*

Take a look at the `FolderHierarchy` table. Each folder has an ID, a name, and an ID of a parent folder, which is the folder from which we can access the given folder.

Show four columns: `ID`, `Name`, `ParentId`, and `Path`. The last column should contain the path to the folder, starting with '/' and followed by all folder names separated by '/'. At the end of the path, there should be the name of the given folder and a slash ('/'). An example for folder B, which is in the root folder A, would be: /A/B/.

The script below generates a temporary table containing the folder hierarchy details. The script to generate this table is also available under Files on Canvas.

```
-- Create the table
DROP TABLE IF EXISTS #FolderHierarchy;
GO

-- Create the table
CREATE TABLE #FolderHierarchy
(
ID INTEGER PRIMARY KEY,
Name VARCHAR(100),
ParentID INTEGER
);
GO

-- Populate the table
INSERT INTO #FolderHierarchy VALUES
(1, 'my_folder', NULL),
(2,'my_documents', 1),
(3, 'events', 2),
(4, 'meetings', 3),
(5, 'conferences', 3),
(6, 'travel', 3),
(7, 'integration', 3),
(8, 'out_of_town', 4),
(9, 'abroad', 8),
(10, 'in_town', 4);
GO
```

SQL SERVER BONUS QUERY 5 *(20 points)*

The table `Destination` contains the names of four cities. You want to travel among the four cities starting from `Warsaw`. We have another table named `Ticket` which lists all possible flight connections and the cost.

Your task is to find the path which will be **cheapest** in terms of the **total tickets' cost**. List all paths starting from `Warsaw` that go through all four cities. Order the paths in **descending** order by `TotalCost`.

Your result set should contain the following columns: `Path` which contains city names separated by `->`, `LastId` which contains the ID of the last city, `TotalCost` which contains the total cost of the tickets, `NumPlacesVisited` which contains the number of places visited; it should equal **4**.

The script below generates the temporary tables. This script is also available under Files on Canvas.

```
DROP TABLE IF EXISTS #Destination;
GO

-- Create the table
CREATE TABLE #Destination
(
ID INTEGER PRIMARY KEY,
Name VARCHAR(100)
);
GO

-- Populate the table
INSERT INTO #Destination VALUES
(1, 'Warsaw'),
(2,'Berlin'),
(3, 'Bucharest'),
(4, 'Prague');
GO

DROP TABLE IF EXISTS #Ticket;
GO

-- Create the table
CREATE TABLE #Ticket
(
CityFrom INTEGER,
CityTo INTEGER,
Cost INTEGER
```

```
);
GO

-- Populate the table
INSERT INTO #Ticket VALUES
(1, 2, 350),
(1, 3, 80),
(1, 4, 220),
(2, 3, 410),
(2, 4, 230),
(3, 2, 160),
(3, 4, 110),
(4, 2, 140),
(4, 3, 75);
GO
```