

Task: CSE 581 Spring 2018 Project 1

Deadline: Monday, April 9 at 12.00 PM Eastern Time

=====

Student Name : Sonal Patil

SUID : 997435672

=====

I. My Guitar Shop Database

In part I, you will use SQL Server Management Studio to create the MyGuitarShop database, to review the tables in the MyGuitarShop database, and to enter SQL statements and run them against this database.

A. Database Setup [2 pts.]

1. Download CreateMyGuitarShop.sql from Blackboard Project 1 directory and open it in SQL server management studio. Execute the entire script by clicking the Execute button in the SQL Editor toolbar or by pressing F5. Show the message in the Message tab, indicating the script is executed successfully. A full screenshot of execution result is required.

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "CreateMyGuitarShop.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (52)) - Microsoft SQL Server Management Studio". The left pane is the Object Explorer, showing the database "MyGuitarShop" under "DESKTOP-69JNVD4\SQLEXPRESS (SQL Server 14.0.1000)". The center pane contains the SQL script "CreateMyGuitarShop.sql" with the following content:

```
-- This script creates the database named my_guitar_shop
USE master;
GO

IF DB_ID('MyGuitarShop') IS NOT NULL
    DROP DATABASE MyGuitarShop;
GO

CREATE DATABASE MyGuitarShop;
GO

USE MyGuitarShop;

-- create the tables for the database
CREATE TABLE Categories (
    CategoryID      INT             PRIMARY KEY     IDENTITY,
    CategoryName    VARCHAR(255)    NOT NULL        UNIQUE
)
```

The "Messages" pane shows the execution results:

```
(4 rows affected)
(10 rows affected)
(485 rows affected)
(512 rows affected)
(41 rows affected)
(47 rows affected)
(3 rows affected)
```

The status bar at the bottom indicates "Query executed successfully." and "0 rows". The Properties pane on the right shows the "Aggregate Status" and "Connection" details.

Navigate through the database objects and view the column definitions for each table. Open a new Query Editor window. Show details in Administrators table and Orders table using SELECT statement. A full screenshot of execution result is required.

Project1_AQ1.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (55)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

Object Explorer

Connect Connect to... New Query Execute Debug

Properties

Aggregate Status

Connection failure:

- Elapsed time: 00:00:02.73
- Finish time: 17-Mar-18 11:39:36 AM
- Name: DESKTOP-69JNVD4\SQL
- Rows returned: 44
- Start time: 17-Mar-18 11:39:35 AM
- State: Open

Connection

Connection name: DESKTOP-69JNVD4\SQL

Connection Details

- Connection elapse: 00:00:02.73
- Connection encrypt: Not encrypted
- Connection finish t: 17-Mar-18 11:39:36 AM
- Connection rows n: 44
- Connection start t: 17-Mar-18 11:39:35 AM
- Connection state: Open
- Display name: DESKTOP-69JNVD4\SQL
- Login name: DESKTOP-69JNVD4\son
- Server name: DESKTOP-69JNVD4\SQL
- Server version: 14.0.1000
- Session Tracing ID: SPID: 55

Name

The name of the connection.

Results Messages

```
USE MyGuitarShop;
SELECT * FROM Administrators;
SELECT * FROM Orders;
```

	AdminID	EmailAddress	Password	FirstName	LastName
1	1	admin@myguitarshop.com	Gz710bd768c237b51f8249b54897940e9022	Admin	User
2	2	joel@murach.com	97195957d5b74d70d79c20949cd91b697aa	Joel	Murach
3	3	mike@murach.com	32975c819cf686282456aee3a137bf596e8	Mike	Murach

	OrderID	CustomerID	OrderDate	ShipAmount	TaxAmount	ShipDate	ShipAddressID	CardType	CardNumber	C
1	1	1	2016-03-28 09:40:28.000	5.00	58.75	2016-03-31 09:41:11.000	1	Visa	4111111111111111	04
2	2	2	2016-03-28 11:23:20.000	5.00	21.27	2016-03-31 11:24:03.000	3	Visa	4012888888881881	08
3	3	1	2016-03-29 09:44:58.000	10.00	102.29	2016-04-01 09:45:41.000	1	Visa	4111111111111111	06
4	4	3	2016-03-30 15:22:31.000	10.00	117.50	2016-04-02 15:23:14.000	4	American Express	3782822463100005	02
5	5	4	2016-03-31 05:43:11.000	5.00	20.93	2016-04-03 05:43:54.000	5	Visa	4111111111111111	08
6	6	5	2016-03-31 18:37:22.000	5.00	20.93	2016-04-03 18:38:05.000	7	Discover	6011111111111117	04
7	7	6	2016-04-01 23:11:12.000	15.00	107.80	2016-04-04 23:11:55.000	8	MasterCard	5555555555554444	12
8	8	7	2016-04-02 11:26:38.000	5.00	47.60	2016-04-05 11:27:21.000	9	Visa	4012888888881881	04

Query executed successfully.

DESKTOP-69JNVD4\SQLEXPRESS ... DESKTOP-69JNVD4\sonalp... MyGuitarShop 00:00:00 | 44 rows

Ln 8 Col 22 Ch 22 INS

B. An introduction to SQL [6 pts.]

- Write a SELECT statement that returns one column from the Customers table named FullName that joins the LastName and FirstName columns.

Format this column with the last name, a comma, a space, and the first name like this:

Doe, John

Add an ORDER BY clause to this statement that sorts the result set by last name in descending sequence. Return only the contacts whose last name begins with a letter from A to E.

Project1_BQ1.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (54)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

Object Explorer

Connect Connect to... New Query Execute Debug

Properties

Aggregate Status

Connection failure:

- Elapsed time: 00:00:00.139
- Finish time: 17-Mar-18 12:07:45 PM
- Name: DESKTOP-69JNVD4\SQL
- Rows returned: 122
- Start time: 17-Mar-18 12:07:45 PM
- State: Open

Connection

Connection name: DESKTOP-69JNVD4\SQL

Connection Details

- Connection elapse: 00:00:00.139
- Connection encrypt: Not encrypted
- Connection finish t: 17-Mar-18 12:07:45 PM
- Connection rows n: 122
- Connection start t: 17-Mar-18 12:07:45 PM
- Connection state: Open
- Display name: DESKTOP-69JNVD4\SQL
- Login name: DESKTOP-69JNVD4\son
- Server name: DESKTOP-69JNVD4\SQL
- Server version: 14.0.1000
- Session Tracing ID: SPID: 54

Name

The name of the connection.

Results Messages

```
/*Project 1 Sonal Patil B.
1. Write a SELECT statement that returns one column from the Customers table named FullName
that joins the LastName and FirstName columns.
Format this column with the last name, a comma, a space, and the first name like this:
Doe, John
Add an ORDER BY clause to this statement that sorts the result set by last name in descending sequence.
Return only the contacts whose last name begins with a letter from A to E.*/

USE MyGuitarShop;

SELECT LastName + ', ' + FirstName AS 'FullName'
FROM Customers
WHERE LastName LIKE '[A-E]%'
ORDER BY LastName DESC;
```

FullName
Esway, Heather
Estell, Roselle
Eschberger, Christiane
Eroman, Irene
Ernaco, Teri
Engelberg, Johanna
Emigh, Stephen
Emard, Franklyn
Ermann, Kristel
Eanes, Sharla
Duemas, Kimberlie
Dubald, Pete
Dymon, Jennie
Dwoskin, Marcell

Comments - here i have combines two columns and gave that column a new name
also used like to where clause to meet the condition and order by clause to give proper order to result set*/

Query executed successfully.

DESKTOP-69JNVD4\SQLEXPRESS ... DESKTOP-69JNVD4\sonalp... MyGuitarShop 00:00:00 | 122 rows

Ln 15 Col 25 Ch 25 INS

2. Write a SELECT statement that returns these column names and data from the Products table:

ProductName
ListPrice
DiscountPercent
DiscountAmount
DiscountPrice

The ProductName column
The ListPrice column
The DiscountPercent column
A column that's calculated from the previous two columns
A column that's calculated from the previous three columns

Sort the result set by discount price in descending sequence.

Project1_BQ2.sql - DESKTOP-69JNVD4\SQLExpress\MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (56)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

Object Explorer

Project1_BQ2.sql - DESKTOP-69JNVD4\sonalpatil (56) -> Project1_BQ1.sql -> VD4\sonalpatil (54) -> Project1_AQ1.sql -> VD4\sonalpatil (55)

2. Write a SELECT statement that returns these column names and data from the Products table:

```

    ProductName      The ProductName column
    ListPrice        The ListPrice column
    DiscountPercent The DiscountPercent column
    DiscountAmount   A column that's calculated from the previous two columns
    DiscountPrice    A column that's calculated from the previous three columns
    Sort the result set by discount price in descending sequence. */

USE MyGuitarShop;

SELECT ProductName, ListPrice, DiscountPercent,
       (ListPrice * DiscountPercent) / 100 AS 'DiscountAmount',
       ListPrice - ((ListPrice * DiscountPercent) / 100) AS 'DiscountPrice'
FROM Products
ORDER BY DiscountPrice DESC;
    
```

/*Comments - here i have performed mathematical calculation on 2/3 columns and gave computed result column a new name also used order by clause to give proper order to result set*/

Results

	ProductName	ListPrice	DiscountPercent	DiscountAmount	DiscountPrice
1	Gibson SG	2517.00	52.00	1308.84	1208.16
2	Gibson Les Paul	1199.00	30.00	359.70	839.30
3	Tama 5-Piece Drum Set with Cymbals	799.99	15.00	119.9985	679.9915
4	Fender Precision	799.99	30.00	239.99	559.993
5	Ludwig 5-piece Drum Set with Cymbals	699.99	30.00	209.997	489.993
6	Fender Stratocaster	699.00	30.00	209.70	489.30
7	Hofner Icon	499.99	25.00	124.9975	374.9925
8	Yamaha FG700S	489.99	38.00	186.1962	303.7938
9	Washburn D10S	299.00	0.00	0.00	299.00
10	Rodríguez Caballero 11	415.00	39.00	161.85	253.15

Query executed successfully.

Properties

Current connection parameters

Aggregate Status

Connection failure:

- Elapsed time: 00:00:00.071
- Finish time: 17-Mar-18 12:16:06 PM
- Name: DESKTOP-69JNVD4\SQL
- Rows returned: 10
- Start time: 17-Mar-18 12:16:06 PM
- State: Open

Connection

Connection name: DESKTOP-69JNVD4\SQL

Connection Details

- Connection elapse: 00:00:00.071
- Connection encryp: Not encrypted
- Connection finish t: 17-Mar-18 12:16:06 PM
- Connection rows n: 10
- Connection start ti: 17-Mar-18 12:16:06 PM
- Connection state: Open
- Display name: DESKTOP-69JNVD4\SQL
- Login name: DESKTOP-69JNVD4\son
- Server name: DESKTOP-69JNVD4\SQL
- Server version: 14.0.1000
- Session Tracing ID: SPID: 56

Name

The name of the connection.

C. The essential SQL skills [44 pts.]

1. Write a SELECT statement that returns the ProductName and ListPrice columns from the Products table. Return one row for each product that has the same list price as another product. Sort the result set by ProductName from A to Z.

(Hint: Use a self-join to check that the ProductID columns aren't equal but ListPrice column is equal.)

Project1_CQ1.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (57)) - Microsoft SQL Server Management Studio

```

USE MyGuitarShop;
SELECT P1.ProductName, P1.ListPrice
FROM Products AS P1 JOIN Products AS P2 ON P1.ProductID <> P2.ProductID AND P1.ListPrice = P2.ListPrice
ORDER BY P1.ProductName ASC;

/*Comments - here i have used self join to meet the condition of having same list price
also used order by clause to give the proper order to result set*/

```

Results

ProductName	ListPrice
Fender Precision	799.99
Tama 5-Piece Drum Set with Cymbals	799.99

Query executed successfully.

Properties

Aggregate Status

Connection failure:

- Elapsed time: 00:00:00.118
- Finish time: 17-Mar-18 12:31:50 PM
- Name: DESKTOP-69JNVD4\SQL
- Rows returned: 2
- Start time: 17-Mar-18 12:31:50 PM
- State: Open

Connection

Connection name: DESKTOP-69JNVD4\SQL

Connection Details

- Connection elapse: 00:00:00.118
- Connection encrypt: Not encrypted
- Connection finish t: 17-Mar-18 12:31:50 PM
- Connection rows n: 2
- Connection start t: 17-Mar-18 12:31:50 PM
- Connection state: Open
- Display name: DESKTOP-69JNVD4\SQL
- Login name: DESKTOP-69JNVD4\son
- Server name: DESKTOP-69JNVD4\SQL
- Server version: 14.0.1000
- Session Tracing ID: 57
- SPID: 57

Name

The name of the connection.

2. Use the UNION operator to generate a result set consisting of three columns from the Orders table:
ShipStatus A calculated column that contains a value of SHIPPED or NOT SHIPPED

OrderID The OrderID column

OrderDate The OrderDate column

If the order has a value in the ShipDate column, the ShipStatus column should contain a value of SHIPPED. Otherwise, it should contain a value of NOT SHIPPED. Sort the final result set by OrderDate from earliest to most recent.

Project1_CQ2.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (52)) - Microsoft SQL Server Management Studio

```

2. Use the UNION operator to generate a result set consisting of three columns from the Orders table:
    ShipStatus A calculated column that contains a value of SHIPPED or NOT SHIPPED
    OrderID The OrderID column
    OrderDate The OrderDate column

    If the order has a value in the ShipDate column, the ShipStatus column should contain a value of SHIPPED.
    Otherwise, it should contain a value of NOT SHIPPED. Sort final result set by OrderDate from earliest to most recent

USE MyGuitarShop;

SELECT 'SHIPPED' AS ShipStatus, OrderID, OrderDate
FROM Orders
WHERE ShipDate IS NOT NULL

UNION

SELECT 'NOT SHIPPED' AS ShipStatus, OrderID, OrderDate
FROM Orders
WHERE ShipDate IS NULL

ORDER BY OrderDate ASC;

```

Results

ShipStatus	OrderID	OrderDate
SHIPPED	26	2016-04-20 08:14:45.000
SHIPPED	25	2016-04-20 08:23:32.000
SHIPPED	27	2016-04-20 09:17:52.000
SHIPPED	28	2016-04-21 17:52:24.000
SHIPPED	29	2016-04-25 23:36:41.000
SHIPPED	30	2016-04-27 16:21:31.000
SHIPPED	31	2016-04-29 06:47:14.000
NOT SHIPPED	32	2016-05-01 01:23:23.000
SHIPPED	33	2016-05-01 09:11:51.000
SHIPPED	34	2016-05-02 11:36:12.000

Query executed successfully.

Properties

Aggregate Status

Connection failure:

- Elapsed time: 00:00:00.090
- Finish time: 18-Mar-18 10:58:56 AM
- Name: DESKTOP-69JNVD4\SQL
- Rows returned: 41
- Start time: 18-Mar-18 10:58:56 AM
- State: Open

Connection

Connection name: DESKTOP-69JNVD4\SQL

Connection Details

- Connection elapse: 00:00:00.090
- Connection encrypt: Not encrypted
- Connection finish t: 18-Mar-18 10:58:56 AM
- Connection rows n: 41
- Connection start t: 18-Mar-18 10:58:56 AM
- Connection state: Open
- Display name: DESKTOP-69JNVD4\SQL
- Login name: DESKTOP-69JNVD4\son
- Server name: DESKTOP-69JNVD4\SQL
- Server version: 14.0.1000
- Session Tracing ID: 52
- SPID: 52

Name

The name of the connection.

3. Write a SELECT statement that returns 1 row for each category that has products with these columns:
- The CategoryName column from the Categories table
 - The count of the products in the Products table
 - The list price of the most expensive product in the Products table
- Sort the result set so the category with the least products appears first.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window with the following content:

```

/*Project 1 Sonal Patil C.
3. Write a SELECT statement that returns 1 row for each category that has products with these columns:
a. The CategoryName column from the Categories table
b. The count of the products in the Products table
c. The list price of the most expensive product in the Products table
Sort the result set so the category with the least products appears first. */

USE MyGuitarShop;

SELECT C.CategoryName, COUNT(*) AS ProductCount, MAX(P.ListPrice) AS MostExpensiveProduct
FROM Categories AS C JOIN Products AS P ON C.CategoryID = P.CategoryID
GROUP BY C.CategoryName
ORDER BY ProductCount ASC;

/*Comments - i have used aggregate functions like count and max to get total count and max value and group
the result for these aggregate function. Also used join as we want 1 row for each category
ordered the final result set using order by clause*/

```

The results pane shows the following data:

CategoryName	ProductCount	MostExpensiveProduct
Basses	2	799.99
Drums	2	799.99
Guitars	6	2517.00

At the bottom of the screen, the taskbar shows various application icons, and the system tray indicates the date and time as 11:05 AM on 25-Mar-18.

4. Write a SELECT statement that returns 1 row for each customer that has orders with these columns:
- The EmailAddress column from the Customers table
 - A count of the number of orders
 - The total amount for each order (Hint: First, subtract the discount amount from the price. Then, multiply by the quantity)

Return only those rows where the items have a more than 400 ItemPrice value.
Sort the result set in ascending sequence by the sum of the line item amounts.

Project1_CQ4.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (61)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Execute Debug

Object Explorer

Connect Connect to Database

MyGuitarShop

Project1_CQ4.sql - DESKTOP-69JNVD4\sonalpatil (54)*

```
/*Project 1 Sonal Patil C.
4. Write a SELECT statement that returns 1 row for each customer that has orders with these columns:
a. The EmailAddress column from the Customers table
b. A count of the number of orders
c. The total amount for each order (Hint: First, subtract the discount amount from the price.
Then, multiply by the quantity)
Return only those rows where the items have a more than 400 ItemPrice value.
Sort the result set in ascending sequence by the sum of the line item amounts. */

USE MyGuitarShop;

SELECT c.EmailAddress, COUNT(o.OrderID) AS 'NumberofOrders',
SUM((oi.ItemPrice - oi.DiscountAmount) * oi.Quantity) AS 'total'
FROM Customers AS c JOIN Orders AS o ON c.CustomerID = o.CustomerID
JOIN OrderItems AS oi ON o.OrderID = oi.OrderID
WHERE oi.ItemPrice > 400
GROUP BY c.EmailAddress
ORDER BY total;

/*Comments - here i have used aggregate functions COUNT & SUM to get count and total sum value with grouping
the result for each customer. Also used join to join 3 tables as we need to access columns from
all these 3 tables. Finally, sorted the result set in given specified order & condition*/
```

Results Messages

EmailAddress	NumberofOrders	total
baryz@gmail.com	1	303.79
willard@hotmail.com	1	303.79
gladys.m@rim.org	1	374.99
allene_itubide@cox.net	1	489.30
amaclead@gmail.com	1	489.30
calabares@gmail.com	1	489.30
mitsue_joller@yahoo.com	1	489.30
jbutt@gmail.com	1	489.30
kiley.caldadera@aol.com	1	489.30
leota@hotmail.com	1	489.30

Query executed successfully.

DESKTOP-69JNVD4\SQLEXPRESS ... DESKTOP-69JNVD4\sonalp... MyGuitarShop 00:00:00 | 33 rows

Properties

Current connection parameters

Aggregate Status

Connection failures: 00:00:00.161

Elapsed time: 00:00:00.161

Finish time: 25-Mar-18 10:57:25 AM

Name: DESKTOP-69JNVD4\SQLEXPRESS

Rows returned: 33

Start time: 25-Mar-18 10:57:25 AM

State: Open

Connection

Connection name: DESKTOP-69JNVD4\SQLEXPRESS

Connection Details

Connection elapsed time: 00:00:00.161

Connection encryption: Not encrypted

Connection finish time: 25-Mar-18 10:57:25 AM

Connection rows returned: 33

Connection start time: 25-Mar-18 10:57:25 AM

Connection state: Open

Display name: DESKTOP-69JNVD4\SQLEXPRESS

Login name: DESKTOP-69JNVD4\sonalpatil

Server name: DESKTOP-69JNVD4\SQLEXPRESS

Server version: 14.0.1000

Session Tracing ID: 61

SPID: 61

Name

The name of the connection.

Item(s) Saved

Ln 9 Col 9 Ch 9 INS

11:06 AM 25-Mar-18

5. Write a SELECT statement that returns the name and discount percent of each product that has a unique discount percent. In other words, don't include products that have the same discount percent as another product.

Sort the results by the ProductName column from A to Z.

Project1_CQ5.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (54)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Execute Debug

Object Explorer

Connect Connect to Database

MyGuitarShop

Project1_CQ5.sql - DESKTOP-69JNVD4\sonalpatil (54)*

```
/*Project 1 Sonal Patil C.
5. Write SELECT statement that returns name and discount percent of each product that has unique discount
percent. In other words, don't include products that have same discount percent as another product.
Sort the results by the ProductName column from A to Z. */

USE MyGuitarShop;

SELECT P1.ProductName, P1.DiscountPercent
FROM Products AS P1
WHERE P1.DiscountPercent NOT IN
(SELECT P2.DiscountPercent
FROM Products AS P2
WHERE P1.ProductName <> P2.ProductName)
ORDER BY ProductName;

/*comments - here i have used a subquery fro search condition which eliminates the duplicate value entries for
discount percent column and finally sorted the result set in specified order*/
```

Results Messages

ProductName	DiscountPercent
Gibson SG	52.00
Hofner Icon	25.00
Rodriguez Caballero 11	39.00
Tama 5-Piece Drum Set with Cymbals	15.00
Washburn D10S	0.00
Yamaha FG700S	38.00

Query executed successfully.

DESKTOP-69JNVD4\SQLEXPRESS ... DESKTOP-69JNVD4\sonalp... MyGuitarShop 00:00:00 | 6 rows

Properties

Current connection parameters

Aggregate Status

Connection failures: 00:00:00.038

Elapsed time: 00:00:00.038

Finish time: 25-Mar-18 11:18:24 AM

Name: DESKTOP-69JNVD4\SQLEXPRESS

Rows returned: 6

Start time: 25-Mar-18 11:18:24 AM

State: Open

Connection

Connection name: DESKTOP-69JNVD4\SQLEXPRESS

Connection Details

Connection elapsed time: 00:00:00.038

Connection encryption: Not encrypted

Connection finish time: 25-Mar-18 11:18:24 AM

Connection rows returned: 6

Connection start time: 25-Mar-18 11:18:24 AM

Connection state: Open

Display name: DESKTOP-69JNVD4\SQLEXPRESS

Login name: DESKTOP-69JNVD4\sonalpatil

Server name: DESKTOP-69JNVD4\SQLEXPRESS

Server version: 14.0.1000

Session Tracing ID: 54

Name

The name of the connection.

Item(s) Saved

Ln 18 Col 90 Ch 81 INS

11:19 AM 25-Mar-18

6. Write a SELECT statement that returns three columns: EmailAddress, OrderID, and the order total for each customer. To do this, you can group the result set by the EmailAddress and OrderID columns. In addition, you must calculate the order total from the columns in the OrderItems table. Write a second SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return two columns: the customer's email address and the largest order for that customer. To do this, you can group the result set by the EmailAddress column.

```

USE MyGuitarShop;

/*1st select statement*/
SELECT c.EmailAddress, o.OrderID, SUM((oi.ItemPrice - oi.DiscountAmount) * oi.Quantity) AS OrderTotal
FROM Customers AS c JOIN Orders AS o ON C.CustomerID = O.CustomerID
JOIN OrderItems AS oi ON O.OrderID = oi.OrderID
GROUP BY c.EmailAddress, o.OrderID;

/*2nd/Main select statement*/
SELECT c.EmailAddress, MAX(OrderTotal) AS LargestOrder
FROM Customers AS C JOIN Orders AS O ON C.CustomerID = O.CustomerID
JOIN (SELECT c.EmailAddress, o.OrderID, SUM((oi.ItemPrice - oi.DiscountAmount) * oi.Quantity) AS OrderTotal
      FROM Customers AS c JOIN Orders AS o ON C.CustomerID = O.CustomerID
      JOIN OrderItems AS oi ON O.OrderID = oi.OrderID
      GROUP BY c.EmailAddress, o.OrderID) AS largest ON largest.OrderID = O.OrderID
GROUP BY C.EmailAddress;

```

Comments - here i have used subquery as 3rd result set for join. Main query join uses 2 joins one of which is a result set generated using a subquery. Grouped the final main result set using customer email address*/

EmailAddress	OrderID	OrderTotal
allan.sherwood@yahoo.com	1	839.30
barny@gmail.com	2	303.79
allan.sherwood@yahoo.com	3	1461.31
christineb@solarone.com	4	1678.60
david.goldstein@hotmail.com	5	299.00
emrv@gmail.com	6	299.00
frankwillson@boglobal.net	7	1539.97
gary_hernandez@yahoo.com	8	679.99
dean.madison@boglobal.net	9	1461.31

```

USE MyGuitarShop;

/*1st select statement*/
SELECT c.EmailAddress, o.OrderID, SUM((oi.ItemPrice - oi.DiscountAmount) * oi.Quantity) AS OrderTotal
FROM Customers AS c JOIN Orders AS o ON C.CustomerID = O.CustomerID
JOIN OrderItems AS oi ON O.OrderID = oi.OrderID
GROUP BY c.EmailAddress, o.OrderID;

/*2nd/Main select statement*/
SELECT c.EmailAddress, MAX(OrderTotal) AS LargestOrder
FROM Customers AS C JOIN Orders AS O ON C.CustomerID = O.CustomerID
JOIN (SELECT c.EmailAddress, o.OrderID, SUM((oi.ItemPrice - oi.DiscountAmount) * oi.Quantity) AS OrderTotal
      FROM Customers AS c JOIN Orders AS o ON C.CustomerID = O.CustomerID
      JOIN OrderItems AS oi ON O.OrderID = oi.OrderID
      GROUP BY c.EmailAddress, o.OrderID) AS largest ON largest.OrderID = O.OrderID
GROUP BY C.EmailAddress;

```

Comments - here i have used subquery as 3rd result set for join. Main query join uses 2 joins one of which is a result set generated using a subquery. Grouped the final main result set using customer email address*/

emailaddress	LargestOrder
alisha@lusinski.com	1678.60
allan.sherwood@yahoo.com	1461.31
allene_tubide@cox.net	489.30
amadeal@gmail.com	489.30
art@venere.org	679.99
barny@gmail.com	303.79
bette_nicka@cox.net	839.30
calbares@gmail.com	489.30
christineb@solarone.com	782.08

7. Write a SELECT statement that returns these columns from the Orders table:

- CardNumber column
- length of CardNumber column
- last 4 digits of CardNumber column
- Column that displays last 4 digits of CardNumber column in format: XXXX-XXXX-XXXX-1234. In other words, use 'X's for 1st 12 digits of CardNumber & actual numbers for last 4 digits of number.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'MyGuitarShop'. The central pane contains a query window with the following code:

```

/*
Project 1 Sonal Patil C.
7. Write a SELECT statement that returns these columns from the Orders table:
    a. The CardNumber column
    b. The length of the CardNumber column
    c. The last four digits of the CardNumber column
    d. A column that displays the last four digits of the CardNumber column in this format: XXXX-XXXX-XXXX-1234.
        In other words, use 'X's for first 12 digits of card number & actual numbers for last four digits of number. */

USE MyGuitarShop;

SELECT CardNumber, LEN(CardNumber) AS 'CardLength', RIGHT(CardNumber,4) AS 'Last4Digits',
       CONCAT('XXXX-XXXX-XXXX',RIGHT(CardNumber,4)) AS 'FormatedCardNumber'
FROM Orders;

/* comments - here i have used few string functions, LEN to get length, RIGHT to get those many chars from right side of string
and finally CONCAT to concatenate given string with existing part of string which gives us new format*/

```

The Results pane shows the output of the query, displaying 14 rows of card information. The Properties pane on the right shows connection details.

CardNumber	CardLength	Last4Digits	FormatedCardNumber
41111111111111	16	1111	XXXX-XXXX-XXXX-1111
4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
41111111111111	16	1111	XXXX-XXXX-XXXX-1111
378282463100005	16	0005	XXXX-XXXX-XXXX-0005
41111111111111	16	1111	XXXX-XXXX-XXXX-1111
60111111111117	16	1117	XXXX-XXXX-XXXX-1117
55555555555444	16	4444	XXXX-XXXX-XXXX-4444
4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
41111111111111	16	1111	XXXX-XXXX-XXXX-1111
41111111111111	16	1111	XXXX-XXXX-XXXX-1111
4012888888881881	16	1881	XXXX-XXXX-XXXX-1881
41111111111111	16	1111	XXXX-XXXX-XXXX-1111
41111111111111	16	1111	XXXX-XXXX-XXXX-1111
41111111111111	16	1111	XXXX-XXXX-XXXX-1111

8. Write a SELECT statement that returns these columns from Orders table (Hint: use system functions):

- OrderDate column
- A column that returns four-digit year that's stored in OrderDate column
- A column that returns only the day of the month that's stored in the OrderDate column.
- A column that returns the result from adding twenty days to the OrderDate column

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'MyGuitarShop'. The central pane contains a query window with the following code:

```

/*
Project 1 Sonal Patil C.
8. Write a SELECT statement that returns these columns from Orders table (Hint: use system functions):
    a. The OrderDate column
    b. A column that returns the four-digit year that's stored in the OrderDate column
    c. A column that returns only the day of the month that's stored in the OrderDate column.
    d. A column that returns the result from adding twenty days to the OrderDate column

*/

USE MyGuitarShop;

SELECT OrderDate, YEAR(OrderDate) AS 'Year', DAY(OrderDate) AS 'Day', DATEADD(DAY,20,OrderDate) AS 'Plus20Days'
FROM Orders;

/* comments - here i have used datetime system functions to manipulate the data stored in the orders tables' orderdate column
YEAR, DAY functions gives us the particular year or day stored in the given date
DATEADD function adds the given amount of days in the given data and returns new date*/

```

The Results pane shows the output of the query, displaying 14 rows of order date information. The Properties pane on the right shows connection details.

OrderDate	Year	Day	Plus20Days
2016-03-28 09:40:28.000	2016	28	2016-04-17 09:40:28.000
2016-03-28 11:23:20.000	2016	28	2016-04-17 11:23:20.000
2016-03-29 09:44:58.000	2016	29	2016-04-18 09:44:58.000
2016-03-30 15:22:31.000	2016	30	2016-04-19 15:22:31.000
2016-03-31 05:43:11.000	2016	31	2016-04-20 05:43:11.000
2016-03-31 18:37:22.000	2016	31	2016-04-20 18:37:22.000
2016-04-01 23:11:12.000	2016	1	2016-04-21 23:11:12.000
2016-04-02 11:26:38.000	2016	2	2016-04-22 11:26:38.000
2016-04-03 12:22:31.000	2016	3	2016-04-23 12:22:31.000
2016-04-03 14:59:20.000	2016	3	2016-04-23 14:59:20.000
2016-04-04 06:24:44.000	2016	4	2016-04-24 06:24:44.000
2016-04-04 08:15:12.000	2016	4	2016-04-24 08:15:12.000
2016-04-04 11:20:31.000	2016	4	2016-04-24 11:20:31.000
2016-04-05 09:24:53.000	2016	5	2016-04-25 09:24:53.000

For question 9-11: To test whether a table has been modified correctly as you do these questions, please write and run an appropriate SELECT statement.

Run this statement first:

```
INSERT INTO Categories (CategoryName)
VALUES ('Brass');
```

9. Write an UPDATE statement that modifies the row you just added to the Categories table. This statement should change the CategoryName column to "Woodwinds", and it should use the CategoryID column to identify the row. Use SELECT statements to verify your result (data changes before and after update).

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, under the 'MyGuitarShop' database, there are several tables listed: FileTables, External Tables, Graph Tables, dbo.Addresses, dboAdministrators, dboCategories, dboCustomers, dboOrderItems, dboOrders, and Columns. The 'Columns' node is expanded, showing detailed information about columns like OrderID, CustomerID, OrderDate, ShipAmount, TaxAmount, ShipDate, ShipAddressID, CardType, CardNumber, CardExpires, and BillingAddressID. The 'Results' tab of the query editor displays two tables. The first table, 'Categories', has five rows with CategoryID 1 through 5 and CategoryName 'Basses', 'Brass', 'Drums', 'Guitars', and 'Keyboards'. The second table, also named 'Categories', has five rows with CategoryID 1 through 5 and CategoryName 'Basses', 'Drums', 'Guitars', 'Keyboards', and 'Woodwinds'. The status bar at the bottom indicates 'Query executed successfully.' and '10 rows'.

```
/*Project 1 Sonal Patil C.
For question 9-11: To test whether table has been modified correctly as you do these questions, please write & run
appropriate SELECT statement. Run this statement first:
INSERT INTO Categories (CategoryName)
VALUES ('Brass');

9. Write an UPDATE statement that modifies the row you just added to the Categories table.
This statement should change CategoryName column to "Woodwinds", and it should use CategoryID column to identify row.
Use SELECT statements to verify your result (data changes before and after update). */

USE MyGuitarShop;

INSERT INTO Categories (CategoryName)
VALUES ('Brass');

SELECT * FROM Categories;

UPDATE Categories
SET CategoryName = 'Woodwinds'
WHERE CategoryID = 7;

SELECT * FROM Categories;

/*Comments - i have used insert to insert new row in category table, update to update row for given id using where clause*/
```

10. Write an INSERT statement that adds this row to the Products table:

ProductID: The next automatically generated ID

CategoryID: 4

ProductCode: dgx_640

ProductName: Yamaha DGX 640 88-Key Digital Piano

Description: Long description to come.

ListPrice: 799.99

DiscountPercent: 0

DateAdded: Today's date/time.

Use a column list for this statement. Use SELECT statements to verify your result (data changes before and after insertion).

Project1_CQ10.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (65)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Execute Debug

MyGuitarShop

Object Explorer

Project1_CQ10.sql - D4\sonalpatil (65) Project1_CQ9.sql - VD4\sonalpatil (64) Project1_CQ8.sql - VD4\sonalpatil (63)

10. Write an INSERT statement that adds this row to the Products table:

```

    ProductID: The next automatically generated ID
    CategoryID: 4
    ProductCode: dgx_640
    ProductName: Yamaha DGX 640 88-Key Digital Piano
    Description: Long description to come.
    ListPrice: 799.99
    DiscountPercent: 0
    DateAdded: Today's date/time.

    Use column list for this statement. Use SELECT statements to verify your result (data changes before & after insertion). */

USE MyGuitarShop;

INSERT INTO Products (CategoryID, ProductCode, ProductName, Description, ListPrice, DiscountPercent, DateAdded)
VALUES (4, 'dgx_640', 'Yamaha DGX 640 88-Key Digital Piano', 'Long description to come.', 799.99, 0, GETDATE());

SELECT * FROM Products;
    /* comments - here i have used insert statement to insert a new row in Products table with given values for given columns*/

```

Results

ProductID	CategoryID	ProductCode	ProductName	Description	ListPrice	DiscountPercent	DateAdded
1	1	strat	Fender Stratocaster	The Fender Stratocaster is the electric guitar design...	699.00	30.00	2015-10-30 09:32:40.000
2	2	les_paul	Gibson Les Paul	This Les Paul guitar offers a carved top and humbu...	1199.00	30.00	2015-12-05 16:33:13.000
3	3	sg	Gibson SG	This Gibson SG electric guitar takes the best of the...	2517.00	52.00	2016-02-04 11:04:31.000
4	4	fg700s	Yamaha FG700S	The Yamaha FG700S solid top acoustic guitar has ...	489.99	38.00	2016-06-01 11:12:59.000
5	5	washburn	Washburn D10S	The Washburn D10S acoustic guitar is superbly cr...	299.00	0.00	2016-07-30 13:58:35.000
6	6	rodriguez	Rodriguez Caballero 11	Featuring a carefully chosen, solid Canadian cedar ...	415.00	39.00	2016-07-30 14:12:41.000
7	7	precision	Fender Precision	The Fender Precision bass guitar delivers the sound...	799.99	30.00	2016-06-01 11:29:35.000
8	8	hofner	Hofner Icon	With authentic details inspired by the original, the H...	499.99	25.00	2016-07-30 14:18:33.000
9	9	ludwig	Ludwig 5-Piece Drum Set with Cymbals	This product includes a Ludwig 5-piece drum set a...	699.99	30.00	2016-07-30 12:46:40.000
10	10	tama	Tama 5-Piece Drum Set with Cymbals	The Tama 5-piece Drum Set is the most affordable ...	799.99	15.00	2016-07-30 13:14:15.000
11	11	dgx_640	Yamaha DGX 640 88-Key Digital Piano	Long description to come.	799.99	0.00	2018-03-25 12:32:19.370

Query executed successfully.

DESKTOP-69JNVD4\SQLEXPRESS... DESKTOP-69JNVD4\sonalp... MyGuitarShop 00:00:00 11 rows

Ln 19 Ch 22 Col 22 INS

Item(s) Saved

Windows Ink Workspace

12:33 PM 25-Mar-18

11. Write a DELETE statement that deletes the row in the Categories table that has an ID of 4. When you execute this statement, it will produce an error since the category has related rows in Products table. To fix that, precede DELETE statement with another DELETE statement that deletes all products in this category. Use SELECT statements to verify your result (data changes before & after insertion).

Project1_CQ11.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Execute Debug

MyGuitarShop

Object Explorer

Project1_CQ11.sql - D4\sonalpatil (53) Project1_CQ10.sql - not connected Project1_CQ9.sql - not connected Project1_CQ8.sql - not connected

11. Write DELETE statement that deletes row in Categories table that has ID of 4.

When you execute this statement, it will produce an error since the category has related rows in the Products table.

To fix that, precede DELETE statement with another DELETE statement that deletes all products in this category.

Use SELECT statements to verify your result (data changes before and after insertion). */

`USE MyGuitarShop;

DELETE FROM Products
WHERE CategoryID = 4;

DELETE FROM Categories
WHERE CategoryID = 4;

SELECT * FROM Categories;`
 /* comments - here i have used delete statement to delete an entry from Categories table but to delete all its footprints from other tables i have used another delete statement before this one.*/

Results

CategoryID	CategoryName
1	Guitars
2	Basses
3	Drums
4	Woodwinds

Query executed successfully.

DESKTOP-69JNVD4\SQLEXPRESS... DESKTOP-69JNVD4\sonalp... MyGuitarShop 00:00:00 4 rows

Ln 13 Ch 22 Col 22 INS

Ready

Windows Ink Workspace

2:31 PM 25-Mar-18

D. Advanced SQL skills (views/stored procedures/functions/scripts) [24 pts.]

Open script CreateMyGuitarShop.sql & run it. That should restore data. Complete questions in SectionD.

1. Create view OrderItemProducts that returns columns from Orders, OrderItems, & Products tables.

- a. columns from Orders table: OrderID, OrderDate, TaxAmount, & ShipDate.
- b. columns from OrderItems table: ItemPrice, DiscountAmount, FinalPrice (discount amount subtracted from item price), Quantity & ItemTotal (calculated total for item).
- c. column from Products table: ProductName.

```

Project1_DQ1.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (53)) - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
MyGuitarShop Execute Debug
Object Explorer Properties
Project1_DQ2.sql - VD4\sonalpatil (56) SQLQuery10.sql - DE...D4\sonalpatil (57)* Project1_DQ1.sql - VD4\sonalpatil (57)
1. Create a view named OrderItemProducts that returns columns from the Orders, OrderItems, and Products tables.
    a. This view should return these columns from Orders table: OrderID, OrderDate, TaxAmount, & ShipDate.
    b. This view should return these columns from OrderItems table: ItemPrice, DiscountAmount, FinalPrice
        (the discount amount subtracted from the item price), Quantity, and ItemTotal (the calculated total for the item).
    c. This view should return the ProductName column from the Products table. */

CREATE VIEW OrderItemProducts AS
SELECT O.OrderID, O.OrderDate, O.TaxAmount, O.ShipDate,
OI.ItemPrice, OI.DiscountAmount, (OI.ItemPrice - OI.DiscountAmount) AS 'FinalPrice', OI.Quantity,
(OI.ItemPrice - OI.DiscountAmount) * OI.Quantity AS 'ItemTotal',
P.ProductName
FROM Orders AS O JOIN OrderItems AS OI ON O.OrderID = OI.OrderID
JOIN Products AS P ON P.ProductID = OI.ProductID;

/*comments - here i have created a view using create view statement which compbines multiple columns from 3 different tables of current database*/

```

Messages: Commands completed successfully.

Query executed successfully.

```

Project1_DQ1.sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (67)) - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
MyGuitarShop Execute Debug
Object Explorer Properties
Project1_DQ4[2].sql - D4\sonalpatil (71) Project1_CQ6.sql - VD4\sonalpatil (61) Project1_DQ1.sql - VD4\sonalpatil (67) Project1_DQ4[1].sql - D4\sonalpatil (70)
1. Create a view named OrderItemProducts that returns columns from the Orders, OrderItems, and Products tables.
    a. This view should return these columns from Orders table: OrderID, OrderDate, TaxAmount, & ShipDate.
    b. This view should return these columns from OrderItems table: ItemPrice, DiscountAmount, FinalPrice
        (the discount amount subtracted from the item price), Quantity, and ItemTotal (the calculated total for the item).
    c. This view should return the ProductName column from the Products table. */

USE MyGuitarShop;

CREATE VIEW OrderItemProducts AS
SELECT O.OrderID, O.OrderDate, O.TaxAmount, O.ShipDate,
OI.ItemPrice, OI.DiscountAmount, (OI.ItemPrice - OI.DiscountAmount) AS 'FinalPrice', OI.Quantity,
(OI.ItemPrice - OI.DiscountAmount) * OI.Quantity AS 'ItemTotal',
P.ProductName
FROM Orders AS O JOIN OrderItems AS OI ON O.OrderID = OI.OrderID
JOIN Products AS P ON P.ProductID = OI.ProductID;

SELECT * FROM OrderItemProducts; /*SELECT statement to see the result set stored in the above view*/
/*comments - here i have created a view using create view statement which compbines multiple columns from 3 different tables of current database*/

```

OrderID	OrderDate	TaxAmount	ShipDate	ItemPrice	DiscountAmount	FinalPrice	Quantity	Item Total	ProductName
1	2016-03-28 09:40:28.000	58.75	2016-03-31 09:41:11.000	1199.00	359.70	839.30	1	839.30	Gibson Les Paul
2	2016-03-28 11:23:20.000	21.27	2016-03-31 11:24:03.000	489.99	186.20	303.79	1	303.79	Hofner Icon
3	2016-03-29 09:44:58.000	102.29	2016-04-01 09:45:41.000	2517.00	1305.84	1208.16	1	1208.16	Fender Stratocaster
4	2016-03-29 09:44:58.000	102.29	2016-04-01 09:45:41.000	415.00	161.85	253.15	1	253.15	Ludwig 5-piece Drum Set with Cymbals
5	2016-03-30 15:22:31.000	117.50	2016-04-02 15:23:14.000	1199.00	359.70	839.30	2	1678.60	Gibson Les Paul
6	2016-03-31 05:43:11.000	20.93	2016-04-03 05:43:54.000	299.00	0.00	299.00	1	299.00	Tama 5-Piece Drum Set with Cymbals
7	2016-03-31 18:37:22.000	20.93	2016-04-03 18:38:05.000	299.00	0.00	299.00	1	299.00	Tama 5-Piece Drum Set with Cymbals
8	2016-04-01 23:11:12.000	107.80	2016-04-04 23:11:55.000	699.99	210.00	489.99	1	489.99	Washburn D10S
9	2016-04-01 23:11:12.000	107.80	2016-04-04 23:11:55.000	799.99	240.00	559.99	1	559.99	Gibson SG
10	2016-04-01 23:11:12.000	107.80	2016-04-04 23:11:55.000	699.99	210.00	489.99	1	489.99	Washburn D10S
11	2016-04-02 11:26:38.000	47.60	2016-04-05 11:27:21.000	799.99	120.00	679.99	1	679.99	Yamaha FG700S
12	2016-04-03 12:22:31.000	102.75	2016-04-06 12:23:14.000	699.00	209.70	489.30	3	1467.90	Rodríguez Caballero 11
13	2016-04-03 14:59:20.000	26.25	2016-04-06 15:00:03.000	499.99	125.00	374.99	1	374.99	Fender Precision
14	2016-04-06 06:24:00.000	34.25	2016-04-07 06:26:27.000	699.00	204.70	494.30	1	494.30	Rodríguez Caballero 11

Results: Messages: Commands completed successfully.

2. Create a view named ProductSummary that uses the view you created in Section D Question 1. This view should return some summary information about each product. Each row should include these columns: ProductName, OrderCount (the number of times the product has been ordered), and OrderTotal (the total sales for the product).

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the MyGuitarShop database and its tables. In the center, a query window displays the following SQL code:

```

/*
Project 1 Sonal Patil D.
2. Create a view named ProductSummary that uses the view you created in Section D Question 1.
This view should return some summary information about each product. Each row should include these columns:
ProductName,
OrderCount (the number of times the product has been ordered),
OrderTotal (the total sales for the product). */

CREATE VIEW ProductSummary AS
SELECT otp.ProductName, SUM(Quantity) AS OrderCount, SUM(ItemTotal) AS OrderTotal
FROM OrderItemProducts AS otp
GROUP BY otp.ProductName;

/*comments - here i have used previously created view to create new view using CREATE VIEW statement and group by clause to give a row
for each product*/

```

The Messages pane at the bottom shows the message "Commands completed successfully." The Properties pane on the right shows connection details.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the MyGuitarShop database and its tables. In the center, a query window displays the following SQL code:

```

/*
Project 1 Sonal Patil D.
2. Create a view named ProductSummary that uses the view you created in Section D Question 1. This view should return some summary information
about each product. Each row should include these columns: ProductName, OrderCount (number of times product has been ordered),
OrderTotal (the total sales for the product). */

USE MyGuitarShop;

CREATE VIEW ProductSummary AS
SELECT otp.ProductName, SUM(Quantity) AS OrderCount, SUM(ItemTotal) AS OrderTotal
FROM OrderItemProducts AS otp
GROUP BY otp.ProductName;

SELECT * FROM ProductSummary; /*SELECT statement to see the result set stored in the above view*/
/*comments - here i have used previously created view to create new view using CREATE VIEW statement and group by clause to give a row
for each product*/

```

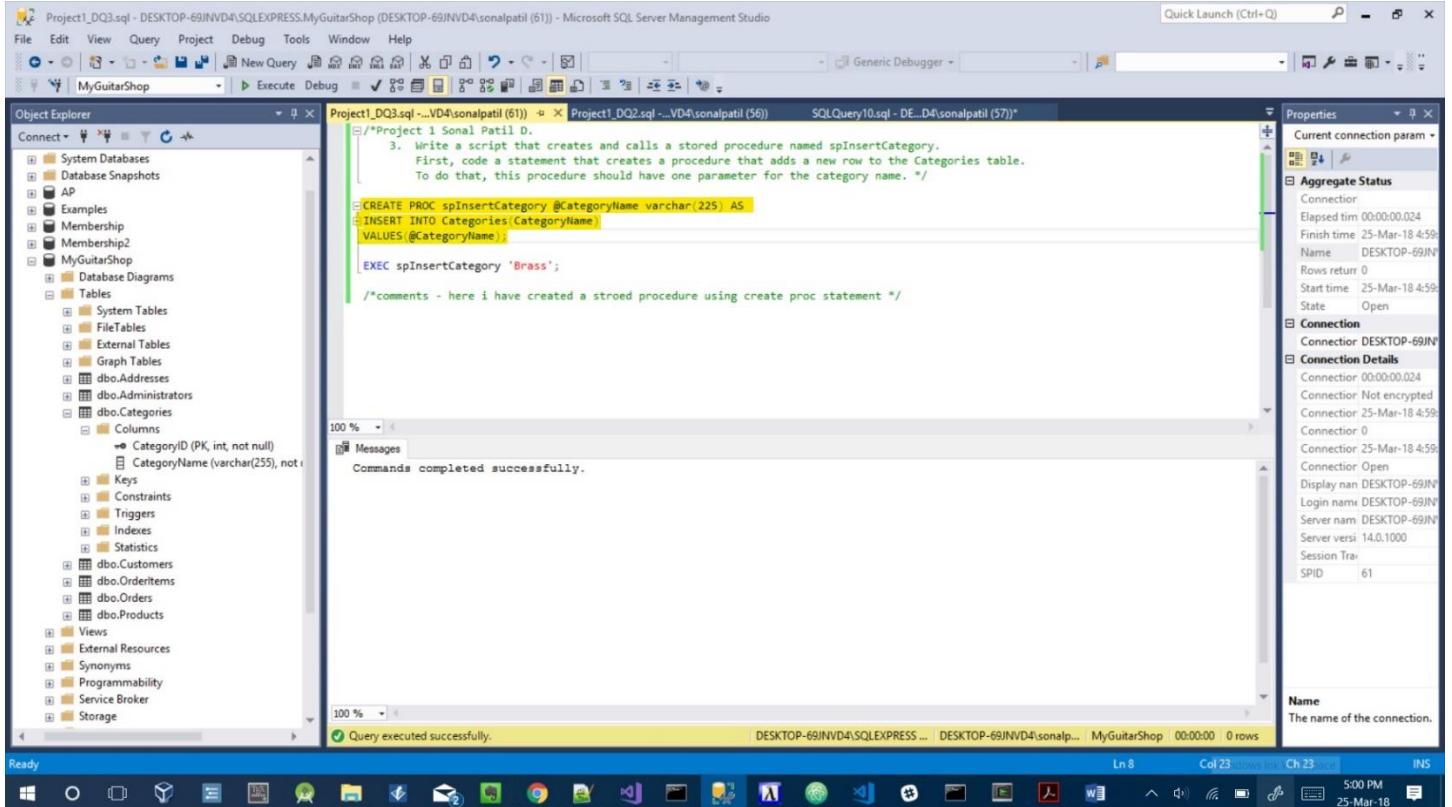
The Results pane on the right displays the following table of data:

	ProductName	OrderCount	OrderTotal
1	Fender Precision	2	749.98
2	Fender Stratocaster	7	8457.12
3	Gibson Les Paul	12	10071.60
4	Gibson SG	9	5039.91
5	Hofner Icon	3	911.37
6	Ludwig 5-piece Drum Set with Cymbals	2	506.30
7	Rodriguez Caballero 11	14	6850.20
8	Tama 5-Piece Drum Set with Cymbals	4	1196.00
9	Washburn D10S	3	1469.97
10	Yamaha FG700S	3	2039.97

The Messages pane at the bottom shows the message "Commands completed successfully." The Properties pane on the right shows connection details.

3. Write a script that creates and calls a stored procedure named spInsertCategory. First, code a statement that creates a procedure that adds a new row to the Categories table. To do that, this procedure should have one parameter for the category name. Code at least two EXEC statements that test this procedure. (Note that this table doesn't allow duplicate category names.)

Created stored procedure -



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'MyGuitarShop' is selected. In the center pane, a query window contains the following T-SQL code:

```


--/*Project 1 Sonal Patil D.
3. Write a script that creates and calls a stored procedure named spInsertCategory.
First, code a statement that creates a procedure that adds a new row to the Categories table.
To do that, this procedure should have one parameter for the category name. */

CREATE PROC spInsertCategory @CategoryName varchar(225) AS
    INSERT INTO Categories(CategoryName)
    VALUES (@CategoryName);

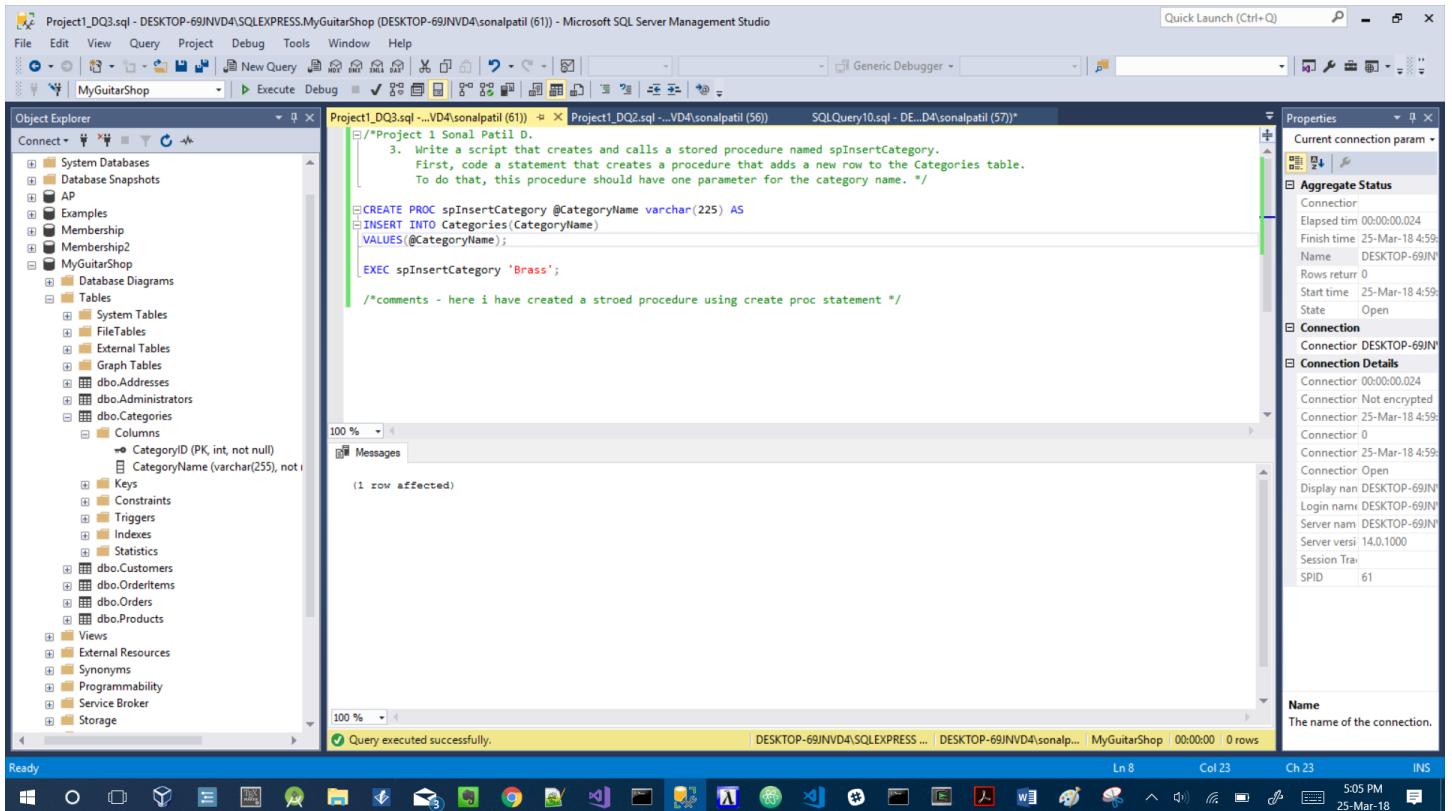
EXEC spInsertCategory 'Brass';

/*comments - here i have created a stored procedure using create proc statement */


```

The 'Messages' pane below the code shows the output: "Commands completed successfully." The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-69JNVD4\SQLEXPRESS... DESKTOP-69JNVD4\sonalpatil MyGuitarShop 00:00:00 0 rows".

Executed created store procedure statement -



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'MyGuitarShop' is selected. In the center pane, a query window contains the same T-SQL code as the previous screenshot. The 'Messages' pane shows the output: "(1 row affected)". The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-69JNVD4\SQLEXPRESS... DESKTOP-69JNVD4\sonalpatil MyGuitarShop 00:00:00 0 rows".

Called Stored Procedure -

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'MyGuitarShop' database is selected. In the center pane, a script window contains the following T-SQL code:

```


--Project 1 Sonal Patil D.
3. Write a script that creates and calls a stored procedure named spInsertCategory.
First, code a statement that creates a procedure that adds a new row to the Categories table.
To do that, this procedure should have one parameter for the category name.
Code at least two EXEC statements that test this procedure. (Note that this table doesn't allow duplicate category names.) */

CREATE PROC spInsertCategory @CategoryName varchar(225) AS
INSERT INTO Categories(CategoryName)
VALUES (@CategoryName);

EXEC spInsertCategory 'Gold';
EXEC spInsertCategory 'Silver';

/*comments - here i have created a stored procedure using create proc statement and executed that procedure to add two new rows in categories table */


```

The 'Messages' pane shows two messages indicating successful execution:

- [2 row affected]
- [2 row affected]

The status bar at the bottom indicates "Query executed successfully." and "0 rows".

Showed result using select statement -

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'MyGuitarShop' database is selected. In the center pane, a script window contains the following T-SQL code:

```


--Project 1 Sonal Patil D.
3. Write a script that creates and calls a stored procedure named spInsertCategory.
First, code a statement that creates a procedure that adds a new row to the Categories table.
To do that, this procedure should have one parameter for the category name.
Code at least two EXEC statements that test this procedure. (Note that this table doesn't allow duplicate category names.) */

CREATE PROC spInsertCategory @CategoryName varchar(225) AS
INSERT INTO Categories(CategoryName)
VALUES (@CategoryName);

EXEC spInsertCategory 'Gold';
EXEC spInsertCategory 'Silver';

SELECT * FROM Categories;


```

The 'Results' pane displays the following table of data:

CategoryID	CategoryName
1	Basses
2	Brass
3	Drums
4	Gold
5	Guitars
6	Keyboards
7	Silver

The status bar at the bottom indicates "Query executed successfully." and "7 rows".

4. [1] Write a script that creates a function named fnDiscountPrice that calculates the discount price of an item in the OrderItems table (discount amount subtracted from item price). To do that, this function should accept one parameter for the item ID, and it should return the value of the discount price for that item.

Project1_DQ4[1].sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (55)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Generic Debugger

MyGuitarShop Execute Debug

Object Explorer

Project1_DQ4[1].sql - DESQQuery10.sql - DE...D4\sonalpatil (57)* SQLQuery14.sql - DE...D4\sonalpatil (58)*

```

CREATE FUNCTION fnDiscountPrice (@itemID INT)
RETURNS MONEY
BEGIN
    RETURN (SELECT (ItemPrice - DiscountAmount) AS 'FinalPrice'
            FROM OrderItems
            WHERE OrderItems.ItemID = @itemID);
END;

```

/*comments - here i have used create function statement which takes itemID as input and return FinalPrice as output matching with given itemID*/

Messages

Commands completed successfully.

Query executed successfully.

Properties

Current connection param

Aggregate Status

Connector Elapsed time 00:00:00.039 Finish time 25-Mar-18 5:35:53 Name DESKTOP-69JN Rows return 0 Start time 25-Mar-18 5:35:53 State Open

Connection

Connector DESKTOP-69JN Connection Details Connector 00:00:00.039 Connector Not encrypted Connector 25-Mar-18 5:35:53 Connector 0 Connector 25-Mar-18 5:35:53 Connector Open Display name DESKTOP-69JN Login name DESKTOP-69JN Server name DESKTOP-69JN Server versi 14.0.1000 Session Trai SPID 55

Name The name of the connection.

Item(s) Saved

Ln 5 Col 1 Ch 1 INS

[2] Write a script that creates and calls a function named fnItemTotal that calculates the total amount of an item in the OrderItems table (discount price multiplied by quantity). To do that, this function should accept one parameter for the item ID, it should use the DiscountPrice function that you created in (1), and it should return the value of the total for that item.

Created the function –

Project1_DQ4[2].sql - DESKTOP-69JNVD4\SQLEXPRESS.MyGuitarShop (DESKTOP-69JNVD4\sonalpatil (58)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

New Query Generic Debugger

MyGuitarShop Execute Debug

Object Explorer

Project1_DQ4[1].sql - DESQQuery10.sql - DE...D4\sonalpatil (57)* Project1_DQ4[2].sql - D4\sonalpatil (58)*

```

CREATE FUNCTION fnItemTotal(@ItemID int)
RETURNS money
BEGIN
    RETURN (SELECT dbo.fnDiscountPrice(@ItemID) * (SELECT Quantity
                                                    FROM OrderItems
                                                    WHERE ItemID = @ItemID) AS 'TotalPrice')
END;

PRINT 'Total Price = $' + CONVERT(varchar, dbo.fnItemTotal(5));

```

/*comments - here i have used create function statement which takes itemID as input and return TotalPrice as output matching with given itemID. This function gives a call to another function written in 1st part of this question which gives us finalprice of the item and then totalprice gets calculated in this function by multiplying it with quantities Example - ItemID = 5 (ItemPrice - DiscountPrice) * Quantity = (1199.00 - 359.70) * 2 = 1678.60 */

Messages

Commands completed successfully.

Query executed successfully.

Properties

Current connection param

Aggregate Status

Connector Elapsed time 00:00:00.059 Finish time 25-Mar-18 6:03:33 Name DESKTOP-69JN Rows return 0 Start time 25-Mar-18 6:03:33 State Open

Connection

Connector 00:00:00.059 Connector Not encrypted Connector 25-Mar-18 6:03:33 Connector 0 Connector 25-Mar-18 6:03:33 Connector Open Display name DESKTOP-69JN Login name DESKTOP-69JN Server name DESKTOP-69JN Server versi 14.0.1000 Session Trai SPID 58

Name The name of the connection.

Ready

Ln 14 Col 1 Ch 1 INS

Called the function -

```
CREATE FUNCTION fnItemTotal(@ItemID int)
RETURNS money
BEGIN
    RETURN (SELECT Quantity
           FROM OrderItems
           WHERE ItemID = @ItemID) * (SELECT ItemPrice - DiscountPrice
                                       FROM OrderItems
                                       WHERE ItemID = @ItemID)
END;
```

PRINT 'Total Price = \$' + CONVERT(varchar, dbo.fnItemTotal(5))

Example - ItemID = 5
= (1199.00 - 359.70) * 2
= 1678.60

Total Price = \$1678.60

Query executed successfully.

Properties pane shows connection details: Connector: DESKTOP-69INVDA\sonalpatil (58), Connection: DESKTOP-69INVDA\sonalpatil (58), Connection Details: Connector 0, Connector Not encrypted, Connector 25-Mar-18 6:01:00, Connector Open, Display name DESKTOP-69INVDA\sonalpatil, Login name DESKTOP-69INVDA\sonalpatil, Server name DESKTOP-69INVDA, Server versi 14.0.1000, Session Tra: SPID 58.

II. Database Design [20 pts.]

Create sample database design for a small bakery and draw one database model for it.

The bakery is interested in keeping track of information about its customers, suppliers, products, ingredients and sales. Bakery sells their products to customers & use different ingredients to create their products. Suppliers provide ingredients to bakery. Customers buy different products.

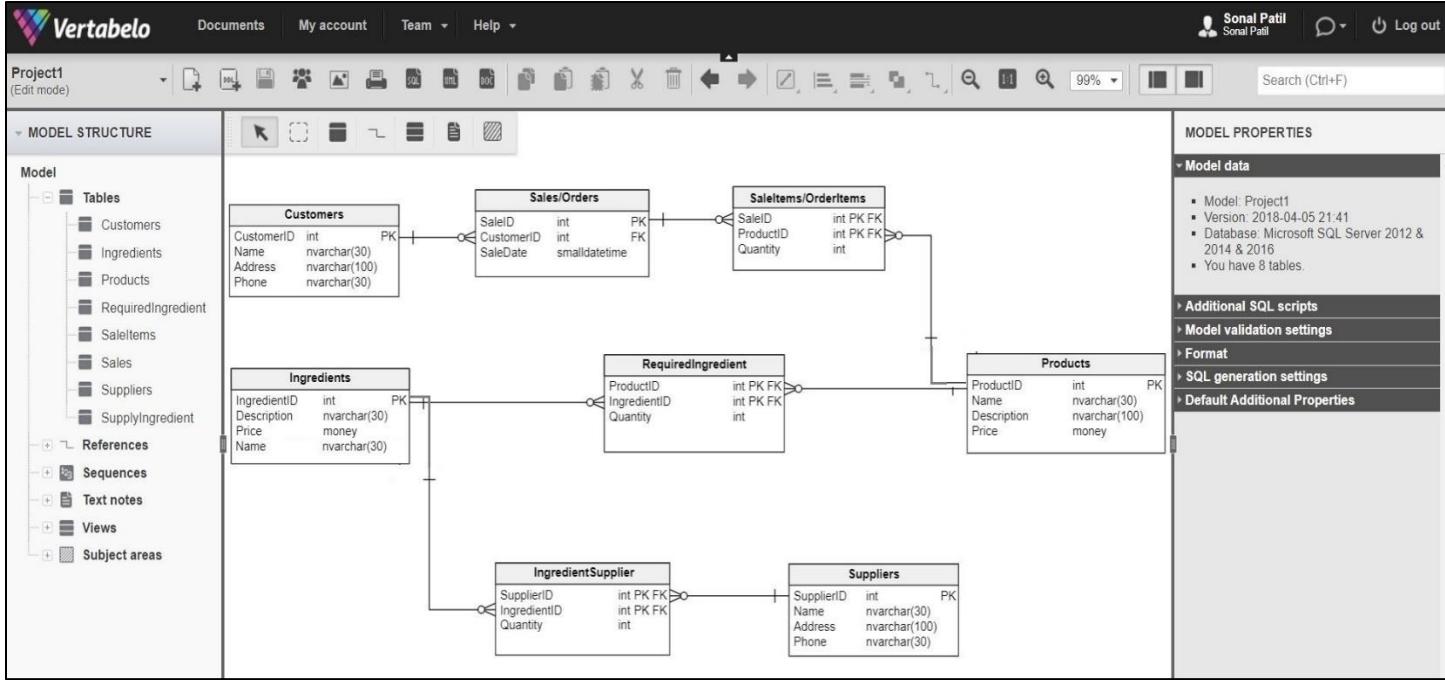
1. Design database that makes sense for business problem & select fields that make the most sense. A full screenshot of your final design is required.
2. Determine the tables, columns, primary keys, datatypes, nullabilities, and show relationships between tables(one-one/one-many/many-many).
3. Explain your design, including relationship between tables.

Here, I have designed total 8 tables out of which 5 are demanded by the requirement of creating a Bakery database and 3 are used to have the relationship between them.

Database design explanation & Full design screenshot -

Customers:

Customers will place one or multiple orders which will contain products that Bakery produces. Uniquely identifying key i.e. Primary key is CustomerID. Each Customer will have only one unique ID. Non-Nullable Values are CustomerID which is a primary key and it cannot be null also Name column must have value. Nullable values are Address and Phone [Optional data].



Sales[/Orders]:

This table will get the Order/SaleID and maps it with the customerID.

Uniquely identifying key i.e. Primary key is SaleID. Each Sale/Order will have only one unique ID. Foreign key is CustomerID. This maps with the unique record from Customers to Sales. This key indicates one:many relationship between Customers and Sales/Orders table. One Customer can have one or more order entries in Orders table.

Non-Nullable Values are SaleID, CustomerID. Right now, in the Sales table, only SaleDate is non-key column, so I chose not to keep it available for Null values.

SaleItems[/OrderItems]:

This table gets the Order from Order table and maps it with the ProductIDs that means when customer orders multiple products, the Order/SaleID will have multiple ProductIDs rows.

Composite key i.e. Multiple Primary keys are SaleID & ProductID.

Foreign keys are SaleID & ProductID. This key indicates one:many relationship between Sales/Orders & SaleItems table and between Product & SaleItems.

Non-Nullable Values are SaleID, ProductID. Right now, in the SaleItems table, only Quantity is non-key column, so I chose not to keep it available for Null values.

Products:

This table keeps track of Product made by Bakery and Ingredients used for making that Product.

Uniquely identifying key i.e. Primary key is ProductID. Each Product will have only one unique ID.

Non-Nullable Values are ProductID which is a primary key and it cannot be null also Name & Price column must have values. Nullable value can be placed for Description column [Optional data].

RequiredIngredients:

This table maps the Ingredients used for creating single product. That means it contains the ProductID and equivalent IngredientIDs used to make that Product.

Composite key i.e. Multiple Primary keys are ProductID & IngredientID.

Foreign keys are ProductID & IngredientID. This key indicates one:many relationship between Products & RequiredIngredients and between RequiredIngredients & Ingredients table. Non-Nullable Values are ProductID, IngredientID. Right now, in the RequiredIngredients table, only Quantity is non-key column, so I chose not to keep it available for Null values.

Ingredients:

This table keeps data stored related to each Ingredient.

Uniquely identifying key i.e. Primary key is IngredientID. Each Ingredient will have only 1 unique ID. Non-Nullable Values are IngredientID which is a primary key and it cannot be null also Name & Price column must have values. Nullable value can be placed for Description column [Optional data].

IngredientSupplier:

This table maps the Ingredient with its supplier.

Composite key i.e. Multiple Primary keys are IngredientID & SupplierID.

Foreign keys are IngredientID & SupplierID. This key indicates one:many relationship between IngredientSupplier & Suppliers and between IngredientSupplier & Ingredient table.

Non-Nullable Values are SupplierID, IngredientID. Right now, in the IngredientSupplier table, only Quantity is non-key column, so I chose not to keep it available for Null values.

Suppliers:

This table keeps data stored related to each supplier.

Uniquely identifying key i.e. Primary key is SupplierID. Each Supplier will have only one unique ID.

Non-Nullable Values are SupplierID which is a primary key and it cannot be null also Name column must have values. Nullable value can be placed for Address & Phone column [Optional data].

4. Normalize your design into its 3rd Normal Form. Please use MS Visio or any similar tool.

The designed database is already in 3rd Normal form since every non-key column of all tables depends only on its own primary key.

Remarks -

This project helped me revise all the concepts I have learned so far in this course.

For first Section, I created a new database called MyGuitarShop using given CreateMyGuitarShop.sql script and then performed some basic & advance level functionality on it.

In the 1st section of this project, I learned & played with the features of SQL server management studio and executed some basic SQL commands to get to know the database structure more. Modified the database tables using INSERT, UPDATE, DELETE statements. Created new views, stored procedures.

In the 2nd section, I learned how to design a database from scratch when you have given the requirements or jobs one real world entity does. Here, how bakery works was given as a part of requirement and then I designed the database matching those requirements.

It also helped me to understand how to transform real world entities data into database tables, how to relate them with each other using relationships available to us. Also gave me knowledge of database design engine 'Vertabelo'.