

Unit 6 – HomeWork

1. What are user defined functions and stored procedures. When are they typically used?

Functions and Stored Procedures are two very different entities in SQL server. They share some common features, such as: all stored in a database, can reduce network traffic, compiled only once and automatic recompilation.

There are two types of Functions: Built-in-Functions (Scalar, Aggregate, Rowset, Ranking Functions) and **User-Defined Function** (Scalar, Inline table valued, multi statement valued User-Defined Functions), and there are 4 types of **Stored Procedures**: User-Defined, System-Defined, Temporary, Extended Stored procedures.

The table below will help decide whether to use User Defined Function (UDF) or Stored Procedure (SP) based on actual requirements.

User Defined Function (UDF)	Stored Procedure (SP)
Return a value which is mandatory	Return value is optional (zero, single, or multiple values)
Allows only SELECT statement	Allows SELECT and DML statements
Can only have input parameters	Can have input/output parameters
Does not support Try Catch blocks	Try Catch blocks can be used
Does not support transaction management	Support Transaction Management
Can use JOIN statement	Does not support JOIN table variables
Used in WHERE/HAVING/SELECT sections	Cannot be used in WHERE/HAVING/SELECT sections
Can use all data types	Cannot use some data types like ntext, image and timestamp

- Building on our Marketing project,
Calculate the Average Number(Count) of Orders per season (Seasons as defined in the SQLProject_When_ToOffer file from the InClass –Live Session 6 folder)

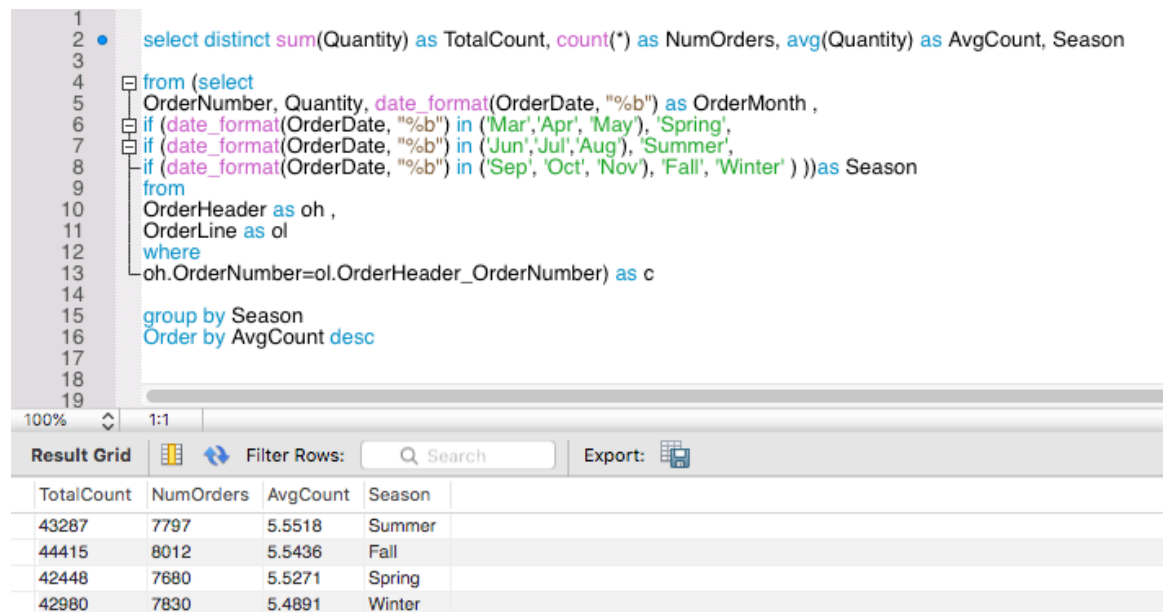
SQL scripts:

```
select distinct sum(Quantity) as TotalCount, count(*) as NumOrders, avg(Quantity) as AvgCount, Season

from (select
OrderNumber, Quantity, date_format(OrderDate, "%b") as OrderMonth ,
if (date_format(OrderDate, "%b") in ('Mar','Apr', 'May'), 'Spring',
if (date_format(OrderDate, "%b") in ('Jun','Jul','Aug'), 'Summer',
if (date_format(OrderDate, "%b") in ('Sep', 'Oct', 'Nov'), 'Fall', 'Winter' ) ))as
Season
from
OrderHeader as oh,
OrderLine as ol
where
oh.OrderNumber=ol.OrderHeader_OrderNumber) as c

group by Season
Order by AvgCount desc
```

SQL Results:



The screenshot shows a SQL IDE interface. The top pane displays the SQL query, and the bottom pane shows the results in a table grid. The query is as follows:

```
1 select distinct sum(Quantity) as TotalCount, count(*) as NumOrders, avg(Quantity) as AvgCount, Season
2
3
4 from (select
5 OrderNumber, Quantity, date_format(OrderDate, "%b") as OrderMonth ,
6 if (date_format(OrderDate, "%b") in ('Mar','Apr', 'May'), 'Spring',
7 if (date_format(OrderDate, "%b") in ('Jun','Jul','Aug'), 'Summer',
8 if (date_format(OrderDate, "%b") in ('Sep', 'Oct', 'Nov'), 'Fall', 'Winter' ) ))as Season
9 from
10 OrderHeader as oh ,
11 OrderLine as ol
12 where
13 oh.OrderNumber=ol.OrderHeader_OrderNumber) as c
14
15 group by Season
16 Order by AvgCount desc
17
18
19
```

The results table has the following data:

TotalCount	NumOrders	AvgCount	Season
43287	7797	5.5518	Summer
44415	8012	5.5436	Fall
42448	7680	5.5271	Spring
42980	7830	5.4891	Winter

3. Is our Ecommerce DB design 1NF, 2NF and 3NF compliant? If yes, Why? If No, why not?

Normalization is a step by step process which organize tables to avoid redundancy and dependency of data. The definitions of three database normal forms are as follows:

1NF (First normal form): each table cell should contain a single value (no multiple values, separated by comma), each record needs to be unique (no repeating column groups) and identify each record uniquely using primary key.

2NF (Second normal form): table in 1NF, move redundant data to a separate table, and create relationship between these tables using foreign keys.

3NF (Third normal form): Table must be in 2NF and has no transitive functional dependencies upon the primary key.

Our Ecommerce DB design is normalized to meet 3NF rule since all the transitive dependencies are resolved by moving the dependency attributes to a new entity type with one-to-many relationship.

