Scalar user defined functions

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Meghan Kwartler
IT Consultant

User defined functions (UDFs)

What?

Routines that

- Can accept input parameters
- Perform an action
- Return result (single scalar value or table)

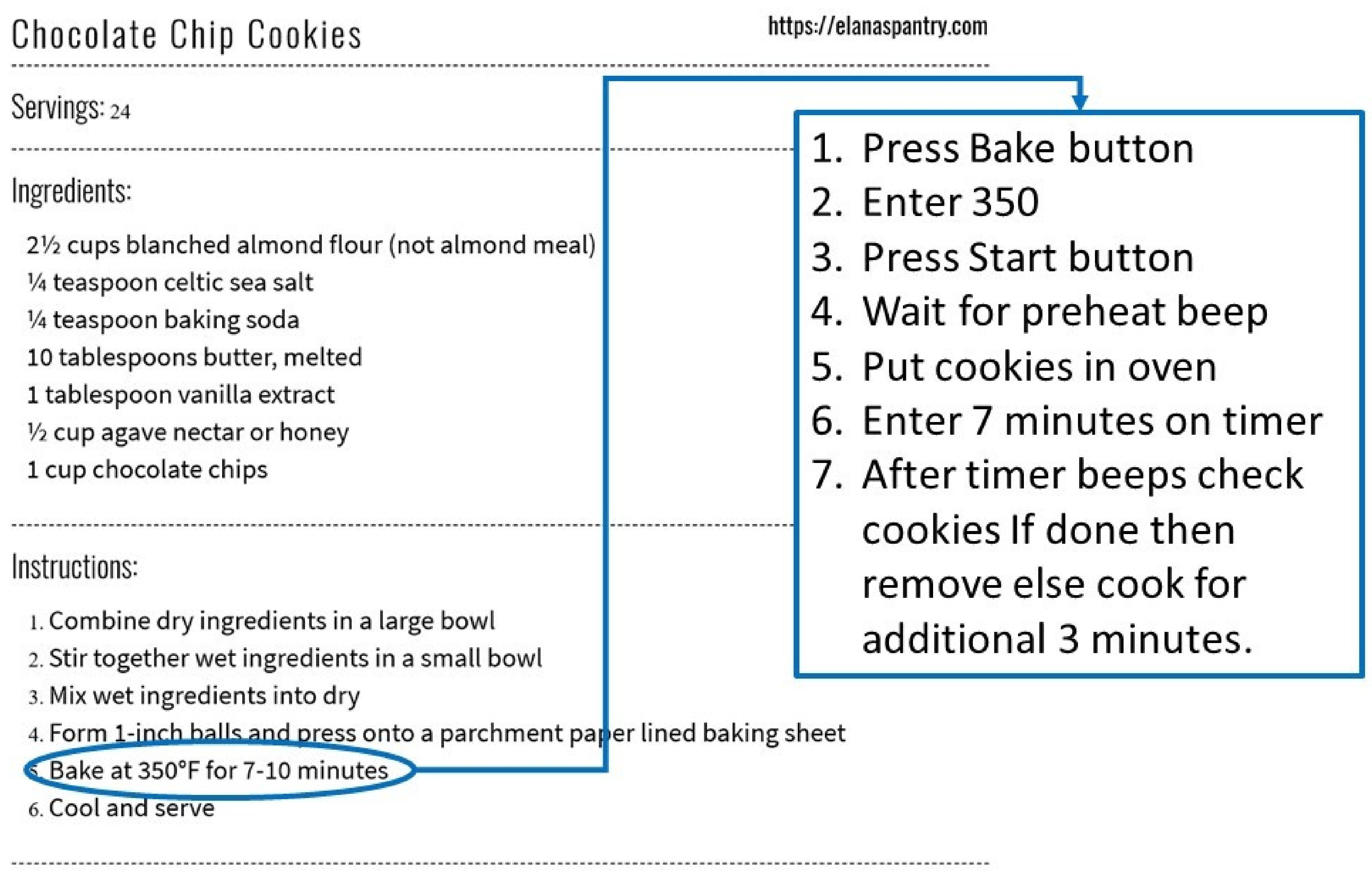
Why?

- Can reduce execution time
- Can reduce network traffic
- Allow for Modular Programming

What is modular programming?

- Software design technique
- Separates functionality into independent, interchangeable modules
- Allows code reuse
- Improves code readability

Functions in recipes



Chocolate Chip Cookies

https://elanaspantry.com

Servings: 24

Ingredients:

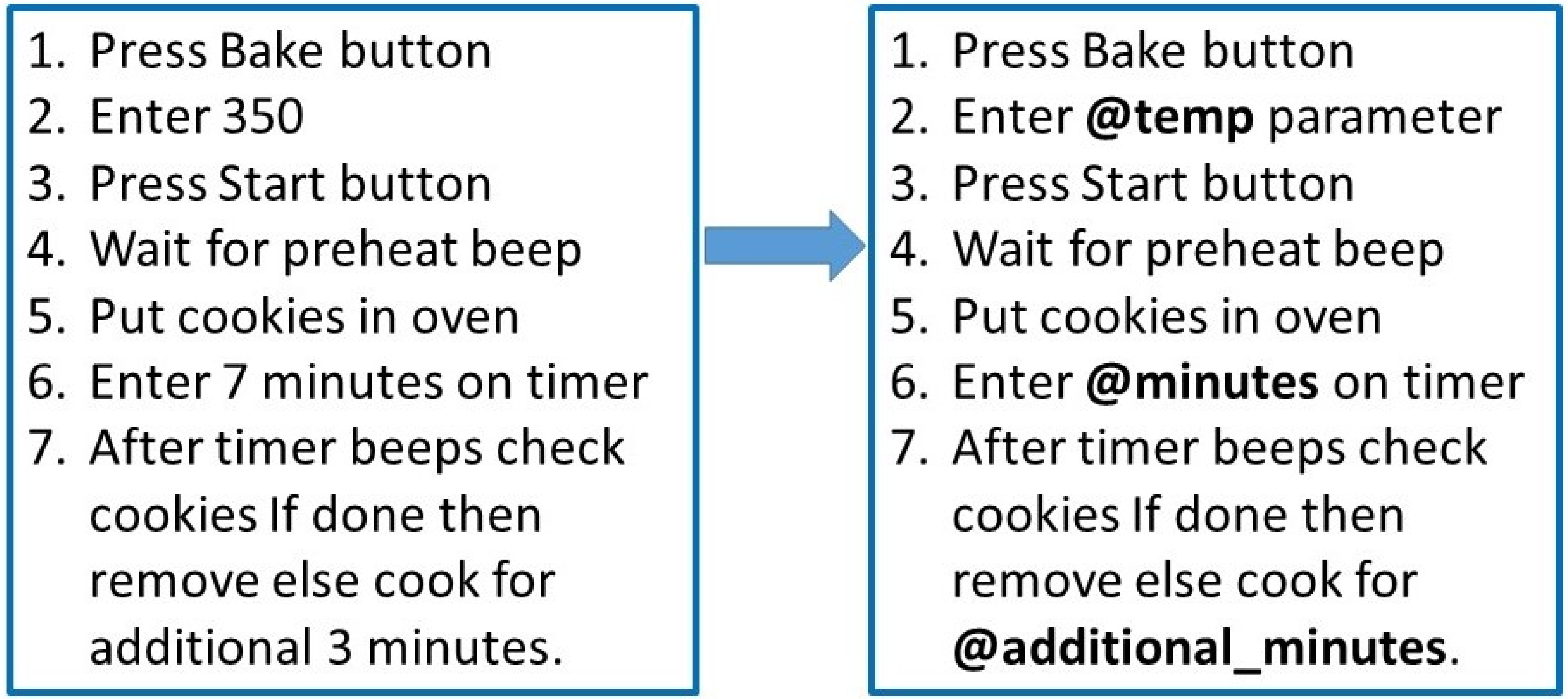
2½ cups blanched almond flour (not almond meal)
¼ teaspoon celtic sea salt
¼ teaspoon baking soda
10 tablespoons butter, melted
1 tablespoon vanilla extract
½ cup agave nectar or honey
1 cup chocolate chips

Instructions:

1. Combine dry ingredients in a large bowl
2. Stir together wet ingredients in a small bowl
3. Mix wet ingredients into dry
4. Form 1-inch balls and press onto a parchment paper lined baking sheet
5. Bake at 350°F for 7-10 minutes
6. Cool and serve

1. Press Bake button
2. Enter 350
3. Press Start button
4. Wait for preheat beep
5. Put cookies in oven
6. Enter 7 minutes on timer
7. After timer beeps check cookies If done then remove else cook for additional 3 minutes.

Bake function input parameters



1. Press Bake button
2. Enter 350
3. Press Start button
4. Wait for preheat beep
5. Put cookies in oven
6. Enter 7 minutes on timer
7. After timer beeps check cookies If done then remove else cook for additional 3 minutes.

1. Press Bake button
2. Enter **@temp** parameter
3. Press Start button
4. Wait for preheat beep
5. Put cookies in oven
6. Enter **@minutes** on timer
7. After timer beeps check cookies If done then remove else cook for **@additional_minutes**.

Scalar UDF with no input parameter

```
-- Scalar function with no input parameters
CREATE FUNCTION GetTomorrow()
    RETURNS date AS BEGIN
RETURN (SELECT DATEADD(day, 1, GETDATE()))
END
```

Scalar UDF with one parameter

```
-- Scalar function with one parameter
CREATE FUNCTION GetRideHrsOneDay (@DateParm date)
    RETURNS numeric AS BEGIN
RETURN (
  SELECT
    SUM(
      DATEDIFF(second, PickupDate, DropoffDate)
    )/ 3600
  FROM
    YellowTripData
  WHERE
    CONVERT (date, PickupDate) = @DateParm
) END;
```

All user defined function names should contain a verb and parameter names must begin with an @ sign.

Scalar UDF with two input parameters

```
-- Scalar function with two input parameters
CREATE FUNCTION GetRideHrsDateRange (
  @StartDateParm datetime, @EndDateParm datetime
) RETURNS numeric AS BEGIN RETURN (
  SELECT
    SUM(
      DATEDIFF(second, PickupDate, DropOffDate)
    )/ 3600
  FROM YellowTripData
  WHERE
    PickupDate > @StartDateParm
    AND DropoffDate < @EndDateParm
) END;
```

It's your turn to create UDFs!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Table valued UDFs

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Meghan Kwartler
IT Consultant

Inline table valued functions (ITVF)

```
CREATE FUNCTION SumLocationStats (
  @StartDate AS datetime = '1/1/2017'
) RETURNS TABLE AS RETURN
SELECT
  PULocationID AS PickupLocation,
  COUNT(ID) AS RideCount,
  SUM(TripDistance) AS TotalTripDistance
FROM YellowTripData
WHERE CAST(PickupDate AS Date) = @StartDate
GROUP BY PULocationID;
```

Scalar functions require the use of the BEGIN END block, regardless of whether it's single statement, but a table valued function doesn't require BEGIN END if the function body is a single statement.

Column names need to be assigned in the SELECT statement because a table is being return.

```sql
CREATE FUNCTION CountTripAvgFareDay (
  @Month char(2),
  @Year char(4)
) RETURNS @TripCountAvgFare TABLE(
  DropOffDate date, TripCount int, AvgFare numeric
) AS BEGIN INSERT INTO @TripCountAvgFare
SELECT
  CAST(DropOffDate as date),
  COUNT(ID),
  AVG(FareAmount) as AvgFareAmt
FROM YellowTripData
WHERE
  DATEPART(month, DropOffDate) = @Month
  AND DATEPART(year, DropOffDate) = @Year
GROUP BY CAST(DropOffDate as date)
RETURN END;
```

Differences - ITVF vs. MSTVF

Inline

- RETURN results of SELECT
- Table column names in SELECT
- No table variable
- No BEGIN END needed
- No INSERT
- Faster performance

Multi statement

- DECLARE table variable to be returned
- BEGIN END block required
- INSERT data into table variable
- RETURN last statement within BEGIN/END block

Your turn!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

UDFs in action

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Meghan Kwartler
IT Consultant

Execute scalar with SELECT

```
-- Select with no parameters
SELECT dbo.GetTomorrow()
```

The dbo. that precedes the function name is the schema where the function exists.
The schema must be specified when executing a UDF.
If a schema is not specified when creating the function SQL will automatically assign it the user's default schema.

```
+------------+
| 2019-02-28 |
-------------+
```

Execute scalar with EXEC & store result

```
-- EXEC & store result in variable
DECLARE @TotalRideHrs AS numeric
EXEC @TotalRideHrs = dbo.GetRideHrsOneDay @DateParm = '1/15/2017'
SELECT
  'Total Ride Hours for 1/15/2017:',
  @TotalRideHrs
```

We use the EXEC keyword to execute the GetRideHrsOneDay() function and assign the result to the @TotalRideHrs variable.

```
+---------------------------------+-------+
| Total Ride Hours for 1/15/2017: | 71626 |
+---------------------------------+-------+
```

SELECT parameter value & scalar UDF

```
-- Declare parameter variable
-- Set to oldest date in YellowTripData
-- Pass to function with select
DECLARE @DateParm as date =
(SELECT TOP 1 CONVERT(date, PickupDate)
    FROM YellowTripData
    ORDER BY PickupDate DESC)
SELECT @DateParm, dbo.GetRideHrsOneDay (@DateParm)
```

When using SELECT to execute the UDF, the parameter is listed within parenthesis after the function name.

```
+------------+-------+
| 2017-01-31 | 75519 |
+------------+-------+
```

```sql
SELECT TOP 10 *
FROM dbo.SumLocationStats ('1/09/2017')
ORDER BY RideCount DESC
```

```
+----------------+-----------+-------------------+
| PickupLocation | RideCount | TotalTripDistance |
|----------------+-----------+-------------------|
| 237            |13254      |  22281.95         |
| 161            |13206      |  28208.49         |
| 236            |13200      |  24224.69         |
| 162            |11859      |  26169.46         |
| 186            |10587      |  22415.43         |
| 230            |10257      |  26139.16         |
| 234            |10234      |  19758.23         |
| 170            |9963       |  20931.97         |
| 132            |9230       | 144778.90         |
| 48             |8361       |  18978.80         |
+----------------+-----------+-------------------+
DECLARE @CountTripAvgFareDay TABLE(
     DropOffDate    date,
     TripCount      int,
     AvgFare        numeric)
INSERT INTO @CountTripAvgFareDay
SELECT TOP 10 *
FROM dbo.CountTripAvgFareDay (01, 2017)
ORDER BY DropOffDate ASC

SELECT * FROM @CountTripAvgFareDay
```

```
+-------------+-----------+---------+
| DropOffDate | TripCount | AvgFare |
|-------------+-----------+---------|
| 2017-01-01  |279198     | 15.37   |
| 2017-01-02  |225224     | 12.65   |
| 2017-01-03  |277980     | 12.27   |
| 2017-01-04  |289050     | 12.33   |
| 2017-01-05  |323885     | 11.89   |
| 2017-01-06  |339158     | 11.72   |
| 2017-01-07  |306508     | 11.31   |
| 2017-01-08  |292649     | 12.33   |
| 2017-01-09  |302120     | 12.49   |
| 2017-01-10  |305611     | 12.27   |
+-------------+-----------+---------+
```

See your functions in action!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Maintaining user defined functions

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Meghan Kwartler
IT Consultant

ALTER Function

```sql
ALTER FUNCTION SumLocationStats (@EndDate as datetime = '1/01/2017')
RETURNS TABLE AS RETURN
SELECT
  PULocationID as PickupLocation,
  COUNT(ID) as RideCount,
  SUM(TripDistance) as TotalTripDistance
FROM YellowTripData
WHERE CAST(DropOffDate as Date) = @EndDate
GROUP BY PULocationID;
```

CREATE OR ALTER

```sql
CREATE OR ALTER FUNCTION SumLocationStats (
 @EndDate AS datetime = '1/01/2017')
 RETURNS TABLE AS RETURN
SELECT
  PULocationID as PickupLocation,
  COUNT(ID) AS RideCount,
  SUM(TripDistance) AS TotalTripDistance
FROM YellowTripData
WHERE CAST(DropOffDate AS Date) = @EndDate
GROUP BY PULocationID;
```

If you want to change a table valued function from a Multi Statement to an Inline or vice versa, you can't use ALTER.

```
-- Delete function
DROP FUNCTION dbo.CountTripAvgFareDay
```

```
-- Create CountTripAvgFareDay as Inline TVF instead of MSTVF
CREATE FUNCTION dbo.CountTripAvgFareDay(
  @Month char(2),
  @Year char(4)
) RETURNS TABLE AS RETURN (
  SELECT
    CAST(DropOffDate as date) as DropOffDate,
    COUNT(ID) as TripCount,
    AVG(FareAmount) as AvgFareAmt
  FROM YellowTripData
  WHERE
    DATEPART(month, DropOffDate) = @Month
    AND DATEPART(year, DropOffDate) = @Year
  GROUP BY CAST(DropOffDate as date));
```

Determinism improves performance

- A function is deterministic when it returns the same result given
  - the same input parameters
  - the same database state

```
SELECT
  OBJECTPROPERTY(
    OBJECT_ID('[dbo].[GetRideHrsOneDay]'),
    'IsDeterministic'
  )
```

```
+---+
| 1 |
+---+
```

```
SELECT
  OBJECTPROPERTY(
    OBJECT_ID('[dbo].[GetTomorrow]'),
    'IsDeterministic'
  )
```

```
+---+
| 0 |
+---+
```

Schemabinding

- Specifies the schema is bound to the database objects that it references
- Prevents changes to the schema if schema bound objects are referencing it

```
CREATE OR ALTER FUNCTION dbo.GetRideHrsOneDay (@DateParm date)
RETURNS numeric WITH SCHEMABINDING
AS
BEGIN
RETURN
(SELECT SUM(DATEDIFF(second, PickupDate, DropoffDate))/3600
FROM dbo.YellowTripData
WHERE CONVERT (date, PickupDate) = @DateParm)
END;
```

Let's practice!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER