Introduction to the course

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

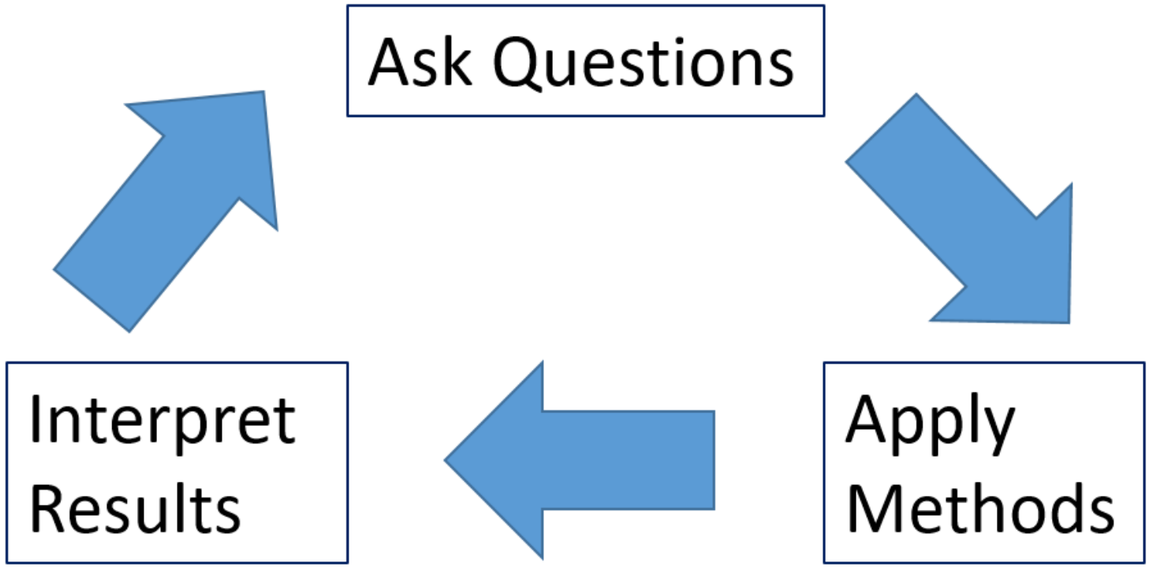
Meghan Kwartler
IT Consultant

Course objectives

- Perform temporal exploratory data analysis (EDA) using SQL date functions
- Create, update, execute User-Defined Functions
- Create, update, execute Stored Procedures
- Prerequisites
  - Introduction to SQL Server
  - Joining Data in SQL
  - Intermediate SQL Server

Temporal EDA

- Transactional datasets
  - CapitalBikeShare
  - YellowTripTaxi
- Exploratory Data Analysis (EDA) Process
  - Iterative
  - No specific checklist for EDA questions
  - Get curious!
  - Reduces re-work effort



SQL functions for EDA

```
-- CONVERT Syntax:
CONVERT ( data_type [ ( length ) ] , expression [ , style ] )
-- Returns expression based on data_type
```

```
-- DATEPART Syntax
DATEPART ( datepart , date )
-- Returns int
```

```
-- DATENAME Syntax
DATENAME ( datepart , date )
-- Returns nvarchar
```

```
-- DATEDIFF Syntax
DATEDIFF ( datepart , startdate , enddate )
-- Returns int; can't use datepart weekday value
```

```
-- datepart values = year, quarter, month, dayofyear, day, week, weekday, hour,
-- minute, second, microsecond, nanosecond
```

```
-- CONVERT
SELECT
  TOP 1 PickupDate,
  CONVERT (DATE, PickupDate) AS DateOnly
FROM YellowTripData
```

```
+-----------------------------+------------+
| PickupDate                  | DateOnly   |
|-----------------------------+------------|
| 2017-01-09 11:52:13.0000000 | 2017-01-09 |
+-----------------------------+------------+
```

```
-- DATEPART
SELECT
  TOP 3 COUNT(ID) AS NumberofRides,
  DATEPART(HOUR, PickupDate) AS Hour
FROM YellowTripData
GROUP BY DATEPART(HOUR, PickupDate)
ORDER BY COUNT(ID) DESC
```

```
+----------------------+
| NumberOfRides | Hour |
|----------------------+
| 616281        | 18   |
| 616281        | 19   |
| 540629        | 17   |
+---------------+------+
```

```
--DATENAME
SELECT
  TOP 3 ROUND(
    SUM(FareAmount),
    0
  ) as TotalFareAmt,
  DATENAME(WEEKDAY, PickupDate) AS DayofWeek
FROM YellowTripData
GROUP BY DATENAME (WEEKDAY, PickupDate)
ORDER BY SUM(FareAmount) DESC;
```

```
+--------------------------+
| TotalFareAmt | DayofWeek |
|--------------------------+
| 19026645     | Sunday    |
| 18749482     | Tuesday   |
| 17213351     | Thursday  |
+--------------+-----------+
```

```
--DATEDIFF
SELECT
  AVG(
    DATEDIFF(SECOND, PickupDate, DropOffDate)/ 60
  ) AS AvgRideLengthInMin
FROM YellowTripData
WHERE DATENAME(WEEKDAY, PickupDate) = 'Sunday';
```

```
+--------------------+
| AvgRideLengthInMin |
|--------------------+
| 13                 |
+--------------------+
```

Let's practice!

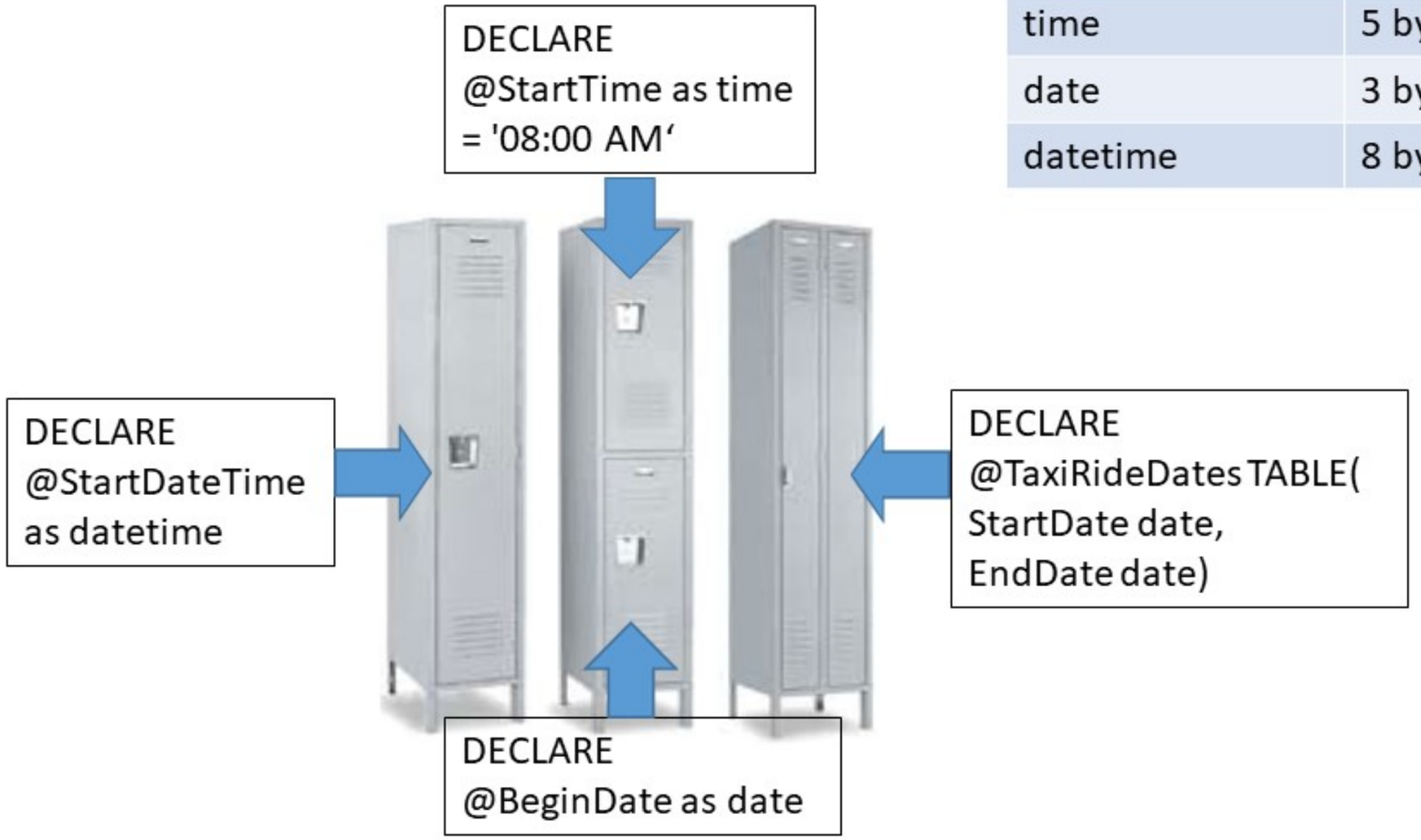
WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Variables for datetime data

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Meghan Kwartler
IT Consultant

DataType	Storage Size
time	5 bytes
date	3 bytes
datetime	8 bytes



```
-- DECLARE variable and assign initial value
 DECLARE @StartTime as time = '08:00 AM'
```

```
-- DECLARE variable and then SET value
DECLARE @StartTime AS time
SET @StartTime = '08:00 AM'
```

```
-- DECLARE variable then SET value
DECLARE @BeginDate as date
SET
  @BeginDate = (
    SELECT TOP 1 PickupDate
    FROM YellowTripData
    ORDER BY PickupDate ASC
  );
```

CASTing

```
-- CAST syntax
CAST ( expression AS data_type [ ( length ) ] )
-- Returns expression based on data_type
```

```
-- DECLARE datetime variable
-- SET value to @BeginDate and @StartTime while CASTing
DECLARE @StartDateTime as datetime
SET @StartDateTime = CAST(@BeginDate as datetime) + CAST(@StartTime as datetime)
```

```
-- DECLARE table variable with two columns
DECLARE @TaxiRideDates TABLE(
    StartDate date,
    EndDate date)
```

```
-- INSERT static values into table variable
INSERT INTO @TaxiRideDates (StartDate, EndDate)
SELECT '3/1/2018', '3/2/2018'
```

```
-- INSERT query result
INSERT INTO @TaxiRideDates(StartDate, EndDate)
SELECT DISTINCT
    CAST(PickupDate  as date),
    CAST(DropOffDate  as date)
    FROM YellowTripData;
```

Your turn to DECLARE, SET, CAST & INSERT!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Date manipulation

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER

Meghan Kwartler
IT Consultant

GETDATE

```
SELECT GETDATE()
```

```
+-------------------------+
| 2019-02-27 13:11:59.590 |
+-------------------------+
```

```
DECLARE @CurrentDateTime AS datetime
SET @CurrentDateTime = GETDATE()
SELECT @CurrentDateTime
```

```
+-------------------------+
| 2019-02-27 13:14:36.517 |
+-------------------------+
```

```
-- DATEADD Syntax:
DATEADD (datepart, number, date)
-- Returns expression based on data_type
```

```
-- One day after 2/27/2019
SELECT DATEADD(day, 1, '2/27/2019')
```

```
+-------------------------+
| 2019-02-28 00:00:00.000 |
+-------------------------+
```

DATEADD and GETDATE

```
-- Yesterday
SELECT DATEADD(d, -1, GETDATE())
```

```
+-------------------------+
| 2019-02-26 09:20:50.013 |
+-------------------------+
```

```
-- Yesterday's Taxi Passenger Count
SELECT SUM(PassengerCount)
FROM YellowTripData
WHERE CAST(PickupDate as date) = DATEADD(d, -1, GETDATE())
```

Remember DATEDIFF?

```
SELECT DATEDIFF(day, '2/27/2019', '2/28/2019')
```

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

```
SELECT DATEDIFF(year, '12/31/2017', '1/1/2019')
```

```
+---+
| 2 |
+---+
```

```
-- First Day of Current Week
SELECT DATEADD(week, DATEDIFF(week, 0, GETDATE()), 0)
```

```
-- First step
GETDATE()
```

```
+-------------------------+
| 2019-02-27 10:33:39.713 |
+-------------------------+
```

```
-- How many weeks between today and 1/1/1900?
SELECT DATEDIFF(week, 0, GETDATE())
```

```
+------+
| 6217 |
+------+
```

```
-- Add zero to the 6217nd week
SELECT DATEADD(week, DATEDIFF(week, 0, GETDATE()),0)
```

```
+-------------------------+
| 2019-02-25 00:00:00.000 |
+-------------------------+
```

Now it's time for you to manipulate some dates!

WRITING FUNCTIONS AND STORED PROCEDURES IN SQL SERVER