Advanced SQL

2018 Tribal Data Workshop

Joe Nowinski

The Plan

- Live demo 1:00 PM – 3:30 PM
- Follow along on GoToMeeting
- Optional practice session 3:45 PM – 5:00 PM
- Laptops available

What is SQL?

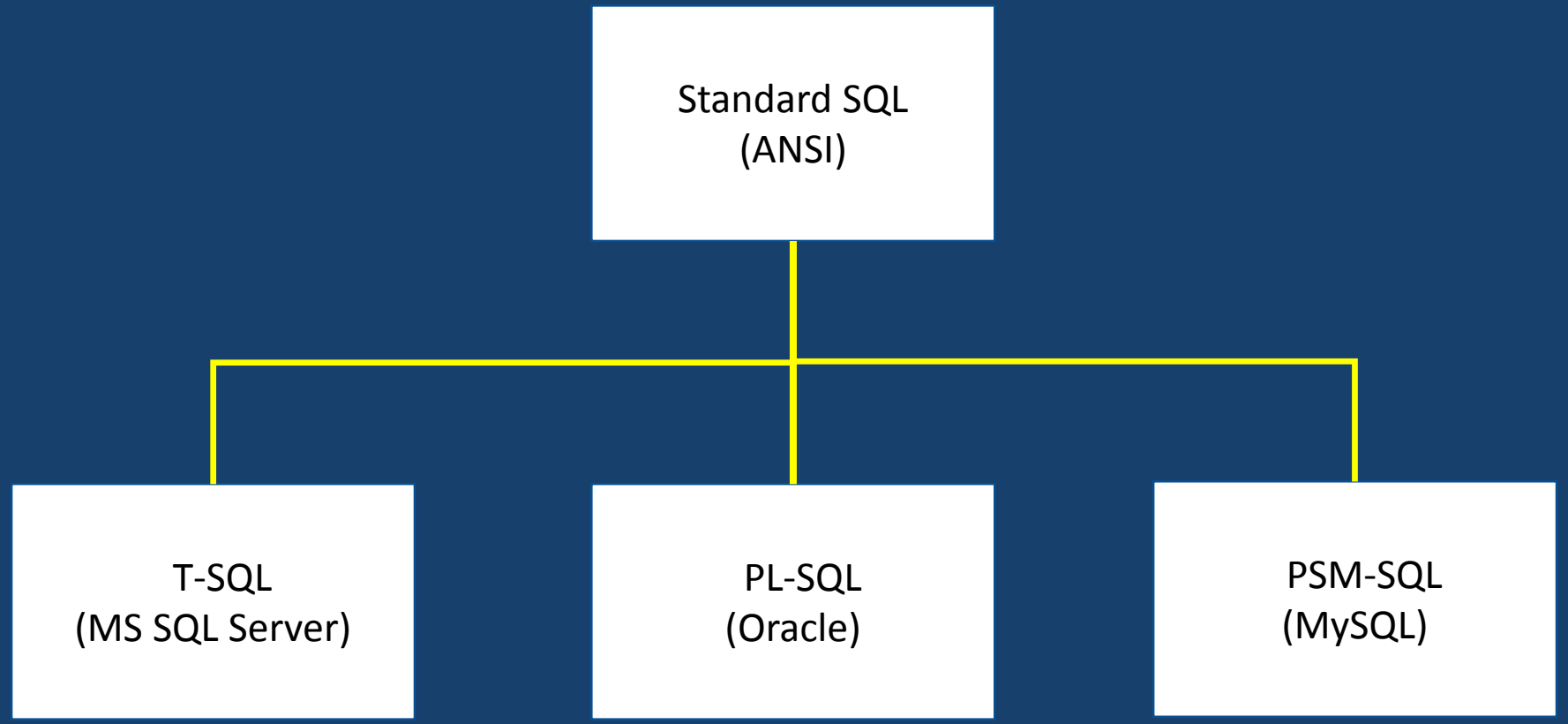
- **S**tructured **Q**uery **L**anguage
- Used to query and manipulate data in a database
- Developed at IBM
- Originally named SEQUEL

SQL Users

SQL is intended as a data base sublanguage for both the professional programmer and the more infrequent database user.

(Chamberlin & Boyce, 1974)

SQL Dialects



Basic SQL

SELECT
FROM
JOIN
WHERE
GROUP BY
ORDER BY

INSERT INTO
UPDATE
DELETE

Advanced SQL

- SQL DML Commands
  - DISTINCT, CASE, GROUP BY, PIVOT, UNION, OVER, MERGE
- SQL DDL Commands
  - Temp tables, views, stored procedures
  - Automated routines to import & cleanse field data
- SQL Programmatic Features
  - Variables, loops, and dynamic SQL

DML Commands

Data Manipulation Language

& functions

DISTINCT

- Specifies that only unique rows can appear in the result-set
- How many unique Management Weeks are in the table?

```
SELECT DISTINCT ManagementWeek
FROM SampleData
```

ManagementWeek	Species	TagCode
1	Chinook	3DD.1234567980
1	Chinook	3DD.1234567981
1	Steelhead	3DD.1234567990
2	Steelhead	3DD.1234567991
2	Steelhead	3DD.1234567992
2	Chinook	3DD.1234567982
2	Chinook	3DD.1234567983
2	Chinook	3DD.1234567985
3	Chinook	3DD.1234567986
3	Steelhead	3DD.1234567993

Aliases

- Used to give a table or a column a temporary name
- Help make column names more readable
- Only exists for the duration of the query

```
SELECT <column> AS <alias>
FROM <table>
```

```
SELECT <column>
FROM <table> AS <alias>
```

CASE

- Evaluates a list of conditions and returns one result
- Similar to IF-THEN-ELSE but used within a query

```
CASE WHEN SpeciesID = 1 THEN 'Chinook'
     WHEN SpeciesID = 3 THEN 'Steelhead'
     ELSE 'Unknown'
END
```

Scalar functions

- Operate on a single value and then return a single value
  - Conversion functions (CAST, TRY_CAST,)
  - Date/time functions (GETDATE, DAY, MONTH, YEAR, DATEPART)
  - Mathematical functions (ABS, EXP, LOG, POWER, ROUND, SQRT)
  - String functions (CONCAT, LEN, SUBSTRING, TRIM)

https://docs.microsoft.com/en-us/sql/t-sql/functions/functions

Example: SUBSTRING & CAST

How far is each PTAGIS site from Bonneville Dam?

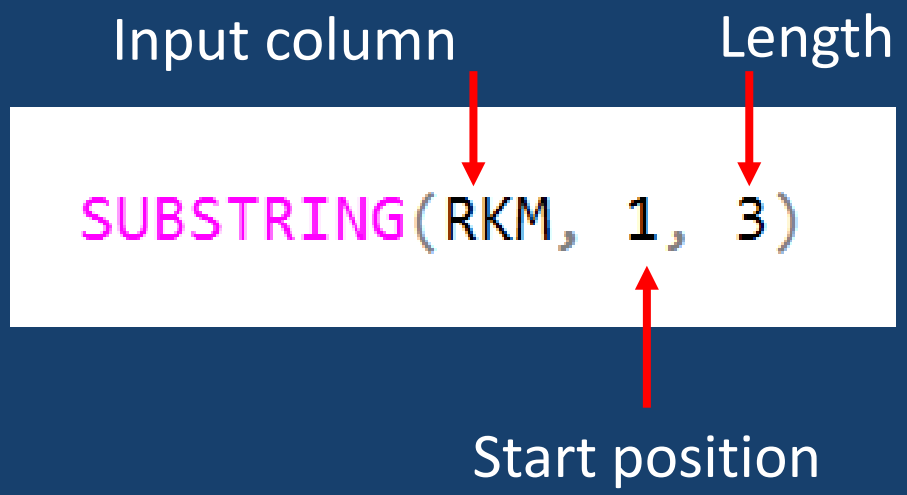
SiteCode	RKM
BO3	234
TD1	308
MC1	470
LMA	522.067
GRA	522.173
IR1	522.308.007

Problem: RKM is data type VARCHAR(27)

Example: SUBSTRING

- SUBSTRING extracts a substring from a string

SiteCode	RKM	
BO3	234	234
TD1	308	308
MC1	470	470
LMA	522.067	522
GRA	522.173	522
IR1	522.308.007	522



Example: CAST

- CAST converts an expression from one data type to another

SiteCode	RKM	
BO3	234	234
TD1	308	308
MC1	470	470
LMA	522.067	Error
GRA	522.173	Error
IR1	522.308.007	Error

Input column

New Data Type

```
CAST(RKM AS INT)
```

Example: Solution

```
SELECT SiteCode
     , RKM
     , SUBSTRING(RKM, 1, 3) AS 'FirstOrderRKM'
     , SUBSTRING(RKM, 5, 3) AS 'SecondOrderRKM'
     , SUBSTRING(RKM, 9, 3) AS 'ThirdOrderRKM'
     , CAST(SUBSTRING(RKM, 1, 3) AS INT)
         + CAST(SUBSTRING(RKM, 5, 3) AS INT)
         + CAST(SUBSTRING(RKM, 9, 3) AS INT) AS 'TotalRKM'
     , CAST(SUBSTRING(RKM, 1, 3) AS INT)
         + CAST(SUBSTRING(RKM, 5, 3) AS INT)
         + CAST(SUBSTRING(RKM, 9, 3) AS INT) - 234 AS 'DistanceFromBonn'
FROM PTAGIS_Sites
```

SiteCode	RKM	FirstOrderRKM	SecondOrderRKM	ThirdOrderRKM	TotalRKM	DistanceFromBonn
BO3	234	234			234	0
TD1	308	308			308	74
MC1	470	470			470	236
LMA	522.067	522	67		589	355
GRA	522.173	522	173		695	461
IR1	522.308.007	522	308	7	837	603

Aggregate functions

- Perform a calculation on a set of values and return a single value
- COUNT, SUM, AVG, MIN, MAX
- Typically used with the GROUP BY clause of the SELECT statement

QUICK REVIEW: GROUP BY

- Used to group the result-set by one or more columns

ManagementWeek	Species	TagCode
1	Chinook	3DD.1234567980
1	Chinook	3DD.1234567981
1	Steelhead	3DD.1234567990
2	Steelhead	3DD.1234567991
2	Steelhead	3DD.1234567992
2	Chinook	3DD.1234567982
2	Chinook	3DD.1234567983
2	Chinook	3DD.1234567985
3	Chinook	3DD.1234567986
3	Steelhead	3DD.1234567993

SampleData Table

```
SELECT Species
, COUNT(TagCode) AS 'FishCount'
FROM SampleData
GROUP BY Species
```

Species	FishCount
Chinook	6
Steelhead	4

Result-set

Example: GROUP BY & COUNT

How many of each species were sampled per week?

```
SELECT ManagementWeek, Species, COUNT(TagCode) AS 'FishCount'
FROM SampleData
GROUP BY ManagementWeek,  Species
ORDER BY ManagementWeek,  Species
```

ManagementWeek	Species	FishCount
1	Chinook	2
2	Chinook	3
3	Chinook	1
1	Steelhead	1
2	Steelhead	2
3	Steelhead	1

Example: PIVOT

- PIVOT rotates a table by turning the unique values from one column into multiple columns in the result-set
- Can be used with aggregate functions

ManagementWeek	Species	FishCount
1	Chinook	2
2	Chinook	3
3	Chinook	1
1	Steelhead	1
2	Steelhead	2
3	Steelhead	1

GROUP BY Result-set

ManagementWeek	Chinook	Steelhead
1	2	1
2	3	2
3	1	1

PIVOT Result-set

Example: PIVOT

Distinct values transformed into columns

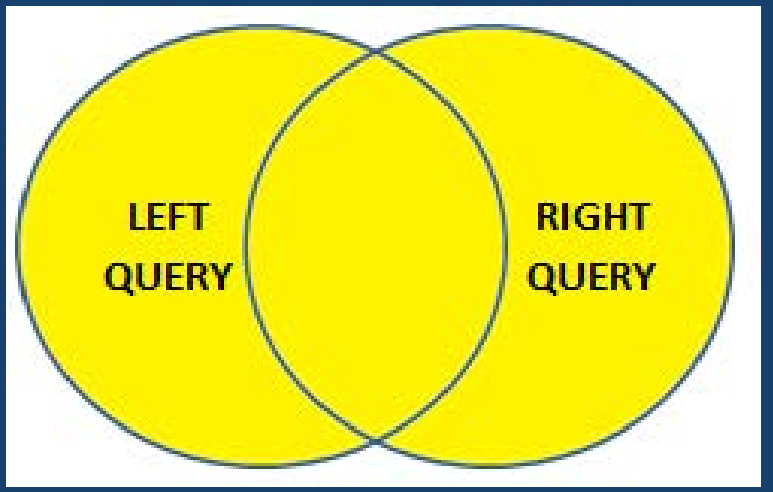
ManagementWeek	Species	TagCode
1	Chinook	3DD.1234567980
1	Chinook	3DD.1234567981
1	Steelhead	3DD.1234567990
2	Steelhead	3DD.1234567991
2	Steelhead	3DD.1234567992
2	Chinook	3DD.1234567982
2	Chinook	3DD.1234567983
2	Chinook	3DD.1234567985
3	Chinook	3DD.1234567986
3	Steelhead	3DD.1234567993

```
SELECT ManagementWeek, Chinook, Steelhead
FROM SampleData
PIVOT
    (
        COUNT(TagCode)
        FOR Species
        IN ([Chinook], [Steelhead])
    )
    AS P
```

Distinct values to transform defined here

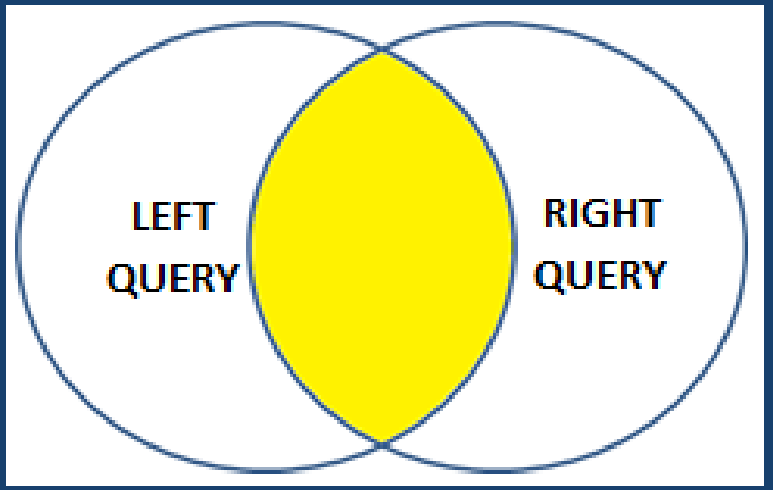
UNION

- Used to combine the result-set of two of more queries
- Data type, order, and number of columns must match
- UNION filters out duplicate rows
- UNION ALL returns all rows



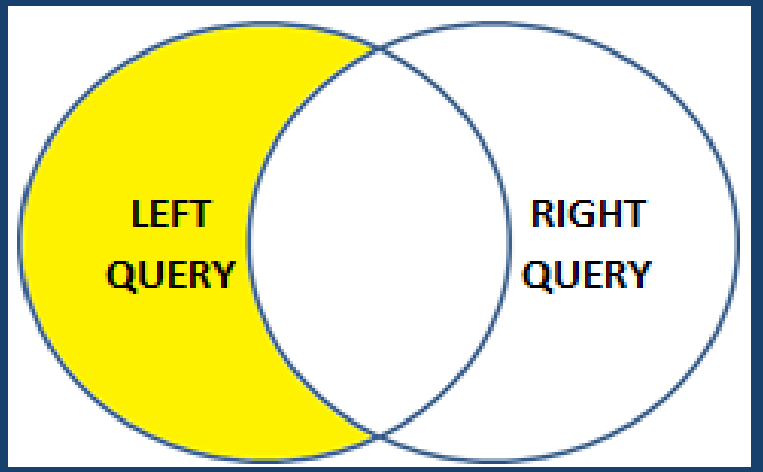
INTERSECT

- Returns common records from left and right queries
- Data type, order, and number of columns must match
- Filters out duplicate rows
- Similar to INNER JOIN



EXCEPT

- Returns records from left query that are not included in right query
- Data type, order, and number of columns must match
- Filters out duplicate rows
- Similar to NOT IN



Example: GROUP BY & UNION

ManagementWeek	Species	FishCount
1	Chinook	2
2	Chinook	3
3	Chinook	1
1	Steelhead	1
2	Steelhead	2
3	Steelhead	1
1	NULL	3
2	NULL	5
3	NULL	2
NULL	Chinook	6
NULL	Steelhead	4
NULL	NULL	10

GROUP BY ManagementWeek, Species

GROUP BY ManagementWeek

GROUP BY Species

No GROUP BY Clause

GROUPING SETS

- A GROUP BY clause that uses GROUPING SETS can generate a result set equivalent to that generated by a UNION ALL of multiple simple GROUP BY clauses.

ManagementWeek	Species	FishCount
1	Chinook	2
2	Chinook	3
3	Chinook	1
1	Steelhead	1
2	Steelhead	2
3	Steelhead	1
1	NULL	3
2	NULL	5
3	NULL	2
NULL	Chinook	6
NULL	Steelhead	4
NULL	NULL	10

```
GROUP BY
    GROUPING SETS
    (
        (ManagementWeek, Species)
        ,(ManagementWeek)
        ,(Species)
        ,()
    )
```

OVER

- The OVER clause defines the partitioning of rows in a result-set
- Alternative to GROUP BY

```
SELECT SiteID, SampleDate, Temperature
, AVG(Temperature) OVER (PARTITION BY SiteID) AS 'AvgTemp'
FROM TemperatureData
```

OVER

- Allows functions to be applied to targeted groups (windows) of rows
- Three arguments:
  - PARTITION BY divides result-sets into partitions
  - ORDER BY controls ordering of rows within a partition
  - ROWS specifies starting and end points of windows with a partition

Video on Window functions & OVER clause

MERGE

- Performs insert, update, and/or delete operations on a target table based on the results of a join with a source table

Target Table

SpeciesID	SampleDate	AdClip	ForkLength	TagCode
3	10/2/2017	1	56	3DD.0077BA7F2B
3	10/3/2017	1	52.2	3DD.0077BA625B
3	10/3/2017	0	60	3DD.0077BAA684

Common Field

Source Table

SpeciesID	SampleDate	AdClip	ForkLength	TagCode
1	10/2/2017	1	56	3DD.0077BA7F2B
1	10/3/2017	1	52.2	3DD.0077BA625B
1	10/3/2017	0	60	3DD.0077BAA684
3	10/6/2017	1	57	3DD.0077BA3811
3	10/6/2017	1	59.5	3DD.0077BA411F
3	10/6/2017	1	60	3DD.0077BA8D86

MERGE

- MERGE . . . ON Source.TagCode = Target.TagCode
- Matching rows in Source table UPDATE rows in Target table
- Non-matching rows in Source table INSERT INTO the Target table
- Non-matching rows in the Target table DELETED from Target table

DDL Commands

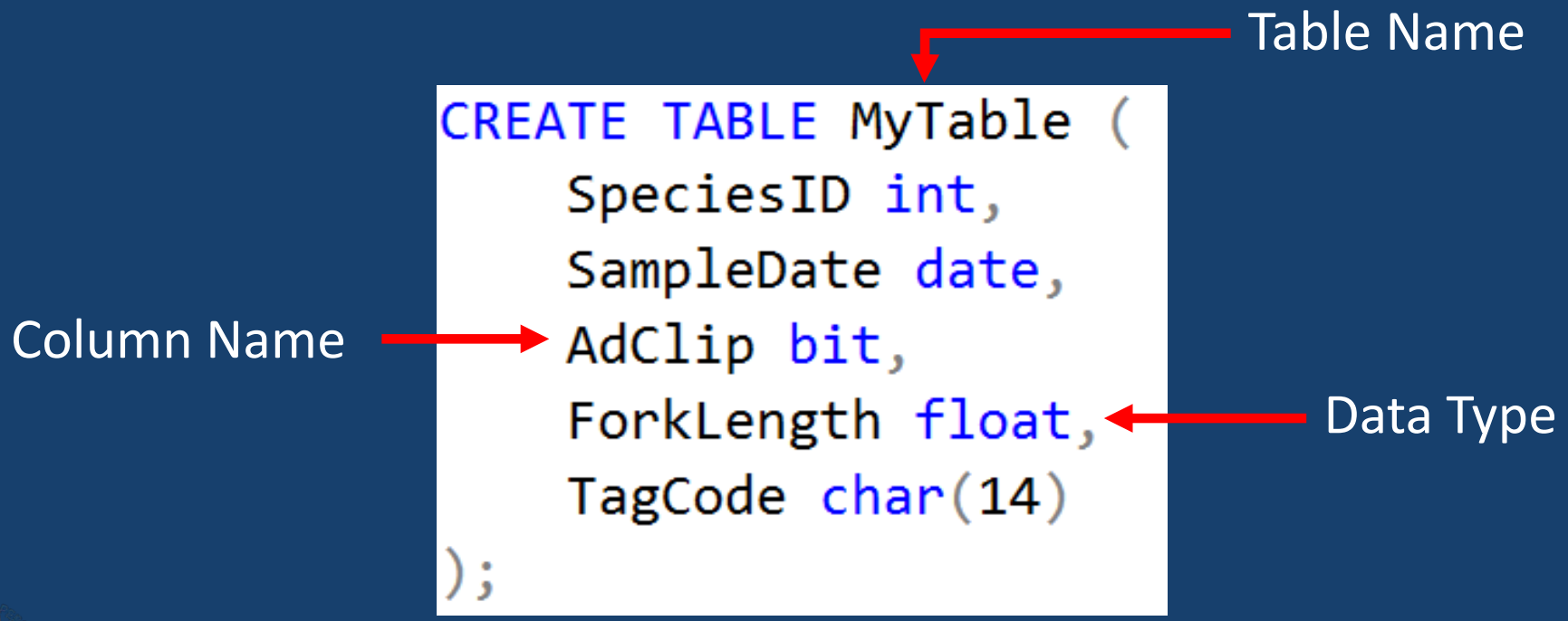
Data Definition Language

DDL Commands

- Data Definition Language (DDL)
- A special group of SQL keywords used to CREATE, ALTER or DROP database objects (e.g., tables, views, stored procedures)
- Alternative to creating database objects with point & click method
- Security restricted in some cases*

CREATE

- Use the CREATE TABLE command to build tables



ALTER

- Use the ALTER TABLE command to add, alter, or drop columns

```
ALTER TABLE FishData
ADD DorsalCondition BIT;
```

```
ALTER TABLE FishData
ALTER COLUMN DorsalCondition INT;
```

```
ALTER TABLE FishData
DROP COLUMN DorsalCondition;
```

DROP

- Use the DROP TABLE command to delete tables

```
DROP TABLE dbo.MyTable;
```

- Use IF with the OBJECT_ID function to check if a table exist

```
IF OBJECT_ID('dbo.MyTable', 'U') IS NOT NULL
DROP TABLE dbo.MyTable;
```

T-SQL System Object Type Codes

Temporary Tables

- Act like regular tables but only exist during a session
- Live in the TempDb (a system database managed by SQL Server)
- CREATE TABLE #TableName
- Users do not need CREATE/DROP table permissions

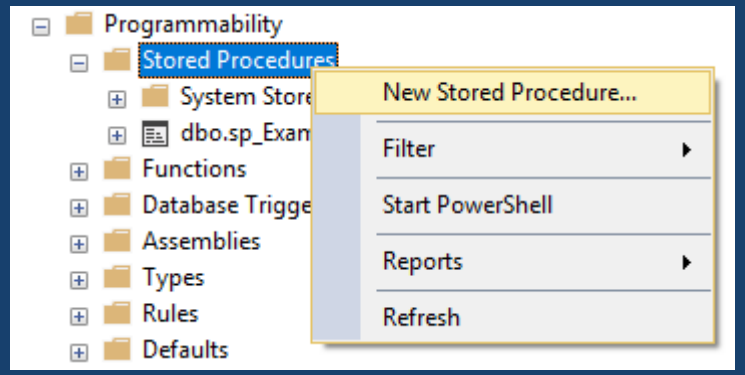
Views

- A virtual table whose columns and rows are defined by a query
- Allow DBA to customize how users view data
- Allow user access to data without granting permissions to the underlying base tables.

Stored Procedures

- SQL query or routine that is saved inside a database
- Convenient alternative to writing the same query over and over again
- Can be executed manually in SSMS, set to run automatically on a schedule, or called by an external application

Creating Stored Procedures



1. Right click Stored Procedures
2. Select New Stored Procedure
3. Add SQL code to the template
4. Click Execute to save

CREATE PROCEDURE Template

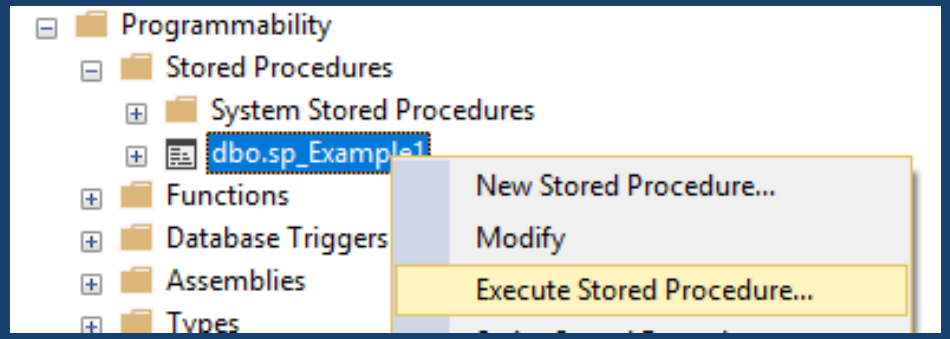
```
CREATE PROCEDURE [dbo].[sp_Example1]
    -- This stored procedure does not have any parameters

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT * FROM FishData WHERE ForkLength BETWEEN 50 AND 60;
END
```

Executing Stored Procedures

Point & Click

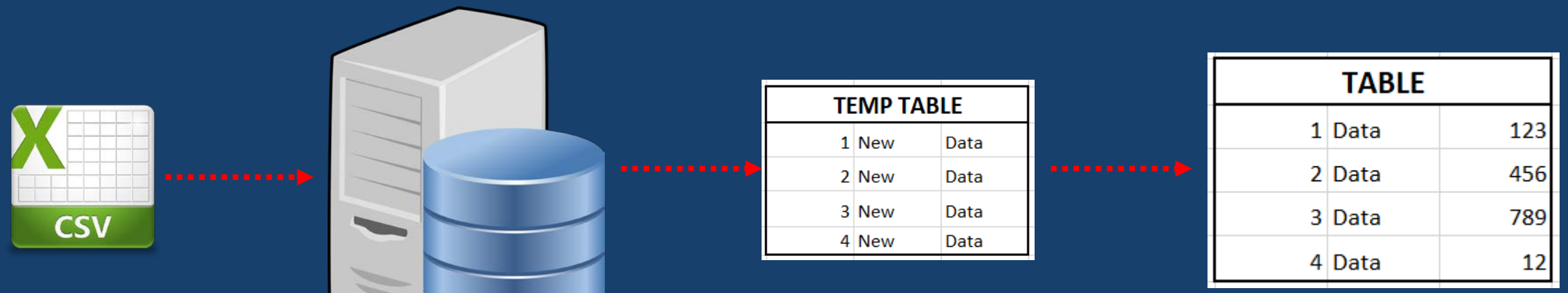


1. Right click stored procedure name
2. Click Execute Stored Procedure
3. Click OK

Or run SQL code

```
EXECUTE sp_Example1;
```

Example: Automated Import Process



File transfer
to DB server

BULK IMPORT
into Temp Table

Data cleanse &
MERGE into
Destination Table

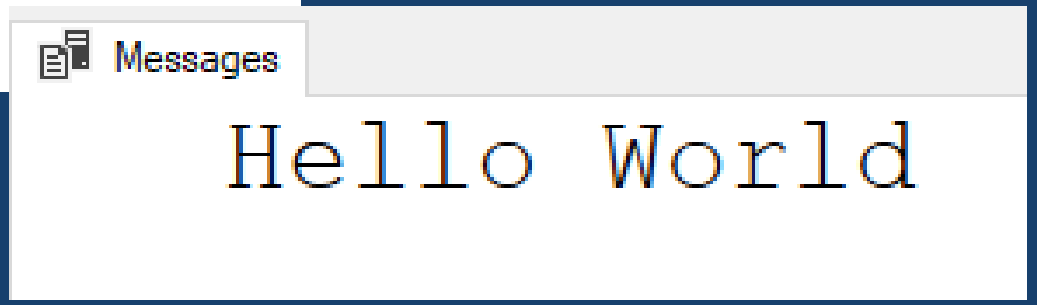
Programmatic Features

Variables, Loops, & Dynamic SQL

Variables

- An object that can hold a single data value of a specific data type
- DECLARE @Name Data Type
- Use the SET keyword to initialize

```
DECLARE @Message AS varchar(20);
SET @Message = 'Hello World';
PRINT @Message;
```



Variables

```
DECLARE @MaxValue AS FLOAT, @MinValue as FLOAT;
SET @MaxValue = 60;
SET @MinValue = 50;



SELECT *
FROM FishData
WHERE ForkLength BETWEEN @MinValue AND @MaxValue;
```

Variables (Input parameters)

60

50

```
CREATE PROCEDURE [dbo].[sp_Example2]
    -- Add the parameters for the stored procedure here
    @MinValue FLOAT
    ,@MaxValue FLOAT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here

    SELECT *
    FROM FishData
    WHERE ForkLength BETWEEN @MinValue AND @MaxValue
    ORDER BY ForkLength DESC;
```

Variables (output parameters)

4

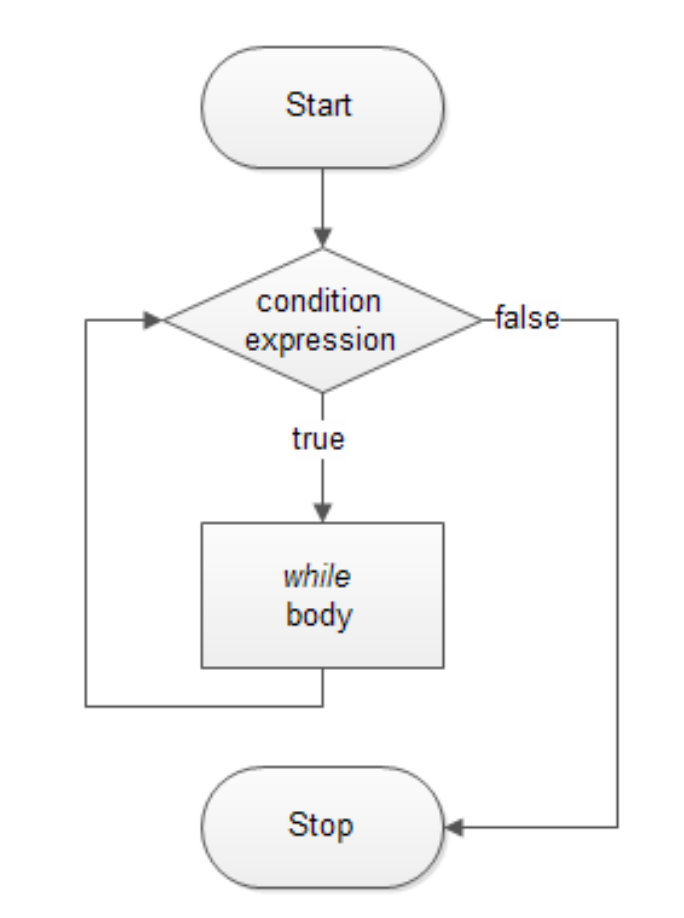
```
CREATE PROCEDURE [dbo].[sp_Example3]
    -- Add the parameters for the stored procedure here
      @MinValue FLOAT
     ,@MaxValue FLOAT
     ,@FishCount INT OUTPUT

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.

    -- Insert statements for procedure here
    SET @FishCount = (SELECT COUNT(Species) FROM FishData
    WHERE ForkLength BETWEEN @MinValue AND @MaxValue);

END
```

WHILE LOOP




```
DECLARE @i AS INT, @iMAX AS INT

SET @i = 1
SET @iMAX = 10

WHILE @i <= @iMAX
BEGIN

    PRINT @i
    SET @i = @i + 1

END
```

Dynamic SQL

- Dynamic SQL is a programming technique that enables you to build SQL statements dynamically at runtime

```
DECLARE @SQL VARCHAR(1000)
, @MaxValue VARCHAR(3)
, @MinValue VARCHAR(3)

SET @MinValue = 50
SET @MaxValue = 60

SET @SQL = 'SELECT * FROM FishData WHERE ForkLength BETWEEN '
            +  @MinValue + ' AND ' + @MaxValue

EXEC(@SQL)
```

Example: Dynamic PIVOT table

- PIVOT tables require hard coding distinct values

```
SELECT ManagementWeek, [1.1], [1.2], [1.3]
FROM SampleData2
PIVOT
(
    COUNT(Age)
    FOR Age
    IN ([1.1], [1.2], [1.3])
)
AS P
```

ManagementWeek	Species	Age
1	Chinook	1.1
1	Chinook	1.1
2	Chinook	1.1
2	Chinook	1.2
3	Chinook	1.2
3	Chinook	1.2
3	Chinook	1.2
4	Chinook	1.3
4	Chinook	1.3
4	Chinook	1.3

Example: Dynamic PIVOT table

- Dynamic SQL can be used to build a column list at run time
- Step 1: Select DISTINCT age values into a table
- Step 2: Use a loop to build a column list string
- Step 3: Add column list string to PIVOT query
- Step 4: Check syntax & execute dynamic SQL statement

Example: Dynamic PIVOT table

Step 1: Select DISTINCT age values into a table

```
IF OBJECT_ID('Ages', 'U') IS NOT NULL
DROP Table Ages

CREATE TABLE Ages
    (
        ID INT IDENTITY(1,1)
        , Age VARCHAR(8)
    )

INSERT INTO Ages (Age)
SELECT DISTINCT Age FROM SampleData2
```

Example: Dynamic PIVOT table

Step 2: Use a loop to build a column list string

```
DECLARE @i AS INT, @iMAX AS INT, @Age AS VARCHAR(8), @AgeList AS VARCHAR(100) = ''

SET @i = 1
SET @iMAX = (SELECT MAX(ID) FROM Ages)

WHILE @i <= @iMAX
BEGIN

    SET @Age =  (SELECT Age FROM Ages WHERE ID = @i)
    SET @Age = '[' + @Age + '],'
    SET @AgeList = @AgeList + @Age
    SET @i = @i + 1

END
```

Example: Dynamic PIVOT table

Step 3: Add column list string to PIVOT query string

```
DECLARE @SQL AS VARCHAR(500)
SET @SQL ='SELECT ManagementWeek,' +  @AgeList +
' FROM
    (
        Select ManagementWeek, Age FROM SampleData2
    ) AS T1
PIVOT
(
    COUNT(Age)
    FOR Age
    IN ('  +  @AgeList + ')
)
AS P'
```

Example: Dynamic PIVOT table

Step 4: Check syntax & execute dynamic SQL statement

```
PRINT @SQL
```

Messages

```
(7 rows affected)
SELECT ManagementWeek,[0.3],[1.1],[1.2],[1.3],[1.4],[2.3],[2.4] FROM
    (
        Select ManagementWeek, Age FROM SampleData2
    ) AS T1
PIVOT
(
    COUNT(Age)
    FOR Age
    IN ([0.3],[1.1],[1.2],[1.3],[1.4],[2.3],[2.4])
)
AS P
```

Links

- Original SEQUEL Whitepaper
- W3Schools.com Intro to Basic SQL
- PRAGIM Technologies SQL Video Tutorials
- Microsoft T-SQL Reference

Links

- Microsoft T-SQL Reserved Keywords
- Microsoft BCP Utility Info
- MS Access SQL vs Standard ANSI SQL