Introduction to T-SQL Queries

Kathi Kellenberger

Redgate Software

Kathi Kellenberger

- **Editor and Advocate Redgate Software**

- /Kathikellenberger
- @auntkathi
- http://auntkathisql.com
- Kathi.Kellenberger@red-gate.com

- Lifelong learner
- Teacher

- Co-leader of Data Platform Women in Technology Group
- Data Platform MVP

Agenda

- Class 1
  - Module 1: Introduction
  - Module 2: Simple select statements
  - Module 3: Filtering
- Class 2
  - Module 4: Expressions
  - Module 5: Joining
- Class 3
  - Module 5: Joining (Continued)
  - Module 6: Grouping
- Class 4
  - Module 7: Subqueries
  - Module 8: UNION

CLASS MATERIALS

- https://github.com/KathiKellenberger/CoderGirlDataAnalysis
  - Slides
  - Demos
  - Resources
- Students should install Azure Data Studio and connect to
  - sqlprojects.com,2433
  - Student
  - Madison18*
  - Instructions will be given in class

Schedule

- Lunch around noon for 30 minutes
- Take a break before or after lab
- Done at 3pm or when we get through Module 4

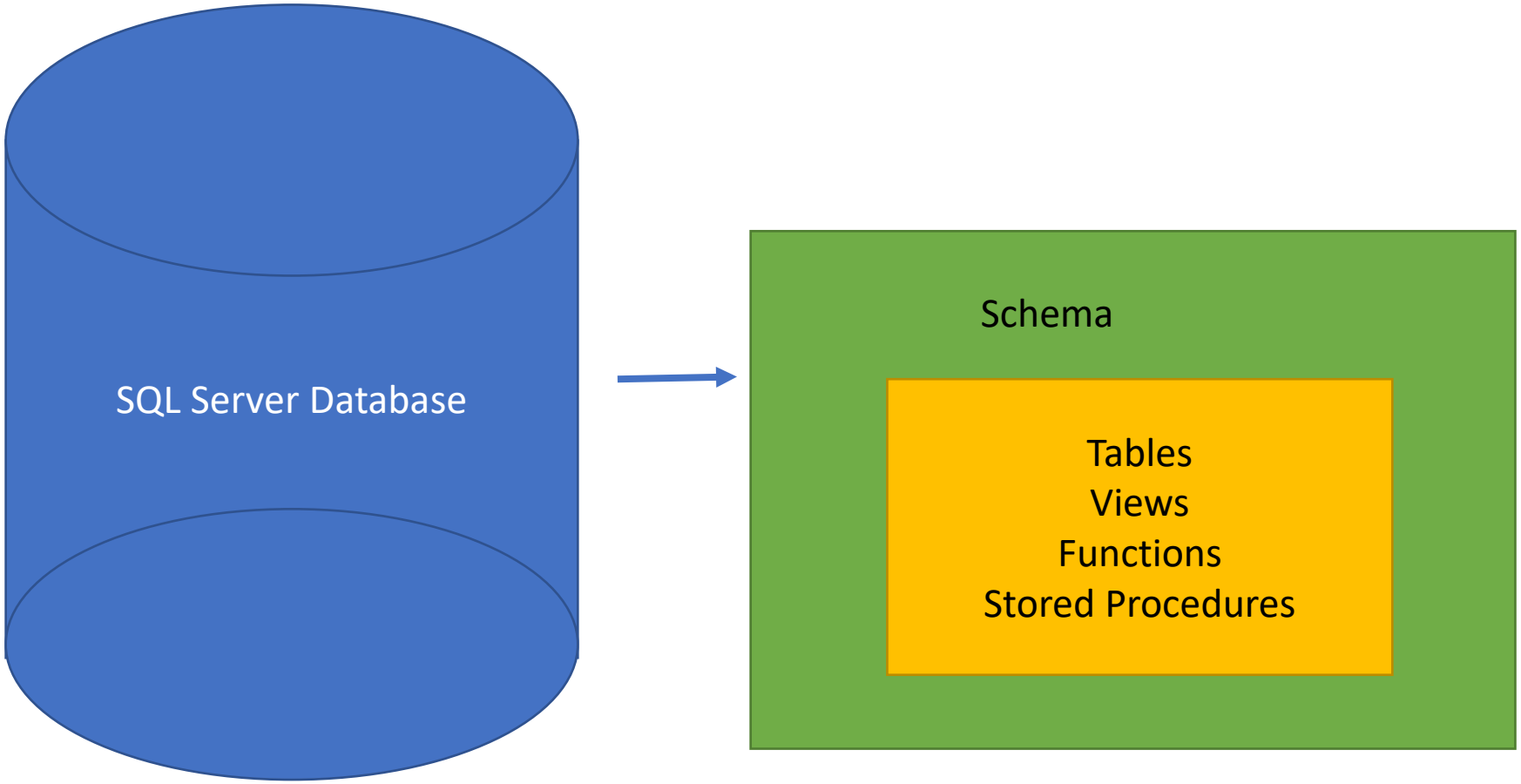
Module 1:
Introduction

What’s a database?

Database

A database is an organized collection of data. It is the collection of schemas, tables, queries, reports, views and other objects. The data are typically organized to model aspects of reality in a way that supports processes requiring information, such as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Database - Wikipedia
https://en.wikipedia.org/wiki/Database


SQL Server Database
Schema
Tables
Views
Functions
Stored Procedures

Tables

UsedCars					
ID	**Make**	**Model**	**Type**	**Year**	**Color**
1	Chevrolet	Malibu	Passenger car	2015	Blue
2	Hyundai	Sonata	Passenger car	2011	Silver
3	Chrysler	Pacifica	Minivan	2017	White
4	Toyota	Prius	Hybrid car	2013	White
5	Hyundai	Elantra	Passenger car	2015	Blue
6	Chevrolet	Silverado	Truck	2013	Red

```
SELECT *
FROM UsedCars
WHERE Make = ‘Hyundai’;
```

T-SQL

- SQL = Structured Query Language
- T-SQL = Transact SQL
- Each vendor has own version
- The basics are the same

PRINT, GO, USE, and comments

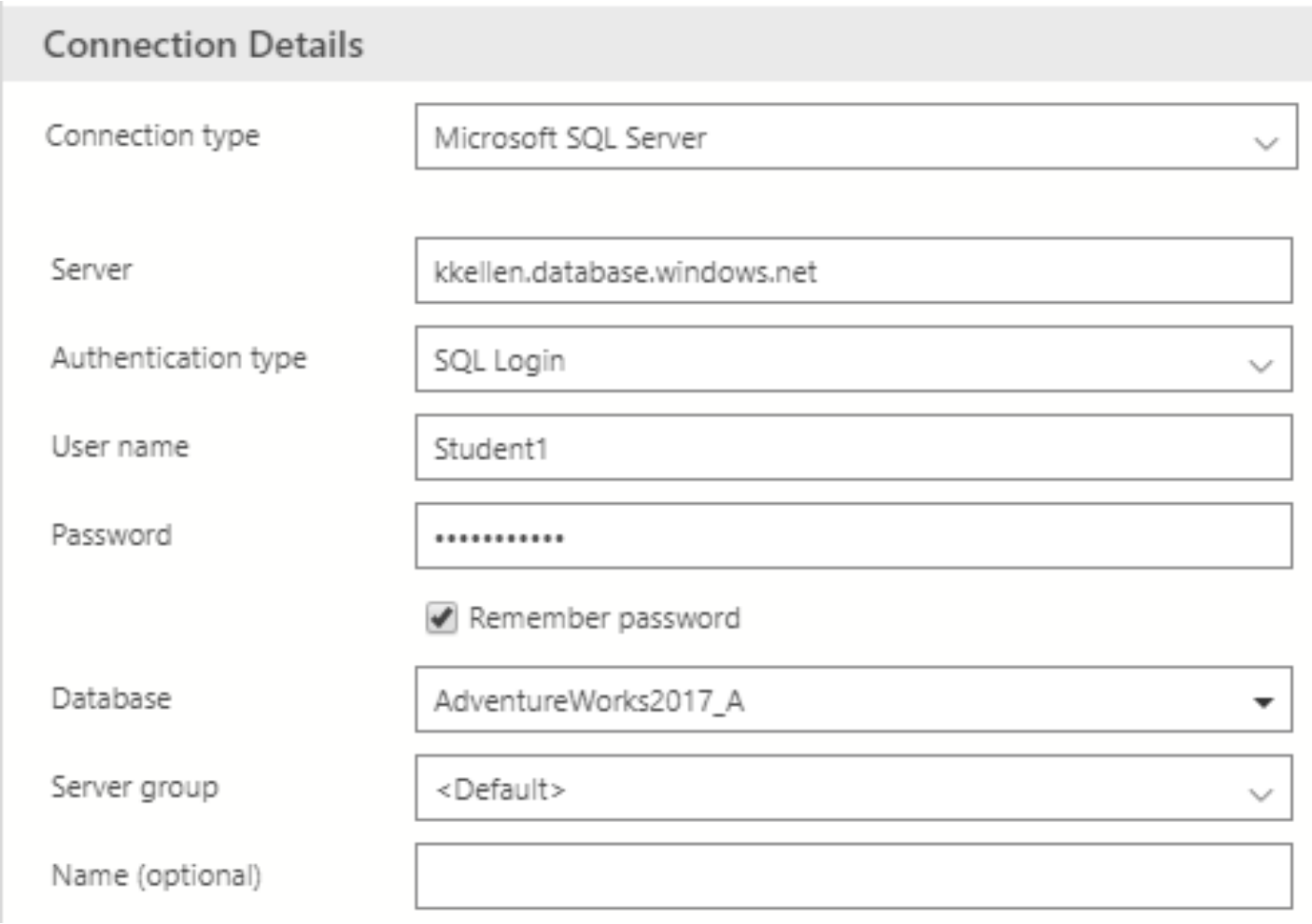
- Print displays a message
- GO is a batch separator
- USE – switch databases (will not work with our Azure dbs)
- -- (two dashes) for a one-line comment
- /* */ for multi line comments
- Use a tick mark aka single quote around strings or dates
- Use a semi-colon at end of statements

Accounts

Login	Password
Student1	Vermilion1*
Student2	Alexander1*
Student3	Washington1*
Student4	Greyside1*

Get started

- Connection
  - Kkellen.database.windows.net
  - Student1 with pw Vermilion1*
  - SQL Authentication
  - Type in a Database
    - AdventureWorks2017_A
    - AdventureWorks2017_B
    - AdventureWorks2017_C
- You can continue to use the databases after today.
- This is Azure's database as a service



Demo 1: Getting around in Azure Data Studio

Lab

- Complete Module 1 Lab 1

- Start back up at

Module 2:
Simple SELECT statements

SELECT

- Keyword for retrieving data from a database
- Return a list of columns or expressions
- Syntax

```
SELECT <expr1>[,<expr2>,<expr3>,…]
```

FROM

- The table where the data can be found
- Syntax

```
SELECT *
FROM <schema>.<table>

SELECT <expr1>[,<expr2>,<expr3>,…]
FROM <schema>.<table>
```

- The schema is often “dbo”
- You join tables together in the FROM clause, but you’ll learn about that in a later module (tomorrow!)

Aliases

- Give a name to an expression or table
- Syntax

```
SELECT <expr1> AS Name1
FROM <tablename> AS tbl

SELECT <expr1> AS [The name]

SELECT <expr1> AS “The name”
```

TOP

- Return a number of rows or a percent of rows
- Syntax

```
SELECT TOP(n) <expr1>[,<expr2>,<expr3>,…]
FROM <schema>.<table>

SELECT TOP(n) PERCENT <expr1>[,<expr2>,<expr3>,…]
FROM <schema>.<table>
```

DISTINCT

- Return a unique set of rows
- Syntax

```
SELECT DISTINCT <expr1>[,<expr2>,<expr3>,…]
FROM <schema>.<table>
```

Demo: SELECT FROM

Lab

- Complete Module 2 Lab 1

- Start at 9:50
- 13 minutes
- In lab info, label the parts to make it easier

Ordering data

- Use the ORDER BY clause
- One or more columns or expressions
- Ascending by default
- Use DESC to reverse order

Demo: ORDER BY

Lab

- Complete Module 2 Lab 2

Module 3:
Filtering

WHERE

- Basic Syntax

```
SELECT <expr1>[,<expr2>,<expr3>,…]
FROM <schema>.<table>
WHERE <expr5> = <expr6>
ORDER BY <expr1>
```

- Dates example

```
SELECT SalesOrderID, ShipDate
FROM Sales.SalesOrderHeader
WHERE ShipDate >= '2011-06-07' and ShipDate < '2011-06-08’
```

Operators

- =, <>, !=
- <, >, <=, >=
- BETWEEN
- LIKE (with wildcards %,_ and more)
- IN
- AND, OR for multiple expressions
- NOT
- Parentheses to enforce logic

Demo: The WHERE clause

Lab

- Complete Module 3 Lab 1

Working with NULL

- Unknown
- Can’t compare anything to NULL
- When trying to compare to NULL, the row is not returned
- Use ISNULL or COALESCE to replace the NULL
- Use IS NULL or IS NOT NULL to compare

Demo: NULL

Lab

- Complete Module 3 Lab 2

Module 4:
Expressions

What’s in an expression?

- Column, really anything
- Column1 + Column2
- Concatenating strings

  <string1> + <string2>
- Math

  <number> <operator> <number>

- Lots of built-in functions!

Functions

- CAST and CONVERT – change a data type
- ISNULL and COALESCE – replace NULL

Demo: Expressions

Lab

- Complete Module 4 Lab 1

String functions

- RTRIM, LTRIM, TRIM remove spaces
- LEFT, RIGHT return a number of characters
- LEN, DATALENGTH return the length
- CHARINDEX find a string
- SUBSTRING return part of a string
- REVERSE returns the string backwards
- UPPER, LOWER returns all upper or lower case
- REPLACE replace part of a string

Demo: String functions

Lab

- Complete Module 4 Lab 2

Working with Dates

- GETDATE, SYSDATETIME returns the server date
- DATEADD adds a time period to a date
- DATEDIFF finds the difference between two dates
- DATENAME, DATEPART returns part of a date
- DAY, MONTH, YEAR returns part of a date
- CONVERT, FORMAT formatting dates

Demo: Working with dates

Lab

- Complete Module 4 Lab 3

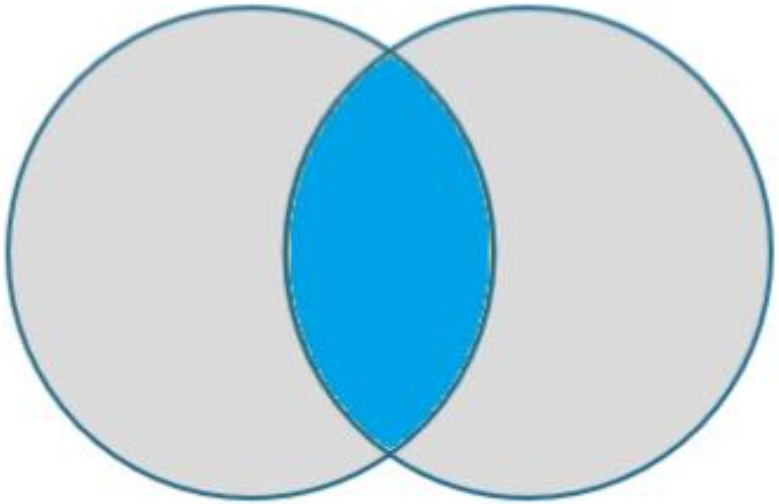
Module 5:
Joining Tables

INNER JOIN

- The columns from two tables where there is a match on a key
- Syntax

SELECT <table1>.<col1>,<table2>.<col2>

FROM <table1> [INNER] JOIN <table2>
ON <table1>.<col1> = <table2>.<col1>



Old join syntax: Comma join (Don’t use!)

```
SELECT Col1, Col1
FROM table1, table2
Where table1.col1 = table2.col1
```

- Used more often by Oracle developers than SQL Server devs

INNER JOIN

Customer	
CustomerID (Primary Key)	**Name**
1	John
2	Sharon
3	Dana
4	Fox

Sale		
SaleID (Primary Key)	**CustomerID (Foreign Key)**	**Amt**
1	3	100
2	1	200
3	3	75
4	3	90
5	1	100

Query results			
SaleID	**CustomerID**	**Name**	**Amt**
1	3	Dana	100
2	1	John	200
3	3	Dana	75
4	3	Dana	90
5	1	John	100

Demo: INNER JOIN

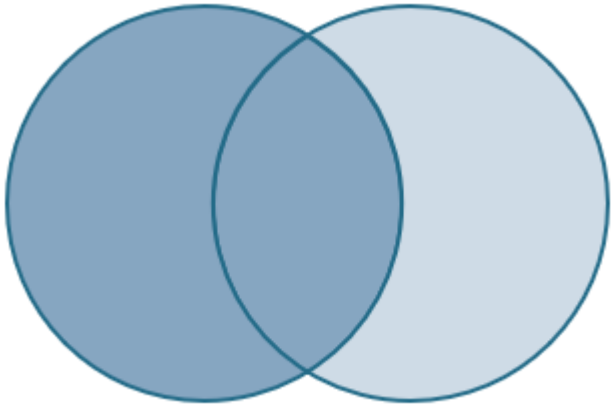
Lab

- Complete Module 5 Lab 1

LEFT OUTER JOIN

- All the rows from first table even if they don’t match
- Once you start down the left path, continue left
- Syntax

```
SELECT <table1>.<col1>, <table2><col2>
FROM <table1>
LEFT [OUTER] JOIN <table2>
ON <table1>.<col1> = <table2>.<col2>
```



LEFT OUTER JOIN

Customer

CustomerID	Name
1	John
2	Sharon
3	Dana
4	Fox

Sale

SaleID	CustomerID	Amt
1	3	100
2	1	200
3	3	75
4	3	90
5	1	100

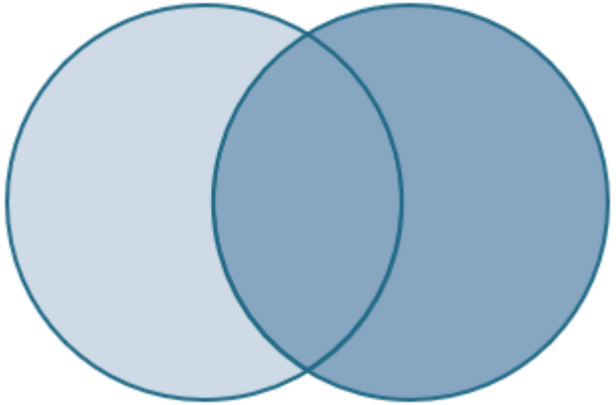
Query results

SaleID	CustomerID	Name	Amt
1	3	Dana	100
2	1	John	200
3	3	Dana	75
4	3	Dana	90
5	1	John	100
NULL	2	Sharon	NULL
NULL	4	Fox	NULL

RIGHT OUTER JOIN

- All the rows from second table even if they don’t match
- Not used as much
- Syntax

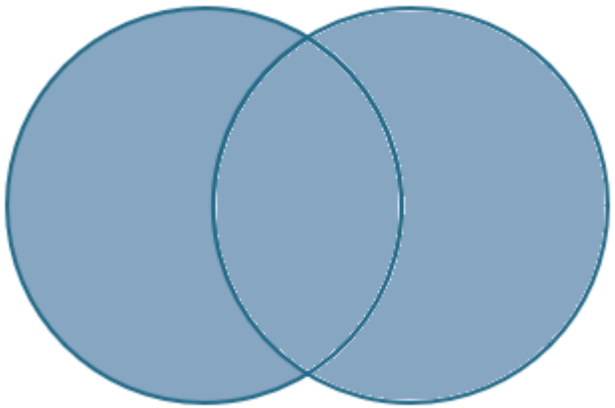
```
SELECT <table1>.<col1>, <table2><col2>
FROM <table1>
RIGHT [OUTER] JOIN <table2>
ON <table1>.<col1> = <table2>.<col2>
```



FULL OUTER JOIN

- All the rows from both tables even if they don’t match
- Rarely used
- Syntax

```
SELECT <table1>.<col1>, <table2><col2>
FROM <table1>
FULL [OUTER] JOIN <table2>
ON <table1>.<col1> = <table2>.<col2>
```



Demo: OUTER JOIN

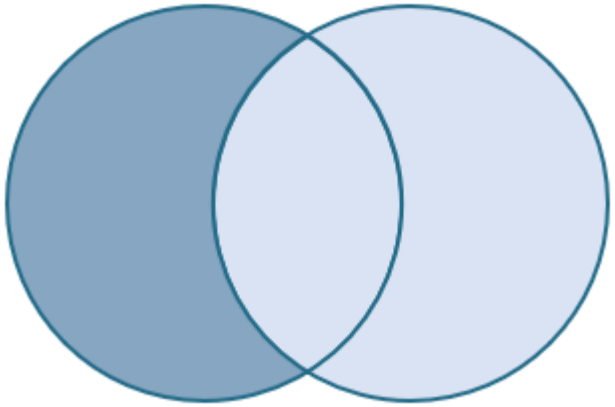
Lab

- Complete Module 5 Lab 2

LEFT OUTER JOIN with NULL RIGHT Filter

- Use to find rows that don’t match
- Filter on a key from the table on the right
- Syntax

```
SELECT <table1>.<col1>,<table2>.<col2>
FROM <table1>
LEFT [OUTER] JOIN <table2>
ON <table1>.<col1 > = <table2>.<col1>
WHERE <table2>.<col1> IS NULL
```



LEFT OUTER JOIN with NULL right table filter

Customer	
CustomerID	**Name**
1	John
2	Sharon
3	Dana
4	Fox

Sale		
SaleID	**CustomerID**	**Amt**
1	3	100
2	1	200
3	3	75
4	3	90
5	1	100

Query results			
SaleID	**CustomerID**	**Name**	**Amt**
NULL	2	Sharon	NULL
NULL	4	Fox	NULL

Demo: OUTER JOIN with FILTER

Lab

- Complete Module 5 Lab 3