# Data Manipulation Language

- SELECT
  - SELECT INTO
  - Temp Tables
- INSERT
  - Basic Insert
  - Using OpenRowset
  - External Data into SQL Server
- UPDATE
- DELETE
- TRUNCATE TABLE

# SELECT Statement

- The SELECT statement is a member of the DML family
- Used to read data from a database
- SELECT syntax covered in previous lessons

# SELECT…INTO and DROP TABLE

- SELECT..INTO is a syntax used to create a new table based on the result set of a SELECT statement
- The INTO keyword identifies the name of the table to create
- The INTO keyword is placed after the last column name in the SELECT clause and before the FROM clause
- Both permanent and temporary tables can be created using this method
- DROP TABLE is a command used to remove tables from a database

```
SELECT
    FirstName
    ,LastName
    ,Country
INTO Customer_Temp
FROM Customer

SELECT *
FROM Customer_Temp
```

Messages

(59 row(s) affected)

| | FirstName | LastName | Country |
|---|---|---|---|
| 1 | Luís | Gonçalves | Brazil |
| 2 | Leonie | Köhler | Germany |
| 3 | François | Tremblay | Canada |
| 4 | Bjørn | Hansen | Norway |
| 5 | František | Wichterlová | Czech Republic |
| 6 | Helena | Holý | Czech Republic |
| 7 | Astrid | Gruber | Austria |
| 8 | Daan | Peeters | Belgium |
| 9 | Kara | Nielsen | Denmark |
| 10 | Eduardo | Martins | Brazil |
| 11 | Alexandre | Rocha | Brazil |
| 12 | Roberto | Almeida | Brazil |

```
DROP TABLE Customer_Temp

SELECT *
FROM Customer_Temp
```

Messages

Command(s) completed successfully.

Messages

Msg 208, Level 16, State 1, Line 15
Invalid object name 'Customer_Temp'.

# Local Temp Tables

- Temp tables are temporary tables that only last for the duration of your SQL Server session
- Local temp tables are prefixed with a hash tag (#)
- Once created temp tables can be referenced within the session
- Local temp tables can only be viewed by the session that created them
- When the session is closed the all temp tables associated with that session are deleted.

```
SELECT *
INTO #TempEmployee
FROM Employee

SELECT *
FROM #TempEmployee
```

Results | Messages

| | EmployeeId | LastName | FirstName | Title | ReportsTo |
|---|---|---|---|---|---|
| 1 | 1 | Adams | Andrew | General Manager | NULL |
| 2 | 2 | Edwards | Nancy | Sales Manager | 1 |
| 3 | 3 | Peacock | Jane | Sales Support Agent | 2 |
| 4 | 4 | Park | Margaret | Sales Support Agent | 2 |
| 5 | 5 | Johnson | Steve | Sales Support Agent | 2 |
| 6 | 6 | Mitchell | Michael | IT Manager | 1 |
| 7 | 7 | King | Robert | IT Staff | 6 |
| 8 | 8 | Callahan | Laura | IT Staff | 6 |

# Global Temp Tables

- Global temp tables are defined by two hash tags in front of the name (##)
- Like local temp tables a global temp table is deleted once the session that created it is closed
- Unlike local temp tables a global temp table can be viewed by other sessions

```
SELECT
    G.Name GenreType
    ,T.Name TrackName
    ,AL.Title AlbumTitle
INTO ##GlobalTempGenre
FROM Genre G
JOIN Track T
    ON T.GenreId = G.GenreId
JOIN Album AL
    ON AL.AlbumId = T.AlbumId


SELECT *
FROM ##GlobalTempGenre
ORDER BY TrackName
```

| | GenreType | TrackName | AlbumTitle |
|---|---|---|---|
| 1 | TV Shows | "?" | Lost, Season 2 |
| 2 | Rock | "40" | War |
| 3 | Classical | "Eine Kleine Nachtmusik" Seren... | Sir Neville Marriner: A Celebration |
| 4 | Alternative & Punk | #1 Zero | Out Of Exile |
| 5 | Pop | #9 Dream | Instant Karma: The Amnesty Inter... |
| 6 | Metal | (Anesthesia) Pulling Teeth | Kill 'Em All |
| 7 | Rock | (Da Le) Yaleo | Supernatural |
| 8 | Reggae | (I Can't Help) Falling In Love Wit... | UB40 The Best Of - Volume Two ... |
| 9 | Rock | (Oh) Pretty Woman | Diver Down |
| 10 | Pop | (There Is) No Greater Love (Teo ... | Frank |

# INSERT Statement

- INSERT statement is used to add one or more rows of data into an existing table
- The data inserted can come from different sources
  - Data values are typed directly into the INSERT statement using the VALUES keyword
  - Data is read from a different table in SQL Server
  - Data comes from an external source such as a text file or Excel spreadsheet
- 2 different ways to insert data in T-SQL
  - Enter data manually using the VALUES clause
  - Insert data from another table using the INSERT INTO clause

# INSERT INTO with VALUES

- Using INSERT INTO with the VALUES keyword allows you to type in directly the data you wish to insert
- The VALUES keyword is followed by parenthesis with the content for each row contained within
- Each row beyond the first  has its own parenthesis and content separated by a comma

```
SELECT *
INTO Genre_Temp
FROM Genre

SELECT *
FROM Genre_Temp
ORDER BY GenreId DESC
```

| | GenreId | Name |
|---|---|---|
| 1 | 25 | Opera |
| 2 | 24 | Classical |
| 3 | 23 | Alternative |
| 4 | 22 | Comedy |
| 5 | 21 | Drama |
| 6 | 20 | Sci Fi & Fantasy |
| 7 | 19 | TV Shows |
| 8 | 18 | Science Fiction |
| 9 | 17 | Hip Hop/Rap |
| 10 | 16 | World |
| 11 | 15 | Electronica/Dance |
| 12 | 14 | R&B/Soul |

```
INSERT INTO Genre_Temp
VALUES(101, 'Movie Soundtrack')
,(102, 'Punk Rock')
,(103, 'Kid Rock')

SELECT *
FROM Genre_Temp
ORDER BY GenreId DESC
```

| | GenreId | Name |
|---|---|---|
| 1 | 103 | Kid Rock |
| 2 | 102 | Punk Rock |
| 3 | 101 | Movie Soundtrack |
| 4 | 25 | Opera |
| 5 | 24 | Classical |
| 6 | 23 | Alternative |
| 7 | 22 | Comedy |
| 8 | 21 | Drama |
| 9 | 20 | Sci Fi & Fantasy |
| 10 | 19 | TV Shows |
| 11 | 18 | Science Fiction |
| 12 | 17 | Hip Hop/Rap |

# INSERT INTO specific Columns

- When the order and number of source columns doesn't exactly match the destination then you must define the destination columns in your INSERT INTO statement
- The destination columns are enclosed in parenthesis and separated by commas
- The destination column order must be the same as the source column order

```
SELECT *
FROM Customer_Temp
ORDER BY CustomerId DESC
```

| | CustomerId | FirstName | LastName | Company | Address |
|---|---|---|---|---|---|
| 1 | 59 | Puja | Srivastava | NULL | 3,Raj Bhavan Road |
| 2 | 58 | Manoj | Pareek | NULL | 12,Community Centre |
| 3 | 57 | Luis | Rojas | NULL | Calle Lira, 198 |

```
INSERT INTO Customer_Temp (CustomerId, FirstName, LastName, Email)
VALUES (101, 'John', 'Smith', 'john@mail.com')
,(102, 'Jane', 'Doe', 'jane@mail.com')

SELECT *
FROM Customer_Temp
ORDER BY CustomerId DESC
```

| | CustomerId | FirstName | LastName | Company | Address |
|---|---|---|---|---|---|
| 1 | 102 | Jane | Doe | NULL | NULL |
| 2 | 101 | John | Smith | NULL | NULL |
| 3 | 59 | Puja | Srivastava | NULL | 3,Raj Bhavan Road |
| 4 | 58 | Manoj | Pareek | NULL | 12,Community Centre |

# INSERT INTO with SELECT

- A SELECT statement can be used to define the insert data source
- The INSERT INTO syntax is the same
- Instead of VALUES you use a SELECT statement to define the insert source data
- The source column number and order must match the destination

```
SELECT TOP 2
    EmployeeId+200 ,LastName ,FirstName ,Email
FROM Employee
```

| | (No column name) | LastName | FirstName | Email |
|---|---|---|---|---|
| 1 | 201 | Adams | Andrew | andrew@chinookcorp.com |
| 2 | 202 | Edwards | Nancy | nancy@chinookcorp.com |

```
INSERT INTO Customer_Temp
(CustomerId,LastName,FirstName,Email)
SELECT TOP 2
    EmployeeId+200
    ,LastName
    ,FirstName
    ,Email
FROM Employee

SELECT CustomerId,FirstName,LastName,Email
FROM Customer_Temp
ORDER BY CustomerId DESC
```

| | CustomerId | FirstName | LastName | Email |
|---|---|---|---|---|
| 1 | 202 | Nancy | Edwards | nancy@chinookcorp.com |
| 2 | 201 | Andrew | Adams | andrew@chinookcorp.com |
| 3 | 59 | Puja | Srivastava | puja_srivastava@yahoo.in |
| 4 | 58 | Manoj | Pareek | manoj.pareek@rediff.com |

# Inserting from External Data Sources using OPENROWSET

- OPENROWSET uses ODBC drivers to connect directly to an external data source such as text, csv, or excel files
- SQL Server needs to be configured to allow OPENROWSET to work with “distributed queries”
- The drivers needed to make OPENROWSET work may need to be downloaded to the server. Especially if the server is running the 64-bit version
- Configuring OPENROWSET to work can be a real pain. This is not a beginners topic and is provided for informational purposes only

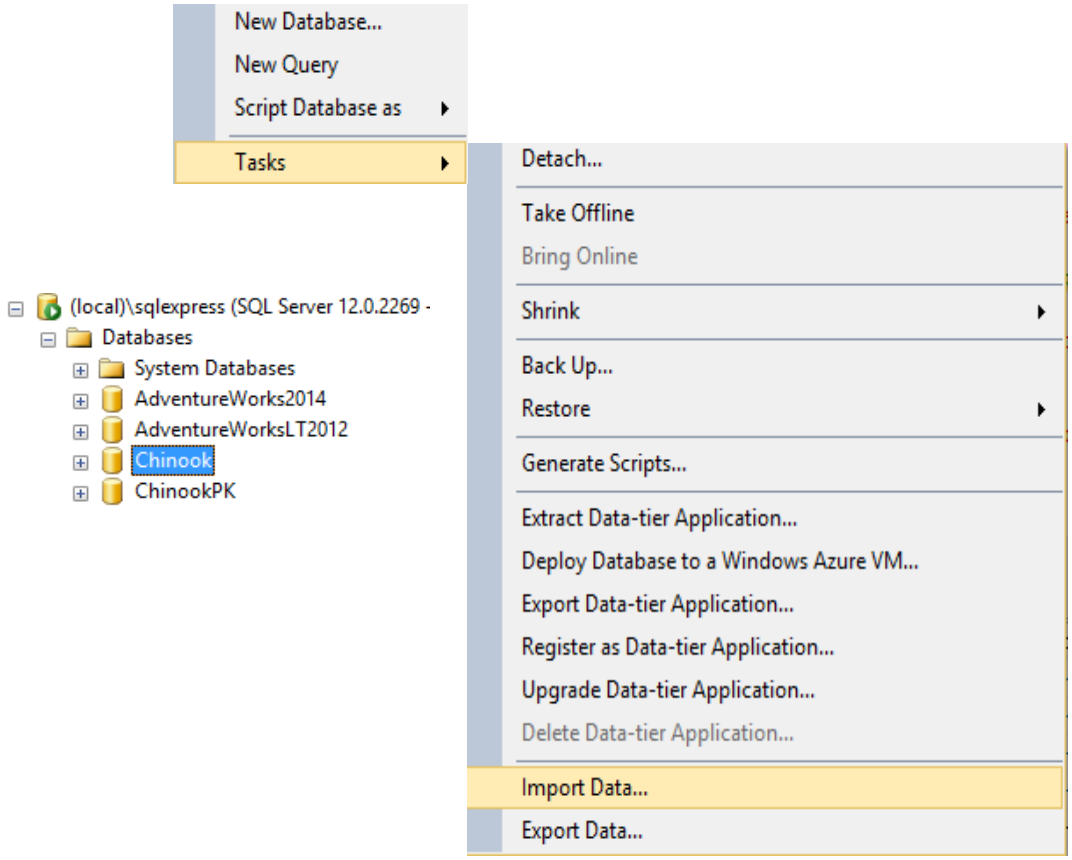
```
--Exposes permissions and allows distributed queries
EXEC sp_configure 'show advanced options', 1
RECONFIGURE
GO
EXEC sp_configure 'ad hoc distributed queries', 1
RECONFIGURE
GO
--The query is using a 64-bit Access text driver
--The driver is called Microsoft Access Database Engine 2010 Redistributable
--and is a free download.  It will not work with 32-bit Office 2010 components installed
 INSERT INTO Product
 SELECT * FROM OPENROWSET('MSDASQL',
    'Driver={Microsoft Access Text Driver (*.txt, *.csv)};
    DefaultDir=C:\Users\Eric\Google Drive\Introduction to SQL\;'
    , 'select * from Product.txt')
--Restricts and hides distributed query permissions
GO
EXEC sp_configure 'ad hoc distributed queries', 0
RECONFIGURE
GO
EXEC sp_configure 'show advanced options', 0
RECONFIGURE
```

Results | Messages

| | ProductID | Name | ProductNumber | Color | StandardCost | ListPrice | Size | Weight |
|---|---|---|---|---|---|---|---|---|
| 1 | 680 | HL Road Frame - Black, 58 | FR-R92B-58 | Black | 1059.31 | 1431.5 | 58 | 1016.04 |
| 2 | 706 | HL Road Frame - Red, 58 | FR-R92R-58 | Red | 1059.31 | 1431.5 | 58 | 1016.04 |
| 3 | 707 | Sport-100 Helmet, Red | HL-U509-R | Red | 13.0863 | 34.99 | NULL | NULL |
| 4 | 708 | Sport-100 Helmet, Black | HL-U509 | Black | 13.0863 | 34.99 | NULL | NULL |
| 5 | 709 | Mountain Bike Socks, M | SO-B909-M | White | 3.3963 | 9.5 | M | NULL |
| 6 | 710 | Mountain Bike Socks, L | SO-B909-L | White | 3.3963 | 9.5 | L | NULL |
| 7 | 711 | Sport-100 Helmet, Blue | HL-U509-B | Blue | 13.0863 | 34.99 | NULL | NULL |
| 8 | 712 | AWC Logo Cap | CA-1098 | Multi | 6.9223 | 8.99 | NULL | NULL |
| 9 | 713 | Long-Sleeve Logo Jersey, S | LJ-0192-S | Multi | 38.4923 | 49.99 | S | NULL |
| 10 | 714 | Long-Sleeve Logo Jersey, M | LJ-0192-M | Multi | 38.4923 | 49.99 | M | NULL |
| 11 | 715 | Long-Sleeve Logo Jersey, L | LJ-0192-L | Multi | 38.4923 | 49.99 | L | NULL |

# Importing Data into SQL Server

- SQL Server has tools for importing data
- Right-click on the database into which you want to import data
- Click on the Import Data… link in the Tasks section
- Follow the wizard instructions to insert data as a new table into your database
- You can then copy the data to other tables in your database using the INSERT INTO syntax



# UPDATE Statement

- The UPDATE statement is used to update one or more rows of existing data in a table
- The SET keyword is required with an UPDATE statement. It defines which columns to update as well as to which values to update them
- A WHERE clause is used to define which rows the UPDATE clause will update
- If no WHERE clause is used then the entire table will be updated by the UPDATE statement

# Simple UPDATE

- The UPDATE keyword is immediately followed by the table name to update
- The SET keyword immediately follows the UPDATE statement
- SET syntax is the column name followed by the equal sign and an expression
- If more than one column is being updated, each column must be separated by a comma

```
SELECT *
INTO #Customer
FROM Customer

SELECT *
FROM #Customer
```

| | CustomerId | FirstName | LastName | Company | Address |
|---|---|---|---|---|---|
| 1 | 1 | Luís | Gonçalves | Embrae... | Av. Brigadeir... |
| 2 | 2 | Leonie | Köhler | NULL | Theodor-Heu... |
| 3 | 3 | François | Tremblay | NULL | 1498 rue Bél... |
| 4 | 4 | Bjørn | Hansen | NULL | Ullevålsveien... |

```
UPDATE #Customer
SET Company = 'Microsoft'
, Address = '1 Microsoft Way'
, City = 'Redmond'

SELECT *
FROM #Customer
```

| | CustomerId | FirstName | LastName | Company | Address | City |
|---|---|---|---|---|---|---|
| 1 | 1 | Luís | Gonçalves | Microsoft | 1 Microsoft Way | Redmond |
| 2 | 2 | Leonie | Köhler | Microsoft | 1 Microsoft Way | Redmond |
| 3 | 3 | François | Tremblay | Microsoft | 1 Microsoft Way | Redmond |
| 4 | 4 | Bjørn | Hansen | Microsoft | 1 Microsoft Way | Redmond |

# UPDATE with WHERE clause

- The WHERE clause is used to filter which rows to update in a table
- A FROM clause is not required if the filter is being performed on the same table being updated

```
SELECT *
INTO #Customer
FROM Customer

SELECT *
FROM #Customer
```

| | CustomerId | FirstName | LastName | Company | Address | City |
|---|---|---|---|---|---|---|
| 1 | 1 | Luís | Gonçalves | Embra... | Av. Brigadeiro F... | São José ... |
| 2 | 2 | Leonie | Köhler | NULL | Theodor-Heuss-... | Stuttgart |
| 3 | 3 | François | Tremblay | NULL | 1498 rue Bélan... | Montréal |

```
UPDATE #Customer
SET Company = 'Microsoft'
, Address = '1 Microsoft Way'
, City = 'Redmond'
WHERE LastName = 'Tremblay'

SELECT *
FROM #Customer
```

| | CustomerId | FirstName | LastName | Company | Address | City |
|---|---|---|---|---|---|---|
| 1 | 1 | Luís | Gonçalves | Embrae... | Av. Brigadeiro ... | São José d... |
| 2 | 2 | Leonie | Köhler | NULL | Theodor-Heus... | Stuttgart |
| 3 | 3 | François | Tremblay | Microsoft | 1 Microsoft Way | Redmond |

# UPDATE with FROM Clause

- Multiple tables can be joined together in an UPDATE clause
- Only one table can be updated at a time
- If the update table is aliased in the FROM clause, you can use the Alias instead of the table name after the UPDATE keyword

```
SELECT
    C.LastName
    ,InvoiceId
    ,BillingAddress
    ,BillingPostalCode
    ,Total
FROM Customer C
JOIN #Invoice I
    ON I.CustomerId = C.CustomerId
WHERE C.LastName = 'Tremblay'
```

| | LastName | InvoiceId | BillingAddress | BillingPostalCode | Total |
|---|---|---|---|---|---|
| 1 | Tremblay | 99 | 1498 rue Bélanger | H2G 1A7 | 3.98 |
| 2 | Tremblay | 110 | 1498 rue Bélanger | H2G 1A7 | 13.86 |
| 3 | Tremblay | 165 | 1498 rue Bélanger | H2G 1A7 | 8.91 |

```
UPDATE I
SET BillingAddress = '1 Microsoft Way'
FROM Customer C
JOIN #Invoice I
    ON I.CustomerId = C.CustomerId
WHERE C.LastName = 'Tremblay'
```

| | LastName | InvoiceId | BillingAddress | BillingPostalCode | Total |
|---|---|---|---|---|---|
| 1 | Tremblay | 99 | 1 Microsoft Way | H2G 1A7 | 3.98 |
| 2 | Tremblay | 110 | 1 Microsoft Way | H2G 1A7 | 13.86 |
| 3 | Tremblay | 165 | 1 Microsoft Way | H2G 1A7 | 8.91 |

# DELETE Statement

- The DELETE statement is used to remove one or more rows of existing data in a table
- DELETE Syntax is similar to UPDATE syntax except there is no SET keyword
- A WHERE clause is used to define which rows the DELETE clause will delete
- If no WHERE clause is used then the entire table content will be deleted by the DELETE clause. The table itself will not be deleted

# Simple DELETE

- Syntax for DELETE is the DELETE keyword followed by the table name
- Use a WHERE clause to identify which rows to delete

```
SELECT *
INTO Genre_Temp
FROM Genre

SELECT *
FROM Genre_Temp
```

| | GenreId | Name |
|---|---|---|
| 1 | 1 | Rock |
| 2 | 2 | Jazz |
| 3 | 3 | Metal |
| 4 | 4 | Alternative & Punk |
| 5 | 5 | Rock And Roll |

```
DELETE Genre_Temp
WHERE GenreId BETWEEN 1 AND 3

SELECT *
FROM Genre_Temp
```

| | GenreId | Name |
|---|---|---|
| 1 | 4 | Alternative & Punk |
| 2 | 5 | Rock And Roll |
| 3 | 6 | Blues |
| 4 | 7 | Latin |

# DELETE with FROM Clause

- Like the UPDATE syntax multiple tables can be joined together in a DELETE clause
- Only one table can be deleted at a time
- If the delete table is aliased in the FROM clause, you can use the Alias instead of the table name after the DELETE keyword

```
SELECT
    C.LastName ,InvoiceId ,BillingAddress ,BillingPostalCode ,Total
FROM Customer C
JOIN #Invoice I
    ON I.CustomerId = C.CustomerId
WHERE C.LastName = 'Tremblay'
```

| | LastName | InvoiceId | BillingAddress | BillingPostalCode | Total |
|---|---|---|---|---|---|
| 1 | Tremblay | 99 | 1498 rue Bélanger | H2G 1A7 | 3.98 |
| 2 | Tremblay | 110 | 1498 rue Bélanger | H2G 1A7 | 13.86 |
| 3 | Tremblay | 165 | 1498 rue Bélanger | H2G 1A7 | 8.91 |
| 4 | Tremblay | 294 | 1498 rue Bélanger | H2G 1A7 | 1.98 |
| 5 | Tremblay | 317 | 1498 rue Bélanger | H2G 1A7 | 3.96 |
| 6 | Tremblay | 339 | 1498 rue Bélanger | H2G 1A7 | 5.94 |

```
DELETE I
FROM Customer C
JOIN #Invoice I
    ON I.CustomerId = C.CustomerId
WHERE C.LastName = 'Tremblay'
AND I.Total < 5.00
```

| | LastName | InvoiceId | BillingAddress | BillingPostalCode | Total |
|---|---|---|---|---|---|
| 1 | Tremblay | 110 | 1498 rue Bélanger | H2G 1A7 | 13.86 |
| 2 | Tremblay | 165 | 1498 rue Bélanger | H2G 1A7 | 8.91 |
| 3 | Tremblay | 339 | 1498 rue Bélanger | H2G 1A7 | 5.94 |

# TRUNCATE TABLE

- The TRUNCATE TABLE command removes all data from a table
- It performs the same way as a DELETE statement without the WHERE clause
- The difference between TRUNCATE TABLE and DELETE is delete is logged while truncate is not logged. This matters if you are working with transaction control language

```
SELECT *
INTO Customer_Temp
FROM Customer

SELECT *
FROM Customer_Temp
```

| | CustomerId | FirstName | LastName | Company |
|---|---|---|---|---|
| 1 | 1 | Luís | Gonçalves | Embraer - Empresa Brasile |
| 2 | 2 | Leonie | Köhler | NULL |
| 3 | 3 | François | Tremblay | NULL |
| 4 | 4 | Bjørn | Hansen | NULL |
| 5 | 5 | František | Wichterlová | JetBrains s.r.o. |

```
TRUNCATE TABLE Customer_Temp

SELECT *
FROM Customer_Temp
```

| | CustomerId | FirstName | LastName | Company |
|---|---|---|---|---|

# Summary

- SELECT
  - SELECT INTO
  - Temp Tables
- INSERT
  - Basic Insert
  - Using OpenRowset
  - External Data into SQL Server
- UPDATE
- DELETE
- TRUNCATE TABLE