Additional WHERE Conditions

- Review
- IS NULL / IS NOT NULL
- BETWEEN
- LIKE
- Subqueries
- IN
- EXISTS
- NOT
- AND / OR

WHERE review

- You can place one or more search conditions within a WHERE clause
- A simple WHERE search condition consists of a column, an operator, and a value
- When searching against fixed text or date values you must enclose the value in single quotes.
- Number datatypes do not require single quotes.
- You can also compare one column against another column.

WHERE Clause Examples
WHERE Lastname = 'Smith'
WHERE BirthDate = '12/25/1982'
WHERE Age = 72
WHERE StartDate = EndDate

Operator	Syntax	Example
Equal	=	WHERE FirstName = 'John'
Not Equal	<> or !=	WHERE Country <> 'United States'
Greater Than	>	WHERE Amount > 1000
Less Than	<	WHERE StartDate < EndDate
Greater Than or Equal	>=	WHERE StartDate >= '1/1/2015'
Less Than or Equal	<=	WHERE Age <= 18

IS [NOT] NULL Keywords

- The IS NULL keywords are used to check if a column field has a Null value
- The IS NOT NULL keywords only return records that do not have a Null value
- Using an equal (=) or not equal (<>) sign is not valid syntax for a Null search in the WHERE clause

```
SELECT *
FROM Customer
WHERE Company IS NULL
```

	CustomerId	FirstName	LastName	Company
1	2	Leonie	Köhler	NULL
2	3	François	Tremblay	NULL
3	4	Bjørn	Hansen	NULL
4	6	Helena	Holý	NULL
5	7	Astrid	Gruber	NULL

```
SELECT *
FROM Customer
WHERE Company IS NOT NULL
```

	CustomerId	FirstName	LastName	Company
1	1	Luís	Gonçalves	Embraer - Empresa
2	5	František	Wichterlová	JetBrains s.r.o.
3	10	Eduardo	Martins	Woodstock Discos
4	11	Alexandre	Rocha	Banco do Brasil S.A
5	12	Roberto	Almeida	Riotur
6	14	Mark	Philips	Telus
7	15	Jennifer	Peterson	Rogers Canada

Searching Columns with Null Values

- Not equal, greater than, and less than operators will not include records with Null values in the result set
- If you wish to include Nulls you need to explicitly search for them

```
SELECT *
FROM Customer
WHERE Company != 'Telus'
```

	CustomerId	FirstName	LastName	Company
1	1	Luís	Gonçalves	Embraer - Empresa B
2	5	František	Wichterlová	JetBrains s.r.o.
3	10	Eduardo	Martins	Woodstock Discos
4	11	Alexandre	Rocha	Banco do Brasil S.A.
5	12	Roberto	Almeida	Riotur
6	15	Jennifer	Peterson	Rogers Canada
7	16	Frank	Harris	Google Inc.

```
SELECT *
FROM Customer
WHERE Company != 'Telus'
OR Company IS NULL
```

	CustomerId	FirstName	LastName	Company
1	1	Luís	Gonçalves	Embraer - Empresa E
2	2	Leonie	Köhler	NULL
3	3	François	Tremblay	NULL
4	4	Bjørn	Hansen	NULL
5	5	František	Wichterlová	JetBrains s.r.o.
6	6	Helena	Holý	NULL
7	7	Astrid	Gruber	NULL
8	8	Daan	Peeters	NULL
9	9	Kara	Nielsen	NULL
10	10	Eduardo	Martins	Woodstock Discos

The BETWEEN Keyword

- Use the BETWEEN keyword to specify a range to search
- Place the AND keyword between the low and high values
- The low and high values will be included in the result set
- BETWEEN works on alphanumeric and date datatypes

```
SELECT
    CustomerId
    ,InvoiceDate
    ,BillingCity
    ,BillingCountry
FROM Invoice
WHERE InvoiceDate BETWEEN '2010-01-08' AND '2010-01-26'
```

Results | Messages

	CustomerId	InvoiceDate	BillingCity	BillingCountry
1	43	2010-01-08 00:00:00.000	Dijon	France
2	45	2010-01-08 00:00:00.000	Budapest	Hungary
3	47	2010-01-09 00:00:00.000	Rome	Italy
4	51	2010-01-10 00:00:00.000	Stockholm	Sweden
5	57	2010-01-13 00:00:00.000	Santiago	Chile
6	7	2010-01-18 00:00:00.000	Vienne	Austria
7	21	2010-01-26 00:00:00.000	Reno	USA

The LIKE keyword

- The LIKE keyword is used to check for character string matches
- LIKE replaces the = sign in a WHERE clause
- LIKE accepts wildcard expressions
  - % (percent) zero or more characters
  - _ (underscore) one single character
  - [ ] (square brackets) one single character in the specified range or set (e.g. [m-z] or [amz])
  - [^] (square brackets with carrot) one single character *not* in the specified range or set

LIKE Example Percent and Underscore

- The first example uses the % wildcard at front and end of the character string
- The second example has 2 underscore (_)wildcards at the front and the % wildcard at the end

```
SELECT
    FirstName
    ,LastName
    ,Email
FROM Customer
WHERE Email LIKE '%gmail%'
```

	FirstName	LastName	Email
1	François	Tremblay	ftremblay@gmail.com
2	Helena	Holý	hholy@gmail.com
3	Heather	Leacock	hleacock@gmail.com
4	Frank	Ralston	fralston@gmail.com
5	Julia	Barnett	jubarnett@gmail.com
6	Martha	Silk	marthasilk@gmail.com
7	Dominique	Lefebvre	dominiquelefebvre@gmail.com
8	Phil	Hughes	phil.hughes@gmail.com

```
SELECT DISTINCT
    Country
FROM Customer
WHERE Country LIKE '__A%'
```

	Country
1	Brazil
2	France
3	Italy
4	Spain
5	USA

LIKE Example Character Range

- In example 1 the first character slot must be in the specified range of A through D
- In example 2 the first character slot can be anything other than the specified range of A through D

```
SELECT
    FirstName
    ,LastName
FROM Customer
WHERE LastName LIKE '[A-D]%'
ORDER BY LastName
```

	FirstName	LastName
1	Roberto	Almeida
2	Julia	Barnett
3	Camille	Bernard
4	Michelle	Brooks
5	Robert	Brown
6	Kathy	Chase
7	Richard	Cunningham
8	Marc	Dubois

```
SELECT
    FirstName
    ,LastName
FROM Customer
WHERE LastName LIKE '[^A-D]%'
ORDER BY LastName
```

	FirstName	LastName
1	João	Fernandes
2	Edward	Francis
3	Wyatt	Girard
4	Luís	Gonçalves
5	John	Gordon
6	Tim	Goyer
7	Patrick	Gray
8	Astrid	Gruber
9	Diego	Gutiérrez

LIKE Example Character Set

- In example 1 the first character slot must be in the specified set of A,B or Z
- IN example 2 the first character slot can be anything other than the specified set of A,B or Z

```
SELECT
    FirstName
    ,LastName
FROM Customer
WHERE LastName LIKE '[ABZ]%'
ORDER BY LastName
```

	FirstName	LastName
1	Roberto	Almeida
2	Julia	Barnett
3	Camille	Bernard
4	Michelle	Brooks
5	Robert	Brown
6	Fynn	Zimmermann

```
SELECT
    FirstName
    ,LastName
FROM Customer
WHERE LastName LIKE '[^ABZ]%'
ORDER BY LastName
```

	FirstName	LastName
1	Kathy	Chase
2	Richard	Cunningham
3	Marc	Dubois
4	João	Fernandes
5	Edward	Francis
6	Wyatt	Girard
7	Luís	Gonçalves
8	John	Gordon

Subqueries

- A subquery is a query nested inside another query
- Subqueries are often used in WHERE clauses
- There are two types of subqueries
- Self-Contained Subquery: The query can run independent of its parent outer query. There are no links between the queries
- Correlated Subquery: The subquery contains one or more references to the outer query. The subquery is evaluated once for each row processed by the outer query
- Examples of each subquery will be shown with the IN and EXISTS keywords

The IN keyword

- The IN keyword is used to compare multiple values against a column or expression
- The IN keyword replaces the = sign in the WHERE clause the values of the IN keyword must be enclosed in parenthesis with each value after the first separated by a comma
- Values must be enclosed in single quotes when searching against date and string datatypes
- The IN keyword can accept a subquery as a value if the subquery consists of a single column

IN Example Character and Number Values

- In example one only the country values in the IN clause are returned in the record set
- In example two only the numbers are returned. Note that the numbers do not require single quotes

```
SELECT
    CustomerId
    ,LastName, Phone, Email, Country
FROM Customer
WHERE Country IN('Brazil','United Kingdom','Sweden')
ORDER BY Country
```

	CustomerId	LastName	Phone	Email	Country
1	1	Gonçalves	+55 (12) 3923-5555	luisg@embraer.com.br	Brazil
2	10	Martins	+55 (11) 3033-5446	eduardo@woodstock.com.br	Brazil
3	11	Rocha	+55 (11) 3055-3278	alero@uol.com.br	Brazil
4	12	Almeida	+55 (21) 2271-7000	roberto.almeida@riotur.gov.br	Brazil
5	13	Ramos	+55 (61) 3363-5547	fernadaramos4@uol.com.br	Brazil
6	51	Johansson	+46 08-651 52 52	joakim.johansson@yahoo.se	Sweden
7	52	Jones	+44 020 7707 0707	emma_jones@hotmail.com	United Kingdom
8	53	Hughes	+44 020 7976 5722	phil.hughes@gmail.com	United Kingdom
9	54	Murray	+44 0131 315 3300	steve.murray@yahoo.uk	United Kingdom

```
SELECT
    AlbumId ,Title ,ArtistId
FROM Album
WHERE AlbumId IN(1,2,3,4,5,6)
```

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
2	Balls to the Wall	2
3	Restless and Wild	2
4	Let There Be Rock	1
5	Big Ones	3
6	Jagged Little Pill	4

IN Example with a Subquery

- The subquery returns a single column with 2 values
- Those 2 values are used in the IN clause when the parent query is executed
- The parent query only returns records that contain the values in the subquery

```
SELECT CustomerId
FROM Customer
WHERE Country = 'India'
```

	CustomerId
1	58
2	59

```
SELECT
    CustomerId
    ,InvoiceId
    ,InvoiceDate
    ,Total
FROM Invoice
WHERE CustomerId IN(
    SELECT CustomerId
    FROM Customer
    WHERE Country = 'India')
ORDER BY  CustomerId, InvoiceId
```

	CustomerId	InvoiceId	InvoiceDate	Total
1	58	120	2010-06-12 00:00:00.000	1.98
2	58	131	2010-07-23 00:00:00.000	13.86
3	58	186	2011-03-23 00:00:00.000	8.91
4	58	315	2012-10-27 00:00:00.000	1.98
5	58	338	2013-01-29 00:00:00.000	3.96
6	58	360	2013-05-03 00:00:00.000	5.94
7	58	412	2013-12-22 00:00:00.000	1.99
8	59	23	2009-04-05 00:00:00.000	3.96
9	59	45	2009-07-08 00:00:00.000	5.94
10	59	97	2010-02-26 00:00:00.000	1.99
11	59	218	2011-08-20 00:00:00.000	1.98
12	59	229	2011-09-30 00:00:00.000	13.86
13	59	284	2012-05-30 00:00:00.000	8.91

The EXISTS keyword

- The EXISTS keyword is used to check against a subquery whether one or more records exist
- A correlated subquery in parenthesis is required after the EXISTS keyword
- The EXISTS subquery doesn't actually produce any data it returns true or false
- The SELECT clause of the subquery will accept any valid column. It is best practice to use the * in an EXISTS subquery

EXISTS Example

- This example checks for the existence Album IDs in the Track table and returns all records in the Album table where a match is found
- Note that the subquery is checking for A.AlbumId which is in the outer query

```
SELECT *
FROM Album_temp A
```

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
3	Restless and Wild	2
4	Let There Be Rock	1

```
SELECT *
FROM Track_temp
```

TrackId	Name	AlbumId
2	Balls to the Wall	2
3	Fast As a Shark	3
8	Inject The Venom	1

```
SELECT *
FROM Album_temp A
WHERE EXISTS (
    SELECT *
    FROM Track_temp T
    WHERE T.AlbumId = A.AlbumId)
```

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
3	Restless and Wild	2

NOT EXISTS Example

- This example checks for Album IDs that don't exist in the Track table and only returns records in the Album table where a match is *not* found
- Other than the addition of the NOT keyword the query is identical to the EXISTS example

```
SELECT *
FROM Album_temp A
```

AlbumId	Title	ArtistId
1	For Those About To Rock We Salute You	1
3	Restless and Wild	2
4	Let There Be Rock	1

```
SELECT *
FROM Track_temp
```

TrackId	Name	AlbumId
2	Balls to the Wall	2
3	Fast As a Shark	3
8	Inject The Venom	1

```
SELECT *
FROM Album_temp A
WHERE NOT EXISTS (
    SELECT *
    FROM Track_temp T
    WHERE T.AlbumId = A.AlbumId)
```

Results | Messages

	AlbumId	Title	ArtistId
1	4	Let There Be Rock	1

The NOT Keyword

- The NOT keyword can be used to negate (i.e. return the opposite) the meaning of a keyword in a WHERE clause
- Place the NOT key word immediately before the keyword you wish to negate

```
SELECT
    CustomerId
    ,InvoiceDate
    ,BillingCity
    ,BillingCountry
FROM Invoice
WHERE InvoiceDate NOT BETWEEN '2010-01-08' AND '2010-01-26'
```

```
SELECT *
FROM Album_temp A
WHERE NOT EXISTS (
    SELECT *
    FROM Track_temp T
    WHERE T.AlbumId = A.AlbumId)
```

```
SELECT
    FirstName
    ,LastName
    ,Email
FROM Customer
WHERE Email NOT LIKE '%gmail%'
```

```
SELECT
    CustomerId
    ,LastName, Phone, Email, Country
FROM Customer
WHERE Country NOT IN('Brazil','United Kingdom','Sweden')
ORDER BY Country
```

The AND and OR operators

- Use the AND OR keywords to string together multiple WHERE conditions
- The example returns the following:
  - Find track names that match the album name, and are in the “Rock” or “Latin” genre, and do not have an artist name that starts with G through Z
  - Or find tracks by the artist “Iron Maiden” and are in the “Blues” genre
- Use parenthesis to group together the search expressions

```
SELECT
    AT.Name AS ArtistName
    ,AB.Title AS AlbumName
    ,T.Name AS TrackName
    ,G.Name AS GenreName
FROM Artist AT
JOIN Album AB
    ON AB.ArtistId = AT.ArtistId
JOIN Track T
    ON T.AlbumId = AB.AlbumId
JOIN Genre G
    ON G.GenreId = T.GenreId
WHERE
    (
    AB.Title = T.Name
    AND G.Name IN('Rock','Latin')
    AND AT.Name NOT LIKE '[G-Z]%'
    )
    OR
    (
    AT.Name = 'Iron Maiden'
    AND G.Name = 'Blues'
    )
ORDER BY ArtistName, TrackName
```

Results | Messages

	ArtistName	AlbumName	TrackName	GenreName
1	AC/DC	Let There Be Rock	Let There Be Rock	Rock
2	Accept	Balls to the Wall	Balls to the Wall	Rock
3	Accept	Restless and Wild	Restless and Wild	Rock
4	Caetano Veloso	Prenda Minha	Prenda Minha	Latin
5	Chico Buarque	Minha Historia	Minha Historia	Latin
6	Chico Science &...	Da Lama Ao Caos	Da Lama Ao Caos	Latin
7	David Coverdale	Into The Light	Into The Light	Rock
8	Deep Purple	Fireball	Fireball	Rock
9	Deep Purple	Stormbringer	Stormbringer	Rock
10	Deep Purple	The Battle Rages On	The Battle Rages On	Rock
11	Falamansa	Deixa Entrar	Deixa Entrar	Latin
12	Iron Maiden	Iron Maiden	01 - Prowler	Blues
13	Iron Maiden	Iron Maiden	02 - Sanctuary	Blues
14	Iron Maiden	Iron Maiden	03 - Remember Tomorrow	Blues
15	Iron Maiden	Iron Maiden	04 - Running Free	Blues
16	Iron Maiden	Iron Maiden	05 - Phantom of the Opera	Blues
17	Iron Maiden	Iron Maiden	06 - Transylvania	Blues
18	Iron Maiden	Iron Maiden	07 - Strange World	Blues
19	Iron Maiden	Iron Maiden	08 - Charlotte the Harlot	Blues
20	Iron Maiden	Iron Maiden	09 - Iron Maiden	Blues

Summary

- Review
- IS NULL / IS NOT NULL
- BETWEEN
- LIKE
- Subqueries
- IN
- EXISTS
- NOT
- AND / OR