SELECT (The “SQL”)

WHERE, BETWEEN, IN, LIKE

The simplified syntax of the SELECT statement

```
SELECT select_list
     FROM table_source
     [WHERE search_condition]
     [ORDER BY order_by_list]
```

The four clauses of the SELECT statement

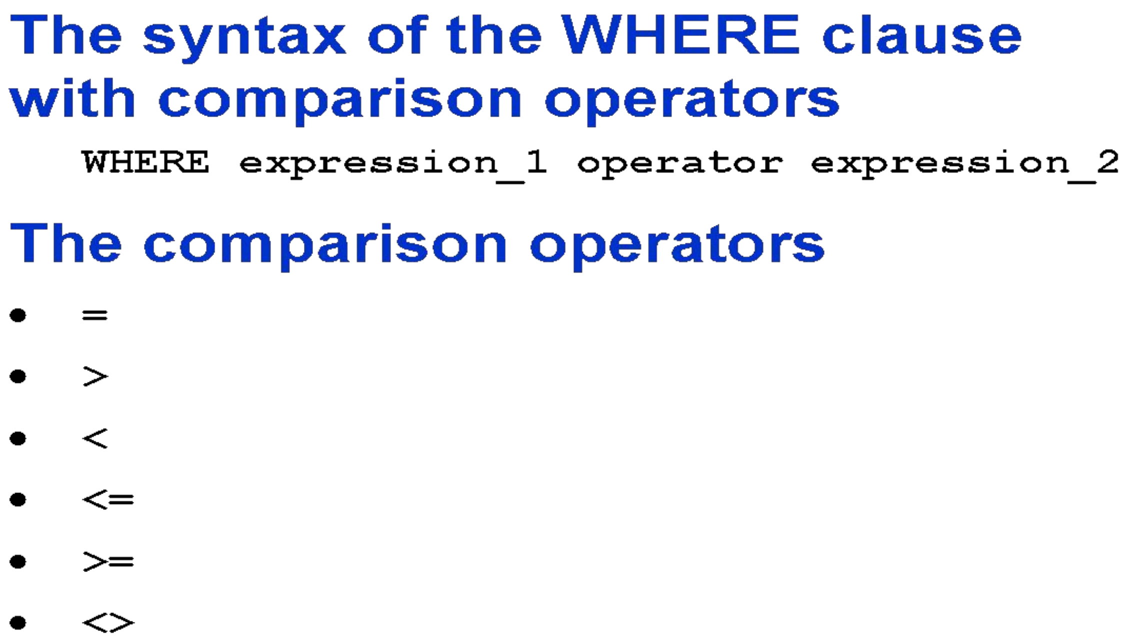
- SELECT
- FROM
- WHERE
- ORDER BY

WHERE

- The WHERE clause is used to filter the data so that the results you get back are only the results that meet the conditions you set
- With a SELECT statement, you should only retrieve the columns that you need…you should also only include the rows that you need, and you do that by using the where clause

Conditional Operators

- You can use comparison operators to compare expressions that result in like data types. So if your field is a integer datatype, you can compare that to another integer. Or if your field is a varchar data type, you can compare that to a string.
- When comparing a string literal or a date literal, you need to enclose it in quotes.



The syntax of the WHERE clause with comparison operators

```
WHERE expression_1 operator expression_2
```

The comparison operators

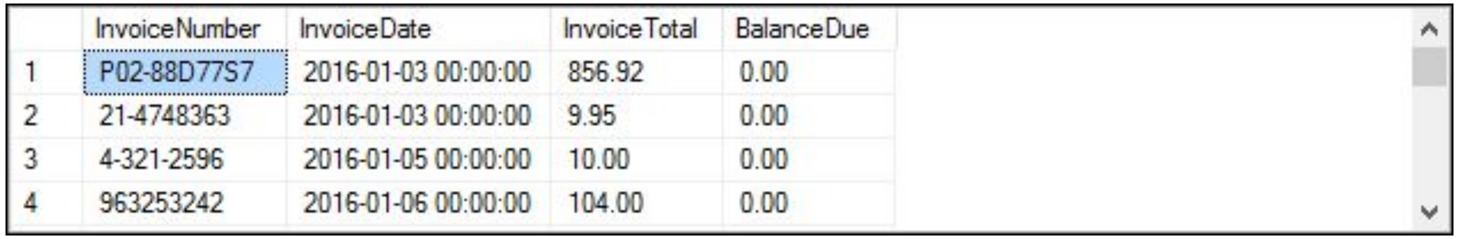
- =
- >
- <
- <=
- >=
- <>

Logical Operators

- You can use these operators to combine two or more search conditions.

A compound condition without parentheses

```
WHERE InvoiceDate > '01/01/2016'
    OR InvoiceTotal > 500
    AND InvoiceTotal - PaymentTotal - CreditTotal > 0
```



	InvoiceNumber	InvoiceDate	InvoiceTotal	BalanceDue
1	P02-88D77S7	2016-01-03 00:00:00	856.92	0.00
2	21-4748363	2016-01-03 00:00:00	9.95	0.00
3	4-321-2596	2016-01-05 00:00:00	10.00	0.00
4	963253242	2016-01-06 00:00:00	104.00	0.00

```
(100 rows)
```

The order of precedence for compound conditions

- NOT
- AND
- OR

Examples of WHERE clauses that retrieve…

Vendors located in Iowa

```
WHERE VendorState = 'IA'
```

Invoices with a balance due (two variations)

```
WHERE InvoiceTotal – PaymentTotal – CreditTotal > 0
WHERE InvoiceTotal > PaymentTotal + CreditTotal
```

Vendors with names from A to L

```
WHERE VendorName < 'M'
```

Invoices on or before a specified date

```
WHERE InvoiceDate <= '2016-05-31'
```

Invoices on or after a specified date

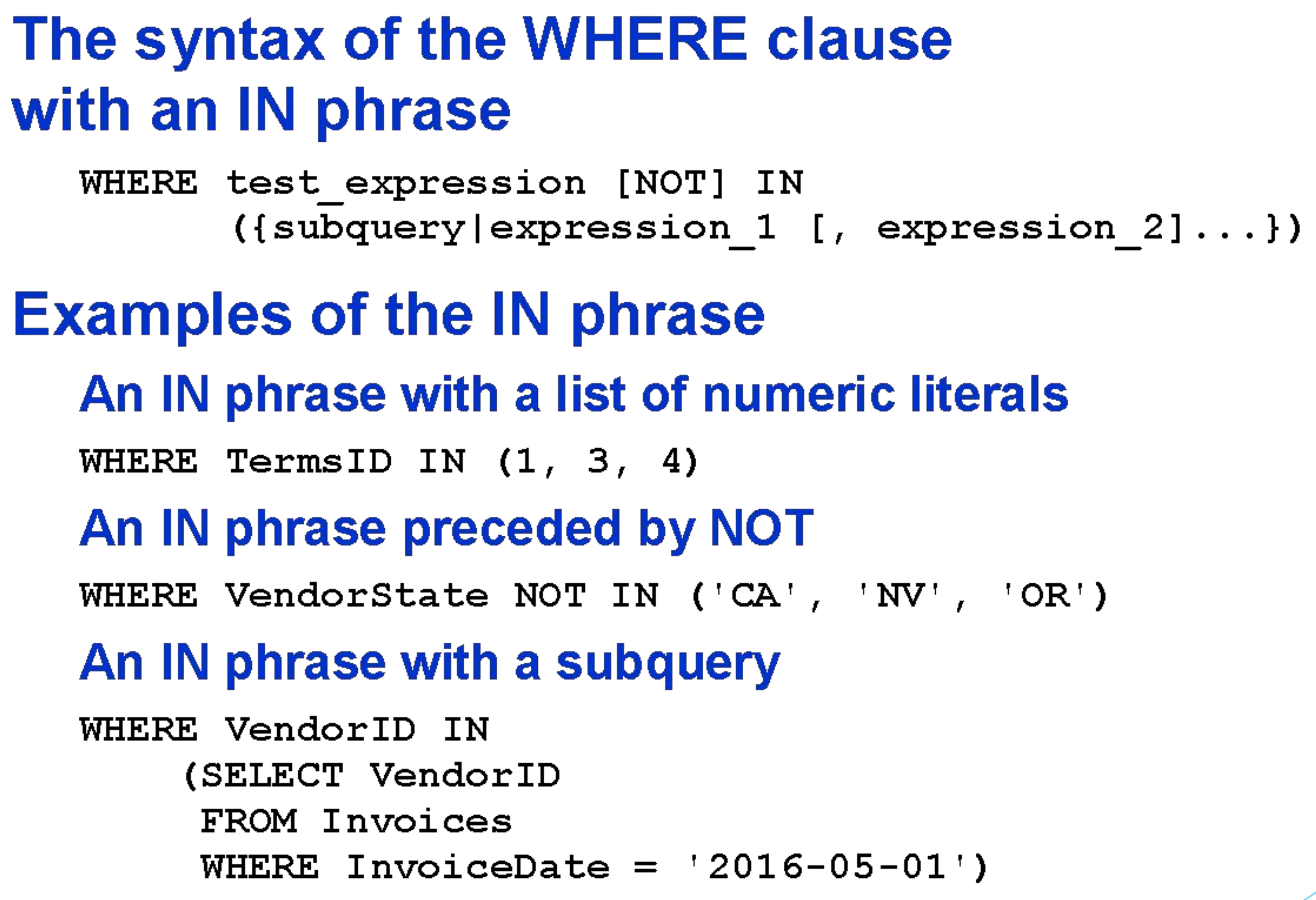
```
WHERE InvoiceDate >= '5/1/16'
```

Invoices with credits that don’t equal zero

```
WHERE CreditTotal <> 0
```

IN

- With IN, the value of the test expression is compared with the list of expressions in the IN phrase. If the test expression is equal to one of the expressions in the list – inside the parentheses – then the row is included in the results



The syntax of the WHERE clause with an IN phrase

```
WHERE test_expression [NOT] IN
      ({subquery|expression_1 [, expression_2]...})
```

Examples of the IN phrase

An IN phrase with a list of numeric literals

```
WHERE TermsID IN (1, 3, 4)
```

An IN phrase preceded by NOT

```
WHERE VendorState NOT IN ('CA', 'NV', 'OR')
```

An IN phrase with a subquery

```
WHERE VendorID IN
    (SELECT VendorID
     FROM Invoices
     WHERE InvoiceDate = '2016-05-01')
```

BETWEEN

- Like using >= and <=

The syntax of the WHERE clause with a BETWEEN phrase

```
WHERE test_expression [NOT] BETWEEN
      begin_expression AND end_expression
```

Examples of the BETWEEN phrase

A BETWEEN phrase with literal values

```
WHERE InvoiceDate BETWEEN '2016-05-01' AND '2016-05-31'
```

A BETWEEN phrase preceded by NOT

```
WHERE VendorZipCode NOT BETWEEN 93600 AND 93799
```

A BETWEEN phrase with a test expression coded as a calculated value

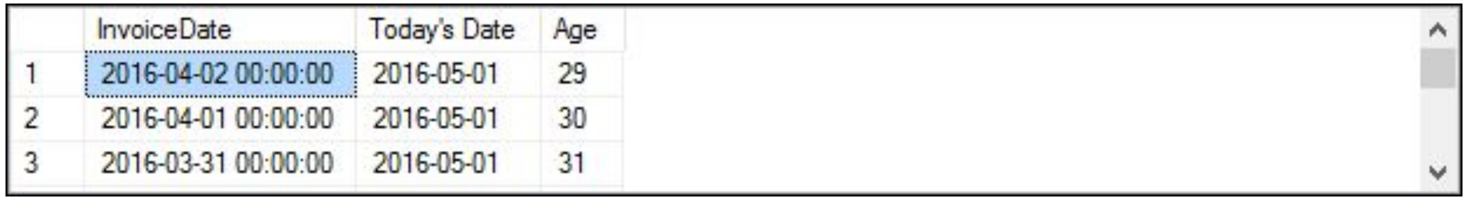
```
WHERE InvoiceTotal – PaymentTotal – CreditTotal
    BETWEEN 200 AND 500
```

A BETWEEN phrase with calculated values

```
WHERE InvoiceDueDate BETWEEN GetDate() AND GetDate() + 30
```

A SELECT statement that computes the age of an invoice

```
SELECT InvoiceDate,
    GETDATE() AS 'Today''s Date',
    DATEDIFF(day, InvoiceDate, GETDATE()) AS Age
FROM Invoices;
```



	InvoiceDate	Today's Date	Age
1	2016-04-02 00:00:00	2016-05-01	29
2	2016-04-01 00:00:00	2016-05-01	30
3	2016-03-31 00:00:00	2016-05-01	31

Warning about date comparisons

- All columns that have the datetime data type include both a date and time, and so does the value returned by the GetDate function.
- When you code a date literal without a time, the time defaults to 12:00 AM (midnight). As a result, a date comparison may not yield the results you expect.

LIKE

- The like determines whether a specified character string matches a specified pattern. Your pattern can include both regular characters and wildcard characters

The syntax of the WHERE clause with a LIKE phrase

```
WHERE match_expression [NOT] LIKE pattern
```

Wildcard symbols

- %
- _
- [ ]
- [-]
- [^]

WHERE clauses that use the LIKE phrase

Example 1

```
WHERE VendorCity LIKE 'SAN%'
```

Cities that will be retrieved

"San Diego" and "Santa Ana"

Example 2

```
WHERE VendorName LIKE 'COMPU_ER%'
```

Vendors that will be retrieved

"Compuserve" and "Computerworld"

Example 3

```
WHERE VendorContactLName LIKE 'DAMI[EO]N'
```

Names that will be retrieved

"Damien" and "Damion"

WHERE clauses that use the LIKE phrase (cont.)

Example 4

```
WHERE VendorState LIKE 'N[A-J]'
```

States that will be retrieved

“NC” and “NJ” but not “NV” or “NY”

Example 5

```
WHERE VendorState LIKE 'N[^K-Y]'
```

States that will be retrieved

“NC” and “NJ” but not “NV” or “NY”

Example 6

```
WHERE VendorZipCode NOT LIKE '[1-9]%'
```

Zip codes that will be retrieved

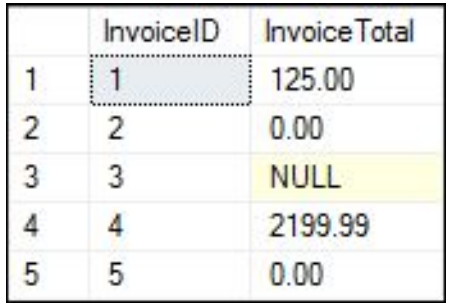
“02107” and “08816”

The syntax of the WHERE clause with the IS NULL clause

```
WHERE expression IS [NOT] NULL
```

The contents of the NullSample table

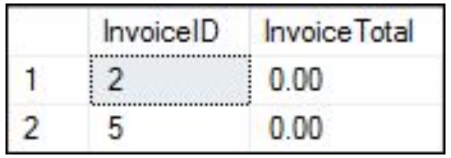
```
SELECT *
FROM NullSample;
```



	InvoiceID	InvoiceTotal
1	1	125.00
2	2	0.00
3	3	NULL
4	4	2199.99
5	5	0.00

A SELECT statement that retrieves rows with zero values

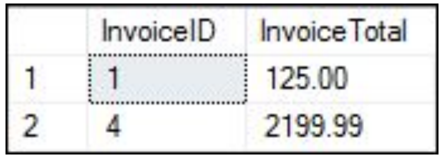
```
SELECT *
FROM NullSample
WHERE InvoiceTotal = 0;
```



	InvoiceID	InvoiceTotal
1	2	0.00
2	5	0.00

A SELECT statement that retrieves rows with non-zero values

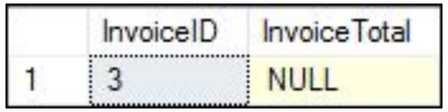
```
SELECT *
FROM NullSample
WHERE InvoiceTotal <> 0;
```



	InvoiceID	InvoiceTotal
1	1	125.00
2	4	2199.99

A SELECT statement that retrieves rows with null values

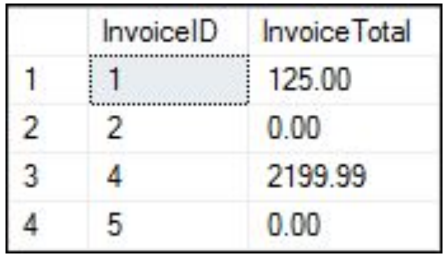
```
SELECT *
FROM NullSample
WHERE InvoiceTotal IS NULL;
```



	InvoiceID	InvoiceTotal
1	3	NULL

A SELECT statement that retrieves rows without null values

```
SELECT *
FROM NullSample
WHERE InvoiceTotal IS NOT NULL;
```



	InvoiceID	InvoiceTotal
1	1	125.00
2	2	0.00
3	4	2199.99
4	5	0.00