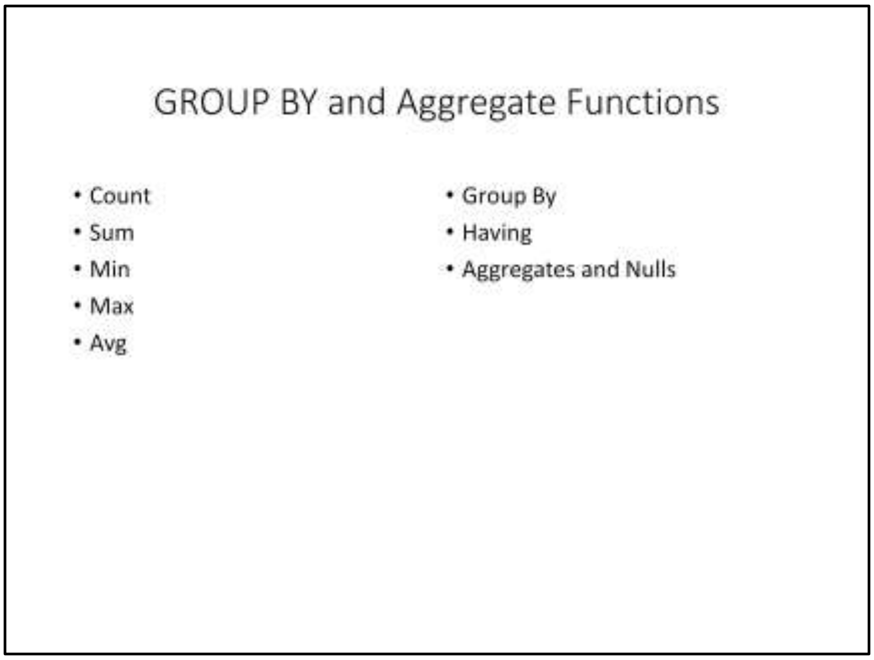
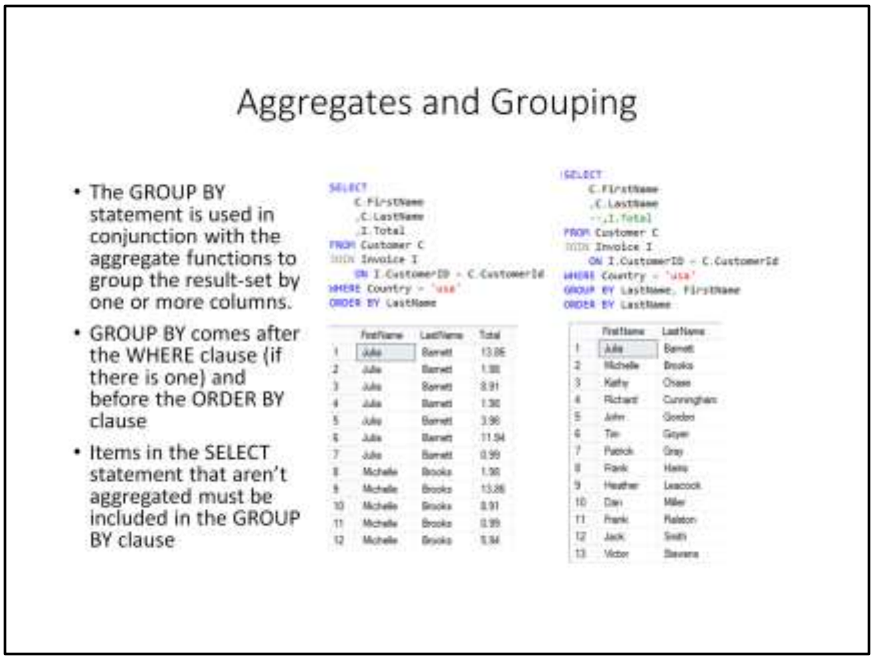


GROUP BY and Aggregate Functions

- Count
- Sum
- Min
- Max
- Avg

- Group By
- Having
- Aggregates and Nulls

This presentation is on the GROUP BY clause as well as some of the aggregate functions associated with the GROUP BY clause.



Aggregates and Grouping

- The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns.
- GROUP BY comes after the WHERE clause (if there is one) and before the ORDER BY clause
- Items in the SELECT statement that aren't aggregated must be included in the GROUP BY clause

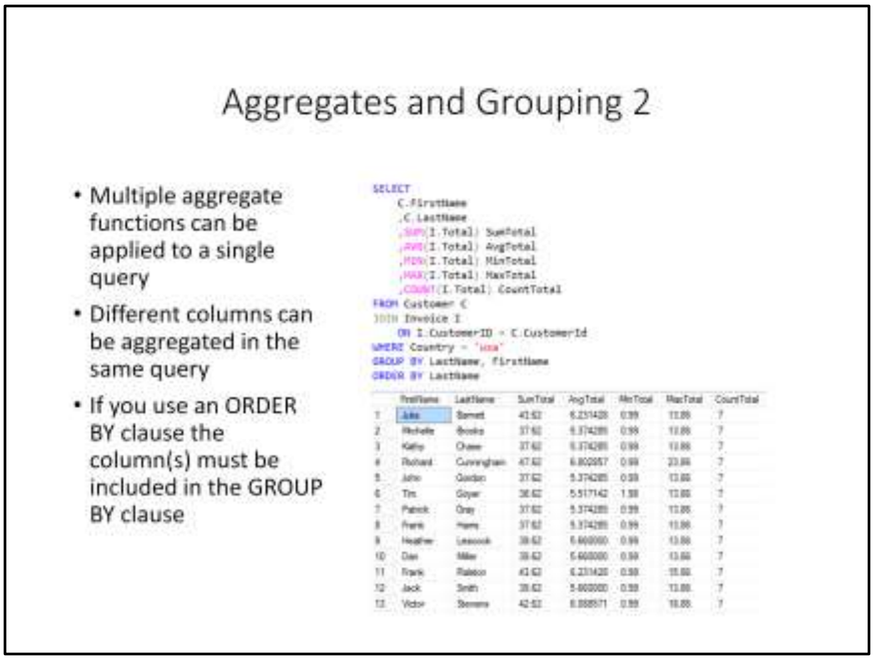
```
SELECT
    C.FirstName
    ,C.LastName
    ,I.Total
FROM Customer C
JOIN Invoice I
    ON I.CustomerID = C.CustomerId
WHERE Country = 'usa'
ORDER BY LastName
```

	FirstName	LastName	Total
1	Julia	Barnett	13.86
2	Julia	Barnett	1.98
3	Julia	Barnett	8.91
4	Julia	Barnett	1.98
5	Julia	Barnett	3.96
6	Julia	Barnett	11.94
7	Julia	Barnett	0.99
8	Michelle	Brooks	1.98
9	Michelle	Brooks	13.86
10	Michelle	Brooks	8.91
11	Michelle	Brooks	0.99
12	Michelle	Brooks	5.94

```
SELECT
    C.FirstName
    ,C.LastName
    --,I.Total
FROM Customer C
JOIN Invoice I
    ON I.CustomerID = C.CustomerId
WHERE Country = 'usa'
GROUP BY LastName, FirstName
ORDER BY LastName
```

	FirstName	LastName
1	Julia	Barnett
2	Michelle	Brooks
3	Kathy	Chase
4	Richard	Cunningham
5	John	Gordon
6	Tim	Goyer
7	Patrick	Gray
8	Frank	Harris
9	Heather	Leacock
10	Dan	Miller
11	Frank	Ralston
12	Jack	Smith
13	Victor	Stevens

Sometimes it is necessary to display aggregate information about multiple records such as the average or sum total. SQL Server achieves this by using the GROUP BY clause. Within the GROUP BY clause you include all of the columns or expressions on which you wish you do your groupings. In the first example you can see that Julia Barnett has 7 records associated with her in the Invoice table. In the second example I have added the GROUP BY clause followed by the first and last name columns. This tells SQL Server to group on the first and last names which you can see in the result set. The result set isn't very useful though because we haven't added any aggregate functions to the SELECT clause. Also notice that I commented out the Total column. This is because Total wasn't included in the GROUP BY clause and it will generate an error if I attempt to run the statement with it uncommented. I could add Total to the GROUP BY clause but that would defeat the purpose as I intend to run aggregate functions on the Total column in the next slide.



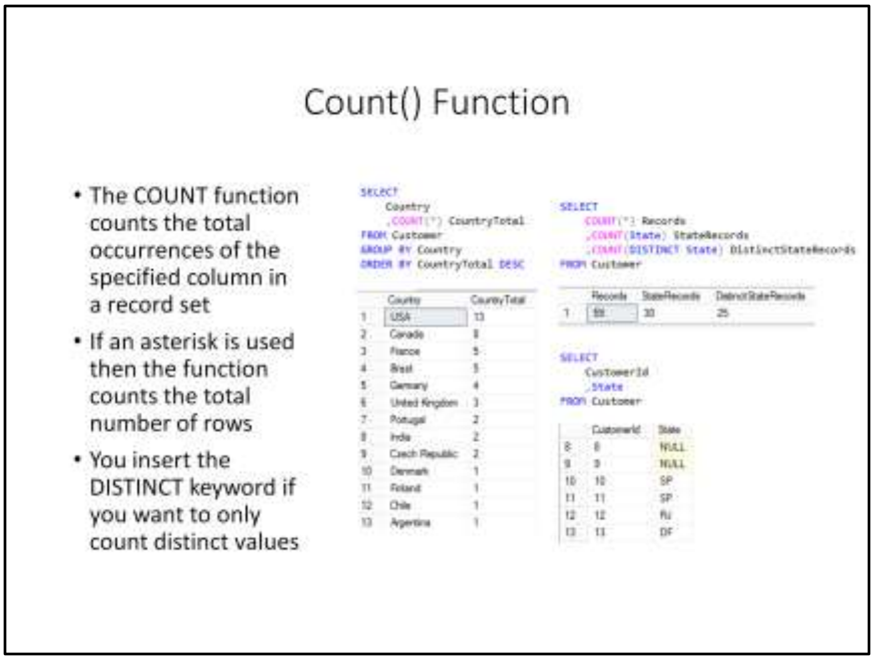
Aggregates and Grouping 2

- Multiple aggregate functions can be applied to a single query
- Different columns can be aggregated in the same query
- If you use an ORDER BY clause the column(s) must be included in the GROUP BY clause

```
SELECT
    C.FirstName
    ,C.LastName
    ,SUM(I.Total) SumTotal
    ,AVG(I.Total) AvgTotal
    ,MIN(I.Total) MinTotal
    ,MAX(I.Total) MaxTotal
    ,COUNT(I.Total) CountTotal
FROM Customer C
JOIN Invoice I
    ON I.CustomerID = C.CustomerId
WHERE Country = 'USA'
GROUP BY LastName, FirstName
ORDER BY LastName
```

	FirstName	LastName	SumTotal	AvgTotal	MinTotal	MaxTotal	CountTotal
1	Julia	Barnett	43.62	6.231428	0.99	13.86	7
2	Michelle	Brooks	37.62	5.374285	0.99	13.86	7
3	Kathy	Chase	37.62	5.374285	0.99	13.86	7
4	Richard	Cunningham	47.62	6.802857	0.99	23.86	7
5	John	Gordon	37.62	5.374285	0.99	13.86	7
6	Tim	Goyer	38.62	5.517142	1.98	13.86	7
7	Patrick	Gray	37.62	5.374285	0.99	13.86	7
8	Frank	Harris	37.62	5.374285	0.99	13.86	7
9	Heather	Leacock	39.62	5.660000	0.99	13.86	7
10	Dan	Miller	39.62	5.660000	0.99	13.86	7
11	Frank	Ralston	43.62	6.231428	0.99	15.86	7
12	Jack	Smith	39.62	5.660000	0.99	13.86	7
13	Victor	Stevens	42.62	6.088571	0.99	18.86	7

I have taken the query from the previous slide and added 5 different aggregate functions based on the Total column. Aggregate functions work on a single column or expression while the GROUP BY clause determines the range of rows on which to perform the aggregates. The firstname, lastname combination of Julia Barnett occurs 7 times in the query so the aggregate functions will work on those 7 rows. Note that any column that isn't part of an aggregate function either needs to be included in the GROUP BY clause, or it needs to be removed from the SELECT and ORDER BY clauses.



Count() Function

- The COUNT function counts the total occurrences of the specified column in a record set
- If an asterisk is used then the function counts the total number of rows
- You insert the DISTINCT keyword if you want to only count distinct values

```
SELECT
    Country
    ,COUNT(*) CountryTotal
FROM Customer
GROUP BY Country
ORDER BY CountryTotal DESC
```

```
     Country          CountryTotal
1    USA              13
2    Canada           8
3    France           5
4    Brazil           5
5    Germany          4
6    United Kingdom   3
7    Portugal         2
8    India            2
9    Czech Republic   2
10   Denmark          1
11   Finland          1
12   Chile            1
13   Argentina        1
```

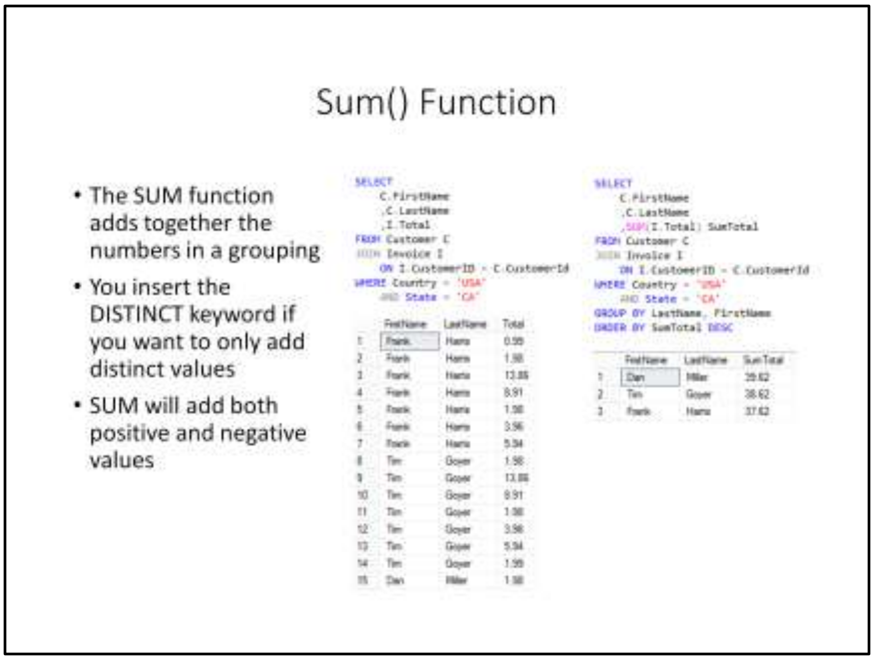
```
SELECT
    COUNT(*) Records
    ,COUNT(State) StateRecords
    ,COUNT(DISTINCT State) DistinctStateRecords
FROM Customer
```

```
     Records   StateRecords   DistinctStateRecords
1    59        30             25
```

```
SELECT
    CustomerId
    ,State
FROM Customer
```

```
     CustomerId   State
8    8            NULL
9    9            NULL
10   10           SP
11   11           SP
12   12           RJ
13   13           DF
```

The COUNT function is used to count the total occurrences of a column or row. If a column is the specified parameter then the column is counted. If an asterisk in used as a parameter then the rows are counted. You can add the DISTINCT keyword before the parameter value when you only want to count unique values in a column. Note that the COUNT function does not count NULL values. The COUNT function is often used to see how many rows are in a table.



Sum() Function

- The SUM function adds together the numbers in a grouping
- You insert the DISTINCT keyword if you want to only add distinct values
- SUM will add both positive and negative values

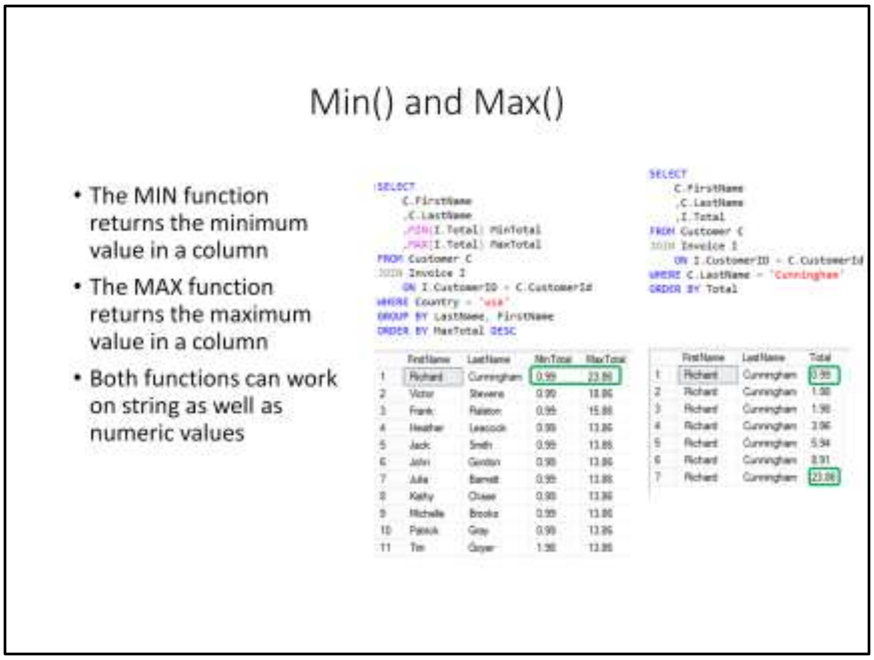
```sql
SELECT
    C.FirstName
    ,C.LastName
    ,I.Total
FROM Customer C
JOIN Invoice I
    ON I.CustomerID = C.CustomerId
WHERE Country = 'USA'
    AND State = 'CA'
```

	FirstName	LastName	Total
1	Frank	Harris	0.99
2	Frank	Harris	1.98
3	Frank	Harris	13.86
4	Frank	Harris	8.91
5	Frank	Harris	1.98
6	Frank	Harris	3.96
7	Frank	Harris	5.94
8	Tim	Goyer	1.98
9	Tim	Goyer	13.86
10	Tim	Goyer	8.91
11	Tim	Goyer	1.98
12	Tim	Goyer	3.96
13	Tim	Goyer	5.94
14	Tim	Goyer	1.99
15	Dan	Miller	1.98

```sql
SELECT
    C.FirstName
    ,C.LastName
    ,SUM(I.Total) SumTotal
FROM Customer C
JOIN Invoice I
    ON I.CustomerID = C.CustomerId
WHERE Country = 'USA'
    AND State = 'CA'
GROUP BY LastName, FirstName
ORDER BY SumTotal DESC
```

	FirstName	LastName	SumTotal
1	Dan	Miller	39.62
2	Tim	Goyer	38.62
3	Frank	Harris	37.62

The SUM function is used to add together the values in a numeric column. The sum total will be broken out by the columns in the GROUP BY clause. The SUM function will add both positive and negative values. NULL values are ignored by the SUM function.



Min() and Max()

- The MIN function returns the minimum value in a column
- The MAX function returns the maximum value in a column
- Both functions can work on string as well as numeric values

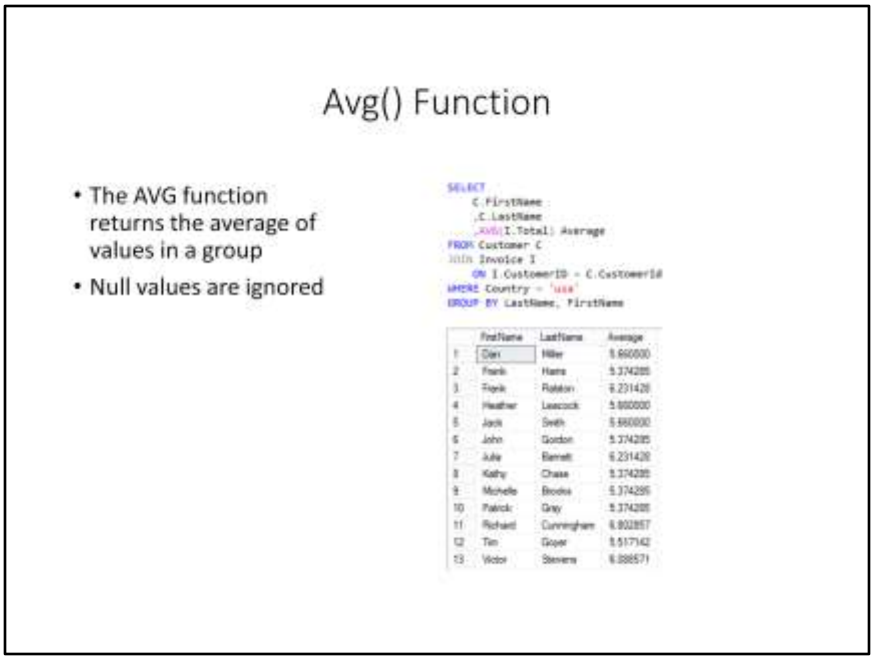
```
SELECT
    C.FirstName
    ,C.LastName
    ,MIN(I.Total) MinTotal
    ,MAX(I.Total) MaxTotal
FROM Customer C
JOIN Invoice I
    ON I.CustomerID = C.CustomerId
WHERE Country = 'usa'
GROUP BY LastName, FirstName
ORDER BY MaxTotal DESC
```

	FirstName	LastName	MinTotal	MaxTotal
1	Richard	Cunningham	0.99	23.86
2	Victor	Stevens	0.99	18.86
3	Frank	Ralston	0.99	15.86
4	Heather	Leacock	0.99	13.86
5	Jack	Smith	0.99	13.86
6	John	Gordon	0.99	13.86
7	Julia	Barnett	0.99	13.86
8	Kathy	Chase	0.99	13.86
9	Michelle	Brooks	0.99	13.86
10	Patrick	Gray	0.99	13.86
11	Tim	Goyer	1.98	13.86

```
SELECT
    C.FirstName
    ,C.LastName
    ,I.Total
FROM Customer C
JOIN Invoice I
    ON I.CustomerID = C.CustomerId
WHERE C.LastName = 'Cunningham'
ORDER BY Total
```

	FirstName	LastName	Total
1	Richard	Cunningham	0.99
2	Richard	Cunningham	1.98
3	Richard	Cunningham	1.98
4	Richard	Cunningham	3.96
5	Richard	Cunningham	5.94
6	Richard	Cunningham	8.91
7	Richard	Cunningham	23.86

The MIN and MAX functions are used to return the minimum or maximum value respectively of a column. The functions can be used both on string and numeric values. Both functions will ignore NULL values.



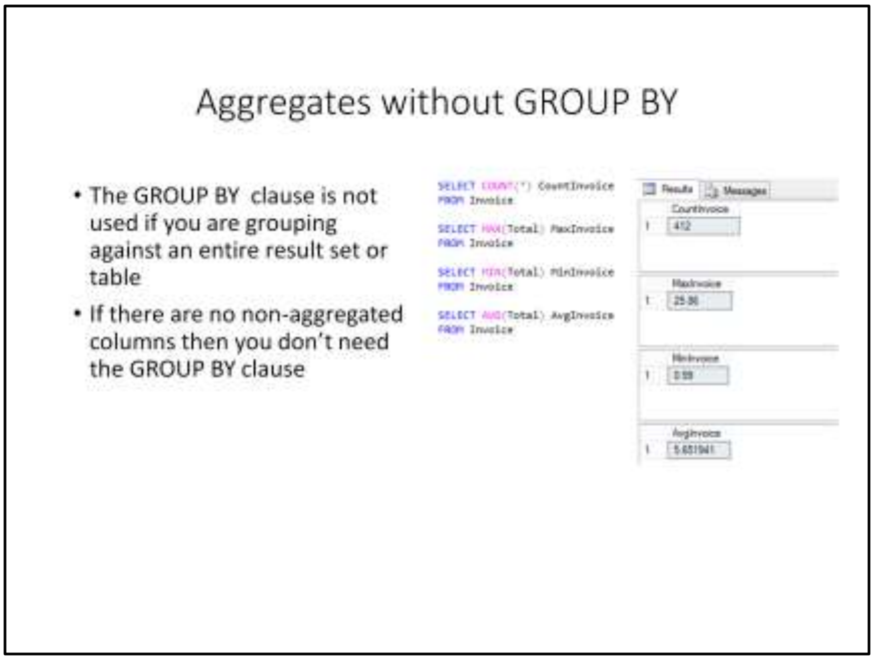
Avg() Function

- The AVG function returns the average of values in a group
- Null values are ignored

```
SELECT
    C.FirstName
    ,C.LastName
    ,AVG(I.Total) Average
FROM Customer C
JOIN Invoice I
    ON I.CustomerID = C.CustomerId
WHERE Country = 'usa'
GROUP BY LastName, FirstName
```

	FirstName	LastName	Average
1	Dan	Miller	5.660000
2	Frank	Harris	5.374285
3	Frank	Ralston	6.231428
4	Heather	Leacock	5.660000
5	Jack	Smith	5.660000
6	John	Gordon	5.374285
7	Julia	Barnett	6.231428
8	Kathy	Chase	5.374285
9	Michelle	Brooks	5.374285
10	Patrick	Gray	5.374285
11	Richard	Cunningham	6.802857
12	Tim	Goyer	5.517142
13	Victor	Stevens	6.088571

The AVG function returns the average of values within a group. As with other aggregate functions NULL values are ignored.



Aggregates without GROUP BY

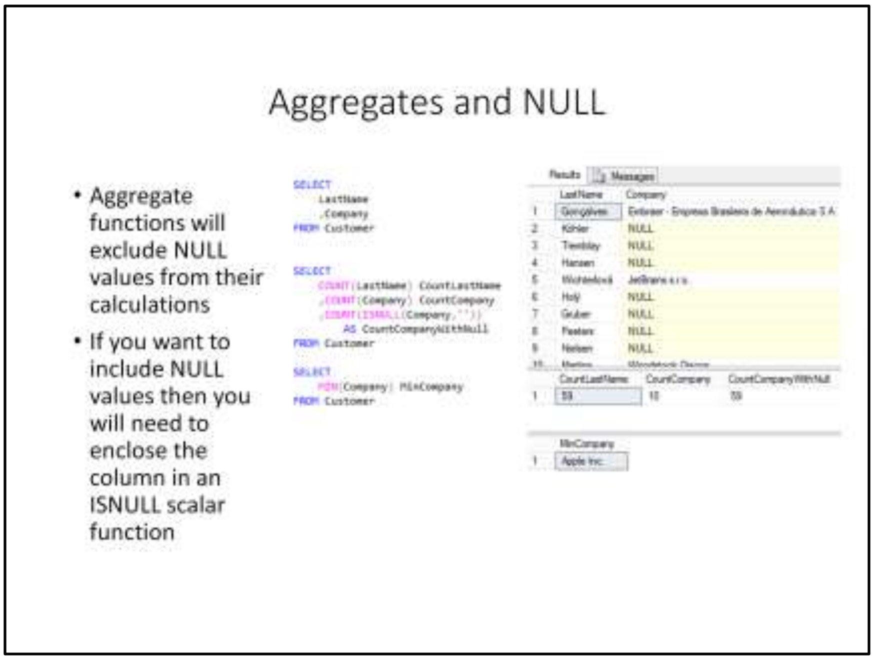
- The GROUP BY clause is not used if you are grouping against an entire result set or table
- If there are no non-aggregated columns then you don't need the GROUP BY clause

```
SELECT COUNT(*) CountInvoice
FROM Invoice

SELECT MAX(Total) MaxInvoice
FROM Invoice

SELECT MIN(Total) MinInvoice
FROM Invoice

SELECT AVG(Total) AvgInvoice
FROM Invoice
```

Results
Messages

```
   CountInvoice
1  412

   MaxInvoice
1  25.86

   MinInvoice
1  0.99

   AvgInvoice
1  5.651941
```

There are instances when a GROUP BY clause is not needed when executing aggregate functions. This is possible only when you don't include any non-aggregated columns in SELECT or ORDER BY clauses. In effect you are aggregating the entire table or result set into a single row.



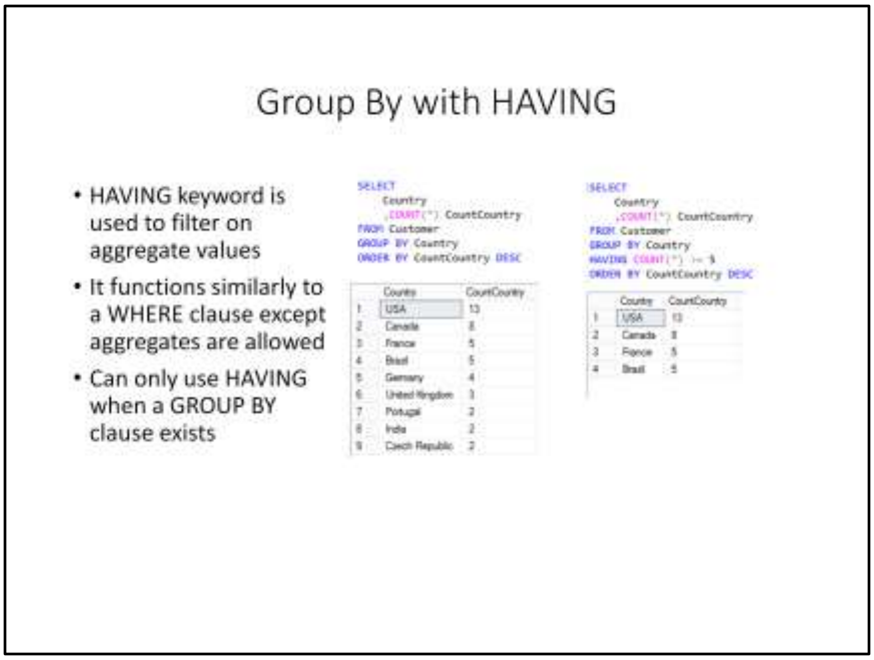
Aggregates and NULL

- Aggregate functions will exclude NULL values from their calculations
- If you want to include NULL values then you will need to enclose the column in an ISNULL scalar function

```
SELECT
    LastName
    ,Company
FROM Customer

SELECT
    COUNT(LastName) CountLastName
    ,COUNT(Company) CountCompany
    ,COUNT(ISNULL(Company,''))
        AS CountCompanyWithNull
FROM Customer

SELECT
    MIN(Company) MinCompany
FROM Customer
```

```
Results | Messages
    LastName     Company
1   Gonçalves    Embraer - Empresa Brasileira de Aeronáutica S.A.
2   Köhler       NULL
3   Tremblay     NULL
4   Hansen       NULL
5   Wichterlová  JetBrains s.r.o.
6   Holý         NULL
7   Gruber       NULL
8   Peeters      NULL
9   Nielsen      NULL
10  Martins      Woodstock Discos

    CountLastName   CountCompany   CountCompanyWithNull
1   59              10             59

    MinCompany
1   Apple Inc.
```

As I have mentioned in previous slides, aggregate functions will not include NULL values in their calculations or output. But what if you do want to include NULL values? In that case you need to use the ISNULL scalar function. In the example on this slide, the Company column has several NULL fields. I am able to count the NULL values by enclosing the Company column in an ISNULL function. In this case I am converting the NULL value to an empty string value.



Group By with HAVING

- HAVING keyword is used to filter on aggregate values
- It functions similarly to a WHERE clause except aggregates are allowed
- Can only use HAVING when a GROUP BY clause exists

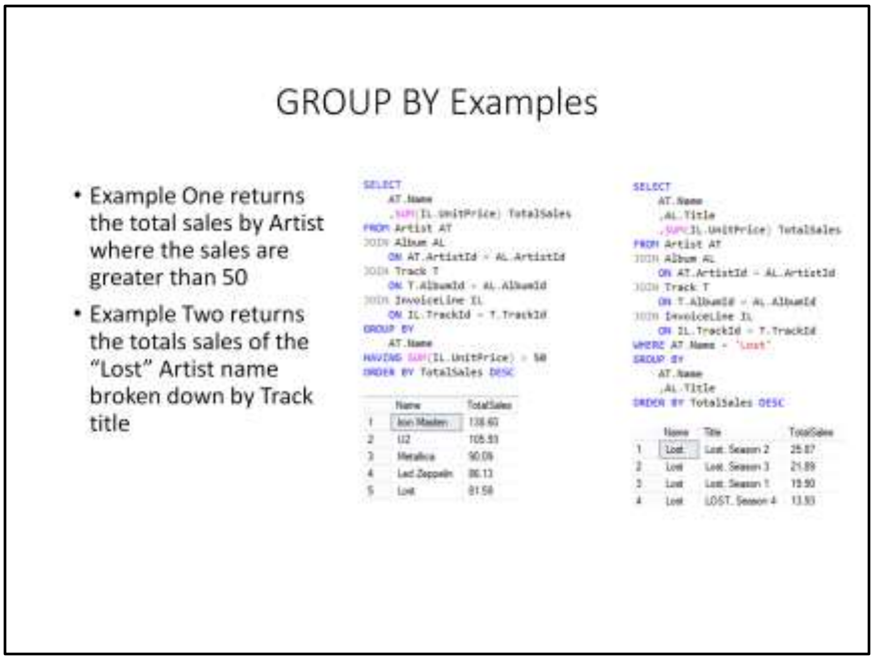
```
SELECT
    Country
    ,COUNT(*) CountCountry
FROM Customer
GROUP BY Country
ORDER BY CountCountry DESC
```

	Country	CountCountry
1	USA	13
2	Canada	8
3	France	5
4	Brazil	5
5	Germany	4
6	United Kingdom	3
7	Portugal	2
8	India	2
9	Czech Republic	2

```
SELECT
    Country
    ,COUNT(*) CountCountry
FROM Customer
GROUP BY Country
HAVING COUNT(*) >= 5
ORDER BY CountCountry DESC
```

	Country	CountCountry
1	USA	13
2	Canada	8
3	France	5
4	Brazil	5

Sometimes you will need to filter on the results of your aggregate functions. However the WHERE clause does not allow filtering on aggregate functions. This is where the HAVING clause comes into play. The HAVING clause performs similarly to the WHERE clause except it can filter aggregate functions. When used the HAVING clause must be placed immediately after the GROUP BY clause. In the example I want to return only those countries that have 5 or more customers. I accomplished this by adding the HAVING clause to the query.



GROUP BY Examples

- Example One returns the total sales by Artist where the sales are greater than 50
- Example Two returns the totals sales of the “Lost” Artist name broken down by Track title

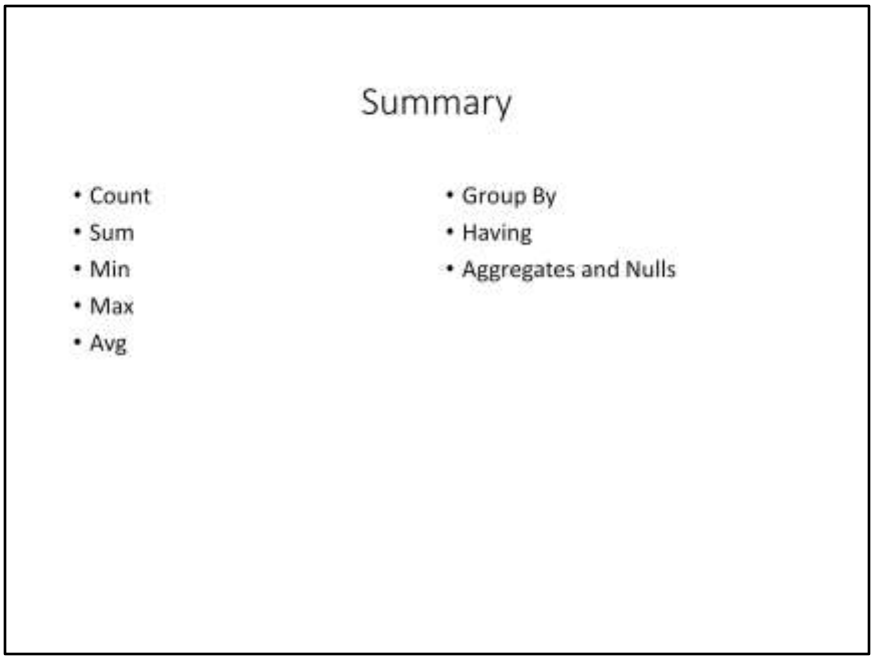
```sql
SELECT
    AT.Name
    ,SUM(IL.UnitPrice) TotalSales
FROM Artist AT
JOIN Album AL
    ON AT.ArtistId = AL.ArtistId
JOIN Track T
    ON T.AlbumId = AL.AlbumId
JOIN InvoiceLine IL
    ON IL.TrackId = T.TrackId
GROUP BY
    AT.Name
HAVING SUM(IL.UnitPrice) > 50
ORDER BY TotalSales DESC
```

	Name	TotalSales
1	Iron Maiden	138.60
2	U2	105.93
3	Metallica	90.09
4	Led Zeppelin	86.13
5	Lost	81.59

```sql
SELECT
    AT.Name
    ,AL.Title
    ,SUM(IL.UnitPrice) TotalSales
FROM Artist AT
JOIN Album AL
    ON AT.ArtistId = AL.ArtistId
JOIN Track T
    ON T.AlbumId = AL.AlbumId
JOIN InvoiceLine IL
    ON IL.TrackId = T.TrackId
WHERE AT.Name = 'Lost'
GROUP BY
    AT.Name
    ,AL.Title
ORDER BY TotalSales DESC
```

	Name	Title	TotalSales
1	Lost	Lost, Season 2	25.87
2	Lost	Lost, Season 3	21.89
3	Lost	Lost, Season 1	19.90
4	Lost	LOST, Season 4	13.93

On this slide I placed two additional examples to help illustrate the GROUP BY concepts. In the first example I want to view all artists that have sold more than $50 worth of tracks. I accomplish this by grouping by the artist name and adding a HAVING clause that only allows sums of the unit price that are greater than 50. The second example filters on the “Lost” artist and breaks out the total sales by Track Title. If you look at the GROUP BY clause, you’ll see that we are grouping by Artist Name and Track Title.



Summary

- Count
- Sum
- Min
- Max
- Avg

- Group By
- Having
- Aggregates and Nulls

This concludes the presentation on GROUP BY and aggregate functions.