

Section 5: Retrieving Data

- Select Statements
 - Top 100, Distinct
- Where Clause
 - And, Or, Not, Null, Not Null
 - Like, In, Between
- Common Wildcard symbols
- Order by
- Group by
- Aggregate Functions
- Alias Names
 - Concatenation
- Having
- Case When



Section 5: Section Introduction



Options





Add to watchlist NasdagGS - NasdagGS Real Time Price, Currency in USD

Conversations

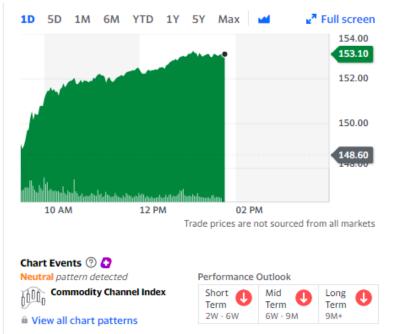
Statistics

153.13 +4.52 (+3.05%)

Company Outlook 😝

As of 1:47PM EDT. Market open.

Summary	Company Outlook	Chart Con	versations 50
Previous Close	148.60	Market Cap	2.531T
Open	149.00	Beta (5Y Monthly)	1.20
Bid	152.99 x 900	PE Ratio (TTM)	29.98
Ask	153.00 x 900	EPS (TTM)	5.11
Day's Range	148.61 - 153.25	Earnings Date	Oct 27, 2021 - Nov 01, 2021
52 Week Range	103.10 - 153.25	Forward Dividend & Yield	0.88 (0.59%)
Volume	61,958,523	Ex-Dividend Date	Aug 06, 2021
Avg. Volume	76,331,758	1y Target Est	166.37
Fair Value 🕜 🖸		Related Research (୭ 🖸
XX.XX	Near Fair Value	🖹 Technical Assess	ment: Bullish in
-1% Est. Return		A Market Update:	AAPL, ECL, ODF
View details			, . ,



Financials

2 Visitors trend 2W ↓ 10W ↑ 9M ↑

Profile

Historical Data

Section 5: Section Introduction

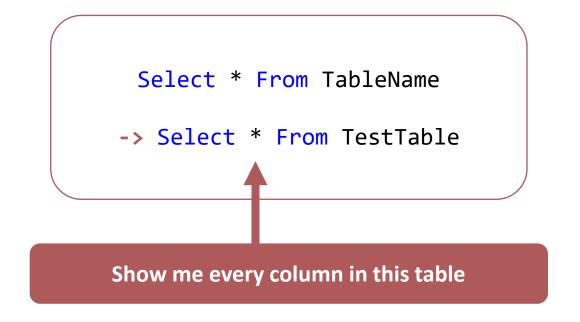


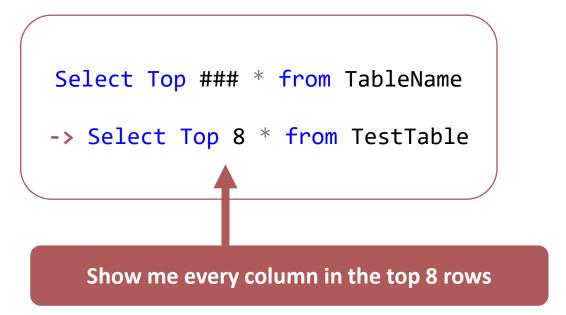
```
Select
PatientName
,PatientState
,Sum(Charges) as Charges
Sum(Visits) as Visits
Sum(Charges)/Sum(Visits) as AvgChargePerVisit
From TestTable
Where Gender = 'F'
and PatientState in ('GA', 'FL')
Group by
PatientName
,PatientState
Having Sum(Charges) >= 900
Order by PatientName
```

Select
From
Where
Group by
Having
Order by

Section 5: Select Statements

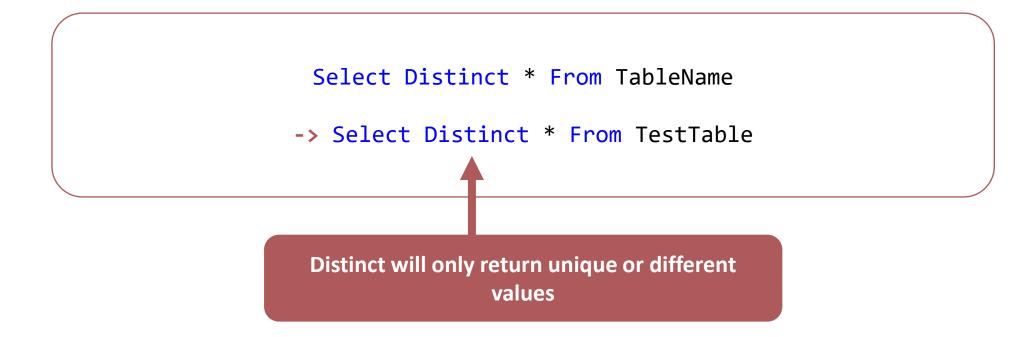






Section 5: Select Statements





Section 5: Select Statements



```
Select Column1
    ,Column2
    ,Column3
from TableName
```

-> Select
PatientName
,Visits
from TestTable

Specify each column

Specify each Distinct column

Section 5: Where Clauses



Where Clause Rules

- 1. Used after the Tablename has been specified in the From
- 2. Used to return values
 with a specific
 condition
- 3. Can also be used in Update and Delete statements, which will be shown next section

```
-> Select
    PatientName
    ,PatientState
    from TestTable
where PatientState = 'GA'
```





Symbol	Description	Example
And	Filters values based on two conditions	<pre>Select * from TableName where Column1 = 'Condition' and Column2 = 'Condition'</pre>
Not	Filter values if condition is not true	Select * from TableName where not Column1 = 'Condition'
Or	Filters values based on one of two conditions	<pre>Select * from TableName where Column1 = 'Condition' or Column2 = 'Condition'</pre>
Like	Search for a patten of values within a column (Uses wildcard symbols)	Select * from TestTable where Column1 like 'Condition1'
Between	Returns values between values in a range	Select * from TableName where Column1 between 'Condition1' and 'Condition2'
In	Specify multiple values within a column	<pre>Select * from TestTable where Column1 in ('Value1','Value2','Value3')</pre>

Section 5: Wildcard Symbols



Functions	Description	Example
0	Finds any character within the brackets	b[eai]d will find bed, bad, bid b[ea]d will find bed, bad, but not bid
_	Finds any single character	b_d will find bad, bed, bid, bod, bud
%	Finds any pattern after or before the symbol	da% will find data, date, day, dance, etc.
-	Finds a range of characters	b[a-e]d will find bad, bbd, bcd, bdd, bed

These wildcard symbols are used in Like conditions

Section 5: Order by



Order By Rules

- An Order by statement sorts the results in ascending or descending order
- 2. If you don't specify whether to sort by ascending or descending, then ascending is the default
- 3. You need to specify which column by either using the column number or name (see video example)

```
-> Select Distinct
   PatientName
   ,PatientState
from TestTable
where Gender = 'M'
Order by PatientState DESC
--ASC
--Order by 2 DESC --ASC
```

Section 5: Aggregate Functions



Common Functions	Description	
Min	Returns the smallest value in the specified column	
Max	Returns the largest value in the specified column	
Count	Returns the number of rows	
Sum	Returns the total sum of a numeric column	
Avg	Returns the average of a numeric column	

Select Sum(Value)
from TableName

-> Select Count(PatientName)
 From TestTable

Section 5: Group by



Group By Rules

- A Group by Statement is necessary in SSMS and Azure Data studio when using aggregate functions
- 2. Groups rows into summary rows.
- 3. When using a group by without aggregates it acts like a distinct. Although they may return similar returns, they are different.

Section 5: Aggregate Functions



A group by is necessary if you are aggregating with another column (see video example)

```
Select Column1
, Sum(Value)
from TableName
   Group by
   Column1
```

-> Select
 PatientState
,Count(PatientName)
 From TestTable
Group by PatientState

Section 5: Alias Names



In the previous example our aggregate column was not named.

```
Select Column1 as 'AliasName'
    , Sum(Value) as 'AliasName'
from TableName
    Group by
    Column1

-> Select
PatientState as 'State'
,Count(PatientName) as 'CountofPatients'
From TestTable
Group by PatientState
```

Section 5: Concatenation



```
Select
Concat(Column1, Column2) as 'AliasName'
,Count(Value) as 'AliasName'
From TableName
Group by Column1, Column2

-> Select
Concat(PatientState,' - ',Gender) as 'StateGender'
,Count(Distinct PatientName) as 'CountofPatients'
From TestTable
Group by PatientState,Gender
```





Symbol	Description	Where Example	Having Example
=	Equal to	<pre>Select * from TableName where Column1 = 'Condition'</pre>	<pre>Select * from TableName Group by Column1 Having Sum(Column1) = 'Condition'</pre>
>	Greater than	Select * from TableName where Column1 > 'Condition'	Select * from TableName Group by Column1 Having Sum(Column1) > 'Condition'
<	Less than	Select * from TableName where Column1 < 'Condition'	Select * from TableName Group by Column1 Having Sum(Column1) < 'Condition'
>=	Greater than or equal	<pre>Select * from TableName where Column1 >= 'Condition'</pre>	<pre>Select * from TableName Group by Column1 Having Sum(Column1) >= 'Condition'</pre>
<=	Less than or equal	<pre>Select * from TableName where Column1 <= 'Condition'</pre>	Select * from TableName Group by Column1 Having Sum(Column1) <= 'Condition'
<> or !=	Not Equal	Select * from TableName where Column1 <> 'Condition'	Select * from TableName Group by Column1 Having Sum(Column1) <> 'Condition'
		<pre>Select * from TableName where Column1 != 'Condition'</pre>	<pre>Select * from TableName Group by Column1 Having Sum(Column1) != 'Condition'</pre>

Data Analytics: Introduction to SQL using Healthcare Data

Section 5: Having



The Having clause is very similar to the where clause. However, where cannot be used with aggregate functions.

Using a Group by is necessary when using the Having clause.

```
Select
Column1, Column2
From TableName
Group by Column1, Column2
Having Condition
-> Select distinct
    PatientName
   ,PatientState
From TestTable
Group by
    PatientName
   ,PatientState
Having Sum(Charges) between '900' and '10000'
```

Section 5: Having Example



```
-> Select top 3
PatientName
PatientState
From TestTable
Group by
PatientName
PatientState
Having Sum(Charges) between
'900' and '10000'
order by 1 desc
```

VS.

```
-> Select top 3
PatientName
PatientState
from TestTable
where Charges between '900'
and '10000'
order by 1 desc
```

Section 5: Case When



Case statement searches
through specified
columns finding
instances where
condition is met and
returning specified
values. Logic is if then

```
Select
     Column
    ,Case When Column = 'Condition' Then 'Value'
          Else Column
           END
    From TestTable
-> Select
    PatientID
    , PatientName
    ,Case When PatientState = 'GA' Then 'Georgia'
          Else PatientState
           END as 'PatientState'
   From TestTable
```

Section 5: Case When



```
-> Select Distinct
   PatientID
    , PatientName
    ,Case When PatientState = 'GA' Then 'Georgia'
 When PatientState = 'AL' Then 'Alabama'
 When PatientState = 'AK' Then 'Alaska'
 When PatientState = 'AZ' Then 'Arizona'
 When PatientState = 'UT' Then 'Utah'
 Else PatientState
          END as 'PatientState'
From TestTable
Where PatientState in ('GA','AL','AK','AZ','UT')
```

Section 5: Section Introduction



```
Select
PatientName
,PatientState
,Sum(Charges) as Charges
Sum(Visits) as Visits
Sum(Charges)/Sum(Visits) as AvgChargePerVisit
From TestTable
Where Gender = 'F'
and PatientState in ('GA', 'FL')
Group by
PatientName
,PatientState
Having Sum(Charges) >= 900
Order by PatientName
```

Select
From
Where
Group by
Having
Order by

Section 5: Self Evaluation



Step 1: Connect to SQLCourse_DB (or create this database if you haven't already

Step 2: Select PatientState, Gender, and average number of visits

Step 3: Write a case statement and spell out each gender

Step 4: Add a second case statement, when the average visits is greater than 10 then insert this value -

"Greater than 10 visits"

Step 5: Sort by PatientState in ascending order

Step 6: Filter to only states with charges between 1,200 and 20,000

Step 7: Give each column an alias name