1. Demonstrate cleaning data with Text Functions using Excel.

i. Removing Unwanted Characters from Text
ii. Finding required Text Patterns with the Text Functions

The data that you obtain from different sources many not be in a form ready for analysis. In this chapter, you will understand how to prepare your data that is in the form of text for analysis.

Initially, you need to clean the data. Data cleaning includes removing unwanted characters from text. Next, you need to structure the data in the form you require for further analysis. You can do the same by −

- Finding required text patterns with the text functions.
- Extracting data values from text.
- Formatting data with text functions.
- Executing data operations with the text functions.

Removing Unwanted Characters from Text

When you import data from another application, it can have nonprintable characters and/or excess spaces. The excess spaces can be −

- leading spaces, and/or
- extra spaces between words.

If you sort or analyze such data, you will get erroneous results.

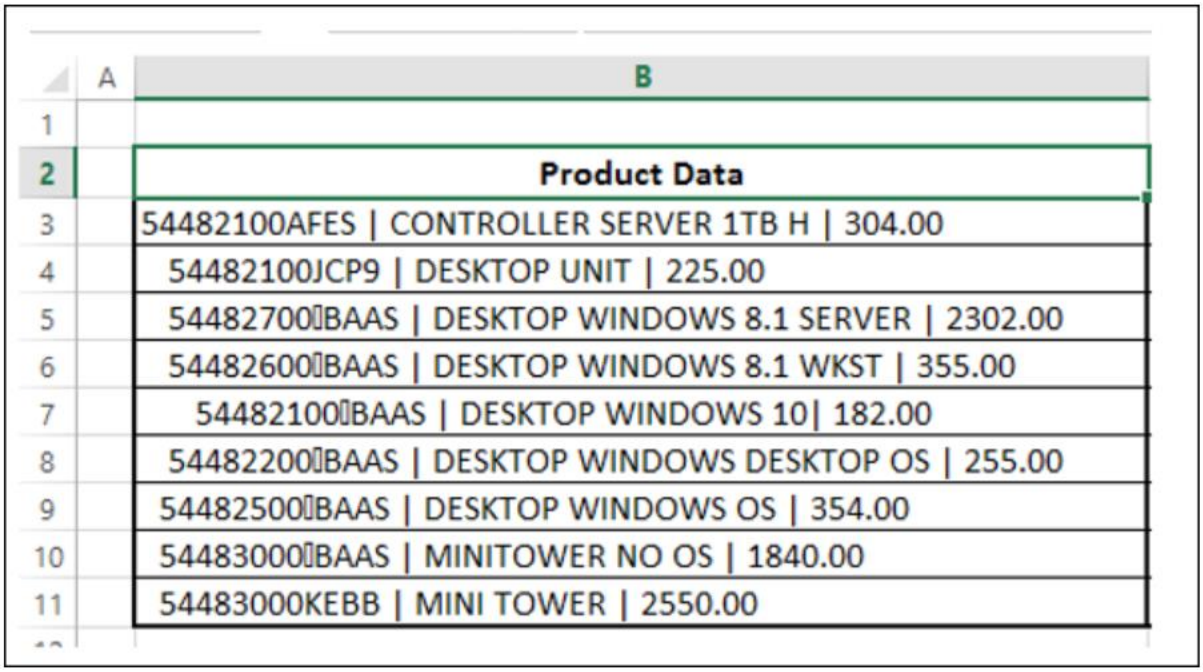
Consider the following example –

Product Data

54482100AFES | CONTROLLER SERVER 1TB H | 304.00
54482100JCP9 | DESKTOP UNIT | 225.00
54482700BAAS | DESKTOP WINDOWS 8.1 SERVER | 2302.00
54482600BAAS | DESKTOP WINDOWS 8.1 WKST | 355.00
54482100BAAS | DESKTOP WINDOWS 10| 182.00
54482200BAAS | DESKTOP WINDOWS DESKTOP OS | 255.00
54482500BAAS | DESKTOP WINDOWS OS | 354.00
54483000BAAS | MINITOWER NO OS | 1840.00
54483000KEBB | MINI TOWER | 2550.00

This is the raw data that you have obtained on product information containing the Product ID, Product description and the price. The character "|" separates the field in each row.

When you import this data into Excel worksheet, it looks as follows −



	A	B
1		
2		**Product Data**
3		54482100AFES | CONTROLLER SERVER 1TB H | 304.00
4		54482100JCP9 | DESKTOP UNIT | 225.00
5		54482700BAAS | DESKTOP WINDOWS 8.1 SERVER | 2302.00
6		54482600BAAS | DESKTOP WINDOWS 8.1 WKST | 355.00
7		54482100BAAS | DESKTOP WINDOWS 10| 182.00
8		54482200BAAS | DESKTOP WINDOWS DESKTOP OS | 255.00
9		54482500BAAS | DESKTOP WINDOWS OS | 354.00
10		54483000BAAS | MINITOWER NO OS | 1840.00
11		54483000KEBB | MINI TOWER | 2550.00

As you observe, the entire data is in a single column. You need to structure this data to perform data analysis. However, initially you need to clean the data.

You need to remove any nonprintable characters and excess spaces that might be present in the data. You can use the CLEAN function and TRIM function for this purpose.

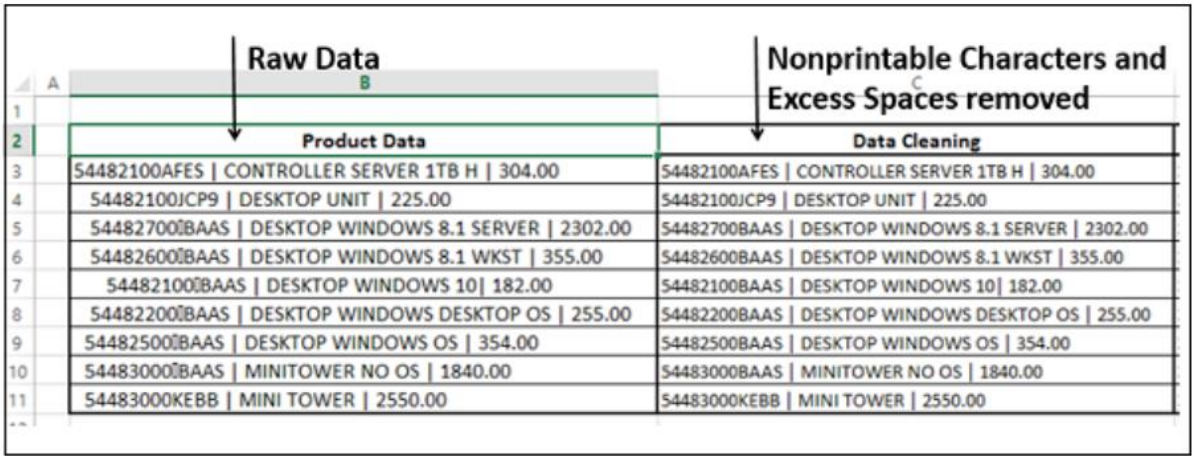
S.No.	Function & Description
1.	**CLEAN** Removes all nonprintable characters from text
2.	**TRIM** Removes spaces from text

- Select the Cells C3 – C11.
- Type =TRIM (CLEAN (B3)) and then press CTRL + Enter.

The formula is filled in the cells C3 – C11.

	A	B	C
1			
2		**Product Data**	**Data Cleaning**
3		54482100AFES | CONTROLLER SERVER 1TB H | 304.00	=TRIM(CLEAN(B3))
4		54482100JCP9 | DESKTOP UNIT | 225.00	=TRIM(CLEAN(B4))
5		54482700BAAS | DESKTOP WINDOWS 8.1 SERVER | 2302.00	=TRIM(CLEAN(B5))
6		54482600BAAS | DESKTOP WINDOWS 8.1 WKST | 355.00	=TRIM(CLEAN(B6))
7		54482100BAAS | DESKTOP WINDOWS 10| 182.00	=TRIM(CLEAN(B7))
8		54482200BAAS | DESKTOP WINDOWS DESKTOP OS | 255.00	=TRIM(CLEAN(B8))
9		54482500BAAS | DESKTOP WINDOWS OS | 354.00	=TRIM(CLEAN(B9))
10		54483000BAAS | MINITOWER NO OS | 1840.00	=TRIM(CLEAN(B10))
11		54483000KEBB | MINI TOWER | 2550.00	=TRIM(CLEAN(B11))

The result will be as shown below −

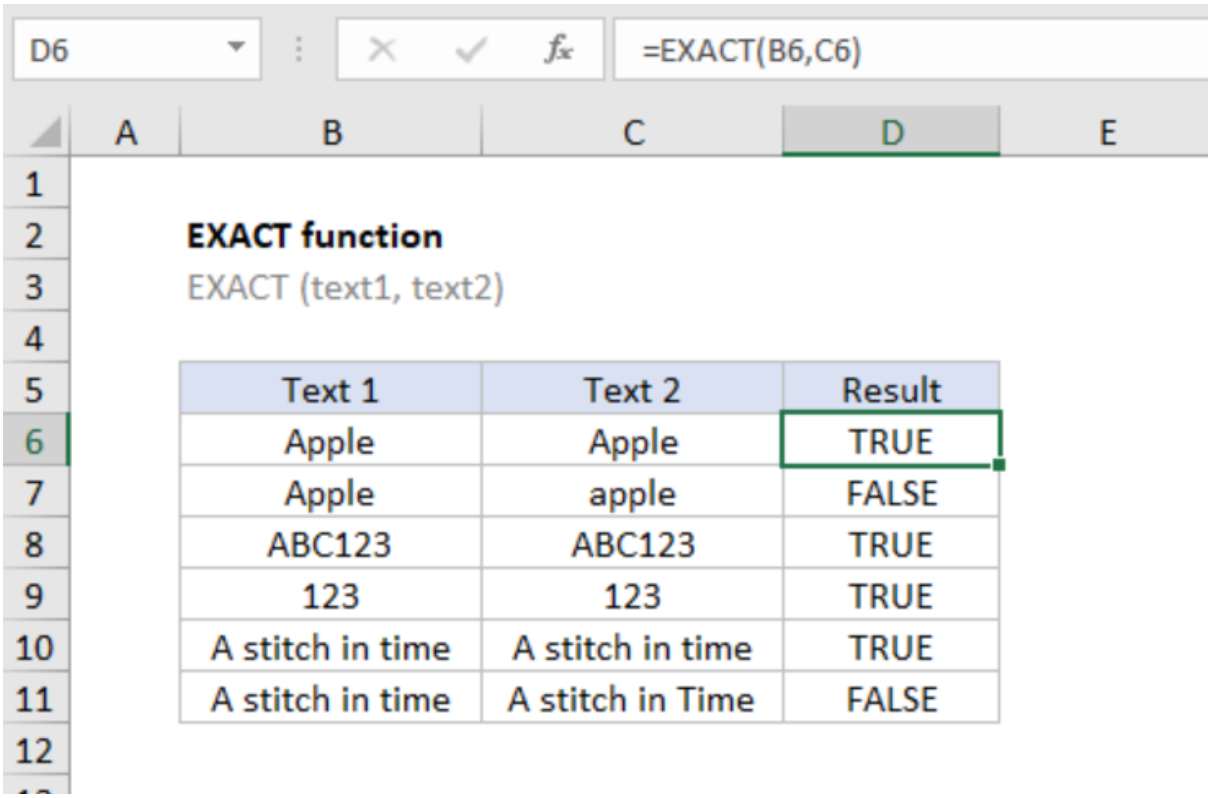


Finding required Text Patterns with the Text Functions

To structure your data, you might have to do certain Text Pattern matching based on which you can extract the Data Values. Some of the Text Functions that are useful for this purpose are –

S.No.	Function & Description
1.	**EXACT** Checks to see if two text values are identical
2.	**FIND** Finds one text value within another (case-sensitive)
3.	**SEARCH** Finds one text value within another (not case-sensitive)

EXACT FUNCTION



	Text 1	Text 2	Result
6	Apple	Apple	TRUE
7	Apple	apple	FALSE
8	ABC123	ABC123	TRUE
9	123	123	TRUE
10	A stitch in time	A stitch in time	TRUE
11	A stitch in time	A stitch in Time	FALSE

The Excel EXACT function compares two text strings, taking into account upper and lower case characters, and returns TRUE if they are the same, and FALSE if not. EXACT is case-sensitive.

Purpose

Compare two text strings

Return value

A Boolean value (TRUE or FALSE)

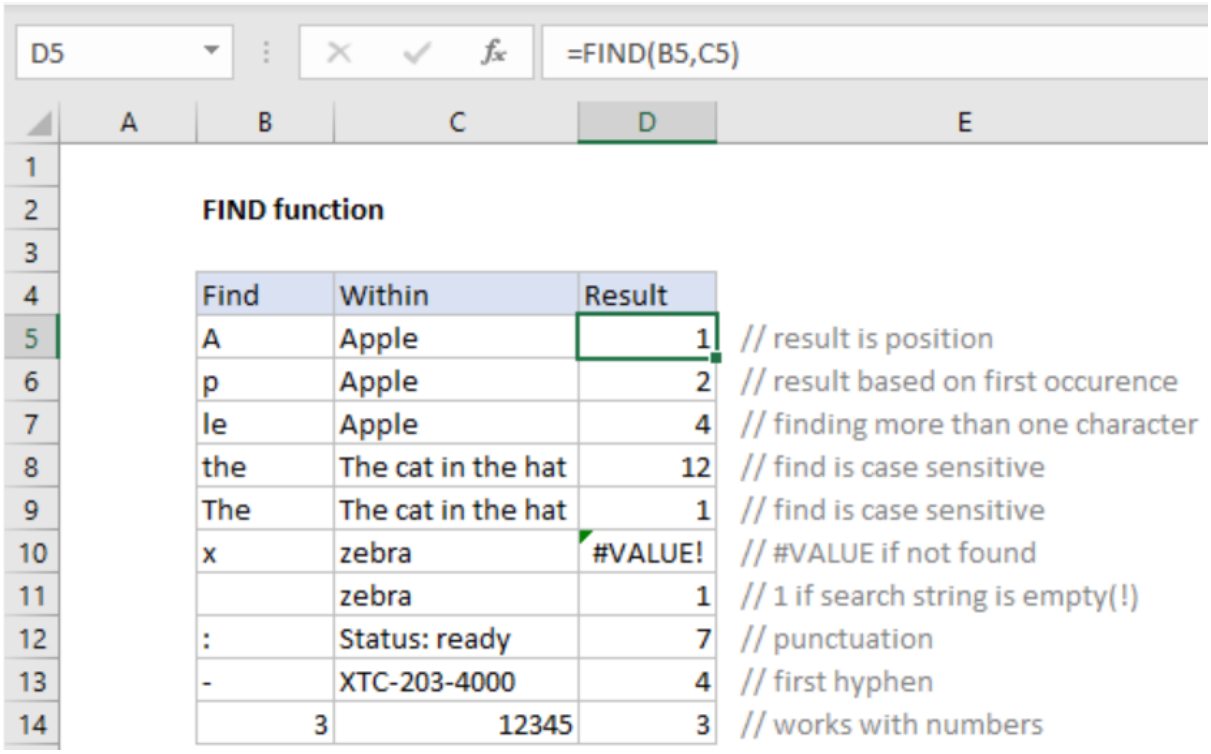
Syntax

=EXACT (text1, text2)

Arguments

- **text1** - The first text string to compare.
- **text2** - The second text string to compare.

FIND FUNCTION



D5 =FIND(B5,C5)

FIND function

	Find	Within	Result	
5	A	Apple	1	// result is position
6	p	Apple	2	// result based on first occurence
7	le	Apple	4	// finding more than one character
8	the	The cat in the hat	12	// find is case sensitive
9	The	The cat in the hat	1	// find is case sensitive
10	x	zebra	#VALUE!	// #VALUE if not found
11		zebra	1	// 1 if search string is empty(!)
12	:	Status: ready	7	// punctuation
13	-	XTC-203-4000	4	// first hyphen
14	3	12345	3	// works with numbers

The Excel FIND function returns the position (as a number) of one text string inside another. When the text is not found, FIND returns a #VALUE error.

Purpose

Get the location of text in a string

Return value

A number representing the location of find_text.

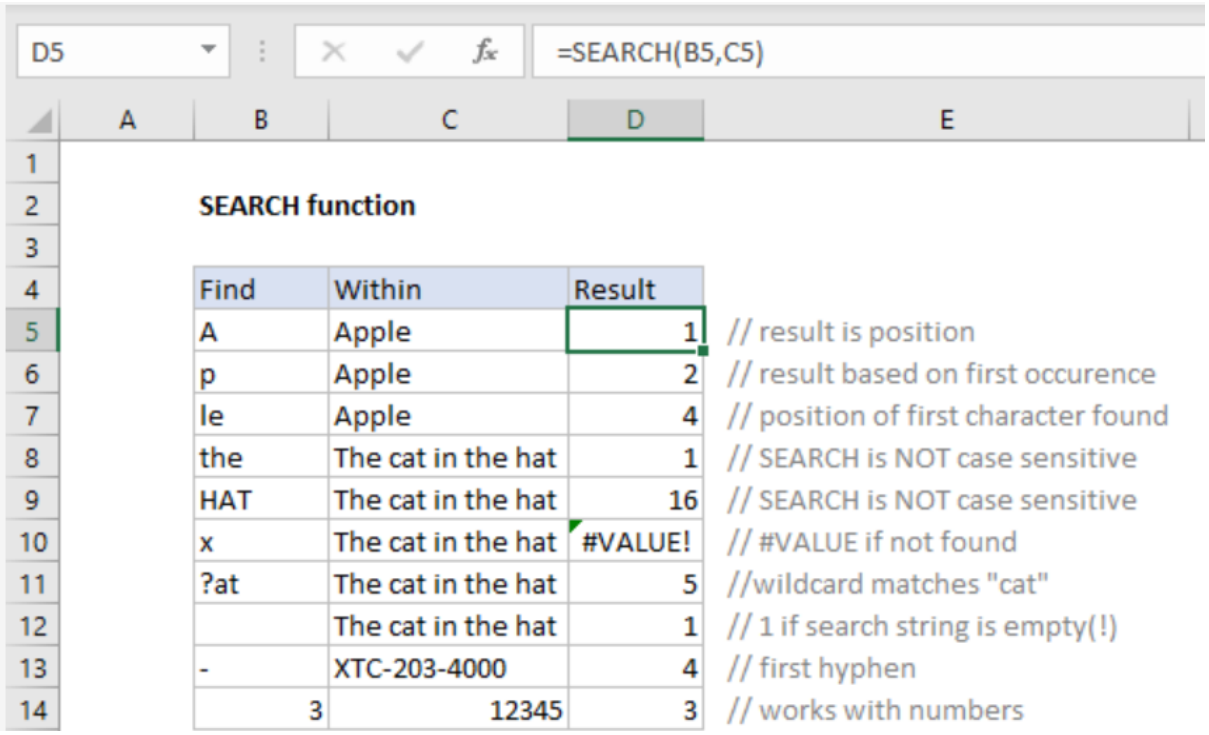
Syntax

=FIND (find_text, within_text, [start_num])

Arguments

- **find_text** - The text to find.
- **within_text** - The text to search within.
- **start_num** - [optional] The starting position in the text to search. Optional, defaults to 1.

SEARCH FUNCTION



The Excel SEARCH function returns the location of one text string inside another. SEARCH returns the position of the first character of find_text inside within_text. Unlike FIND, SEARCH allows wildcards, and is *not* case-sensitive.

Purpose

Get the location of text in a string

Return value

A number representing the location of find_text.

Syntax

=SEARCH (find_text, within_text, [start_num])

Arguments

- **find_text** - The text to find.
- **within_text** - The text to search within.
- **start_num** - [optional] Starting position in the text to search. Optional, defaults to 1.

2. Demonstrate Conditional Formatting in Excel.

i. Highlight cells rules

ii. Top / Bottom Rules

iii. Data Bars

In Microsoft Excel, you can use **Conditional Formatting** for data visualization. You have to specify formatting for a cell range based on the contents of the cell range. The cells that meet the specified conditions would be formatted as you have defined.

Example

In a range containing the sales figures of the past quarter for a set of salespersons, you can highlight those cells representing who have met the defined target, say, $2500.

You can set the condition as total sales of the person >= **$2500** and specify a color code green. Excel checks each cell in the range to determine whether the condition you specified, i.e., total sales of the person >= $2500 is satisfied.

Excel applies the format you chose, i.e. the green color to all the cells that satisfy the condition. If the content of a cell does not satisfy the condition, the formatting of the cell remains unchanged. The result is as expected, only for the salespersons who have met the target, the cells are highlighted in green – a quick visualization of the analysis results.

You can specify any number of conditions for formatting by specifying **Rules**. You can pick up the rules that match your conditions from

- Highlight cells rules
- Top / Bottom rules

You can also define your own rules. You can −

- Add a rule
- Clear an existing rule
- Manage the defined rules

Further, you have several formatting options in Excel to choose the ones that are appropriate for your Data Visualization −

- Data Bars
- Color Scales
- Icon Sets

Conditional formatting has been promoted over the versions Excel 2007, Excel 2010, Excel 2013. The examples you find in this chapter are from Excel 2013.

In the following sections, you will understand the conditional formatting rules, formatting options and how to work with rules.

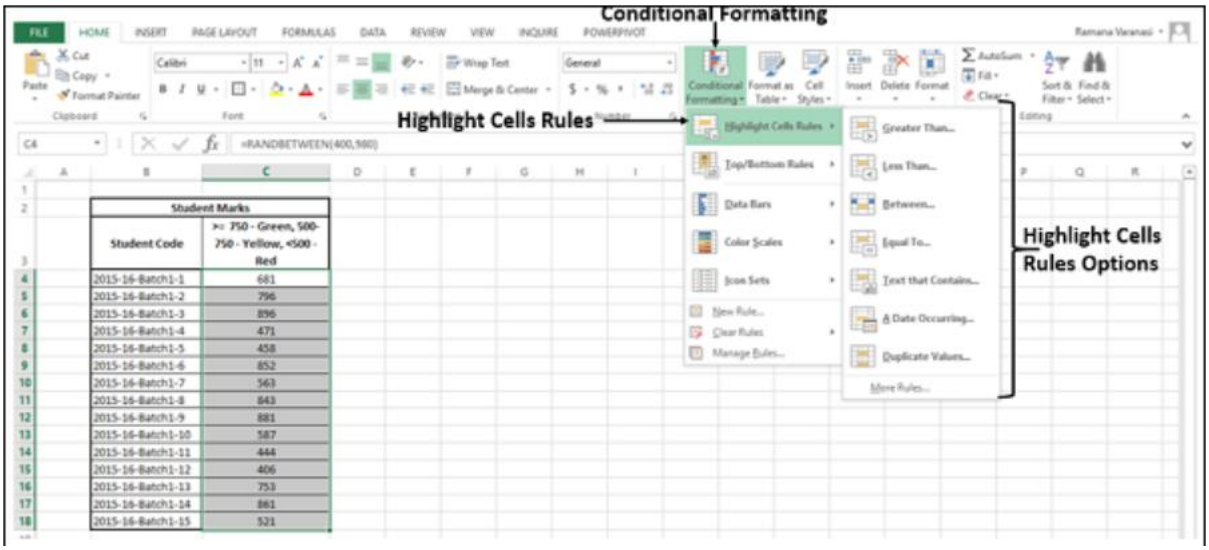
Highlight Cells Rules

You can use **Highlight Cells** rule to assign a format to cells whose contents meet any of the following criteria −

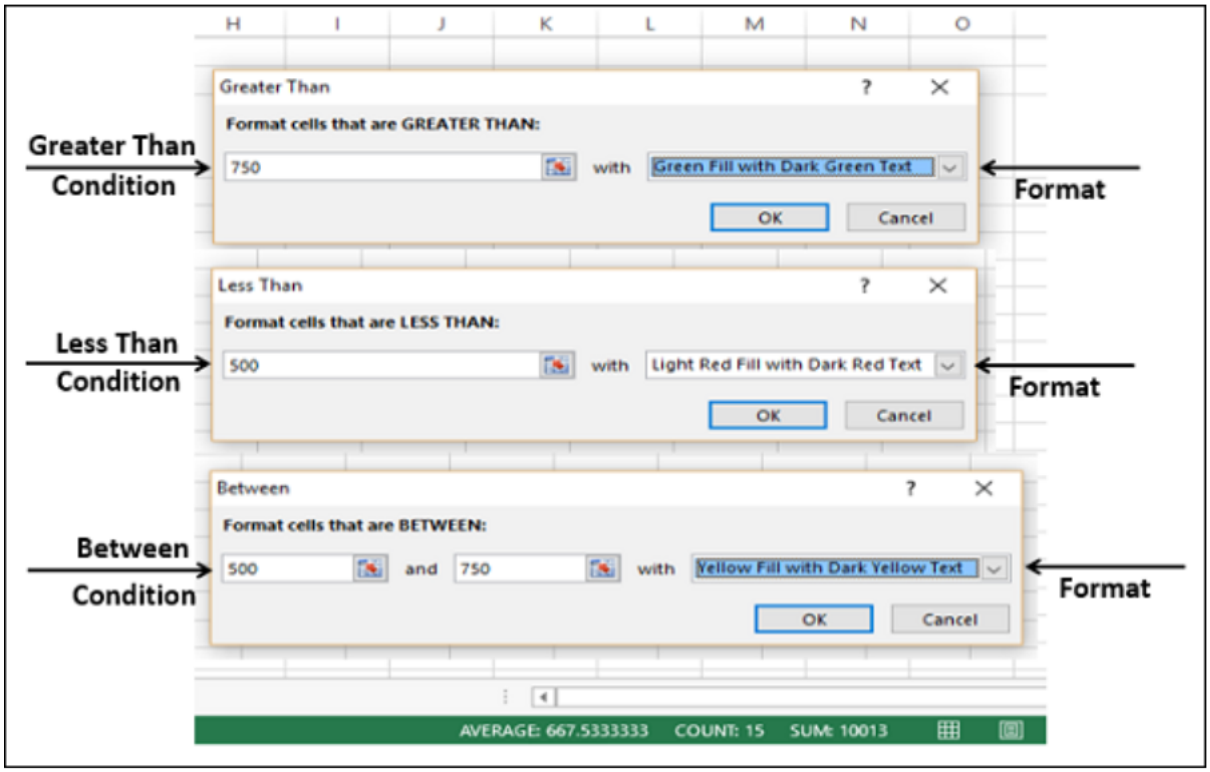
- Numbers within a given numerical range −
  - Greater Than
  - Less Than
  - Between
  - Equal To
- Text that contains a given text string.
- Date occurring within a given range of dates relative to the current date −
  - Yesterday
  - Today
  - Tomorrow
  - In the last 7 days
  - Last week
  - This week
  - Next week
  - Last month
  - This Month
  - Next month
- Values that are duplicate or unique.

Follow the steps to conditionally format cells −

- Select the range to be conditionally formatted.
- Click **Conditional Formatting** in the **Styles** group under **Home** tab.
- Click **Highlight Cells Rules** from the drop-down menu.



- Click **Greater Than** and specify >750. Choose green color.
- Click **Less Than** and specify < 500. Choose red color.
- Click **Between** and specify 500 and 750. Choose yellow color.



The data will be highlighted based on the given conditions and the corresponding formatting.

	A	B	C
1			
2		Student Marks	
3		Student Code	>= 750 - Green, 500-750 - Yellow, <500 - Red
4		2015-16-Batch1-1	681
5		2015-16-Batch1-2	796
6		2015-16-Batch1-3	896
7		2015-16-Batch1-4	471
8		2015-16-Batch1-5	458
9		2015-16-Batch1-6	852
10		2015-16-Batch1-7	563
11		2015-16-Batch1-8	843
12		2015-16-Batch1-9	881
13		2015-16-Batch1-10	587
14		2015-16-Batch1-11	444
15		2015-16-Batch1-12	406
16		2015-16-Batch1-13	753
17		2015-16-Batch1-14	861
18		2015-16-Batch1-15	521

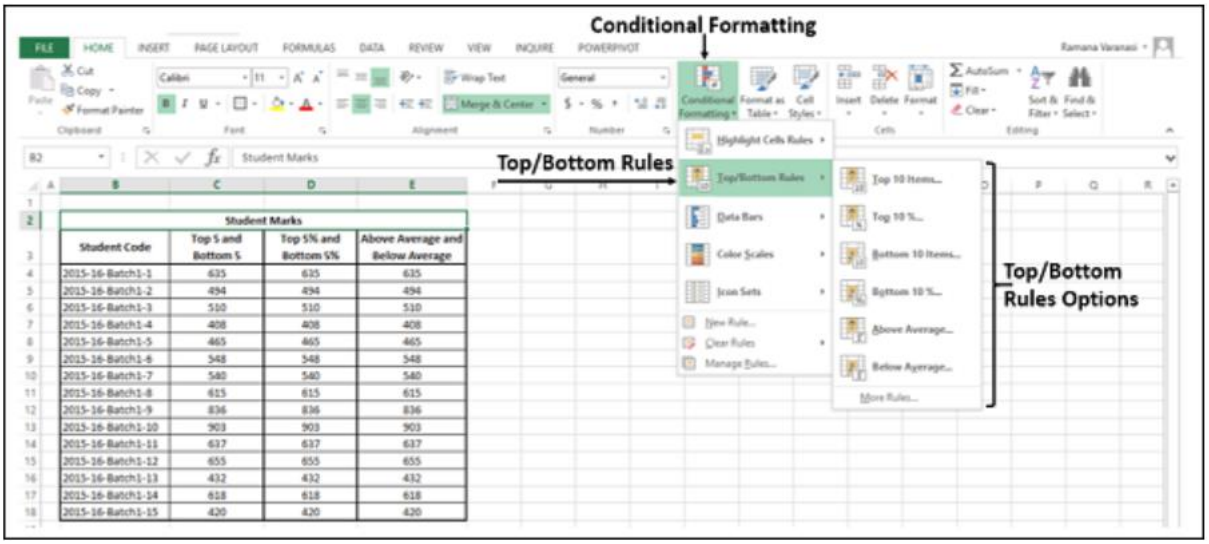
Top / Bottom Rules

You can use **Top / Bottom Rules** to assign a format to cells whose contents meet any of the following criteria −

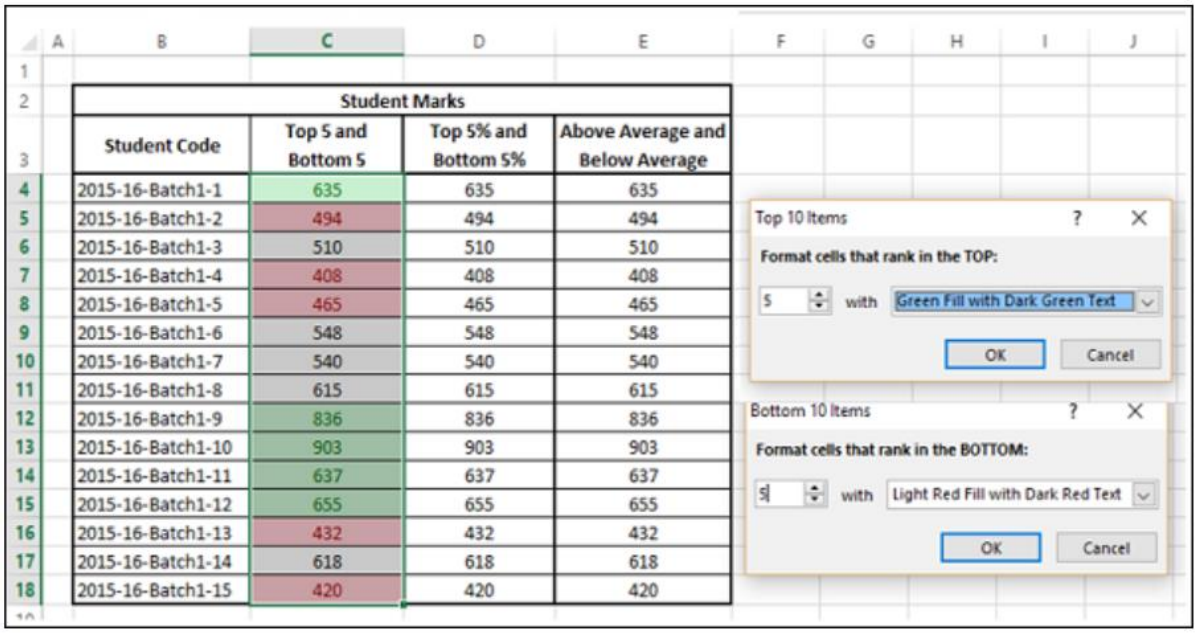
- **Top 10 items** − Cells that rank in the top N, where 1 <= N <= 1000.
- **Top 10%** − Cells that rank in the top n%, where 1 <= n <= 100.
- **Bottom 10 items** − Cells that rank in the bottom N, where 1 <= N <= 1000.
- **Bottom 10%** − Cells that rank in the bottom n%, where 1 <= n <= 100.
- **Above average** − Cells that are above average for the selected range.
- **Below average** − Cells that are below average for the selected range.

Follow the steps given below to assign the Top/Bottom rules.

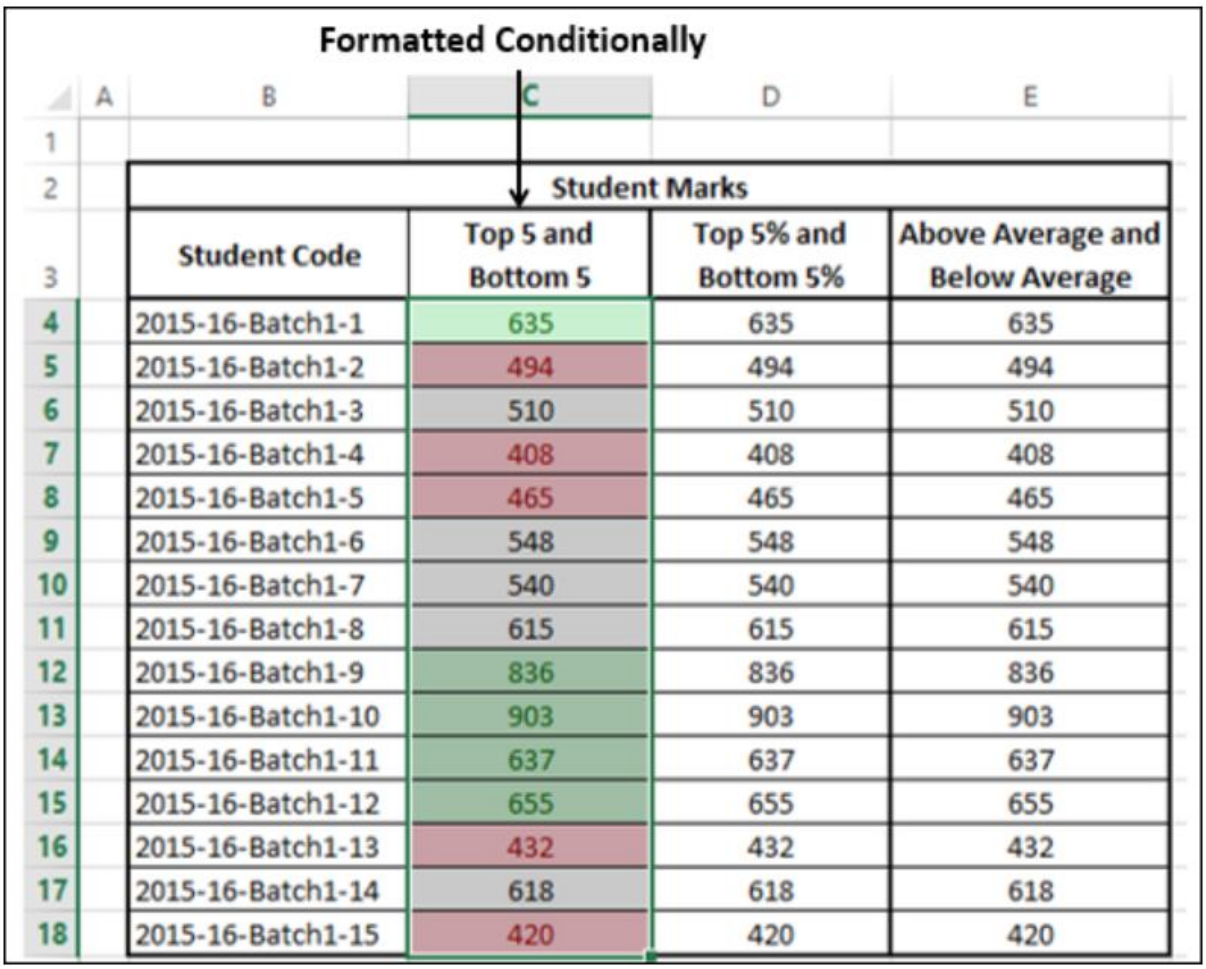
- Select the range to be conditionally formatted.
- Click **Conditional Formatting** in the **Styles** group under **Home** tab.
- Click **Top/Bottom Rules** from the drop-down menu. Top/Bottom rules options appear.



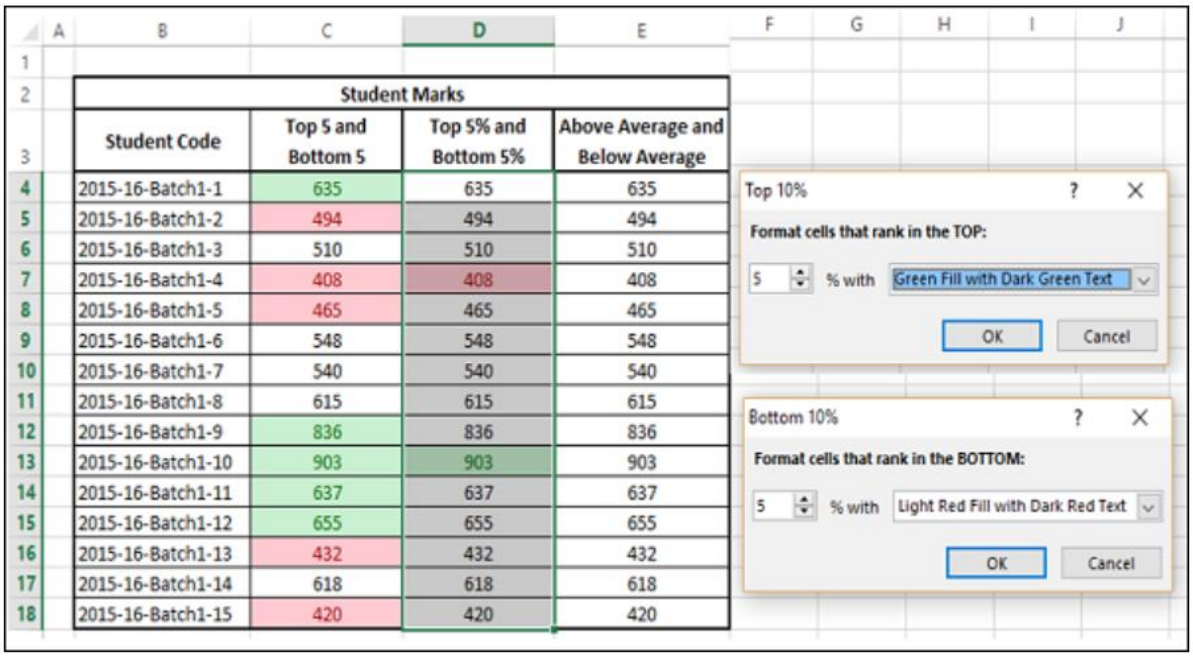
- Click **Top Ten Items** and specify 5. Choose green color.
- Click **Bottom Ten Items** and specify 5. Choose red color.



The data will be highlighted based on the given conditions and the corresponding formatting.

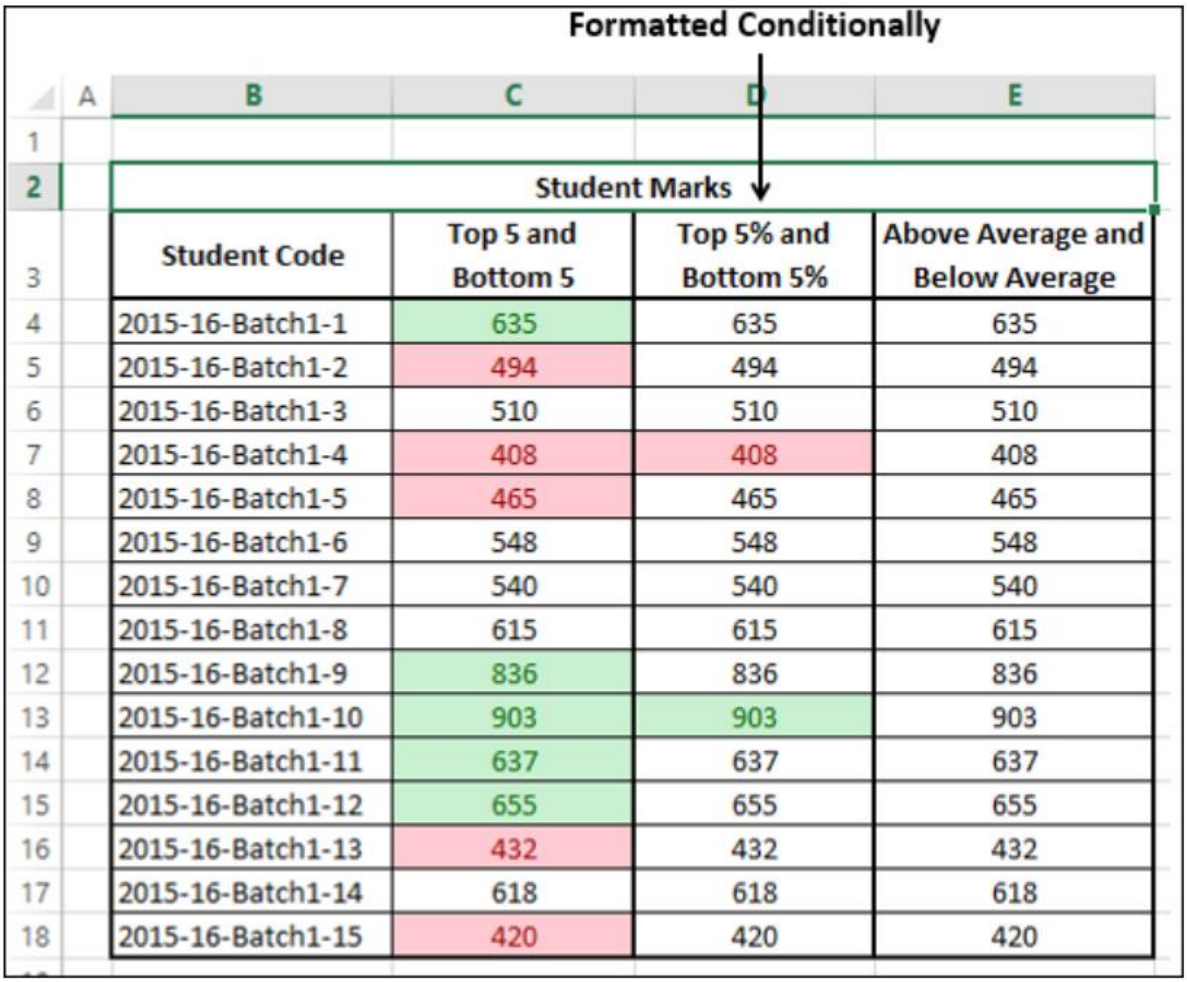


- Repeat the first three steps given above.
- Click **Top Ten%** and specify 5. Choose green color.
- Click **Bottom Ten%** and specify 5. Choose red color.



	Student Marks			
	Student Code	Top 5 and Bottom 5	Top 5% and Bottom 5%	Above Average and Below Average
4	2015-16-Batch1-1	635	635	635
5	2015-16-Batch1-2	494	494	494
6	2015-16-Batch1-3	510	510	510
7	2015-16-Batch1-4	408	408	408
8	2015-16-Batch1-5	465	465	465
9	2015-16-Batch1-6	548	548	548
10	2015-16-Batch1-7	540	540	540
11	2015-16-Batch1-8	615	615	615
12	2015-16-Batch1-9	836	836	836
13	2015-16-Batch1-10	903	903	903
14	2015-16-Batch1-11	637	637	637
15	2015-16-Batch1-12	655	655	655
16	2015-16-Batch1-13	432	432	432
17	2015-16-Batch1-14	618	618	618
18	2015-16-Batch1-15	420	420	420

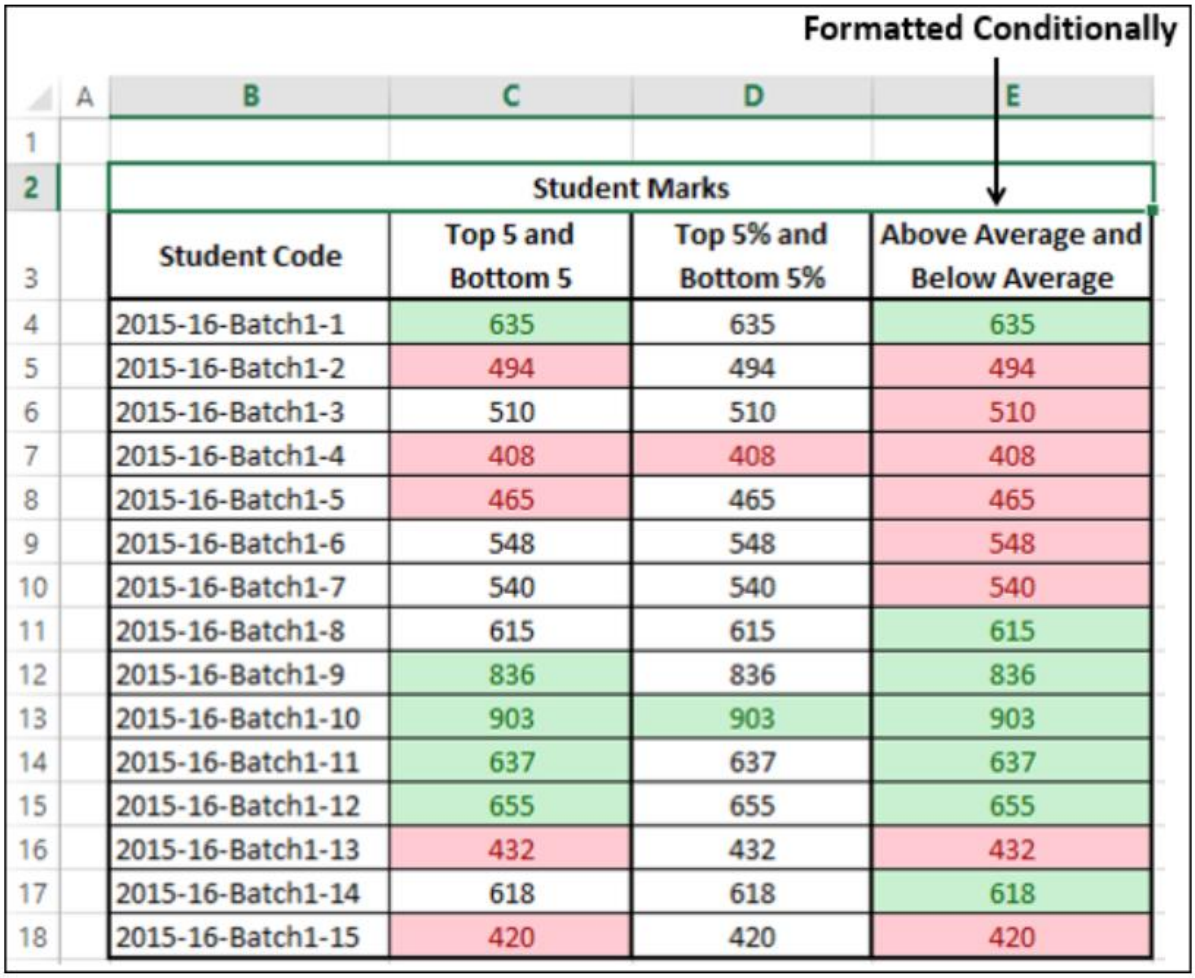
The data will be highlighted based on the given conditions and the corresponding formatting.



- Repeat the first three steps given above.
- Click **Above Average**. Choose green color.

- Click **Below Average**. Choose red color.

The data will be highlighted based on the given conditions and the corresponding formatting.

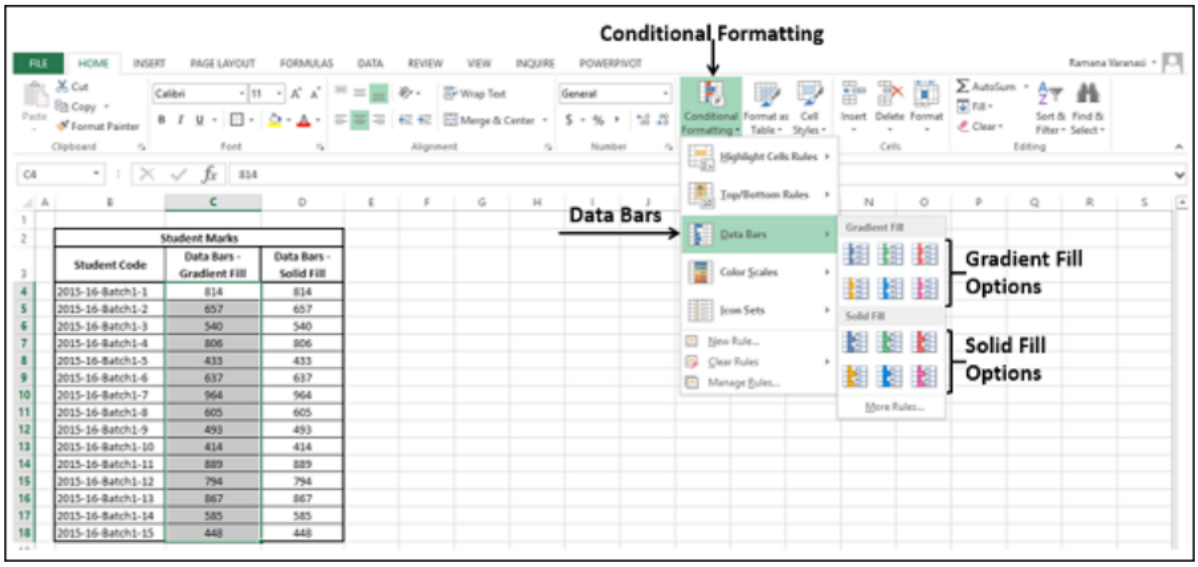


Data Bars

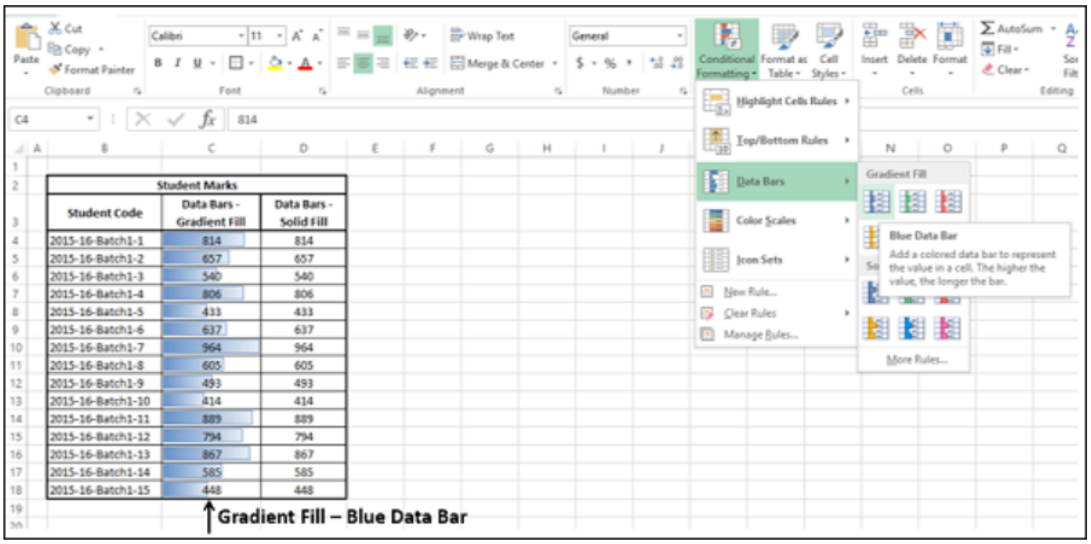
You can use colored **Data Bars** to see the value in a cell relative to the values in the other cells. The length of the data bar represents the value in the cell. A longer bar represents a higher value, and a shorter bar represents a lower value. You have six solid colors to choose from for the data bars – blue, green, red, yellow, light blue and purple.

Data bars are helpful in visualizing the higher, lower and intermediate values when you have large amounts of data. Example - Day temperatures across regions in a particular month. You can use gradient fill color bars to visualize the value in a cell relative to the values in other cells. You have six **Gradient Colors** to choose from for the Data Bars – Blue, Green, Red, Yellow, Light Blue and Purple.

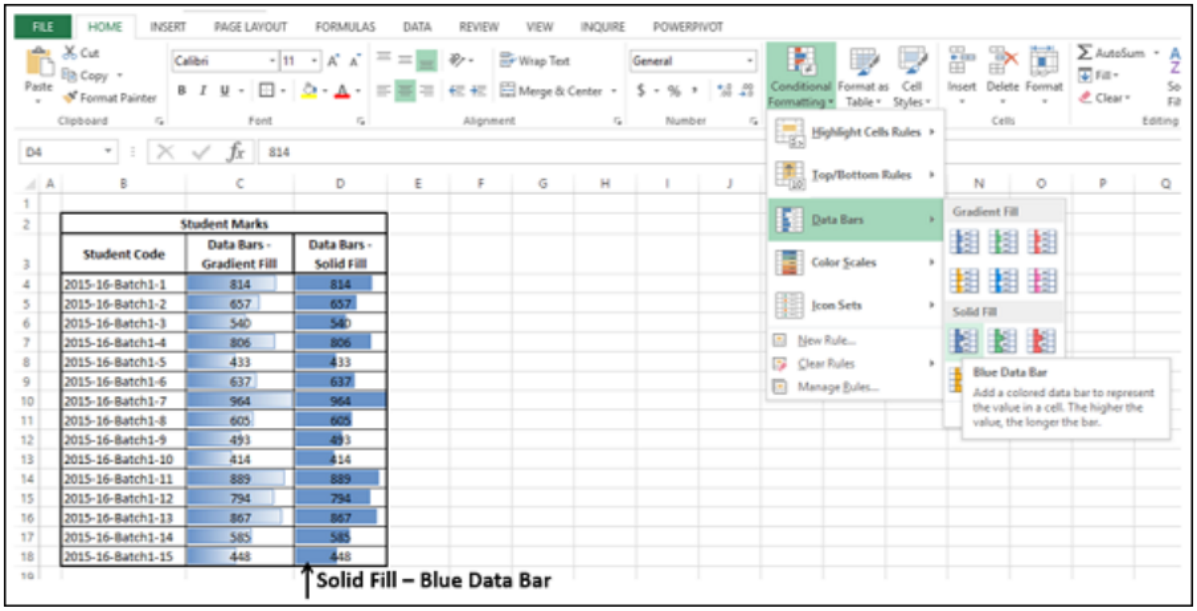
- Select the range to be formatted conditionally.
- Click **Conditional Formatting** in the **Styles** group under **Home** tab.
- Click **Data Bars** from the drop-down menu. The **Gradient Fill** options and **Fill** options appear.



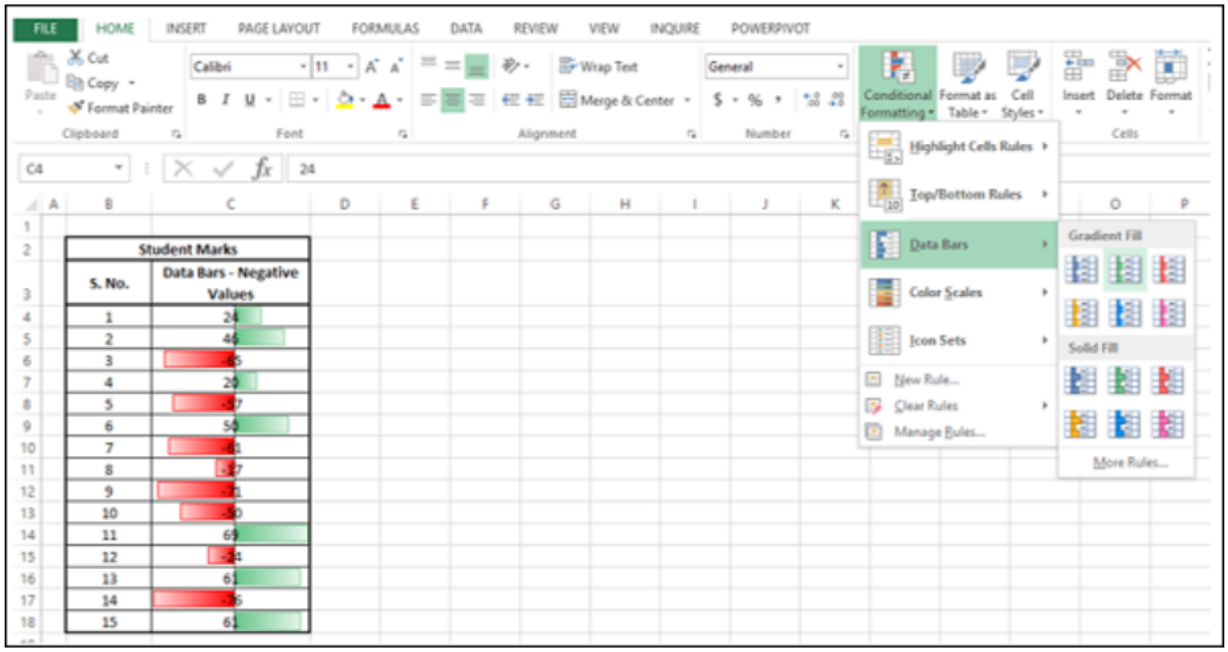
Click the blue data bar in the **Gradient Fill** options.



- Repeat the first three steps.
- Click the blue data bar in the **Solid Fill** options.



You can also format data bars such that the data bar starts in the middle of the cell, and stretches to the left for negative values and stretches to the right for positive values.



3. Visualizing data with charts in Excel

i. Sparklines

ii. Waterfall chart

iii. Pivot Chart

iv. Gantt Chart

v. Thermometer chart

vi. Band Chart

Waterfall Chart

Waterfall Chart is one of the most popular visualization tools used in small and large businesses. Waterfall charts are ideal for showing how you have arrived at a net value such as net income, by breaking down the cumulative effect of positive and negative contributions.

Excel 2016 provides Waterfall Chart type. If you are using earlier versions of Excel, you can still create a Waterfall Chart using Stacked Column Chart.

The columns are color coded so that you can quickly tell positive from negative numbers. The initial and the final value columns start on the horizontal axis, while the intermediate values are floating columns. Because of this look, Waterfall Charts are also called Bridge Charts.

Consider the following data.

	A	B	C
1			
2		Net Cash Flow	
3		80000	Start
4		-5003	Apr
5		-16700	May
6		48802	Jun
7		-11198	Jul
8		-35260	Aug
9		18220	Sep
10		-23840	Oct
11		43250	Nov
12		-18280	Dec
13		26670	Jan
14		15000	Feb
15		24750	Mar

- Prepare the data for Waterfall Chart
- Ensure the column Net Cash Flow is to the left of the Months Column (This is because you will not include this column while creating the chart)
- Add 2 columns – Increase and Decrease for positive and negative cash flows respectively
- Add a column Start - the first column in the chart with the start value in the Net Cash Flow
- Add a column End - the last column in the chart with the end value in the Net Cash Flow
- Add a column Float – that supports the intermediate columns
- Compute the values for these columns as follows

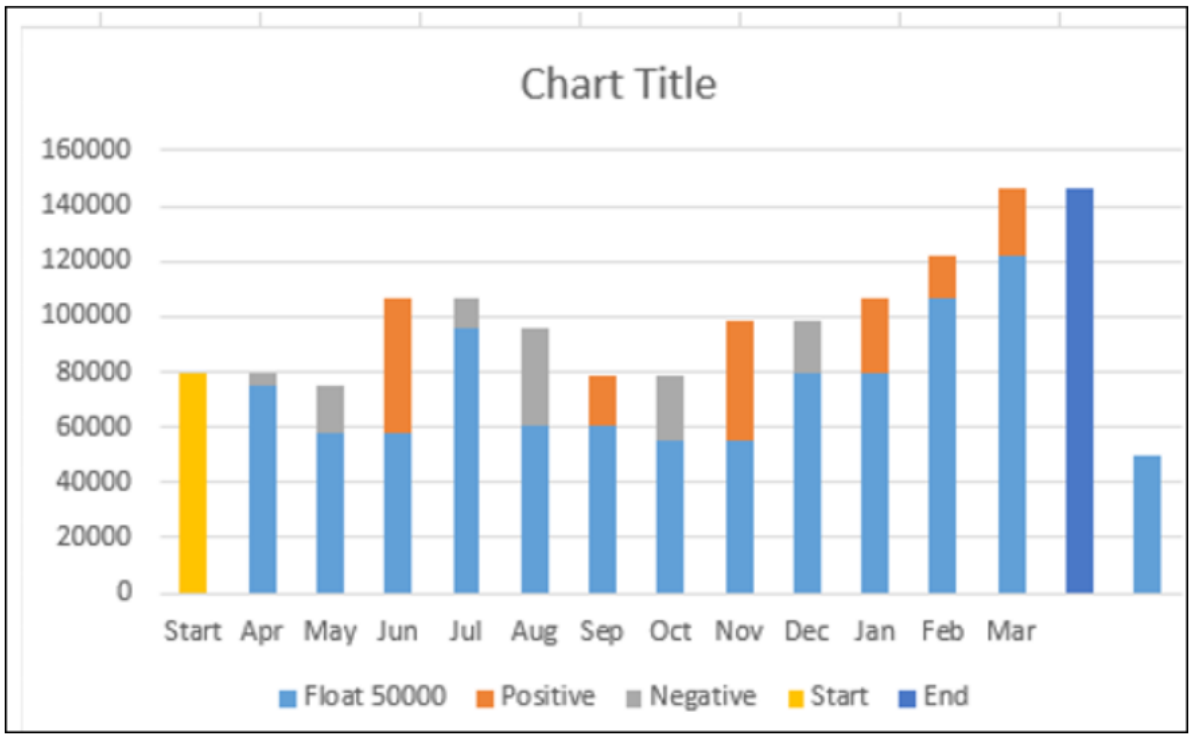
	B	C	D	E	F	G	H
1							
2	Net Cash Flow		Float	Positive	Negative	Start	End
3			50000				
4	80000	Start				=B4	
5	-5003	Apr	=SUM(G4,E5)-F5	=MAX(B5,0)	=-MIN(B5,0)		
6	-16700	May	=SUM(D5,E5)-F6	=MAX(B6,0)	=-MIN(B6,0)		
7	48802	Jun	=SUM(D6,E6)-F7	=MAX(B7,0)	=-MIN(B7,0)		
8	-11198	Jul	=SUM(D7,E7)-F8	=MAX(B8,0)	=-MIN(B8,0)		
9	-35260	Aug	=SUM(D8,E8)-F9	=MAX(B9,0)	=-MIN(B9,0)		
10	18220	Sep	=SUM(D9,E9)-F10	=MAX(B10,0)	=-MIN(B10,0)		
11	-23840	Oct	=SUM(D10,E10)-F11	=MAX(B11,0)	=-MIN(B11,0)		
12	43250	Nov	=SUM(D11,E11)-F12	=MAX(B12,0)	=-MIN(B12,0)		
13	-18280	Dec	=SUM(D12,E12)-F13	=MAX(B13,0)	=-MIN(B13,0)		
14	26670	Jan	=SUM(D13,E13)-F14	=MAX(B14,0)	=-MIN(B14,0)		
15	15000	Feb	=SUM(D14,E14)-F15	=MAX(B15,0)	=-MIN(B15,0)		
16	24750	Mar	=SUM(D15,E15)-F16	=MAX(B16,0)	=-MIN(B16,0)		
17							=SUM(D16,E16)-F17
18			50000				

- In the Float column, insert a row in the beginning and at the end. Place n arbitrary value 50000. This just to have some space to the left and right of the chart

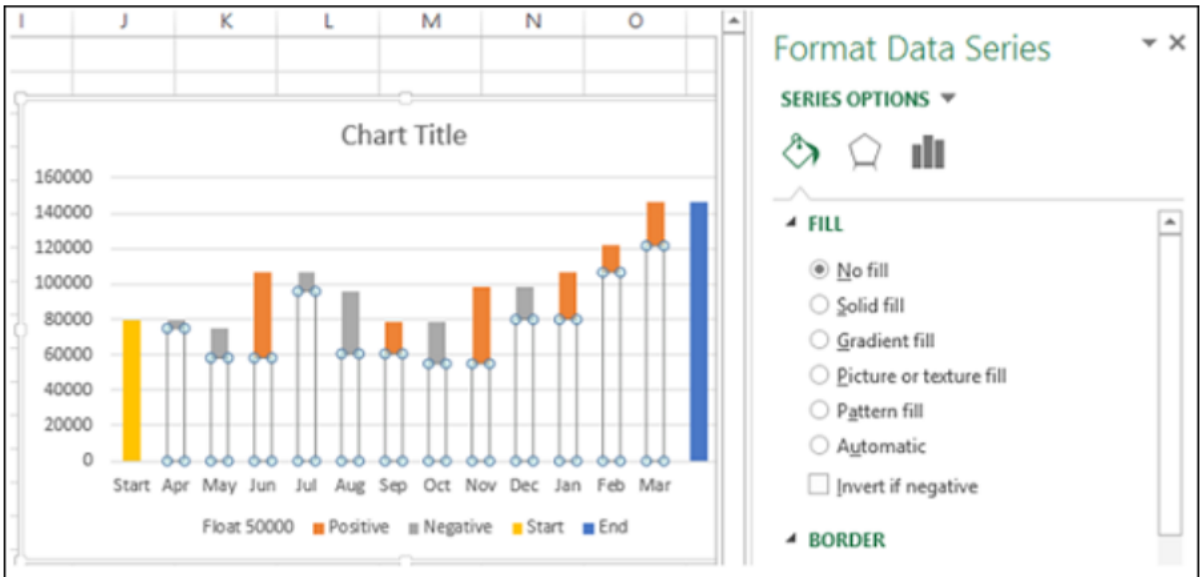
The data will be as follows.

	B	C	D	E	F	G	H
1							
2	Net Cash Flow		Float	Positive	Negative	Start	End
3			50000				
4	80000	Start				80000	
5	-5003	Apr	74997	0	5003		
6	-16700	May	58297	0	16700		
7	48802	Jun	58297	48802	0		
8	-11198	Jul	95901	0	11198		
9	-35260	Aug	60641	0	35260		
10	18220	Sep	60641	18220	0		
11	-23840	Oct	55021	0	23840		
12	43250	Nov	55021	43250	0		
13	-18280	Dec	79991	0	18280		
14	26670	Jan	79991	26670	0		
15	15000	Feb	106661	15000	0		
16	24750	Mar	121661	24750	0		
17							146411
18			50000				

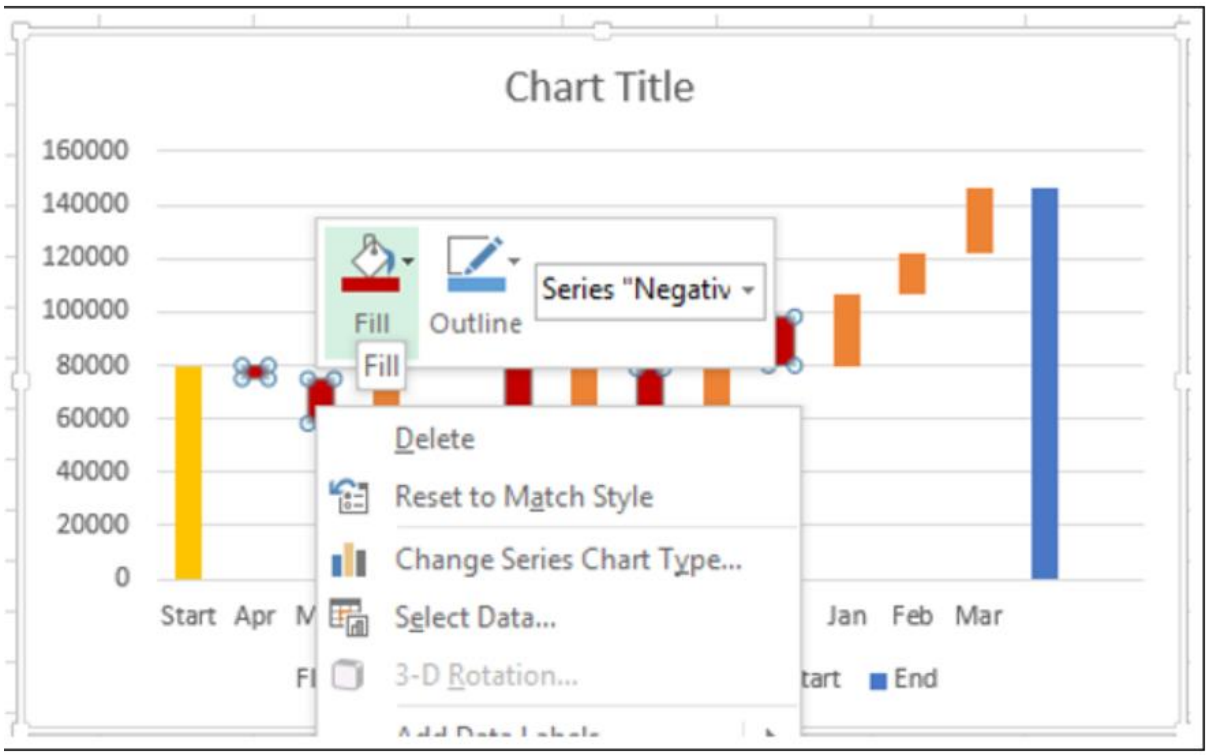
- Select the cells C2:H18 (Exclude Net Cash Flow column)
- Create Stacked Column Chart



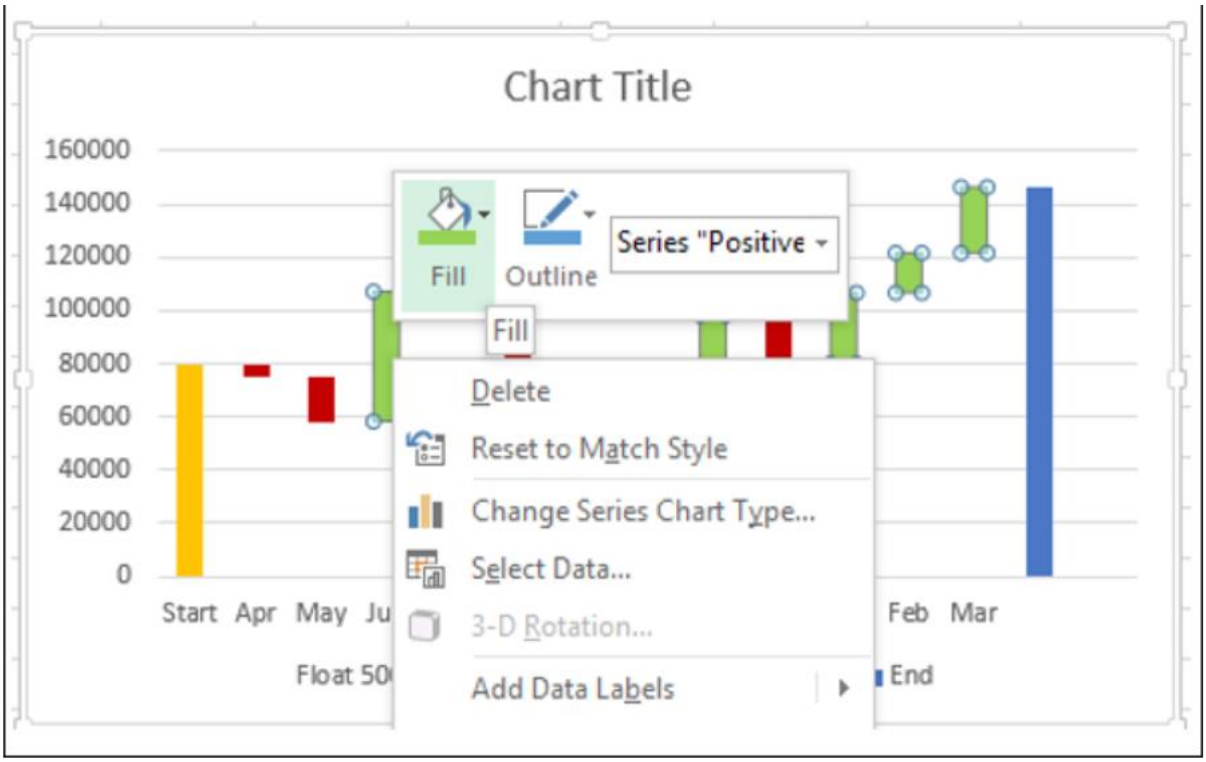
- Right click on the Float Series.
- Click Format Data Series.
- In Format Data Series options, select No fill.



- Right click on Negative Series.
- Select Fill Color as Red.



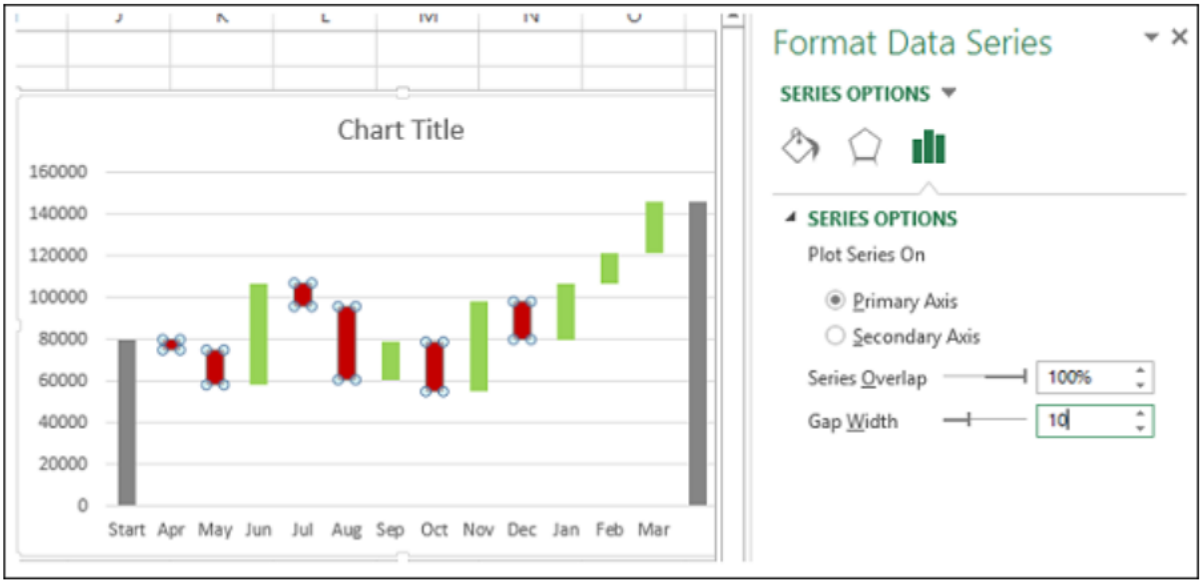
- Right click on Positive Series.
- Select Fill Color as Green.



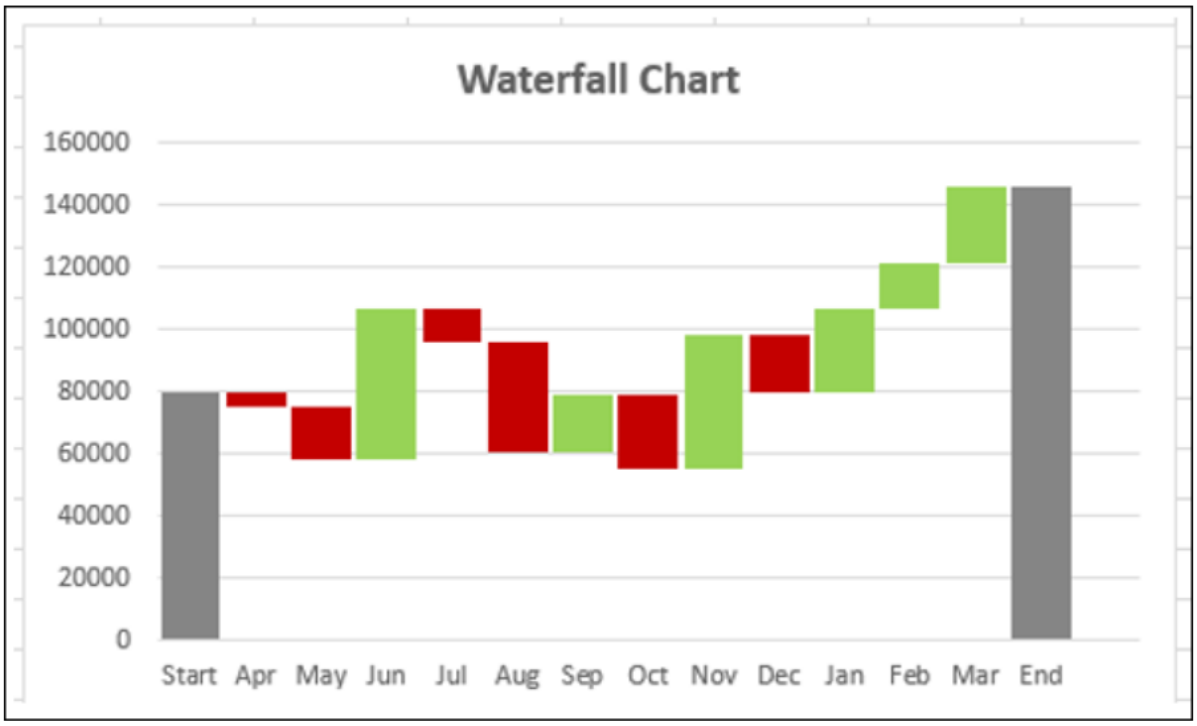
- Right click on Start Series.
- Select Fill Color as Grey.
- Right click on End Series.
- Select Fill Color as Grey.

- Delete the Legend.

- Right click on any Series
- In Format Data Series options, select Gap Width as 10% under Series Options



Give the Chart Title. The Waterfall Chart will be displayed.

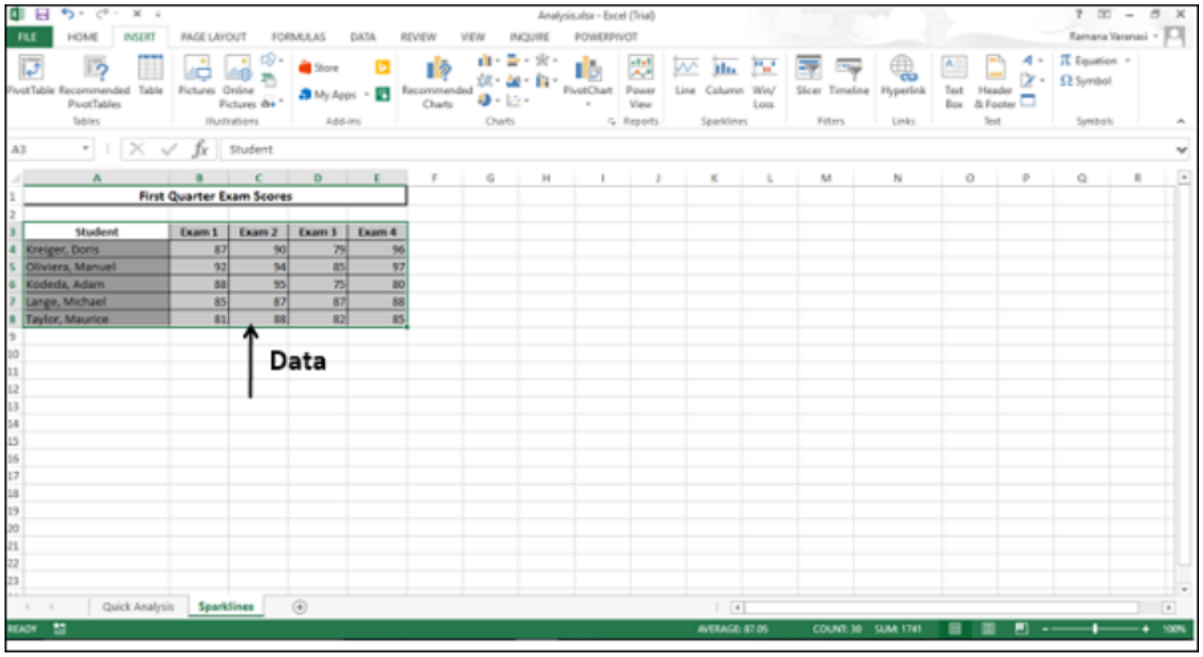


Sparklines

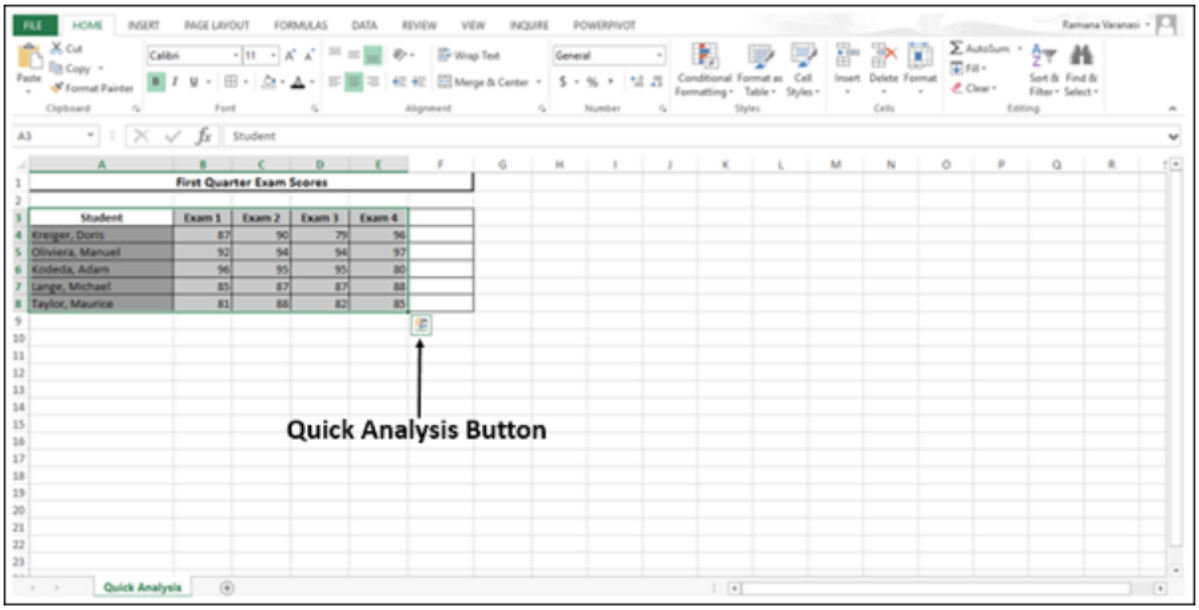
Sparklines are tiny charts placed in single cells, each representing a row of data in your selection. They provide a quick way to see trends.

You can add Sparklines with Quick Analysis tool.

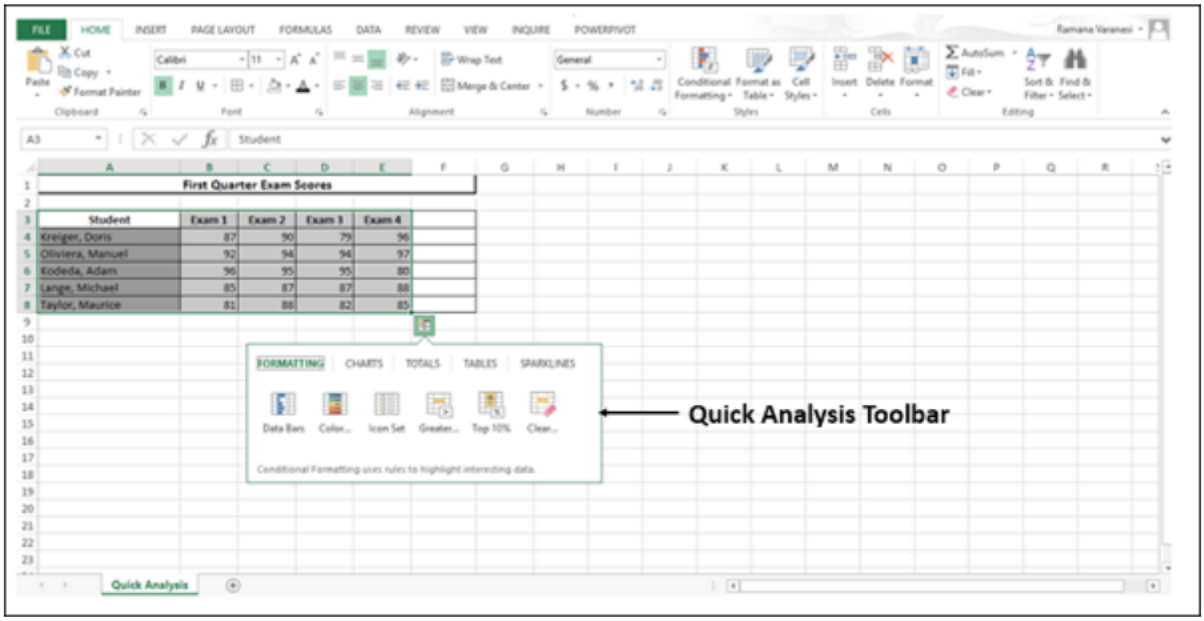
- Select the data for which you want to add Sparklines.
- Keep an empty column to the right side of the data for the Sparklines.



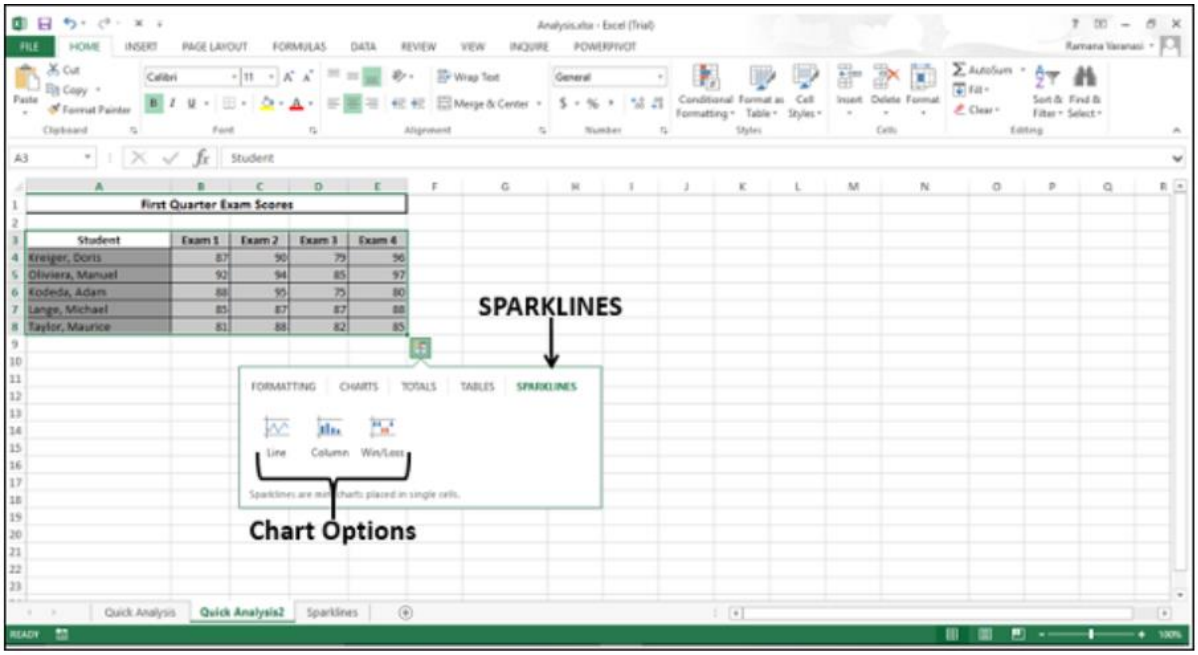
Quick Analysis button appears at the bottom right of your selected data.



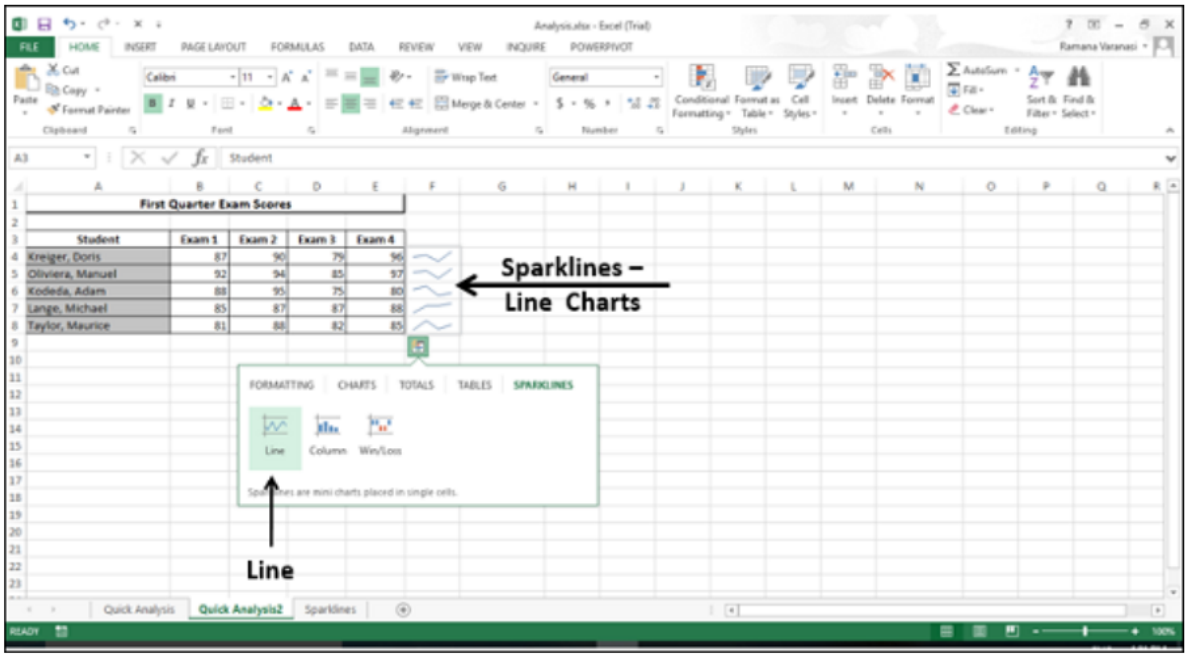
- Click on the Quick Analysis button. The Quick Analysis Toolbar appears with various options.



Click **SPARKLINES**. The chart options displayed are based on the data and may vary.



Click **Line**. A Line Chart for each row is displayed in the column to the right of the data.



ivotCharts

Pivot Charts are used to graphically summarize data and explore complicated data.

A PivotChart shows Data Series, Categories, and Chart Axes the same way a standard chart does. Additionally, it also gives you interactive filtering controls right on the chart so that you can quickly analyze a subset of your data.

PivotCharts are useful when you have data in a huge PivotTable, or many complex worksheet data that includes text and numbers. A PivotChart can help you make sense of this data.

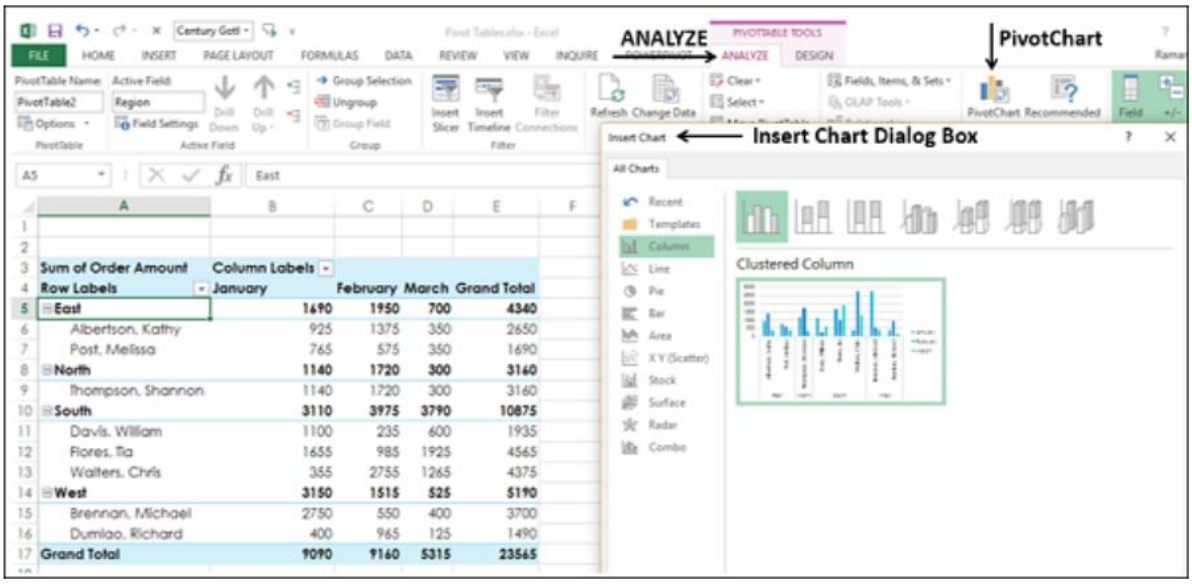
You can create a PivotChart from

- A PivotTable.
- A Data Table as a standalone without PivotTable.

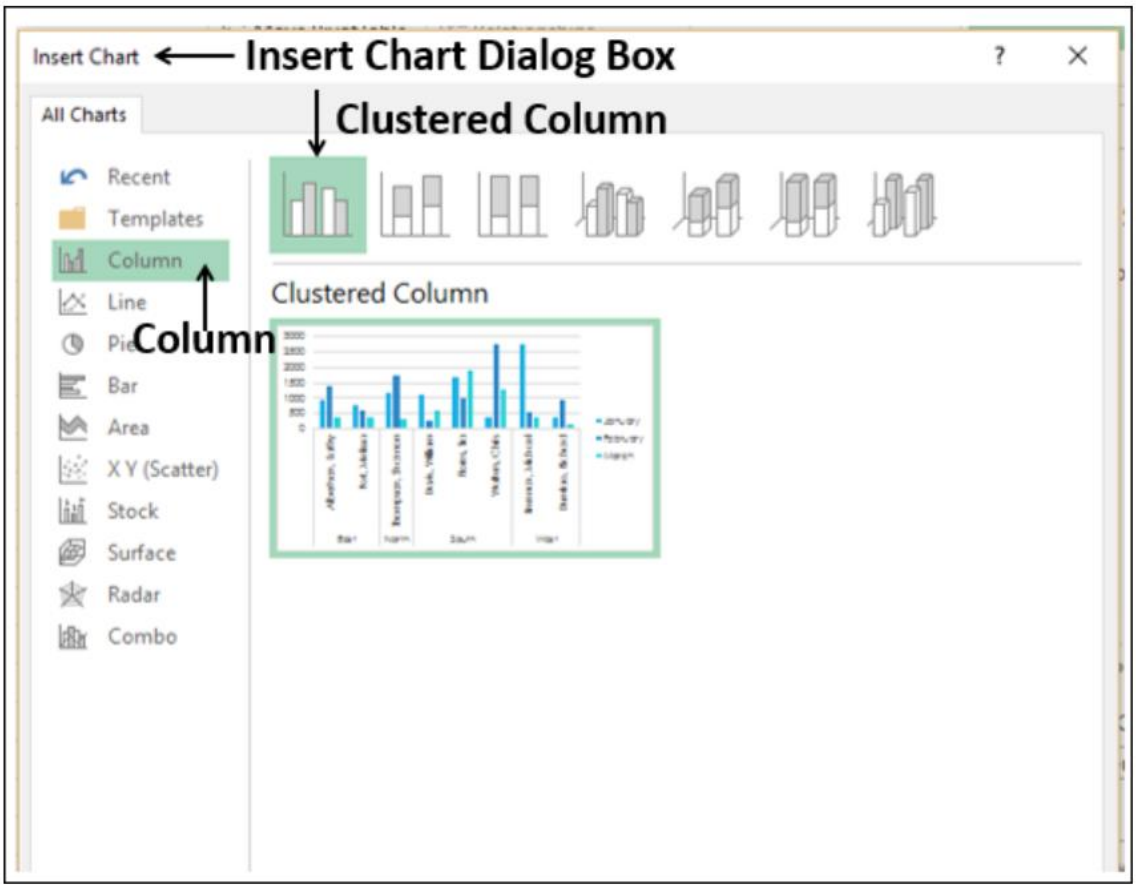
PivotChart from PivotTable

To create a PivotChart follow the steps given below −

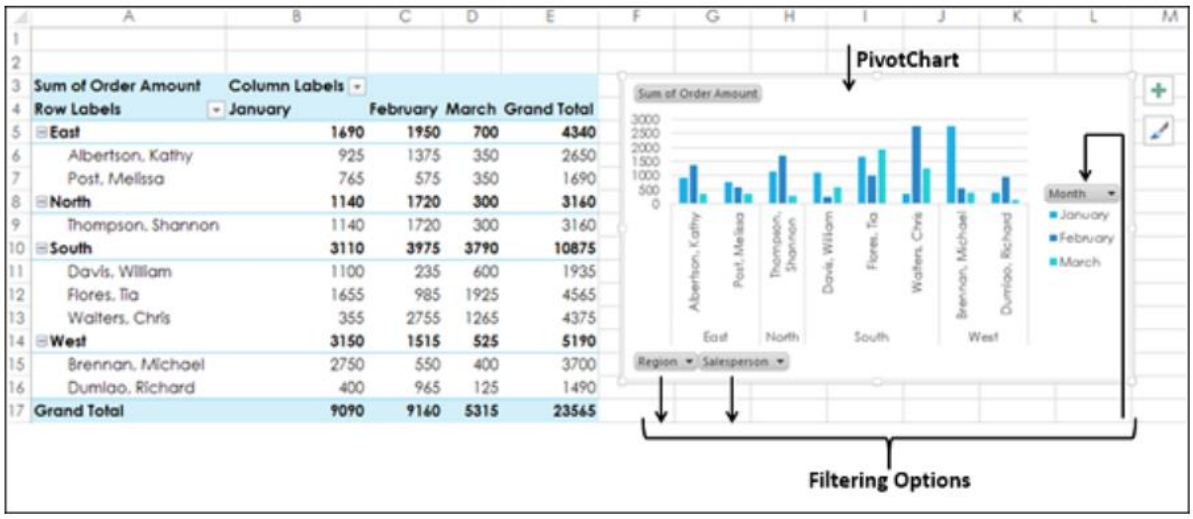
- Click the PivotTable.
- Click ANALYZE under PIVOTTABLE TOOLS on the Ribbon.
- Click on PivotChart. The Insert Chart dialog box appears.



Select Clustered Column from the option Column.



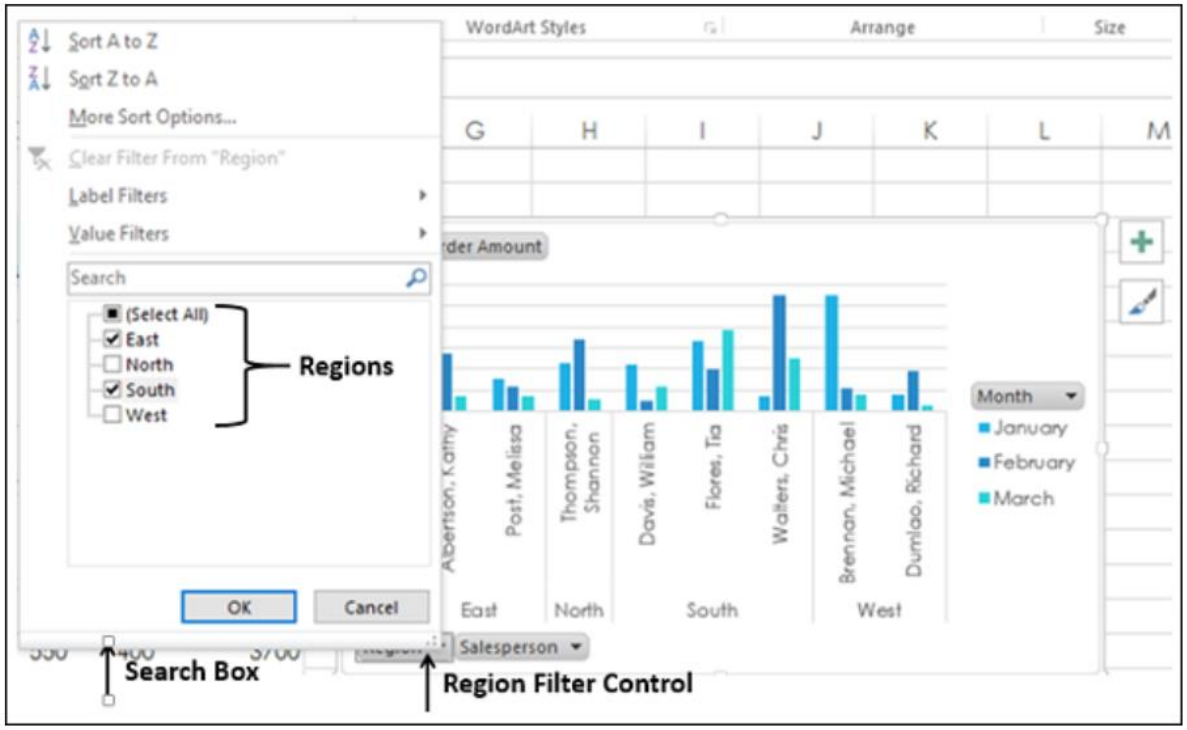
Click OK. The PivotChart is displayed.



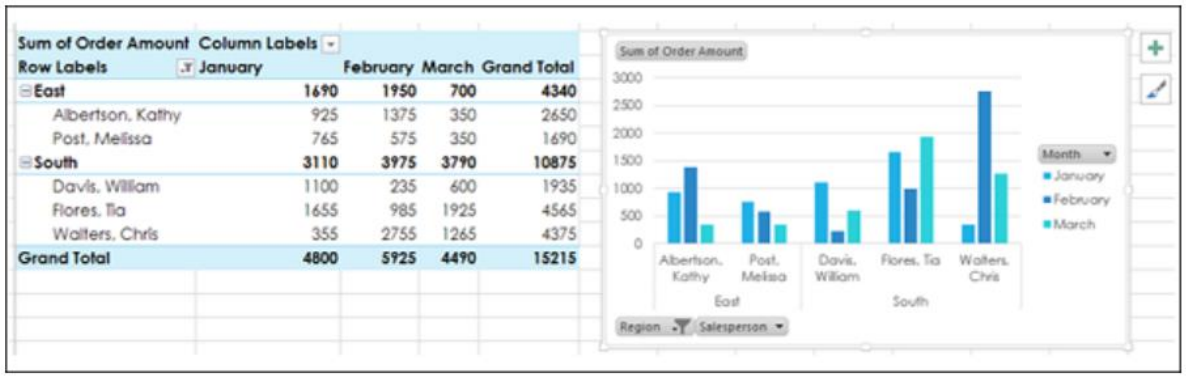
Sum of Order Amount	Column Labels			
Row Labels	January	February	March	Grand Total
East	1690	1950	700	4340
Albertson, Kathy	925	1375	350	2650
Post, Melissa	765	575	350	1690
North	1140	1720	300	3160
Thompson, Shannon	1140	1720	300	3160
South	3110	3975	3790	10875
Davis, William	1100	235	600	1935
Flores, Tia	1655	985	1925	4565
Walters, Chris	355	2755	1265	4375
West	3150	1515	525	5190
Brennan, Michael	2750	550	400	3700
Dumlao, Richard	400	965	125	1490
Grand Total	9090	9160	5315	23565

The PivotChart has three filters – Region, Salesperson and Month.

- Click the Region Filter Control option. The Search Box appears with the list of all Regions. Check boxes appear next to Regions.
- Select East and South options.



The filtered data appears on both the PivotChart and the PivotTable.



Sum of Order Amount	Column Labels			
Row Labels	January	February	March	Grand Total
East	1690	1950	700	4340
Albertson, Kathy	925	1375	350	2650
Post, Melissa	765	575	350	1690
South	3110	3975	3790	10875
Davis, William	1100	235	600	1935
Flores, Tia	1655	985	1925	4565
Walters, Chris	355	2755	1265	4375
Grand Total	4800	5925	4490	15215

Band Chart

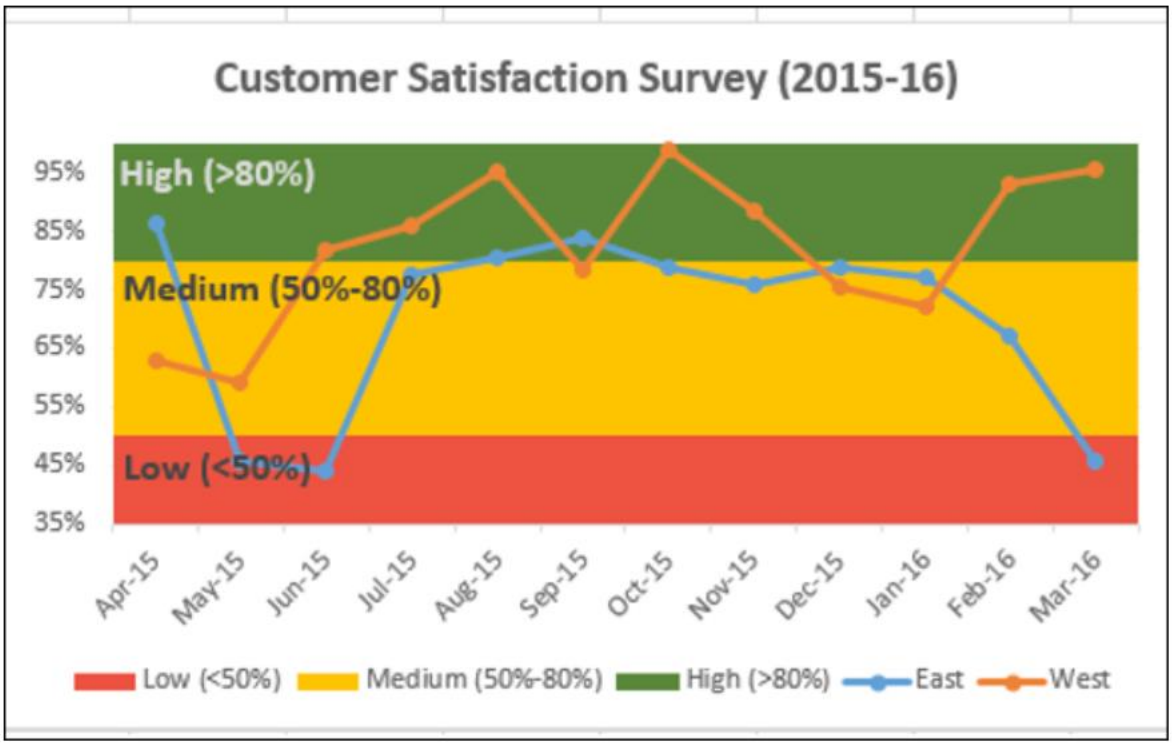
You might have to present customer survey results of a product from different regions. Band Chart is suitable for this purpose. A Band Chart is a Line Chart with an added shaded area to display the upper and lower boundaries of groups of data.

Suppose your customer survey results from the east and west regions, month wise are −

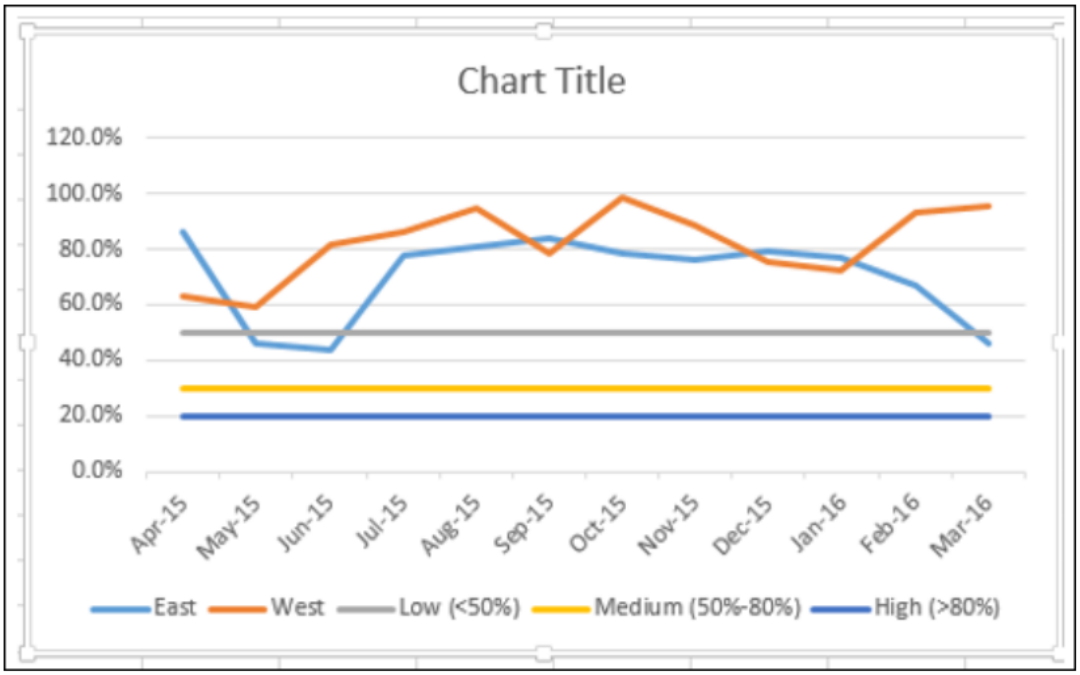
	A	B	C	D	E	F	G
1							
2		Month	East	West	Low (<50%)	Medium (50%-80%)	High (>80%)
3		Apr-15	86.4%	63.0%	50%	30%	20%
4		May-15	45.8%	58.9%	50%	30%	20%
5		Jun-15	44.1%	81.6%	50%	30%	20%
6		Jul-15	77.6%	86.1%	50%	30%	20%
7		Aug-15	80.7%	95.0%	50%	30%	20%
8		Sep-15	83.7%	78.2%	50%	30%	20%
9		Oct-15	78.8%	98.9%	50%	30%	20%
10		Nov-15	76.0%	88.3%	50%	30%	20%
11		Dec-15	79.0%	75.5%	50%	30%	20%
12		Jan-16	77.0%	72.1%	50%	30%	20%
13		Feb-16	67.1%	93.1%	50%	30%	20%
14		Mar-16	45.8%	95.7%	50%	30%	20%

Here, in the data < 50% is Low, 50% - 80% is Medium and > 80% is High.

With Band Chart, you can display your survey results as follows −

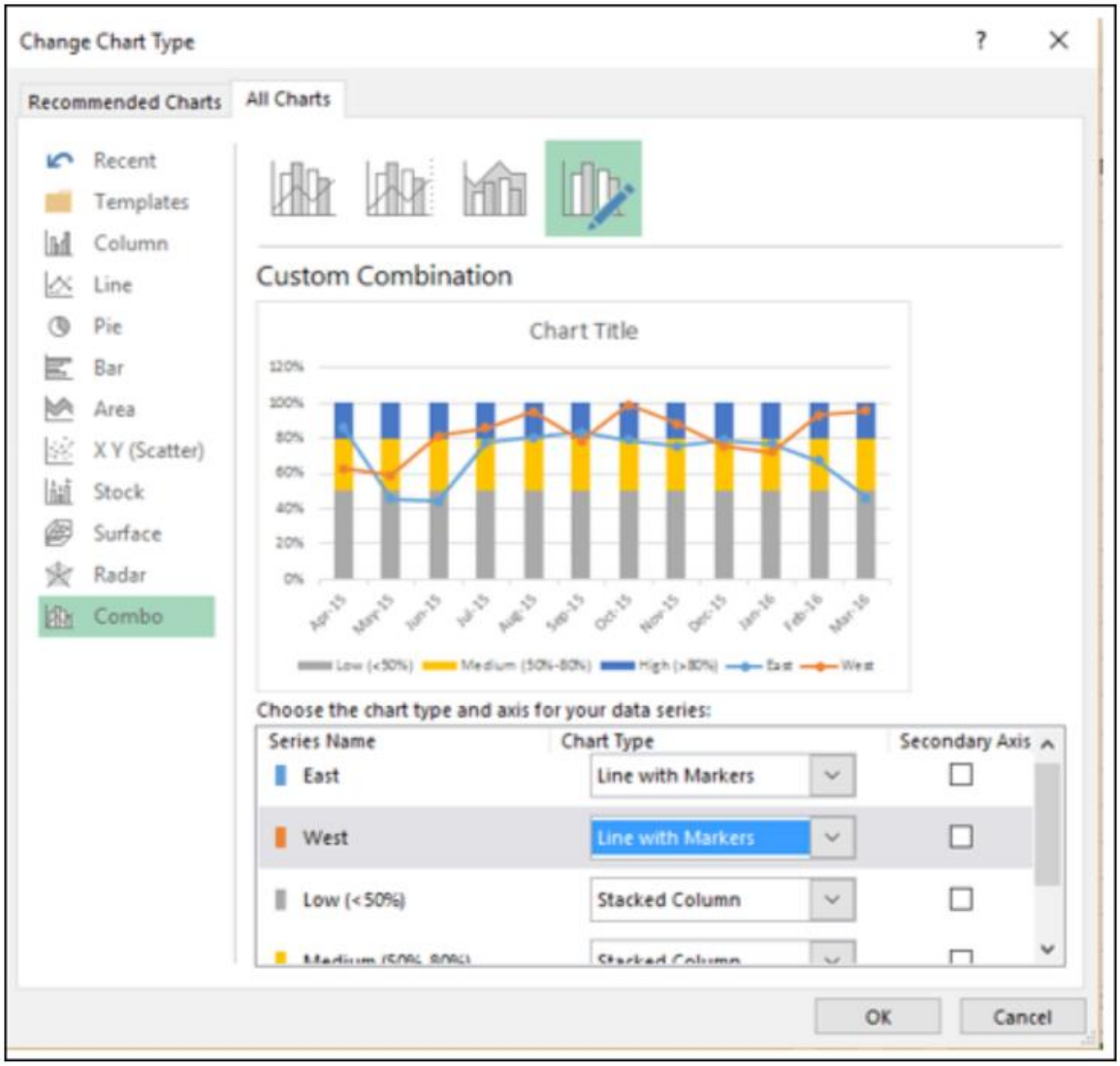


Create a Line Chart from your data.

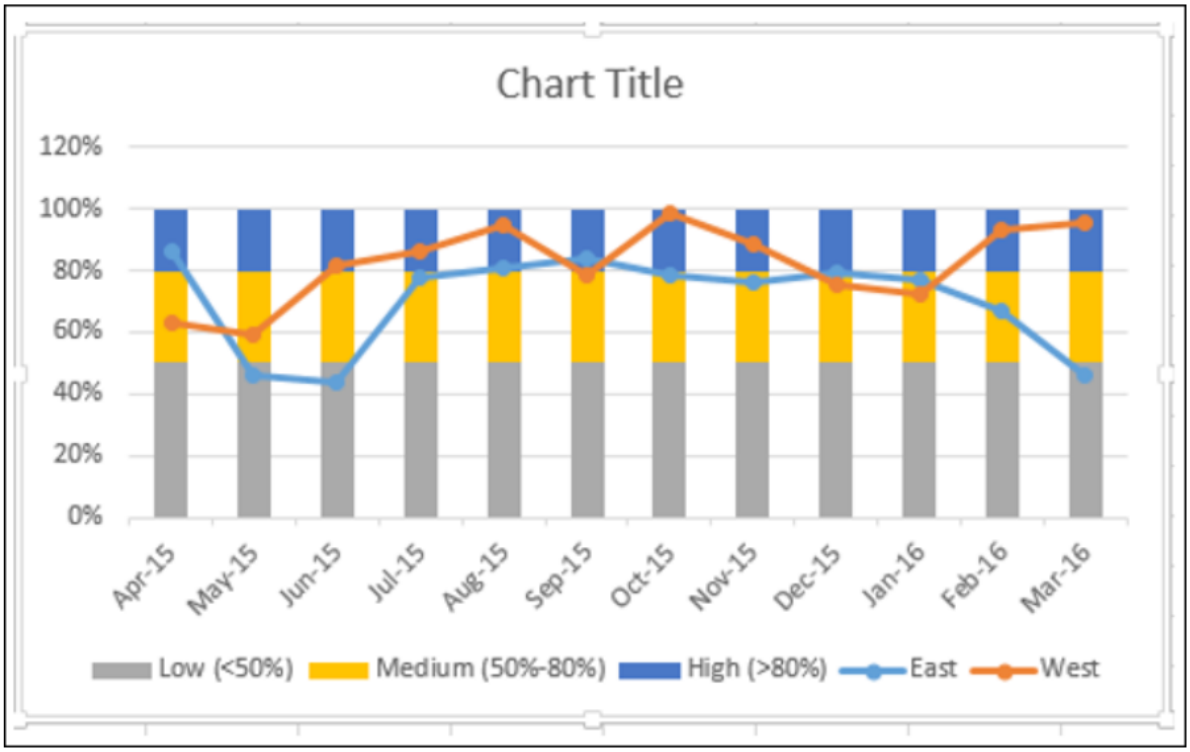


Change the chart type to −

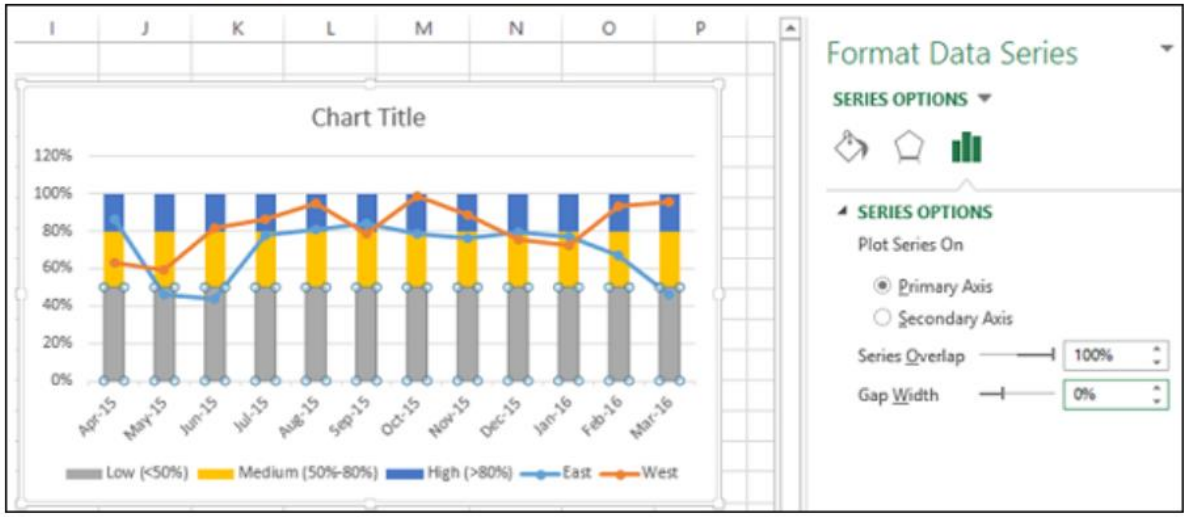
- East and West Series to Line with Markers.
- Low, Medium and High Series to Stacked Column.


Change Chart Type
Recommended Charts
All Charts
Recent
Templates
Column
Line
Pie
Bar
Area
X Y (Scatter)
Stock
Surface
Radar
Combo
Custom Combination
Chart Title
120%
100%
80%
60%
40%
20%
0%
Apr-15
May-15
Jun-15
Jul-15
Aug-15
Sep-15
Oct-15
Nov-15
Dec-15
Jan-16
Feb-16
Mar-16
Low (<50%)
Medium (50%-80%)
High (>80%)
East
West
Choose the chart type and axis for your data series:
Series Name
Chart Type
Secondary Axis
East
Line with Markers
West
Line with Markers
Low (< 50%)
Stacked Column
OK
Cancel

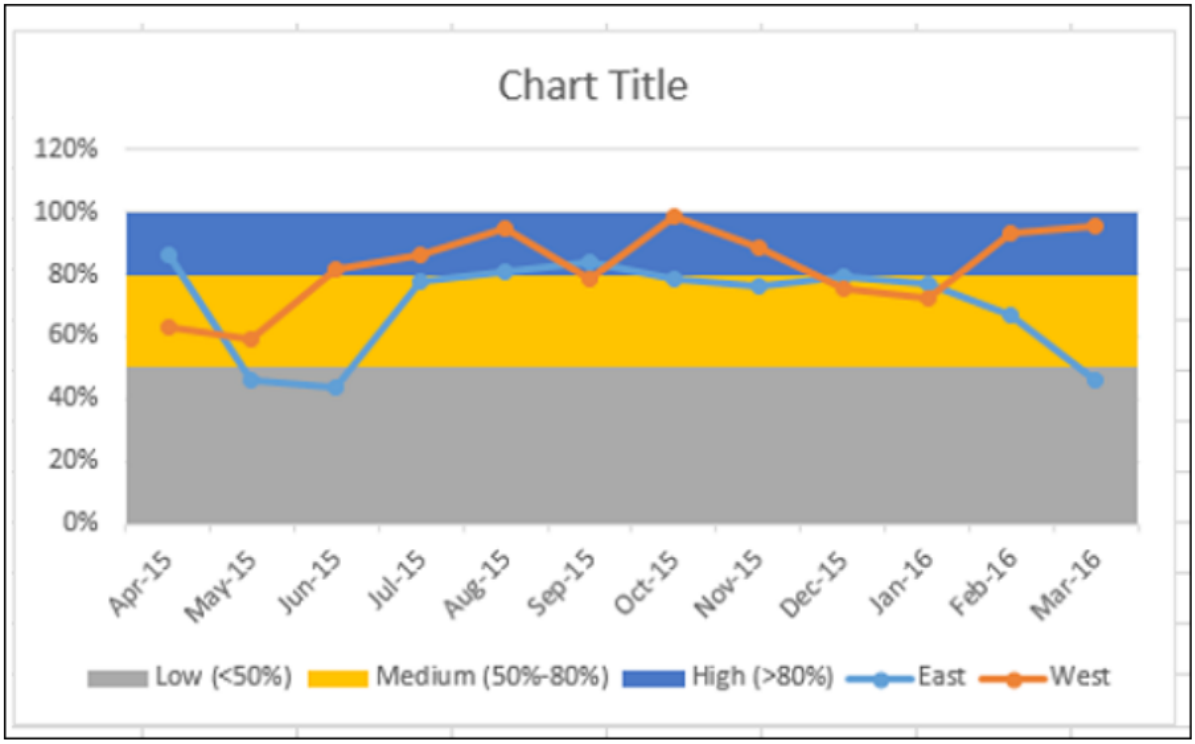
Your chart looks as follows.



- Click on one of the columns.
- Change gap width to 0% in Format Data Series.



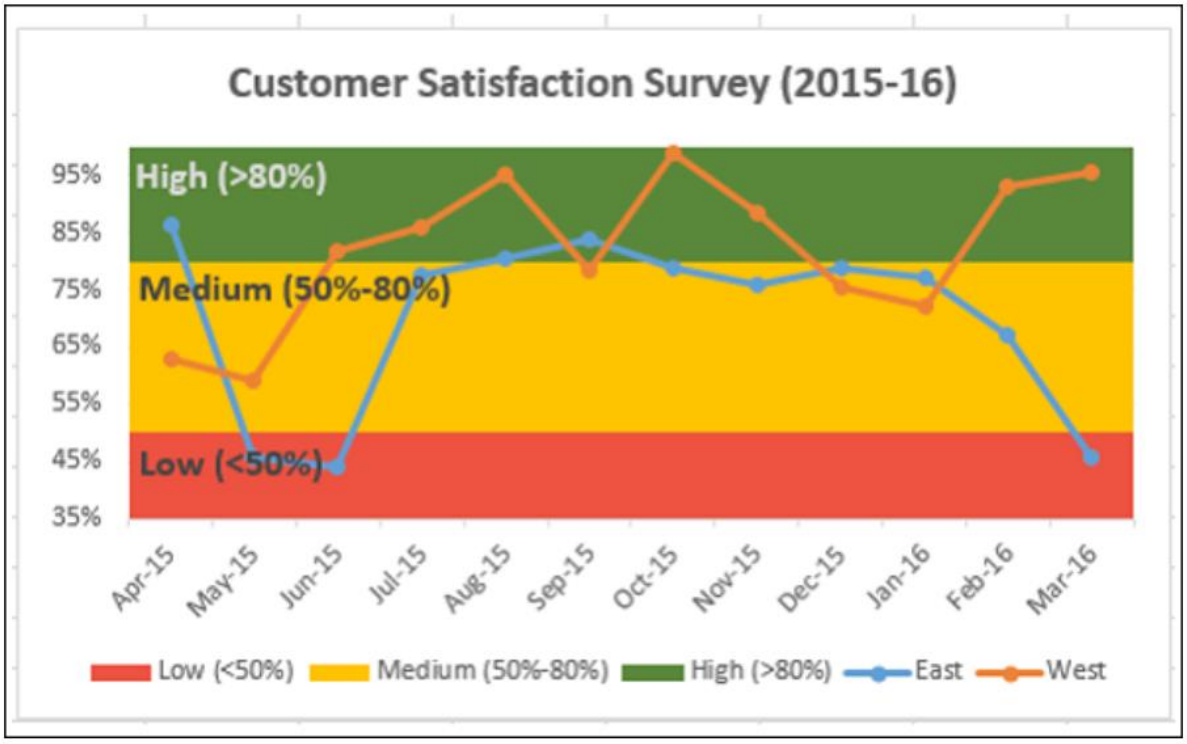
You will get Bands instead of columns.



To make the chart more presentable −

- Add Chart Title.
- Adjust Vertical Axis range.
- Change the colors of the bands to Green-Yellow-Red.
- Add Labels to bands.

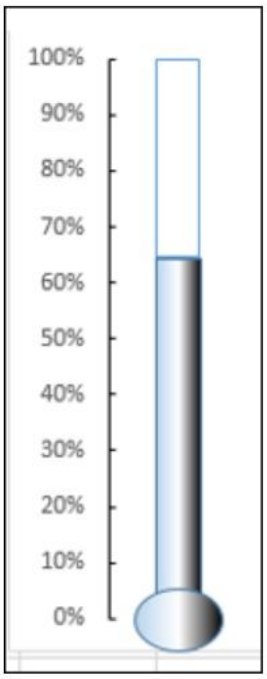
The final result is the Band Chart with the defined boundaries and the survey results represented across the bands. One can quickly and clearly make out from the chart that while the survey results for the region West are satisfactory, those for the region East have a decline in the last quarter and need attention.



Thermometer Chart

When you have to represent a target value and an actual value, you can easily create a Thermometer Chart in Excel that emphatically shows these values.

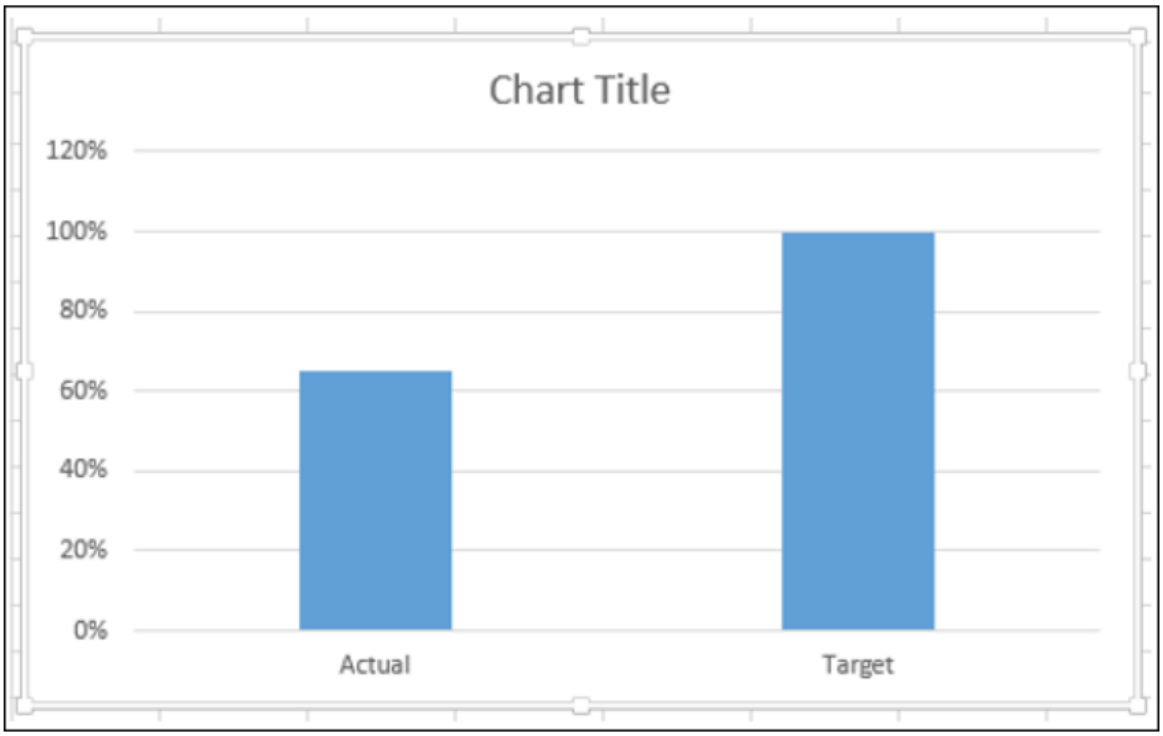
With Thermometer chart, you can display your data as follows −



Arrange your data as shown below −

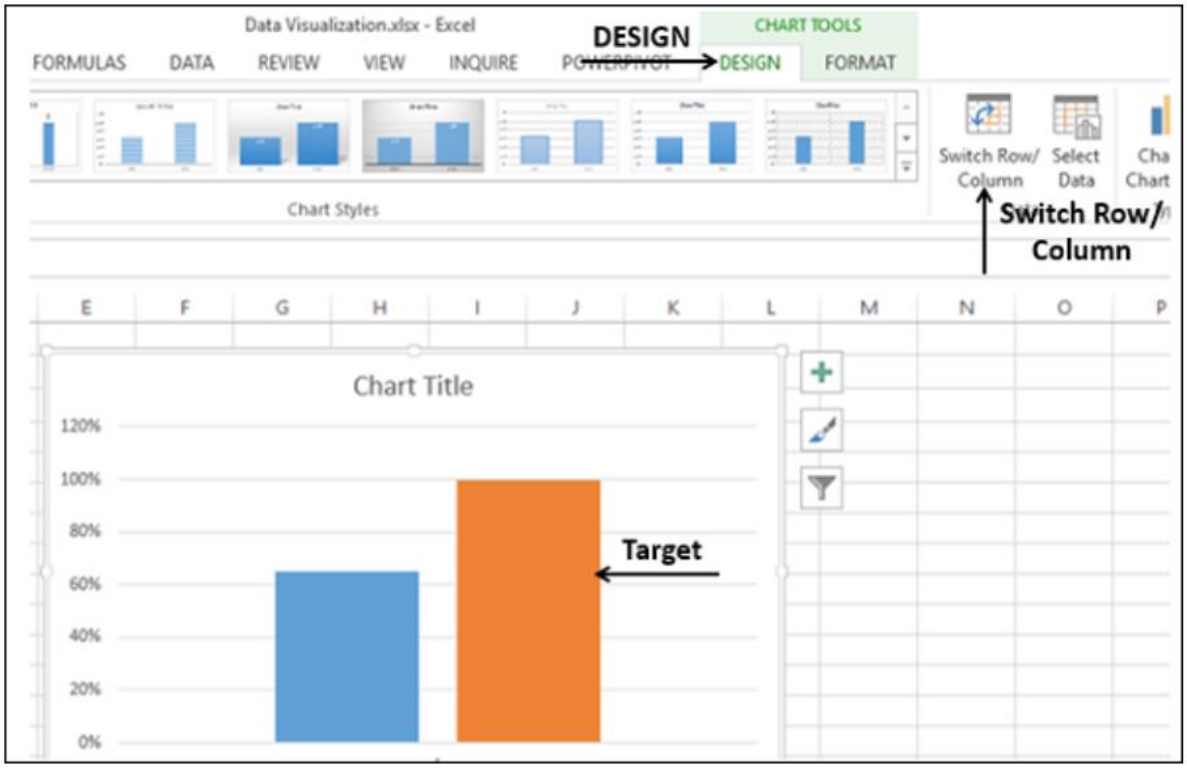
	A	B	C
1			
2		**Actual**	**Target**
3		65%	100%

- Select the data.
- Create a Clustered Column chart.

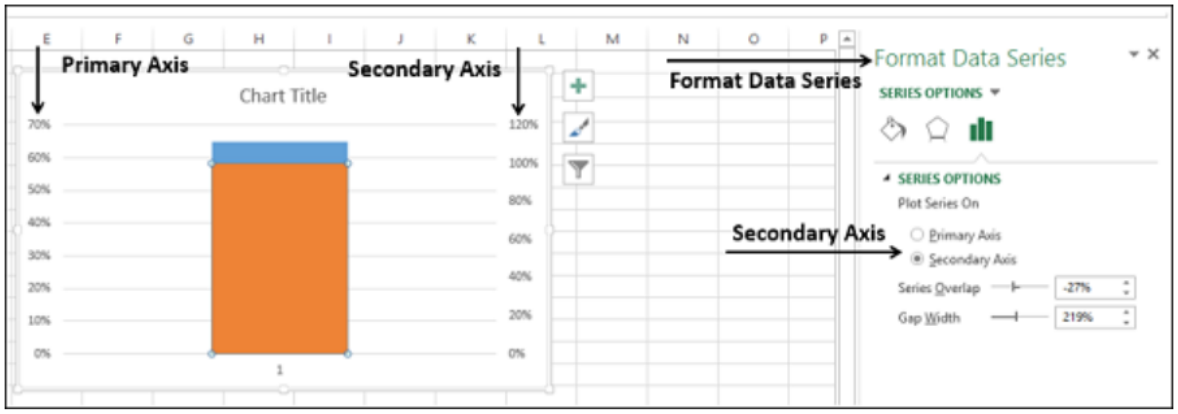


As you observe, the right side Column is Target.

- Click on a Column in the chart.
- Click on Switch Row/Column on the Ribbon.

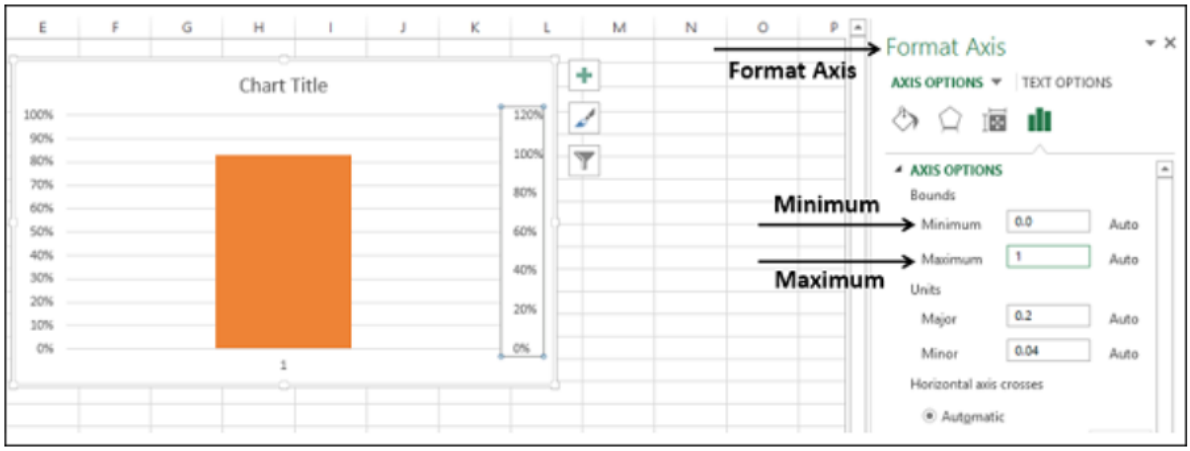


- Right click on the Target Column.
- Click on Format Data Series.
- Click on Secondary Axis.



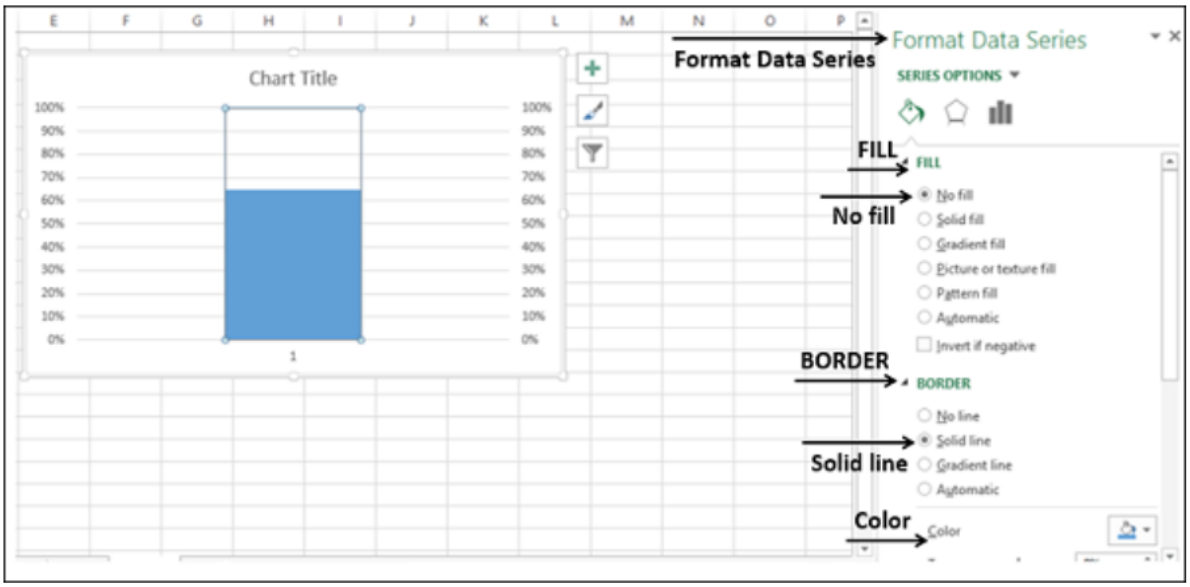
As you observe the Primary Axis and Secondary Axis have different ranges.

- Right click the Primary Axis.
- In the Format Axis options, under Bounds, type 0 for Minimum and 1 for Maximum.
- Repeat the same for Secondary Axis.

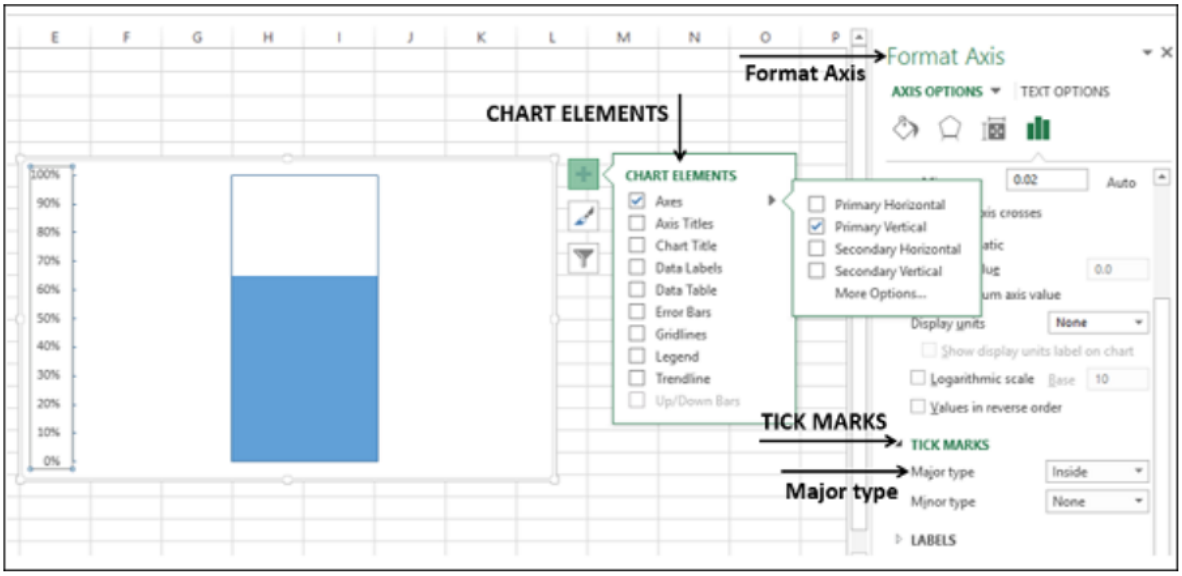


Both Primary Axis and Secondary Axis will be set to 0% - 100%. The Target Column hides the Actual Column.

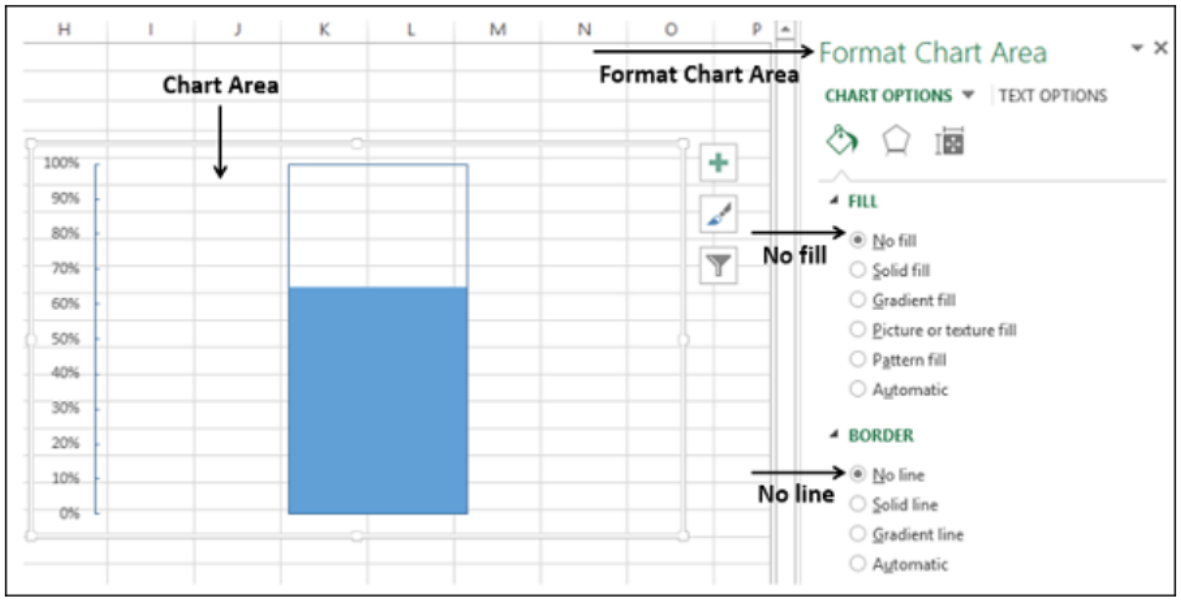
- Right click the visible column (Target)
- In the Format Data Series, select
  - No fill for FILL
  - Solid line for BORDER
  - Blue for Color



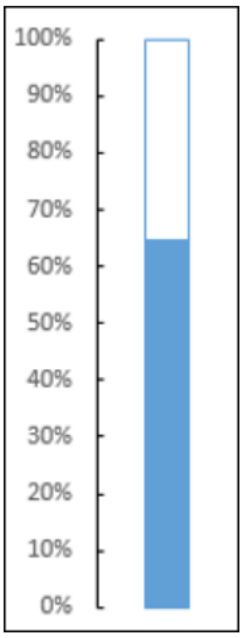
- In Chart Elements, unselect
  - Axis → Primary Horizontal
  - Axis → Secondary Vertical
  - Gridlines
  - Chart Title
- In the chart, right click on Primary Vertical Axis
- In Format Axis options, click on TICK MARKS
- For Major type, select Inside



- Right click on the Chart Area.
- In the Format Chart Area options, select
  - No fill for FILL
  - No line for BORDER

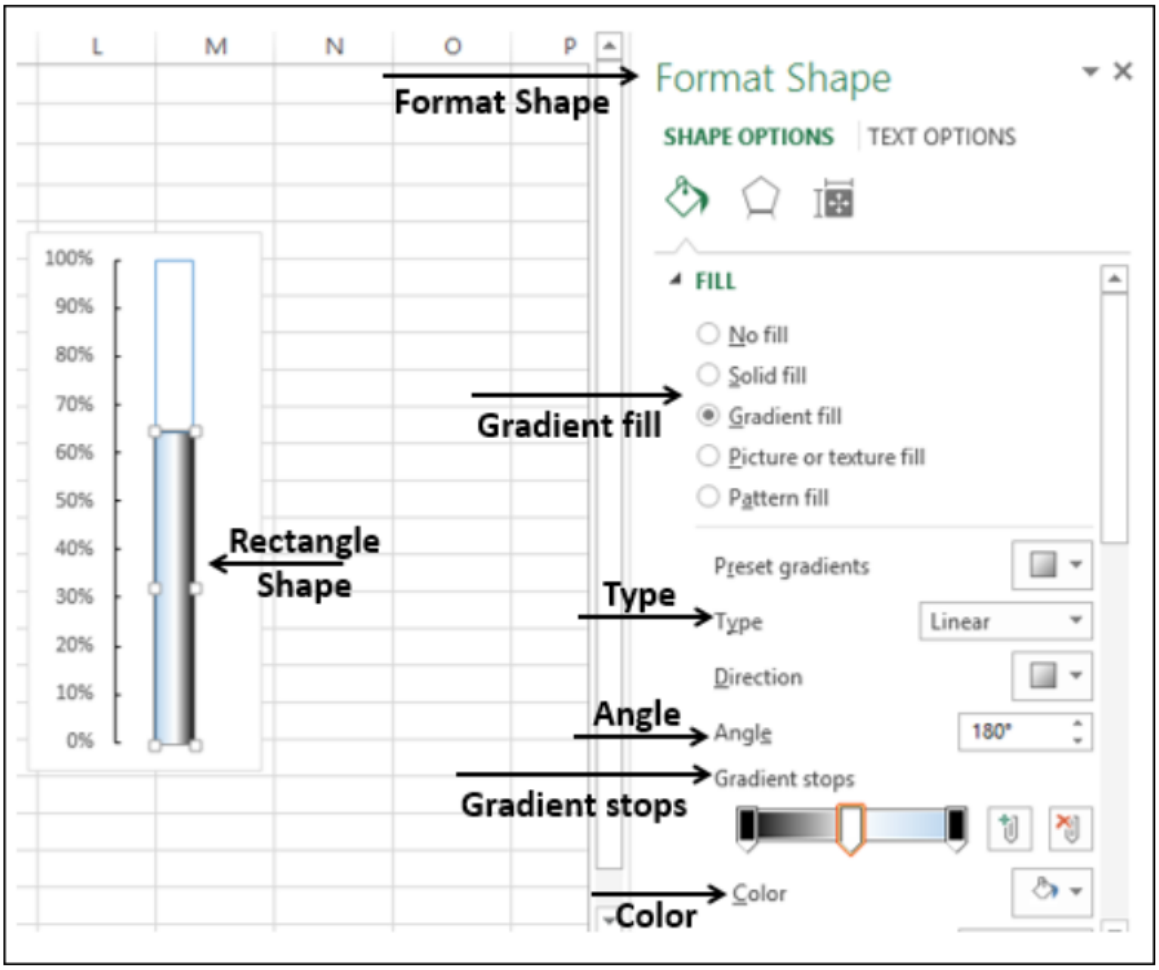


Resize the chart area, to get the shape of a thermometer.



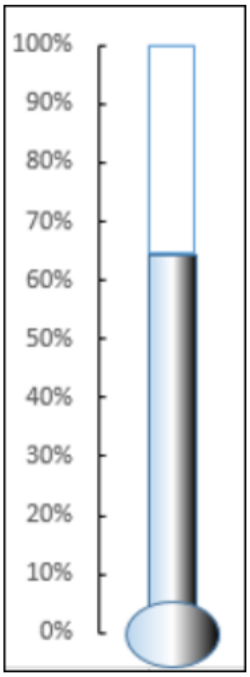
You got your thermometer chart, with the actual value as against target value being shown. You can make this thermometer chart more impressive with some formatting.

- Insert a rectangle shape superimposing the blue rectangular part in the chart.
- In Format Shape options, select −
  - Gradient fill for FILL
  - Linear for Type
  - 180^0 for Angle
- Set the Gradient stops at 0%, 50% and 100%.
- For the Gradient stops at 0% and 100%, choose the color black.
- For the Gradient stop at 50%, choose the color white.


L
M
N
O
P
Format Shape
Format Shape
SHAPE OPTIONS
TEXT OPTIONS
FILL
No fill
Solid fill
Gradient fill
Picture or texture fill
Pattern fill
Gradient fill
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Rectangle Shape
Preset gradients
Type
Type
Linear
Direction
Angle
Angle
180°
Gradient stops
Gradient stops
Color
Color

- Insert an oval shape at the bottom.
- Format shape with same options.

The result is the Thermometer Chart that we started with.



Gantt Chart

A Gantt chart is a chart in which a series of horizontal lines shows the amount of work done in certain periods of time in relation to the amount of work planned for those periods.

In Excel, you can create a Gantt chart by customizing a Stacked Bar chart type so that it depicts tasks, task duration, and hierarchy. An Excel Gantt chart typically uses days as the unit of time along the horizontal axis.

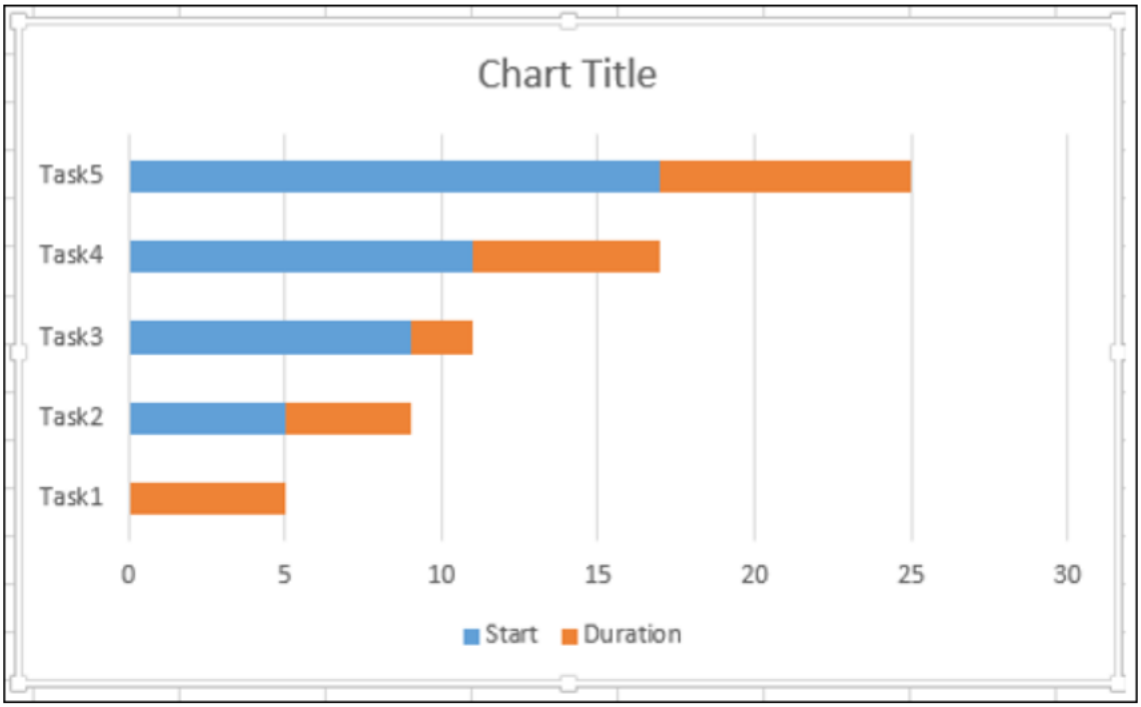
Consider the following data where the column −

- Task represents the Tasks in the project
- Start represents number of days from the Start Date of the project
- Duration represents the duration of the Task

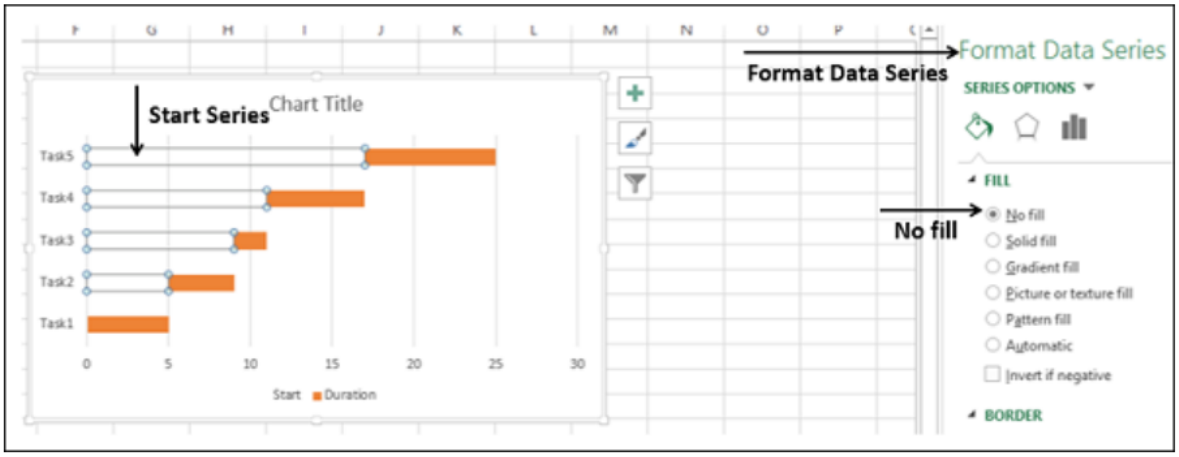
Note that Start of any Task is Start of previous Task + Duration. This is the case when the Tasks are in hierarchy.

	A	B	C	D
1				
2		Task	Start	Duration
3		Task1	0	5
4		Task2	5	4
5		Task3	9	2
6		Task4	11	6
7		Task5	17	8

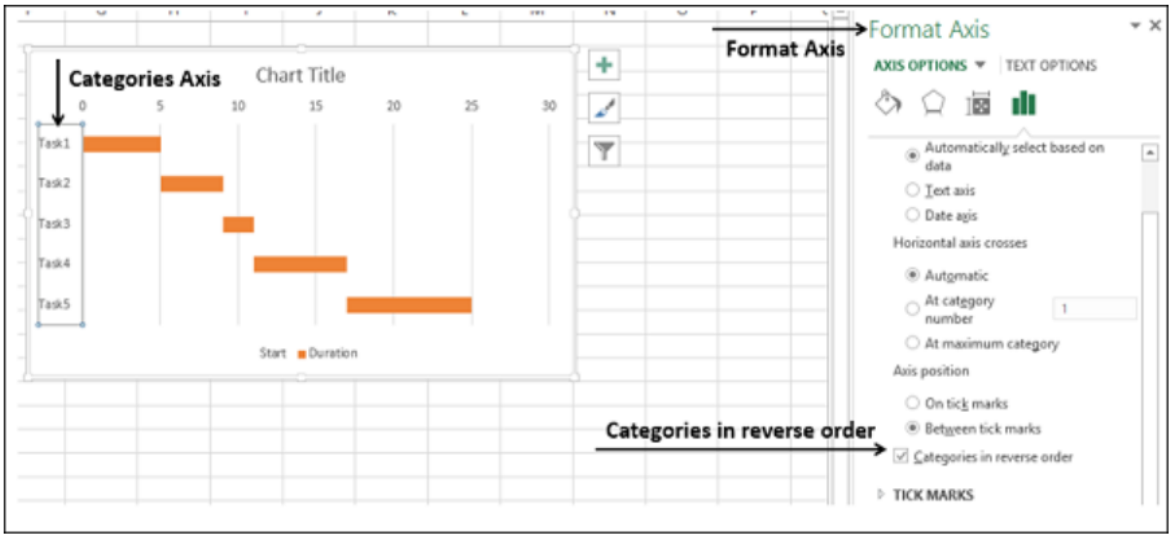
- Select the data.
- Create Stacked Bar Chart.



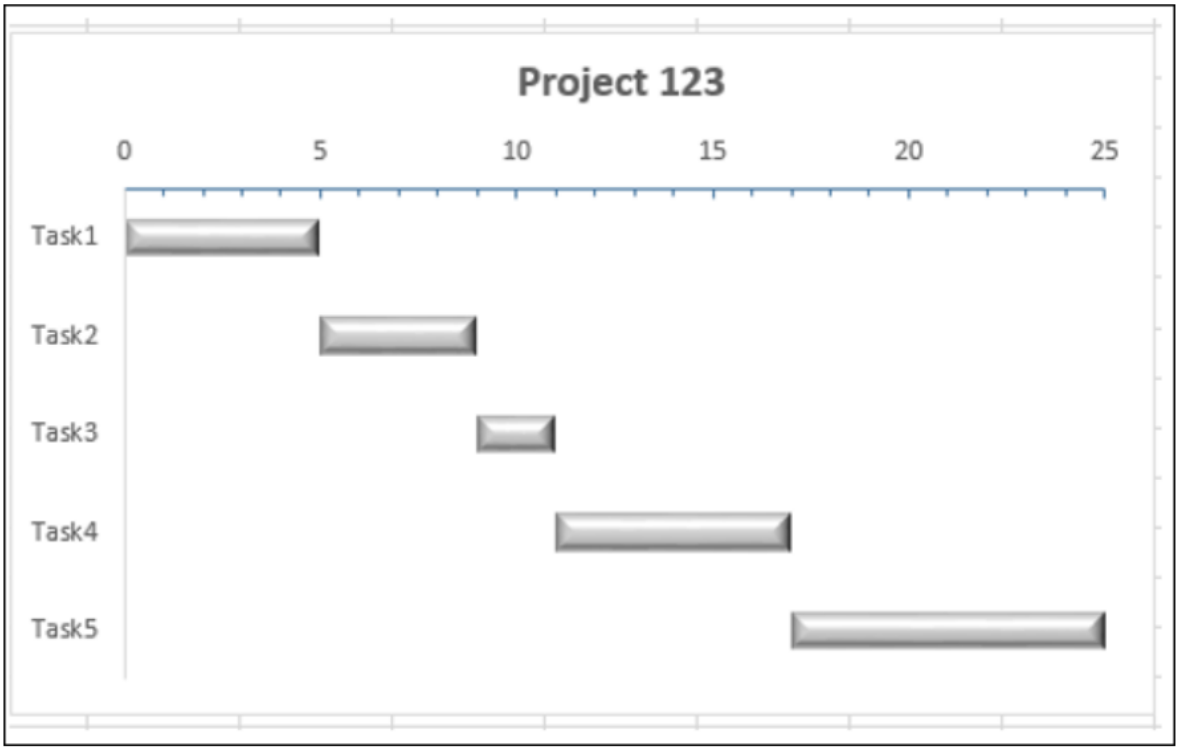
- Right-click on Start Series.
- In Format Data Series options, select No fill.



- Right-click on Categories Axis.
- In Format Axis options, select Categories in reverse order.



- In Chart Elements, deselect
  - Legend
  - Gridlines
- Format the Horizontal Axis to
  - Adjust the range
  - Major Tick Marks at 5 day intervals
  - Minor Tick Marks at 1 day intervals
- Format Data Series to make it look impressive
- Give a Chart Title



SQL

Let’s First Understand the Dataset

What is the best way to learn data analysis? By performing it side by side on a dataset! For this purpose, I have created a dummy dataset of a retail store. The customer data table is represented by *ConsumerDetails*.

Our dataset consists of the following columns:

- **Name** – The name of the consumer
- **Locality** – The locality of the customer
- **Total_amt_spend** – The total amount of money spent by the consumer in the store
- **Industry** – It signifies the industry from which the consumer belongs to

```
mysql> select * from ConsumerDetails;
+----------+-------------+-----------------+---------------+
| Name     | Locality    | Total_amt_spend | Industry      |
+----------+-------------+-----------------+---------------+
| Raj      | Raj Nagar   |             750 | Manufacturing |
| Ajay     | Vijay Nagar |             500 | Creative      |
| Sagar    | Shivam Nagar|             900 | News          |
| Akul     | Preet Vihar |             350 | Teaching      |
| Rohan    | kakar Vihar |            1150 | Tech          |
| Shantanu | Shanti Vihar|            2110 | Defense       |
| Natasha  | shakti nagar|            2200 | Aviation      |
| Kapil    | shakti nagar|             700 | Aviation      |
| Tanamy   | sikkim nagar|             900 | Defense       |
| Tarun    | nikepur     |            3000 | Manufacturing |
+----------+-------------+-----------------+---------------+
```

4. Demonstrate Counting Rows and Items using MySQL.

i. Count function

ii. Distinct function

Counting Rows and Items

- **Count Function**

We will begin our analysis with the simplest query, i.e, counting the number of rows in our table. We will do this by using the function – COUNT().

```
mysql> select count(*) from ConsumerDetails;
+----------+
| count(*) |
+----------+
|       10 |
+----------+
1 row in set (0.00 sec)
```

Great! Now we know the number of rows in our table which is 10. It may seem to be funny using this function on a small test dataset but it can help a lot when your rows run into the millions!

- **Distinct Function**

A lot of times, our data table is filled with duplicate values. To attain the unique value, we use the DISTINCT function.

In our dataset, how can we find the unique industries that customers belong to? You guessed it right. We can do this by using the DISTINCT function.

```
mysql> select distinct Industry from ConsumerDetails;
+---------------+
| Industry      |
+---------------+
| Manufacturing |
| Creative      |
| News          |
| Teaching      |
| Tech          |
| Defense       |
| Aviation      |
+---------------+
7 rows in set (0.00 sec)
```

You can even count the number of unique rows by using the count along with distinct. You can refer to the below query:

```
mysql> select count(distinct Industry) from ConsumerDetails;
+--------------------------+
| count(distinct Industry) |
+--------------------------+
|                        7 |
+--------------------------+
1 row in set (0.00 sec)
```

5. Demonstrate Aggregation functions using MySQL.

i. Calculate the sum

ii. Calculate the average

iii. Calculate the standard deviation

iv. Extreme value identification (min, max)

Aggregation Functions

Aggregation functions are the base of any kind of data analysis. They provide us with an overview of the dataset. Some of the functions we will be discussing are – SUM(), AVG(), and STDDEV().

- **Calculate sum**

We use the **SUM()** function to calculate the sum of the numerical column in a table.

Let's find out the sum of the amount spent by each of the customers:

```
mysql> select sum(Total_amt_spend) as sum_all from ConsumerDetails ;
+---------+
| sum_all |
+---------+
|   12560 |
+---------+
1 row in set (0.00 sec)
```

In the above example, **sum_all** is the variable in which the value of the sum is stored. The sum of the amount of money spent by consumers is Rs. 12,560.

- **Calculate the average**

To calculate the average of the numeric columns, we use the **AVG()** function. Let's find the average expenditure by the consumers for our retail store:

```
mysql> select avg(Total_amt_spend) as avg_total_spend from ConsumerDetails;
+-----------------+
| avg_total_spend |
+-----------------+
|       1256.0000 |
+-----------------+
1 row in set (0.00 sec)
```

The average amount spent by customers in the retail store is Rs. 1256.

- **Calculate standard deviation**

If you have looked at the dataset and then the average value of expenditure by the consumers, you'll have noticed there's something missing. The average does not quite provide the complete picture so let's find another important metric – Standard Deviation. The function is **STDDEV()**.

```
mysql> select stddev(Total_amt_spend) as std_total_spend from ConsumerDetails;
+-------------------+
| std_total_spend   |
+-------------------+
| 829.7132034624976 |
+-------------------+
1 row in set (0.00 sec)
```

The standard deviation comes out to be 829.7 which means there is a high disparity between the expenditures of consumers!

- Max

The maximum numeric value can be identified by using the MAX() function. Let's see how to apply it:

```
mysql> select max(Total_amt_spend) as max_spend from ConsumerDetails;
+-----------+
| max_spend |
+-----------+
|      3000 |
+-----------+
1 row in set (0.00 sec)
```

The maximum amount of money spent by the consumer in the retail store is Rs. 3000.

- Min

Similar to the max function, we have the MIN() function to identify the minimum numeric value in a given column:

```
mysql> select min(Total_amt_spend) as min_spend from ConsumerDetails;
+-----------+
| min_spend |
+-----------+
|       350 |
+-----------+
1 row in set (0.00 sec)
```

The minimum amount of money spent by the retail store consumer is Rs. 350.

6. Demonstrate the usage of the following clauses in MySQL

i. Order by

ii. Group by

iii. Having

Sorting Data

Sorting data helps us put our data into perspective. We can perform the sorting process by using the keyword – ORDER BY.

- **ORDER BY**

The keyword can be used to sort the data into ascending or descending order. The ORDER BY keyword sorts the data in ascending order by default.

Let us see an example where we sort the data according to the column Total_amt_spend in ascending order:

```
mysql> select * from ConsumerDetails order by Total_amt_spend;
+----------+-------------+-----------------+---------------+
| Name     | Locality    | Total_amt_spend | Industry      |
+----------+-------------+-----------------+---------------+
| Akul     | Preet Vihar |             350 | Teaching      |
| Ajay     | Vijay Nagar |             500 | Creative      |
| Kapil    | shakti nagar|             700 | Aviation      |
| Raj      | Raj Nagar   |             750 | Manufacturing |
| Sagar    | Shivam Nagar|             900 | News          |
| Tanamy   | sikkim nagar|             900 | Defense       |
| Rohan    | kakar Vihar |            1150 | Tech          |
| Shantanu | Shanti Vihar|            2110 | Defense       |
| Natasha  | shakti nagar|            2200 | Aviation      |
| Tarun    | nikepur     |            3000 | Manufacturing |
+----------+-------------+-----------------+---------------+
```

Awesome! To order the dataset into descending order, we can follow the below command:

```
mysql> select * from ConsumerDetails order by Total_amt_spend desc;
+----------+-------------+-----------------+---------------+
| Name     | Locality    | Total_amt_spend | Industry      |
+----------+-------------+-----------------+---------------+
| Tarun    | nikepur     |            3000 | Manufacturing |
| Natasha  | shakti nagar|            2200 | Aviation      |
| Shantanu | Shanti Vihar|            2110 | Defense       |
| Rohan    | kakar Vihar |            1150 | Tech          |
| Sagar    | Shivam Nagar|             900 | News          |
| Tanamy   | sikkim nagar|             900 | Defense       |
| Raj      | Raj Nagar   |             750 | Manufacturing |
| Kapil    | shakti nagar|             700 | Aviation      |
| Ajay     | Vijay Nagar |             500 | Creative      |
| Akul     | Preet Vihar |             350 | Teaching      |
+----------+-------------+-----------------+---------------+
```

We have finally arrived at one of the most powerful analysis tools in SQL – Grouping of data which is performed using the GROUP BY statement. The most useful application of this statement is to find the distribution of categorical variables. This is done by using the GROUP BY statement along with aggregation functions like – COUNT, SUM, AVG, etc.

Let’s try to understand this better by taking up a problem statement. The retail store wants to find the Number of Customers corresponding to the industries they belong to:

```
mysql> select count(*), Industry from ConsumerDetails group by Industry;
+----------+---------------+
| count(*) | Industry      |
+----------+---------------+
|        2 | Aviation      |
|        1 | Creative      |
|        2 | Defense       |
|        2 | Manufacturing |
|        1 | News          |
|        1 | Teaching      |
|        1 | Tech          |
+----------+---------------+
7 rows in set (0.01 sec)
```

We notice that the count of customers belonging to the various industries is more or less the same. So, let us move forward and find the sum of spendings by customers grouped by the industry they belong to:

```
mysql> select sum(total_amt_spend) as category_sum, Industry from ConsumerDetails group by Industry;
+--------------+---------------+
| category_sum | Industry      |
+--------------+---------------+
|         2900 | Aviation      |
|          500 | Creative      |
|         3010 | Defense       |
|         3750 | Manufacturing |
|          900 | News          |
|          350 | Teaching      |
|         1150 | Tech          |
+--------------+---------------+
7 rows in set (0.00 sec)
```

We can observe that the maximum amount of money spent is by the customers belonging to the ***Manufacturing*** industry. This seems a bit easy, right? Let us take a step ahead and make it more complicated.

Now, the retailer wants to find the industries whose ***total_sum*** is greater than 2500. To solve this problem, we will again group by the data according to the industry and then use the HAVING clause.

- **HAVING**

The HAVING clause is just like the WHERE clause but only for filtering the grouped by data. Remember, it will always come after the GROUP BY statement.

```
mysql> select sum(total_amt_spend) as category_sum, Industry from ConsumerDetails group by Industry having category_sum >2500;
+--------------+---------------+
| category_sum | Industry      |
+--------------+---------------+
|         2900 | Aviation      |
|         3010 | Defense       |
|         3750 | Manufacturing |
+--------------+---------------+
3 rows in set (0.00 sec)
```

We have only 3 categories that satisfy the conditions – ***Aviation***, ***Defense***, and ***Manufacturing***. But to make it more clearer, I will also add the ORDER BY keyword to make it more intuitive:

```
mysql> select sum(total_amt_spend) as category_sum, Industry from ConsumerDetails group by Industry having category_sum >2500 order by category_sum desc;
+--------------+---------------+
| category_sum | Industry      |
+--------------+---------------+
|         3750 | Manufacturing |
|         3010 | Defense       |
|         2900 | Aviation      |
+--------------+---------------+
3 rows in set (0.00 sec)
```

7. Demonstrate the use of following joins in MySQL.

i. Inner join

ii. Left Outer Join

iii. Right Outer Join

iv. Full Outer Join

SQL OUTER JOIN – left outer join

SQL left outer join is also known as SQL left join. Suppose, we want to join two tables: A and B. SQL left outer join returns all rows in the left table (A) and all the matching rows found in the right table (B). It means the result of the SQL left join always contains the rows in the left table.

The following illustrate SQL left outer syntax of joining 2 tables: table_A and table_B:

```
SELECT column1, column2...
FROM table_A
LEFT JOIN table_B ON join_condition
WHERE row_condition
```

SQL OUTER JOIN – left outer join example

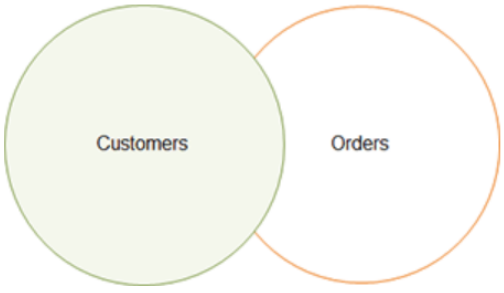
The following query selects all customers and their orders:

```
SELECT c.customerid,
       c.companyName,
       orderid
FROM customers c
LEFT JOIN orders o ON o.customerid = c.customerid
ORDER BY orderid
```

customerid	companyName	orderid
FISSA	FISSA Fabrica Inter. Salchichas S.A.	NULL
PARIS	Paris spécialités	NULL
VINET	Vins et alcools Chevalier	10248
TOMSP	Toms Spezialitäten	10249
HANAR	Hanari Carnes	10250
VICTE	Victuailles en stock	10251

All rows in the *customers* table are listed. In case, there is no matching row in the *orders* table found for the row in the *customers* table, the orderid column in the orders table is populated with NULL values.

We can use Venn diagram to visualize how SQL LEFT OUTER JOIN works.



SQL OUTER JOIN – right outer join

SQL right outer join returns all rows in the right table and all the matching rows found in the left table. The syntax of the SQL right outer join is as follows:

```
SELECT column1, column2...
FROM table_A
RIGHT JOIN table_B ON join_condition
WHERE row_condition
```

SQL right outer join is also known as SQL right join.

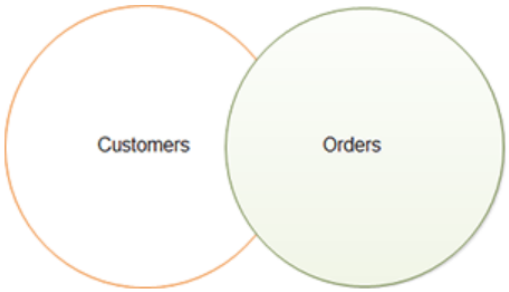
SQL OUTER JOIN – right outer join example

The following example demonstrates the SQL right outer join:

```
SELECT c.customerid,
       c.companyName,
       orderid
FROM customers c
RIGHT JOIN orders o ON o.customerid = c.customerid
ORDER BY orderid
```

The query returns all rows in the *orders* table and all matching rows found in the *customers* table.

The following Venn diagram illustrates how the SQL right outer join works:



SQL OUTER JOIN – full outer join

The syntax of the SQL full outer join is as follows:

```
SELECT column1, column2...
FROM table_A
FULL OUTER JOIN table_B ON join_condition
WHERE row_condition
```

SQL full outer join returns:

- all rows in the left table table_A.
- all rows in the right table table_B.
- and all matching rows in both tables.

Some database management systems do not support SQL full outer join syntax e.g., MySQL. Because SQL full outer join returns a result set that is a combined result of both SQL left join and SQL right join. Therefore you can easily emulate the SQL full outer join using SQL left join and SQL right join with UNION operator as follows:

```
SELECT column1, column2...
FROM table_A
LEFT JOIN table_B ON join_condition
UNION
SELECT column1, column2...
FROM table_A
RIGHT JOIN table_B ON join_condition
```

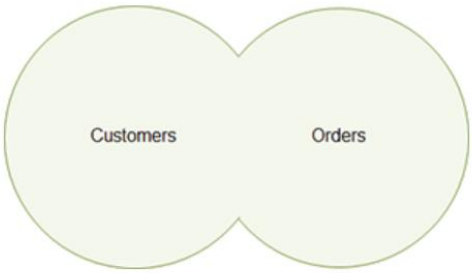
SQL OUTER JOIN – full outer join example

The following query demonstrates the SQL full outer join:

```
SELECT c.customerid,
       c.companyName,
       orderid
FROM customers c
FULL OUTER JOIN orders o ON o.customerid = c.customerid
ORDER BY orderid
```

customerid	companyName	orderid
FISSA	FISSA Fabrica Inter. Salchichas S.A.	NULL
PARIS	Paris spécialités	NULL
VINET	Vins et alcools Chevalier	10248
TOMSP	Toms Spezialitäten	10249
HANAR	Hanari Carnes	10250
VICTE	Victuailles en stock	10251
SUPRD	Suprêmes délices	10252
HANAR	Hanari Carnes	10253
CHOPS	Chop-suey Chinese	10254
RICSU	Richter Supermarkt	10255

The following Venn diagram illustrates how SQL full outer join works:



SQL INNER JOIN syntax

The following illustrates `INNER JOIN` syntax for joining two tables:

```
SELECT
    column1, column2
FROM
    table_1
INNER JOIN table_2 ON join_condition;
```

Let's examine the syntax above in greater detail:

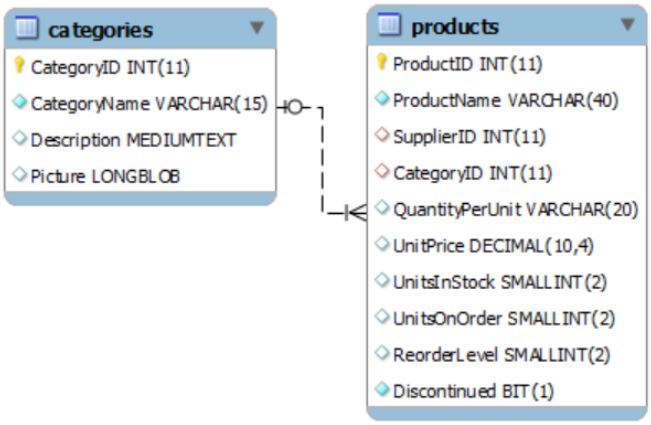
- The `table_1` and `table_2` are called joined-tables.
- For each row in the `table_1`, the query find the corresponding row in the `table_2` that meet the join condition. If the corresponding row found, the query returns a row that contains data from both tables. Otherwise, it examines next row in the `table_1`, and this process continues until all the rows in the `table_1` are examined.

For joining more than two tables, the same logic applied.

SQL INNER JOIN examples

SQL INNER JOIN – querying data from two tables example

In this example, we will use the `products` and `categories` tables in the sample database. The following picture illustrates the database diagram.



In the diagram above:

- One category can have many products.
- One product belongs to one and only one category.

Therefore, there is a many-to-one relationship between the rows in the `categories` table and rows in the `products` table. The link between the two tables is the `categoryid` column.

We need to query the following data from both tables:

- `productID` , `productName` from the `products` table.
- `categoryName` from the `categories` table.

The following query retrieves data from both tables:

```
SELECT
    productID, productName, categoryName
FROM
    products
INNER JOIN
    categories ON categories.categoryID = products.categoryID;
```

productID	productName	categoryName
1	Chai	Beverages
2	Chang	Beverages
3	Aniseed Syrup	Condiments
4	Chef Anton's Cajun Seasoning	Condiments
5	Chef Anton's Gumbo Mix	Condiments
6	Grandma's Boysenberry Spread	Condiments
7	Uncle Bob's Organic Dried Pears	Produce
8	Northwoods Cranberry Sauce	Condiments

The join condition is specified in the `INNER JOIN` clause after the `ON` keyword as the expression:

```
categories.categoryID = products.categoryID
```

For each row in the `products` table, the query finds a corresponding row in the `categories` table that has the same `categoryid.` If there is a match between two rows in both tables, it returns a row that contains columns specified in the SELECT clause i.e., product id, product name and category name; otherwise, it checks the next row in `products` table to find the matching row in the `categories` table. This process continues until the last row of the products table is examined.

8. Demonstrate the use of the following functions in MySQL

i. CASE WHEN

ii. COALESCE

iii. NULLIF

iv. LEAST/GREATEST

CASE WHEN

Definition and Usage

The CASE statement goes through conditions and return a value when the first condition is met (like an IF-THEN-ELSE statement). So, once a condition is true, it will stop reading and return the result.

If no conditions are true, it will return the value in the ELSE clause.

If there is no ELSE part and no conditions are true, it returns NULL.

Syntax

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

Parameter Values

Parameter	Description
condition1, condition2, ...conditionN	Required. The conditions. These are evaluated in the same order as they are listed
result1, result2, ...resultN	Required. The value to return once a condition is true

Example

Go through conditions and return a value when the first condition is met:

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN "The quantity is greater than 30"
    WHEN Quantity = 30 THEN "The quantity is 30"
    ELSE "The quantity is under 30"
END
FROM OrderDetails;
```

Result:

Number of Records: 2155

OrderID	Quantity	QuantityText
10248	12	The quantity is under 30
10248	10	The quantity is under 30
10248	5	The quantity is under 30
10249	9	The quantity is under 30
10249	40	The quantity is greater than 30
10250	10	The quantity is under 30
10250	35	The quantity is greater than 30
10250	15	The quantity is under 30

COALESCE

Definition and Usage

The COALESCE() function returns the first non-null value in a list.

Syntax

```
COALESCE(val1, val2, ...., val_n)
```

Parameter Values

Parameter	Description
val1, val2, val_n	Required. The values to test

Example

Return the first non-null value in a list:

```
SELECT COALESCE(NULL, NULL, NULL, 'W3Schools.com', NULL, 'Example.com');
```

Result:

Number of Records: 1

COALESCE(NULL, NULL, NULL, 'W3Schools.com', NULL, 'Example.com')
W3Schools.com

NULLIF

Definition and Usage

The NULLIF() function compares two expressions and returns NULL if they are equal. Otherwise, the first expression is returned.

Syntax

```
NULLIF(expr1, expr2)
```

Parameter Values

Parameter	Description
expr1, expr2	Required. The two expressions to be compared

Example

Compare two expressions:

```
SELECT NULLIF(25, 25);
```

Result:

Number of Records: 1

NULLIF(25, 25)

LEAST/GREATEST

Definition and Usage

The LEAST() function returns the smallest value of the list of arguments.

Syntax

```
LEAST(arg1, arg2, arg3, ...)
```

Parameter Values

Parameter	Description
arg1, *arg2*, *arg3*, ...	Required. The list of arguments to be evaluated

Example

Return the smallest value of the list of arguments:

```
SELECT LEAST(3, 12, 34, 8, 25);
```

Result:

Number of Records: 1

LEAST(3, 12, 34, 8, 25)
3

Definition and Usage

The GREATEST() function returns the greatest value of the list of arguments.

Syntax

```
GREATEST(arg1, arg2, arg3, ...)
```

Parameter Values

Parameter	Description
arg1, *arg2*, *arg3*, ...	Required. The list of arguments to be evaluated

Example

Return the greatest value of the list of arguments:

```
SELECT GREATEST(3, 12, 34, 8, 25);
```

Result:

Number of Records: 1

GREATEST(3, 12, 34, 8, 25)
34

9. Demonstrate the use of the following operators in MySQL

i. AND, OR, NOT

ii. LIKE

iii. BETWEEN

iv. IN, NOT IN

v. EXISTS, NOT EXISTS

vi. IS NULL, IS NOT NULL

The MySQL AND, OR and NOT Operators

The `WHERE` clause can be combined with `AND`, `OR`, and `NOT` operators.

The `AND` and `OR` operators are used to filter records based on more than one condition:

- The `AND` operator displays a record if all the conditions separated by `AND` are TRUE.
- The `OR` operator displays a record if any of the conditions separated by `OR` is TRUE.

The `NOT` operator displays a record if the condition(s) is NOT TRUE.

AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

The MySQL LIKE Operator

The `LIKE` operator is used in a `WHERE` clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the `LIKE` operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

The percent sign and the underscore can also be used in combinations!

LIKE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

Here are some examples showing different `LIKE` operators with '%' and '_' wildcards:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

The SQL BETWEEN Operator

The `BETWEEN` operator selects values within a given range. The values can be numbers, text, or dates.

The `BETWEEN` operator is inclusive: begin and end values are included.

BETWEEN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

IN() function

MySQL IN() function finds a match in the given arguments.

Syntax:

```
expr IN (value,...)
```

The function returns 1 if expr is equal to any of the values in the IN list, otherwise, returns 0. If all values are constants, they are evaluated according to the type of expr and sorted. The search for the item then is done using a binary search. This means IN is very quick if the IN value list consists entirely of constants. Otherwise, type conversion takes place according to the rules.

NOT IN() function

MySQL NOT IN() makes sure that the expression proceeded does not have any of the values present in the arguments.

Syntax:

```
expr NOT IN (value,...)
```

The SQL EXISTS Operator

The `EXISTS` operator is used to test for the existence of any record in a subquery.

The `EXISTS` operator returns TRUE if the subquery returns one or more records.

EXISTS Syntax

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

EXISTS and NOT EXISTS are most likely to used when your playing with the subqueries in MySQL.
So these will be used when you want to perform operation based on checking row existance of the other table or even same table.

If sub query will return any value then EXISTS will be TRUE and NOT EXISTS will be FALSE. And if sub query will return nothing then EXISTS will be FALSE and NOT EXISTS will be TRUE.

MySQL: IS NULL Condition

This MySQL tutorial explains how to use the MySQL **IS NULL condition** with syntax and examples.

Description

The MySQL IS NULL Condition is used to test for a NULL value in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax

The syntax for the IS NULL Condition in MySQL is:

```
expression IS NULL
```

Parameters or Arguments

expression
The value to test if it is a NULL value.

Note

- If *expression* is a NULL value, the condition evaluates to TRUE.
- If *expression* is not a NULL value, the condition evaluates to FALSE.

MySQL: IS NOT NULL

This MySQL tutorial explains how to use the MySQL **IS NOT NULL condition** with syntax and examples.

Description

The MySQL IS NOT NULL condition is used to test for a NOT NULL value in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax

The syntax for the IS NOT NULL Condition in MySQL is:

```
expression IS NOT NULL
```

Parameters or Arguments

expression
The value to test if it is a not NULL value.

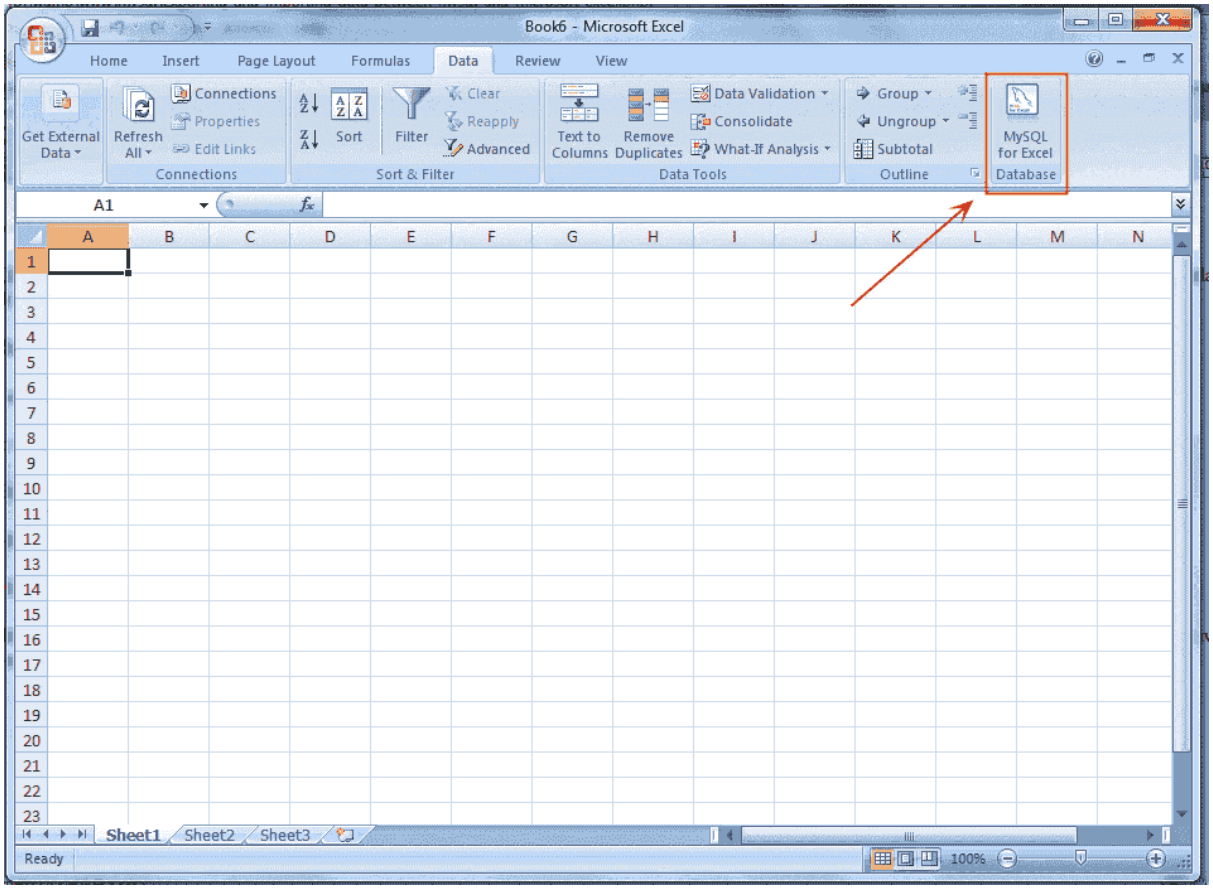
Note

- If *expression* is NOT a NULL value, the condition evaluates to TRUE.
- If *expression* is a NULL value, the condition evaluates to FALSE.

10. Exporting data from MySQL to a file for further processing in Excel.

MySQL for Excel is loaded and executed by selecting the Data menu tab in Excel, and then choosing the "MySQL for Excel" Database icon. This opens a

new Excel sidebar with the available MySQL for Excel options. The navigation bar with the MySQL for Excel icon is shown in the following screen shot:



Edit MySQL Data in Excel

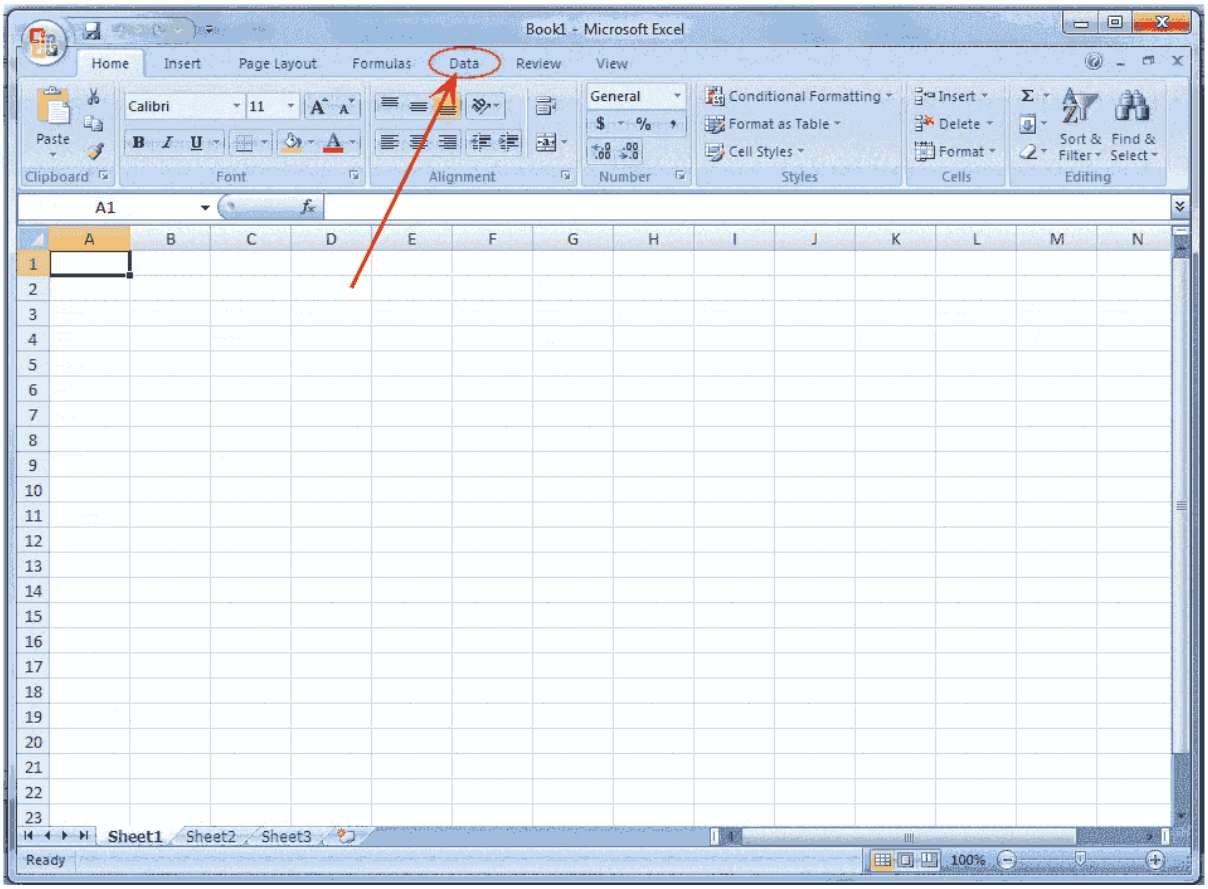
MySQL for Excel enables you to load and edit MySQL data directly from Microsoft Excel, or yon can do it manually by pressing Commit Changes.

The example below uses the location table of the example employee database, but the screen will look the same for any table. Within MySQL for Excel, Open a MySQL Connection, click the employee schema, Next, select the location table, click Edit MySQL Data, then choose Import to import the data into a new Microsoft Excel worksheet for editing.

Here is the step by step guide to editing and commit the data:

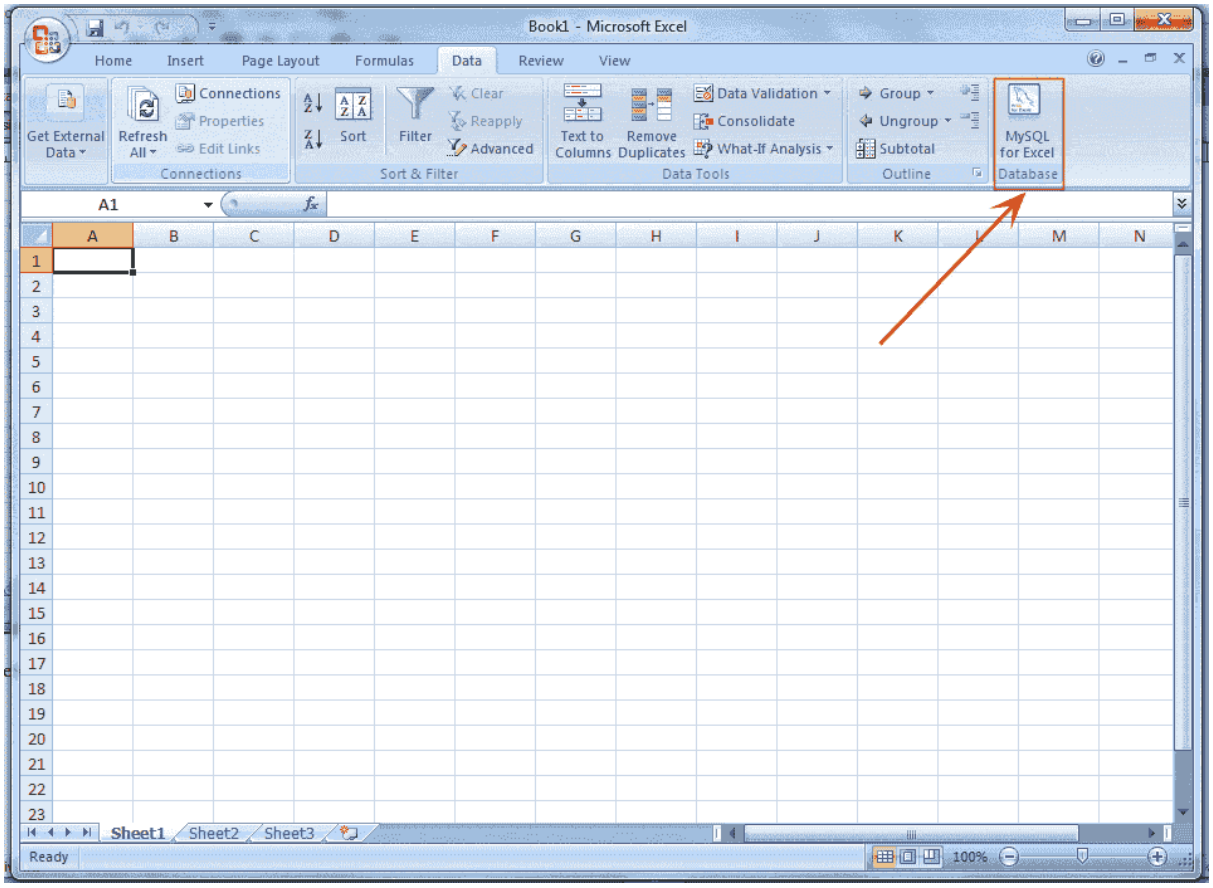
Step 1:

Load Microsoft Office Excel 7



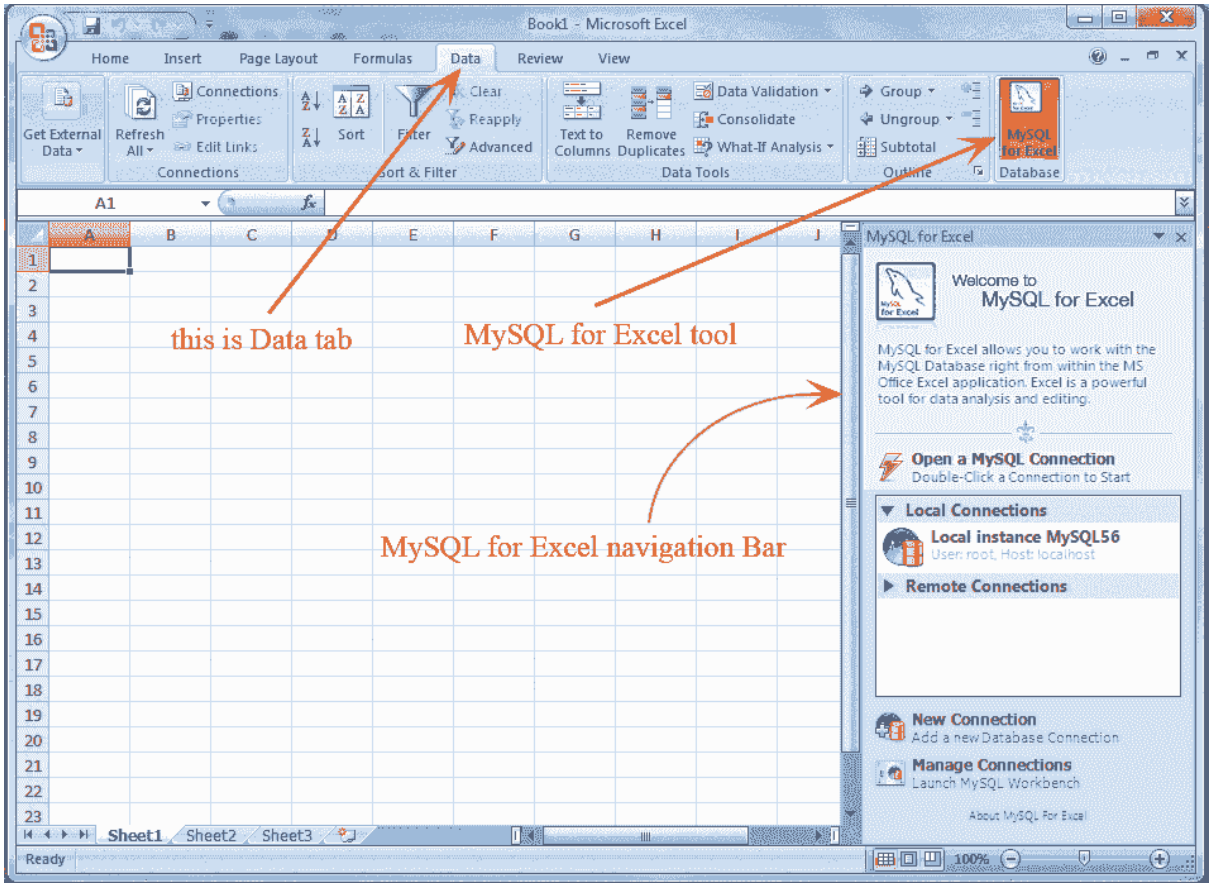
Step 2:

Click on **Data** Tab, see the above picture, the" MySQL for Excel" Database icon will appear shown below.



Step 3:

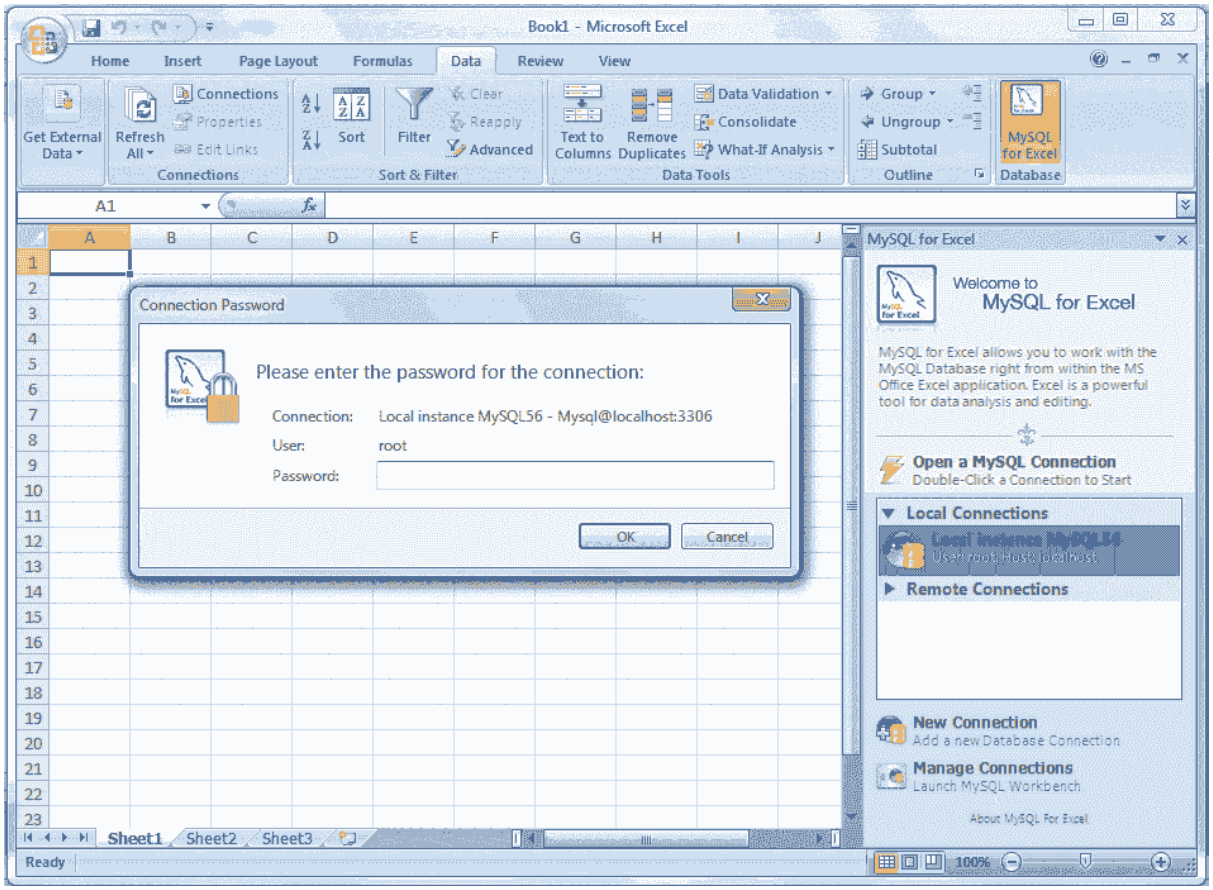
Click on "MySQL for Excel" Database icon. It opens a new Excel sidebar with the available MySQL for Excel options. The navigation bar with the MySQL for Excel icon is shown in the following picture.:



Here our Database is **employee** and we are working with **location** table, but the screen will look the same for any table.

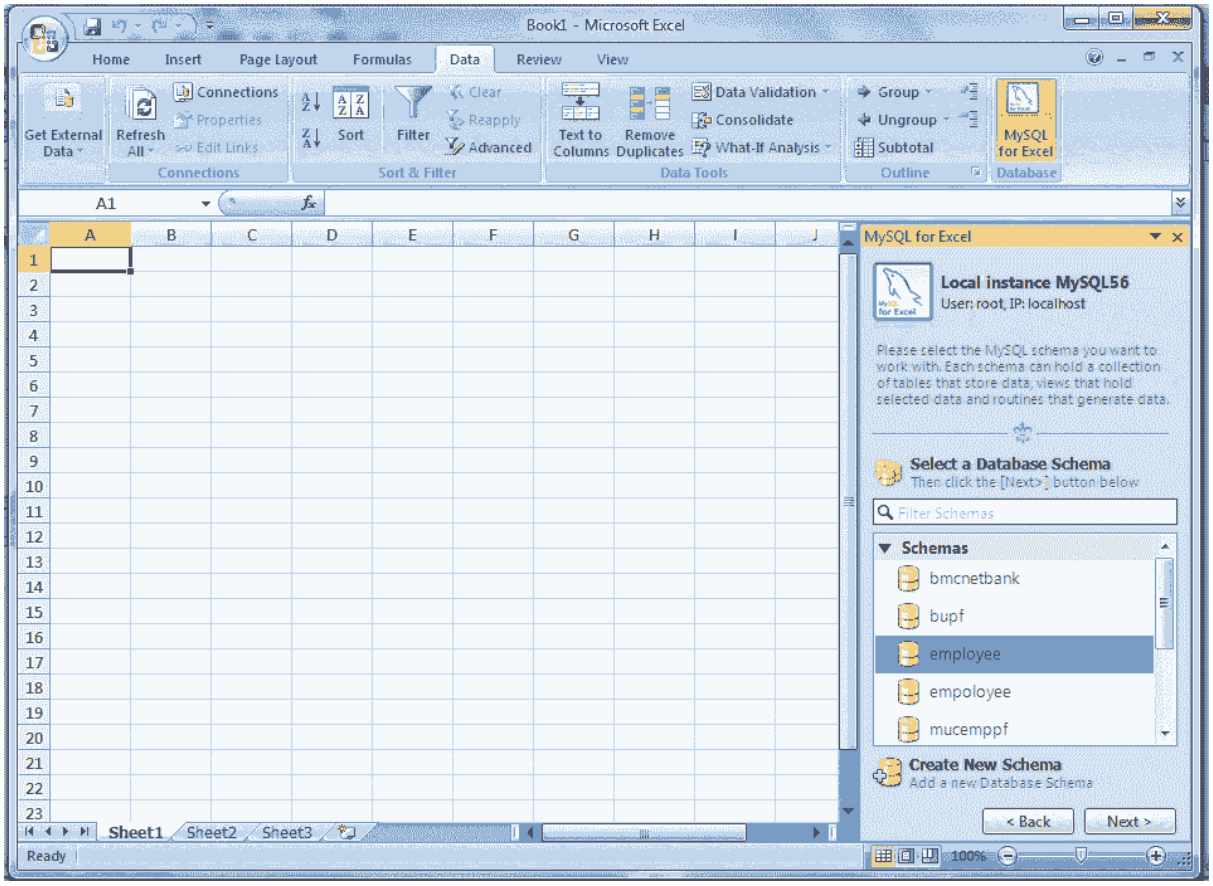
Step 4:

Within MySQL for Excel sidebar open a MySQL connection by double clicking. Here our connection is **Local instance MySQL5.6** and with the following screen will appear for accepting the password.



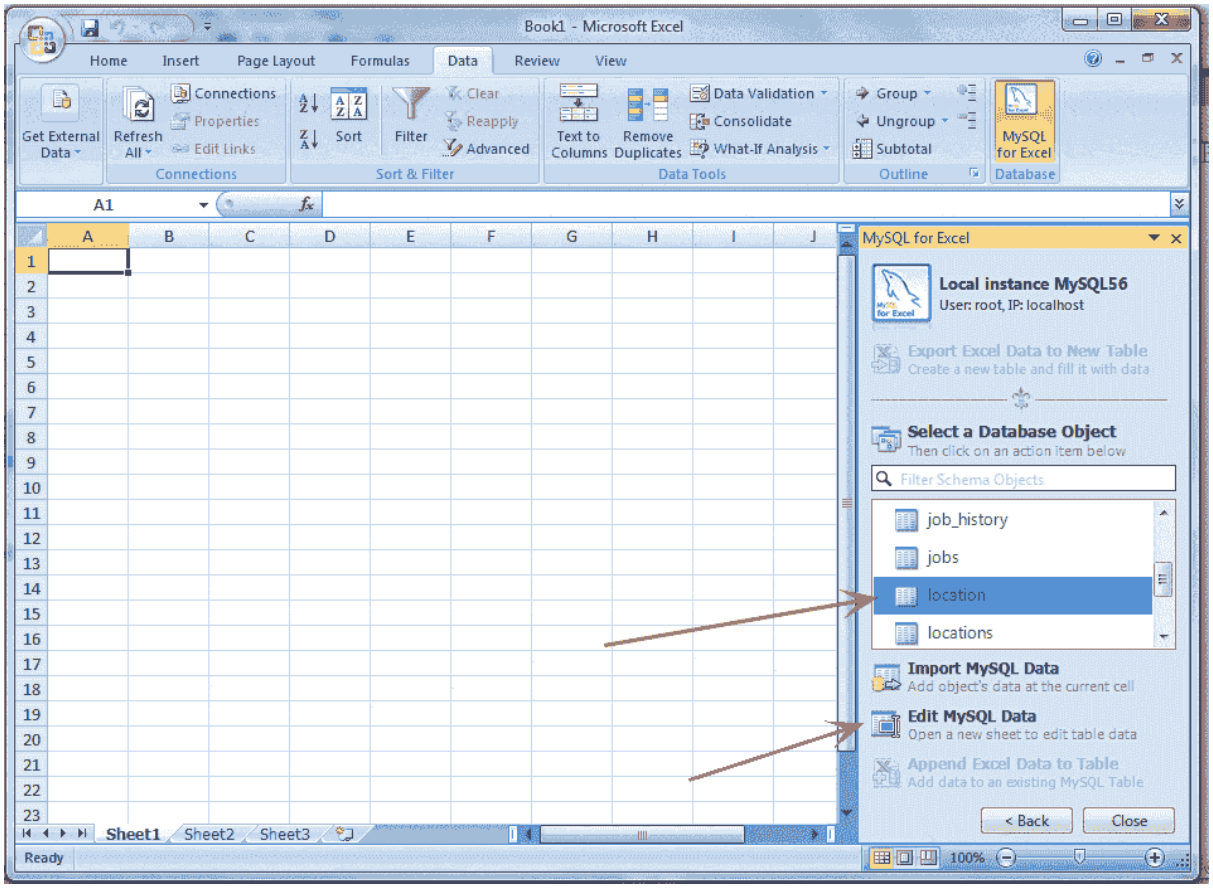
Step 5:

Enter password for connection with MySQL server. The databases will show in MySQL for Excel sidebar. Our Database is employee. See the following picture.



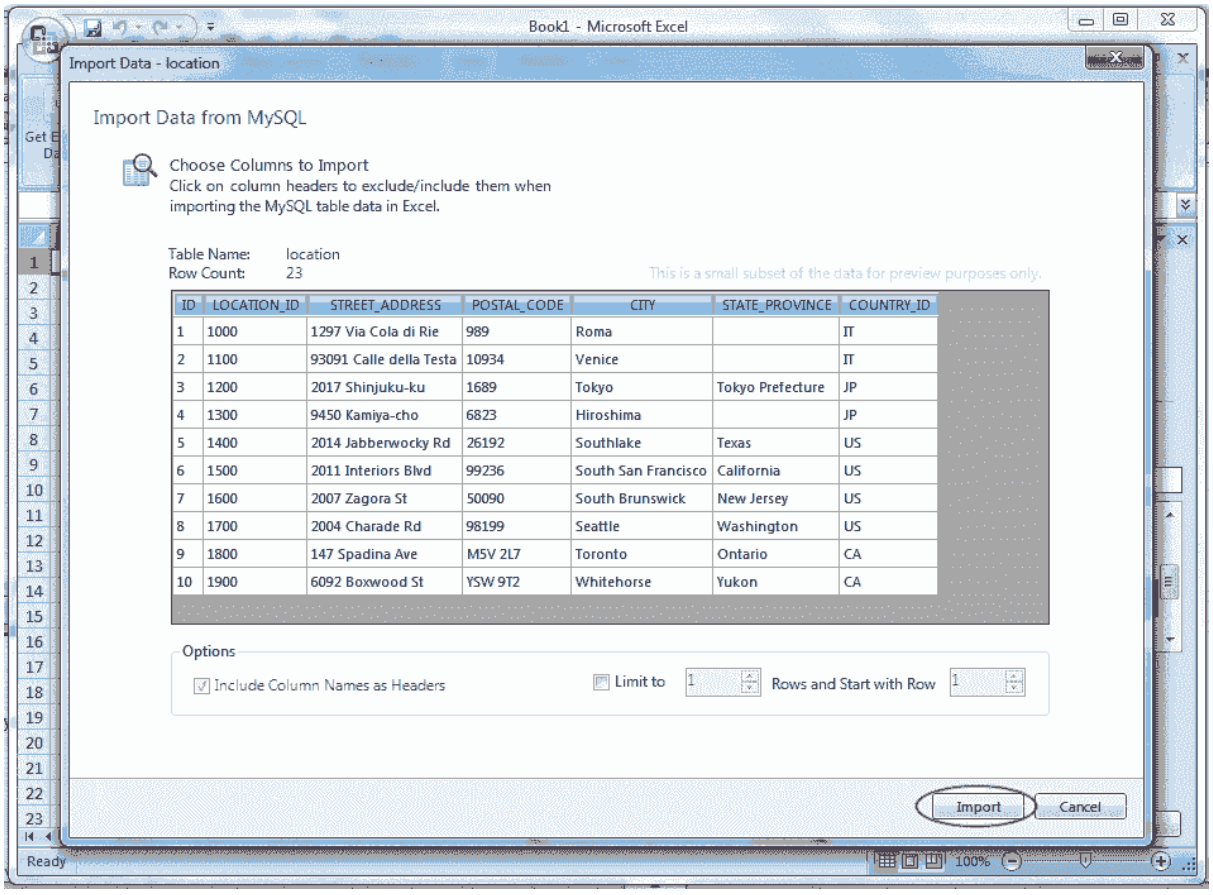
Step 6:

Double click on your desire database, and the tables within the database will display. Our table is location. See the following picture.



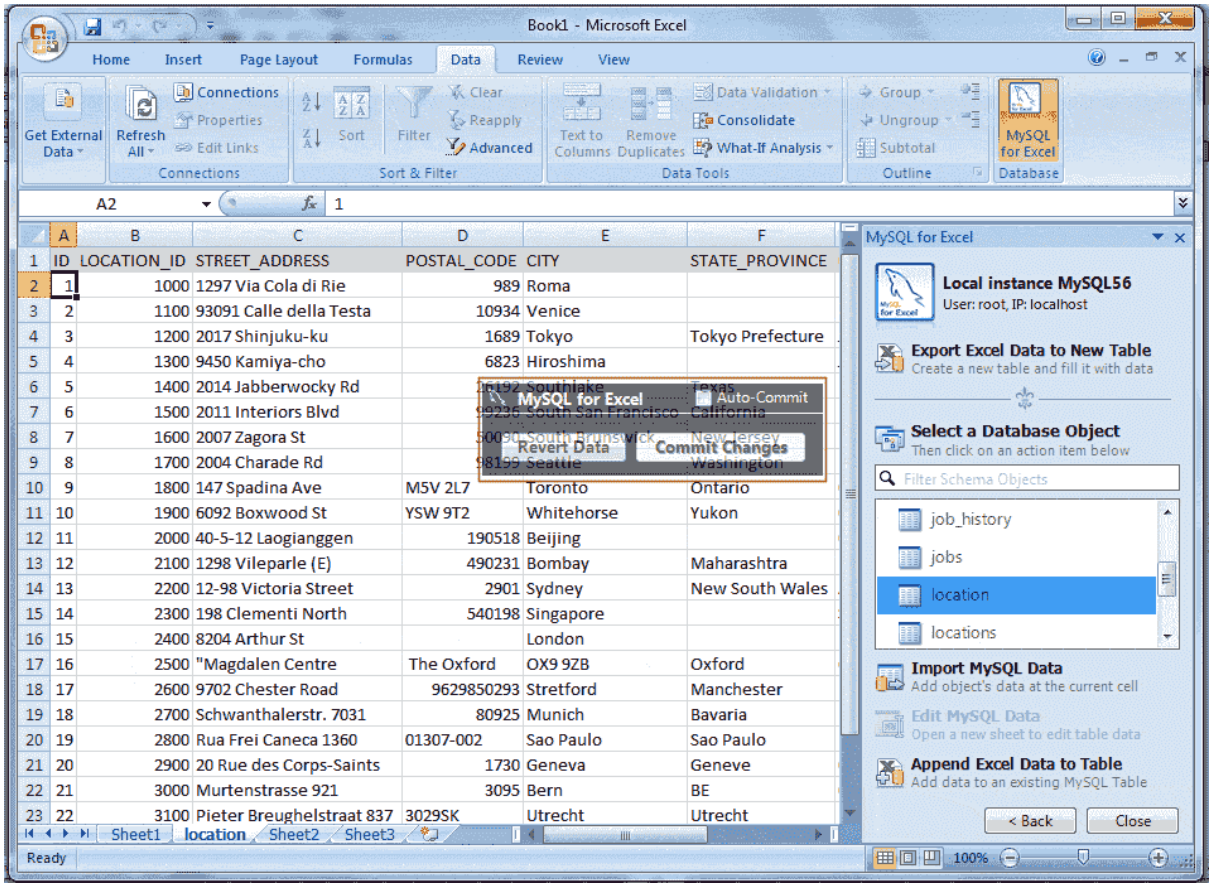
Step 7:

Select the table which you want to edit, click "Edit MySQL Data" inside the navigation bar as shown above, and see the following screen shot.



Step 8:

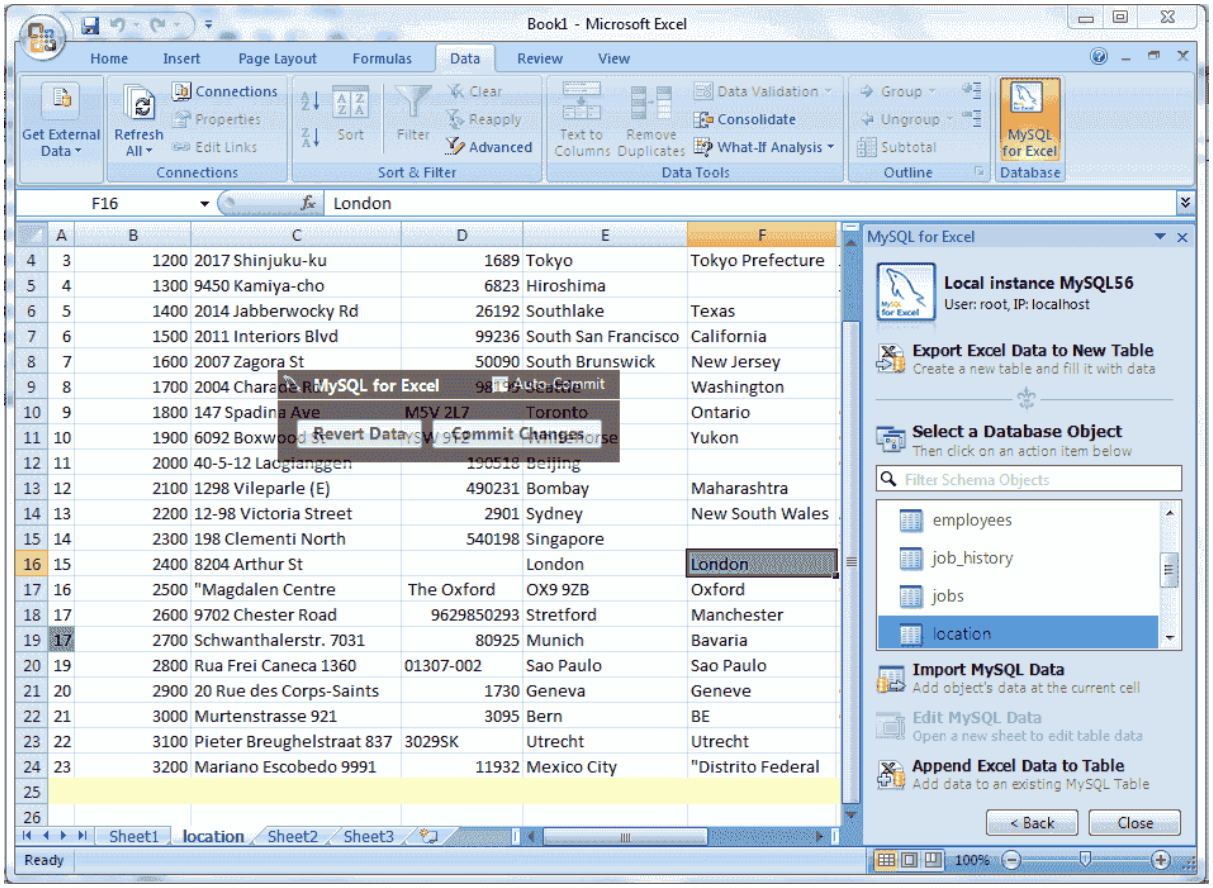
Click on Import button as mention in the above picture, and watch the following screen shot. The data of the selected table will appear and if you place the cursor on the data range the Revert Data and Commit Changes button (specified by a red color rectangle) will appear otherwise not.



	A	B	C	D	E	F
1	ID	LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE
2	1	1000	1297 Via Cola di Rie	989	Roma	
3	2	1100	93091 Calle della Testa	10934	Venice	
4	3	1200	2017 Shinjuku-ku	1689	Tokyo	Tokyo Prefecture
5	4	1300	9450 Kamiya-cho	6823	Hiroshima	
6	5	1400	2014 Jabberwocky Rd	16192	Southlake	Texas
7	6	1500	2011 Interiors Blvd	99236	South San Francisco	California
8	7	1600	2007 Zagora St	50090	South Brunswick	New Jersey
9	8	1700	2004 Charade Rd	98199	Seattle	Washington
10	9	1800	147 Spadina Ave	M5V 2L7	Toronto	Ontario
11	10	1900	6092 Boxwood St	YSW 9T2	Whitehorse	Yukon
12	11	2000	40-5-12 Laogianggen	190518	Beijing	
13	12	2100	1298 Vileparle (E)	490231	Bombay	Maharashtra
14	13	2200	12-98 Victoria Street	2901	Sydney	New South Wales
15	14	2300	198 Clementi North	540198	Singapore	
16	15	2400	8204 Arthur St		London	
17	16	2500	"Magdalen Centre	The Oxford	OX9 9ZB	Oxford
18	17	2600	9702 Chester Road	9629850293	Stretford	Manchester
19	18	2700	Schwanthalerstr. 7031	80925	Munich	Bavaria
20	19	2800	Rua Frei Caneca 1360	01307-002	Sao Paulo	Sao Paulo
21	20	2900	20 Rue des Corps-Saints	1730	Geneva	Geneve
22	21	3000	Murtenstrasse 921	3095	Bern	BE
23	22	3100	Pieter Breughelstraat 837	3029SK	Utrecht	Utrecht

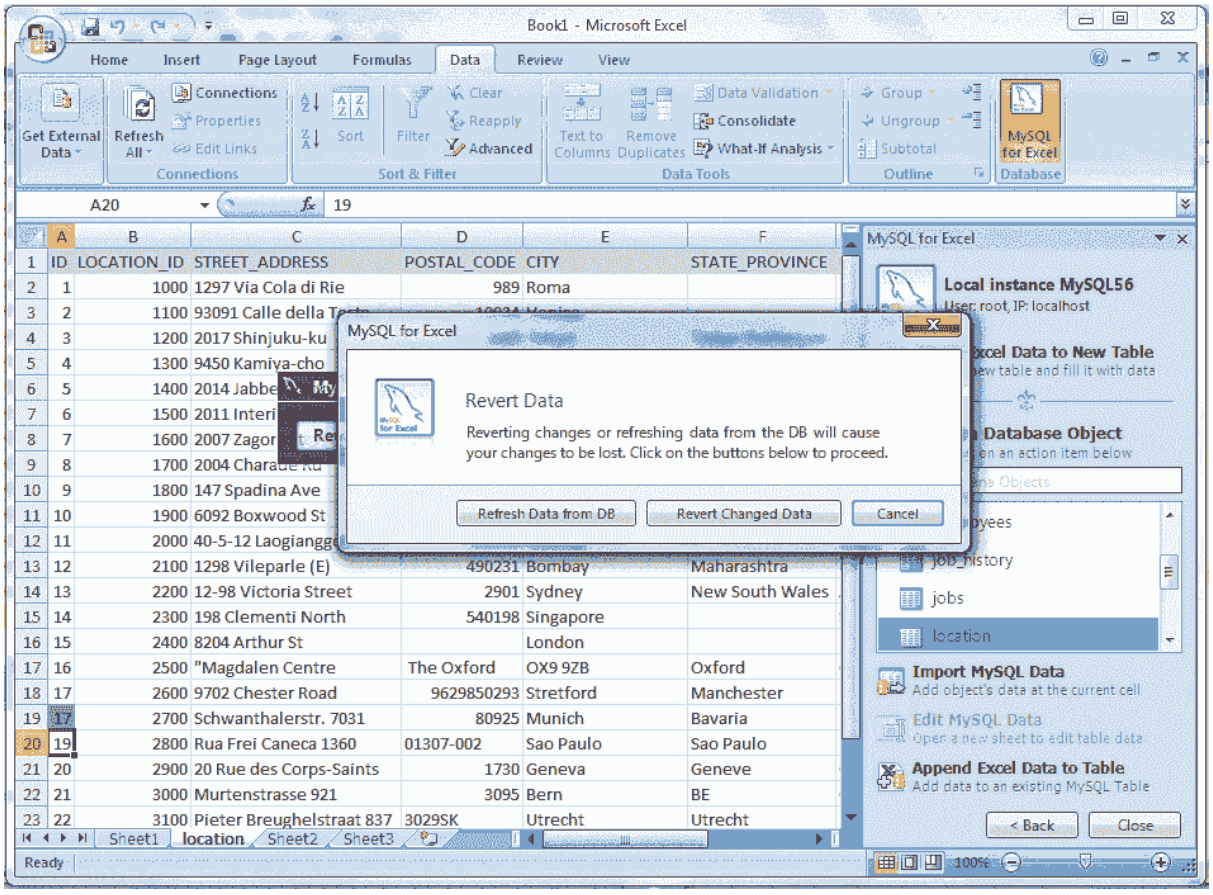
Step 9:

Here at our table, the first column ID is a primary key. If we change the value of ID in A19 cell 17 instead of 18 and press enter key the color of this cell will be green. Now we have changed the value of F16 cell is London and press enter key the cell color will change into green, and the color yellow at the last, indicate that the cells that accept new data. Data entered here is inserted into the MySQL table.



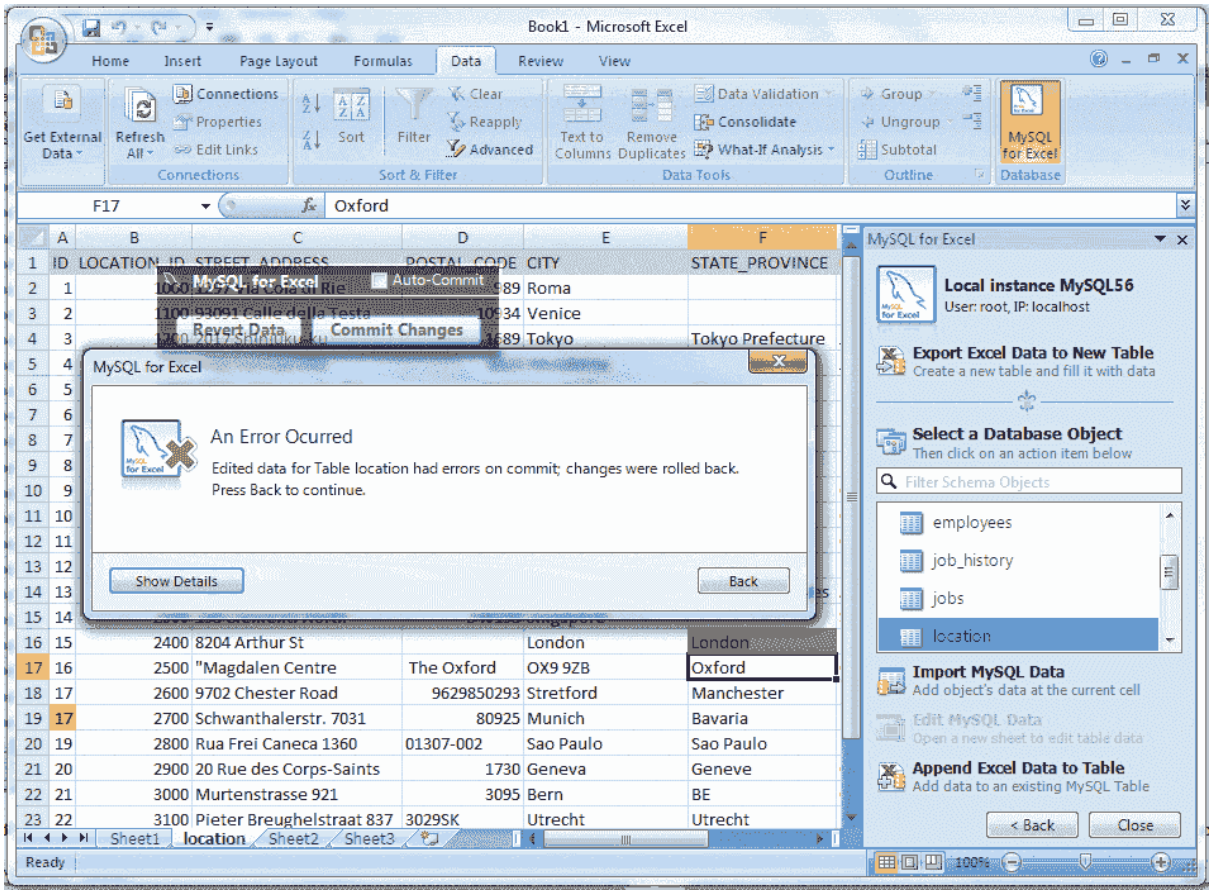
Step 10:

Now if we click the Revert Data button, watch the following screen shot.

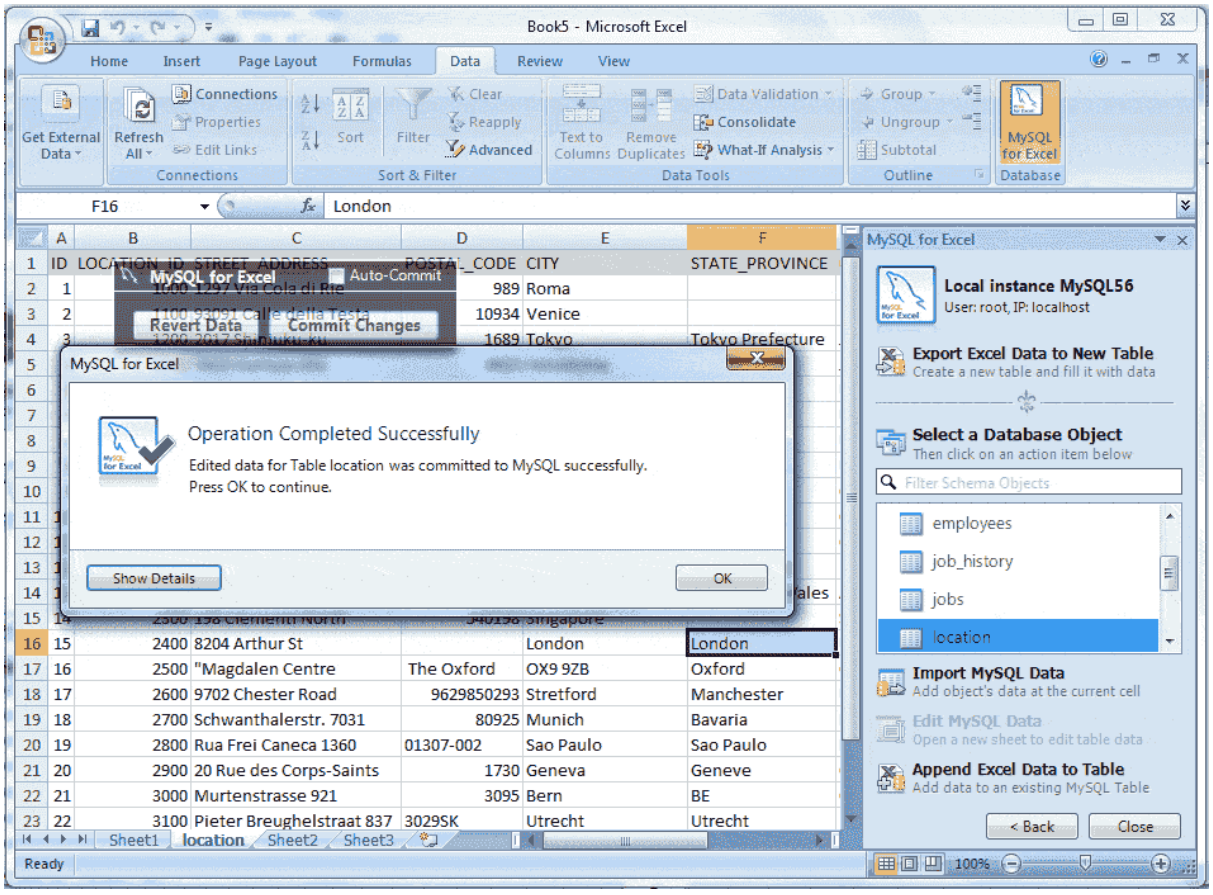


	A	B	C	D	E	F
1	ID	LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE
2	1	1000	1297 Via Cola di Rie	989	Roma	
3	2	1100	93091 Calle della Te			
4	3	1200	2017 Shinjuku-ku			
5	4	1300	9450 Kamiya-cho			
6	5	1400	2014 Jabbe			
7	6	1500	2011 Interi			
8	7	1600	2007 Zagor			
9	8	1700	2004 Charade Rd			
10	9	1800	147 Spadina Ave			
11	10	1900	6092 Boxwood St			
12	11	2000	40-5-12 Laogiangge			
13	12	2100	1298 Vileparle (E)	490231	Bombay	Maharashtra
14	13	2200	12-98 Victoria Street	2901	Sydney	New South Wales
15	14	2300	198 Clementi North	540198	Singapore	
16	15	2400	8204 Arthur St		London	
17	16	2500	"Magdalen Centre	The Oxford	OX9 9ZB	Oxford
18	17	2600	9702 Chester Road	9629850293	Stretford	Manchester
19	17	2700	Schwanthalerstr. 7031	80925	Munich	Bavaria
20	19	2800	Rua Frei Caneca 1360	01307-002	Sao Paulo	Sao Paulo
21	20	2900	20 Rue des Corps-Saints	1730	Geneva	Geneve
22	21	3000	Murtenstrasse 921	3095	Bern	BE
23	22	3100	Pieter Breughelstraat 837	3029SK	Utrecht	Utrecht

Here in above picture if you click the Refresh Data from DB the data will be refresh and display the original data from DB, and if you click on Revert Changed Data, you will loose your changes just made. In this stage, if you click on Commit Changes button look the following screen shot.



Here in the above picture, you are looking an error message, and the color of A19 cell changed from green to red, that is because the ID column is the primary key and the uniqueness have been violated here. Now we returned the value of A19 cell in its original value, i.e. 18, and click on Commit Changes button, and now look the below screen shot.

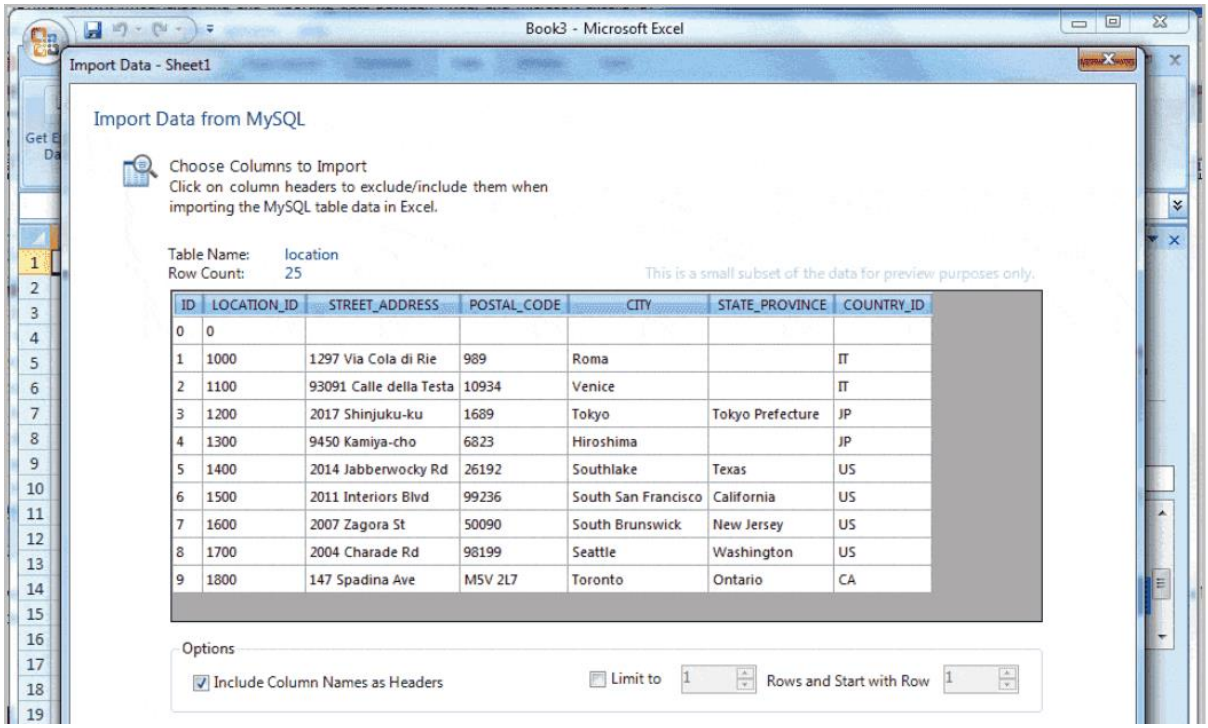


In the above picture shows the color of cell F16 have converted from green to blue, that means Commit Changes done successfully. Now you can close the Excel window saving or not but the database table has been updated. You can see it to do the step again. If you checked the Auto Commit check box, the effect immediately is seen in your sheet and data will be updated in MySQL database.

Import MySQL Data into Excel

Data can be imported from MySQL into a Microsoft Excel spreadsheet by using the Import MySQL Data option after selecting either a table, view, or procedure to import.

First of all, you do the first 6 steps describe above in "Edit MySQL Data in Excel" then select your table which you want to import. Here our table is location. So select the location table and then click "Import MySQL Data" and look the appeared screen shot here in the below.



Choosing columns to import

By default, all columns are selected and will be imported. Specific columns may be selected (or unselected) using the standard Microsoft Windows method of either Control + Mouse click to select the individual columns, or Shift + Mouse click to select a range of columns.

The background color white indicates, the column or columns have been selected and they are ready to be imported, on the other hand, the gray color indicates that the columns are not selected and the column will not be imported.

Right-clicking anywhere in the preview grid opens a context-menu with either a Select None or Select All option, depending on the current status.

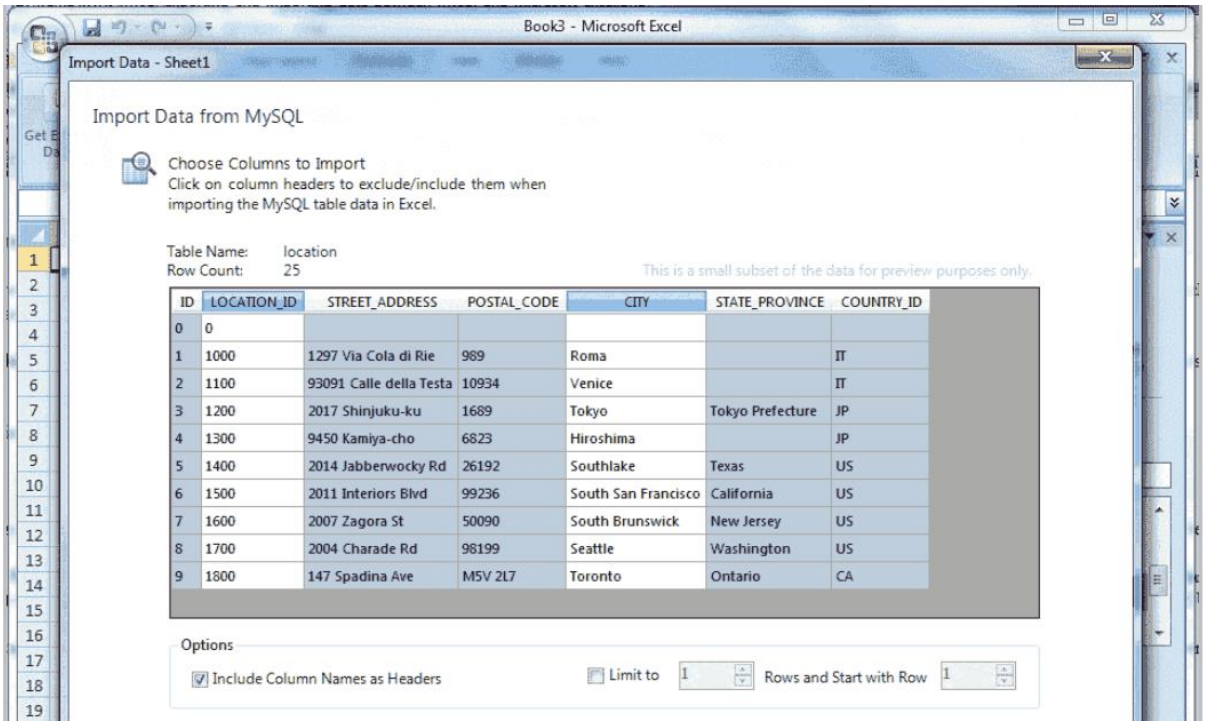
Importing Table

Include Column Names as Headers: By default, this option is enabled, and this treats the column names at the top of the Microsoft Excel spreadsheet as a "headers" row and will be inserted as a header.

Limit to and **Rows and Start with Row** : By default, this option is disabled if enabled, this limits the range of imported data. The Limit to option defaults to 1 and this limit can be changed by defines the number of rows to import. The Start with Row option defaults to 1, i.e. starting from the first row, and it

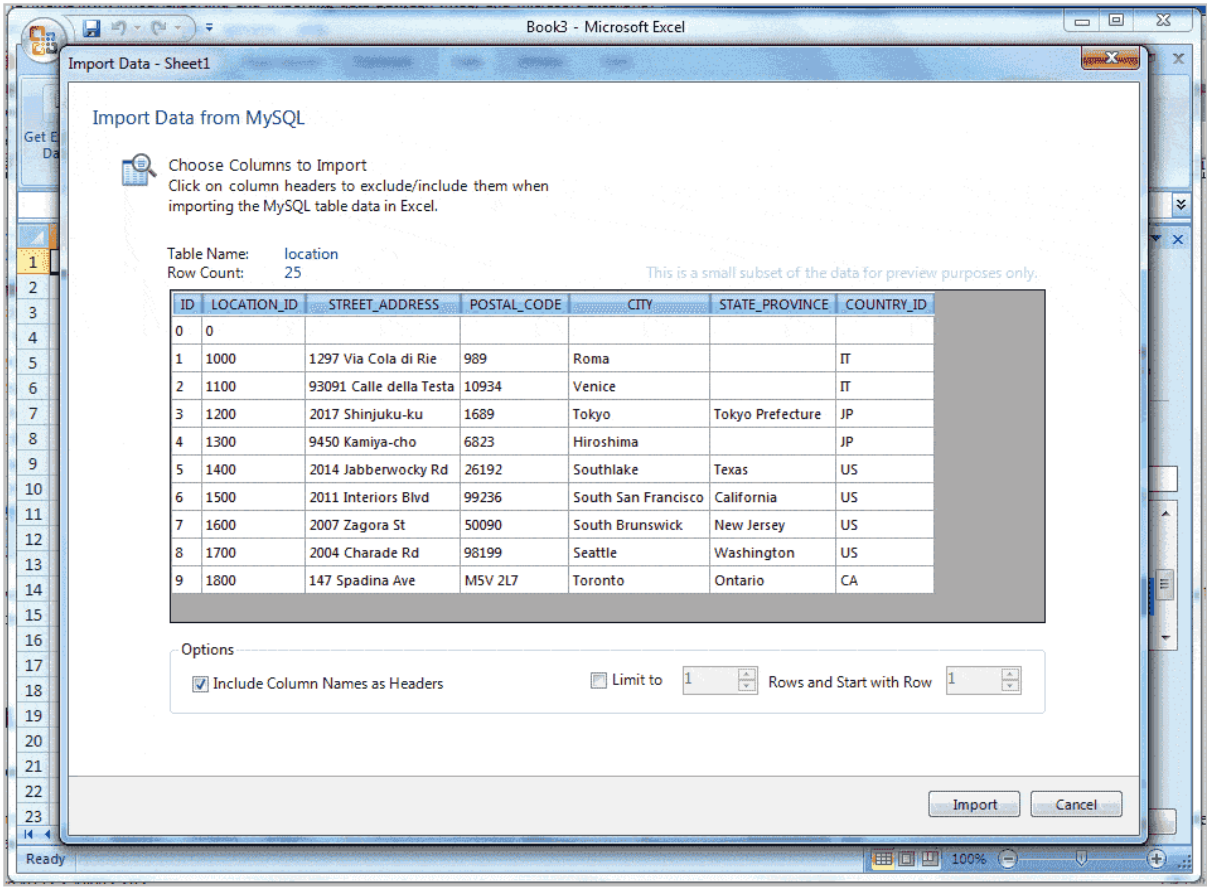
can be changed by defines the number from where the import begins. Each option has a maximum value of COUNT(rows) in the table.

Now assumed that we want to import the columns LOATION_ID and CITY. Click the mouse on LOCATION_ID column and then press and hold CTRL key and click on CITY column, and look the following screen shot.

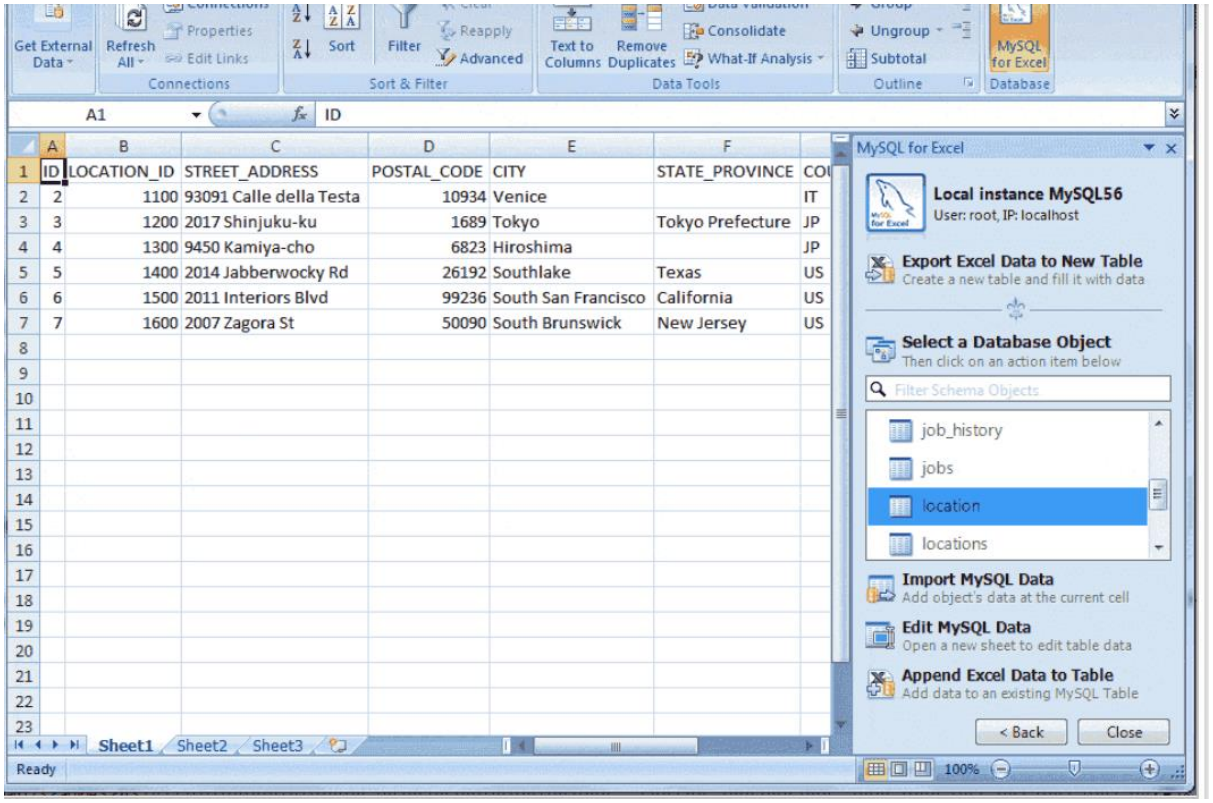


Now if we click the Import button all the rows for this two columns will be imported in Microsoft Excel Worksheet.

Assumes that, we want to import only 6 rows beginning from the 3rd row. Now look the following screen shot.



Here in the above picture shows all the columns have selected and the value of Limit to is 6, that means a number of 6 rows will be imported and the beginning of importing will start from the 3rd row because we have set the value of Start with Row is 3. Now click the Import button and look the following screen shot.


Get External Data
Refresh All
Properties
Edit Links
Connections
Sort
Filter
Reapply
Advanced
Sort & Filter
Text to Columns
Remove Duplicates
Consolidate
What-If Analysis
Data Tools
Ungroup
Subtotal
Outline
MySQL for Excel
Database
A1
ID
ID LOCATION_ID STREET_ADDRESS POSTAL_CODE CITY STATE_PROVINCE CO
2 2 1100 93091 Calle della Testa 10934 Venice IT
3 3 1200 2017 Shinjuku-ku 1689 Tokyo Tokyo Prefecture JP
4 4 1300 9450 Kamiya-cho 6823 Hiroshima JP
5 5 1400 2014 Jabberwocky Rd 26192 Southlake Texas US
6 6 1500 2011 Interiors Blvd 99236 South San Francisco California US
7 7 1600 2007 Zagora St 50090 South Brunswick New Jersey US
Sheet1
Sheet2
Sheet3
Ready
100%
MySQL for Excel
Local instance MySQL56
User: root, IP: localhost
Export Excel Data to New Table
Create a new table and fill it with data
Select a Database Object
Then click on an action item below
Filter Schema Objects
job_history
jobs
location
locations
Import MySQL Data
Add object's data at the current cell
Edit MySQL Data
Open a new sheet to edit table data
Append Excel Data to Table
Add data to an existing MySQL Table
< Back
Close