HELP PAGE

INTRODUCTION TO BQL

Enter BQLX<Go>, then press <Help>

Bloomberg

CONTENTS

CONTENTS

INTRODUCTION TO BQL

BQLX <GO> provides information on using the Bloomberg Query Language (BQL) to retrieve data and perform analysis in Microsoft® Excel.

BQL is a new API based on normalized, curated, point-in-time data. Using BQL, you can perform custom calculations directly in the Bloomberg Cloud. Analyses that previously required you to download thousands of data points and perform complicated Microsoft® Excel manipulation now require only one formula. You just specify the data you want and the calculation you want to perform—including, but not limited to, arithmetic and statistical operations, filtering, grouping, scoring, and dates-based analysis—so you can synthesize large amounts of data and extract the exact information you need.

In a single BQL query, you construct a universe of tickers (such as a list of peers) and define the data you want (such as earnings per share). You can apply analysis and calculations on the data directly on the server side, such as calculating an average or multiplying two values. You can also add optional parameters that fit the data to your model, such as the time period (if you want historical data) and currency (if you want to convert it).

There are two different formulas you can use in Excel to perform a BQL query:

- **BQL**(): Uses the standard Bloomberg formula structure (i.e., "Ticker", "Field", "[optional parameters]"). BQL() is the simplified formula recommended for data retrieval. For more: *BQL() Formulas*.
- **BQL.Query**(): Uses a raw string of BQL code, containing *get*() and *for*() clauses. BQL.Query() exposes the full capabilities of BQL in terms of screening, aggregation, and server-side analysis. For more: *BQL.Query() Formulas*.

Date printed: 09/20/2019

GETTING STARTED

BQL for Equities: *Resources*
BQL for Funds: *Resources*
BQL for Fixed Income: *Resources*
BQL for Economics: *Resources*
BQL for Portfolios: *Resources*

EXCEL FORMULA REFERENCE

BQL() FORMULAS

BQL() SYNTAX

The syntax for BQL() formulas follows the standard Bloomberg API syntax: ticker(s) (aka the *Universe*), data item(s) (aka the *Expression*), optional parameters, and optional local variables (aka *Local Variables*).

Each input can either be a literal value or a cell reference. For more information on cell referencing: *DAPI Help Page > Cell Referencing*

Syntax: =BQL("Universe", "Expression", "Optional Parameters", "Optional Local Variables")

- **Universe** can contain one or more comma-separated tickers or the *members*() function and an index ticker, which lets you create a universe containing all the members of the specified index.
- **Expression** can contain one or more comma-separated data items and, if desired, optional parameters that control the format or calculation of the data item.
- **Optional Parameters** that apply to all specified data items are each surrounded by their own set of quotation marks and separated by commas.
- **Optional Local Variables** can contain one or more variable name and value pairs. Each name and value pair is surrounded by its own set of quotation marks and separated by commas. Each name begins with #.

For example:

- To retrieve the current last prices for IBM and Microsoft:

  =BQL("IBM US Equity, MSFT US Equity", "px_last")
- To retrieve a historical time series of last prices for IBM over the last month, converted to EUR:

  =BQL("IBM US Equity", "px_last", "dates=range(-1M, 0D)", "currency=EUR")

  or

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D), currency=EUR)")
- To retrieve a historical time series of last prices and low prices for IBM over the last month, converted to EUR:

  =BQL("IBM US Equity", "px_last, px_low", "dates=range(-1M, 0D)", "currency=EUR")
- To retrieve a historical time series of last prices and low prices for IBM over the last month using #lastPrice as a variable for last price and #lowPrice as a variable for low price:

  =BQL("IBM US Equity", "#lastPrice, #lowPrice", "dates=range(-1M, 0D)", "#lastPrice=px_last", "#lowPrice=px_low")

Note: BQL() formulas are case-insensitive.

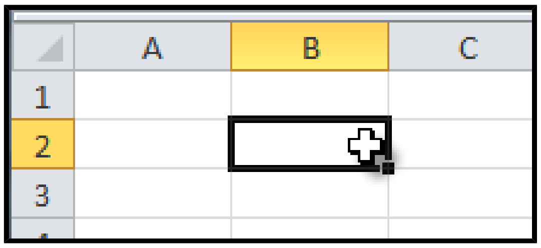
|Hint| You can use the *BQL Builder* tool to easily start creating BQL() formulas. For more: *Using the BQL Builder*.

USING THE BQL BUILDER

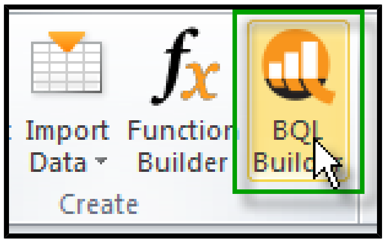
You can quickly build a BQL() formula in your spreadsheet with the *BQL Builder* tool.

Steps:

1. On your spreadsheet, select the cell where you want to input your BQL() formula.

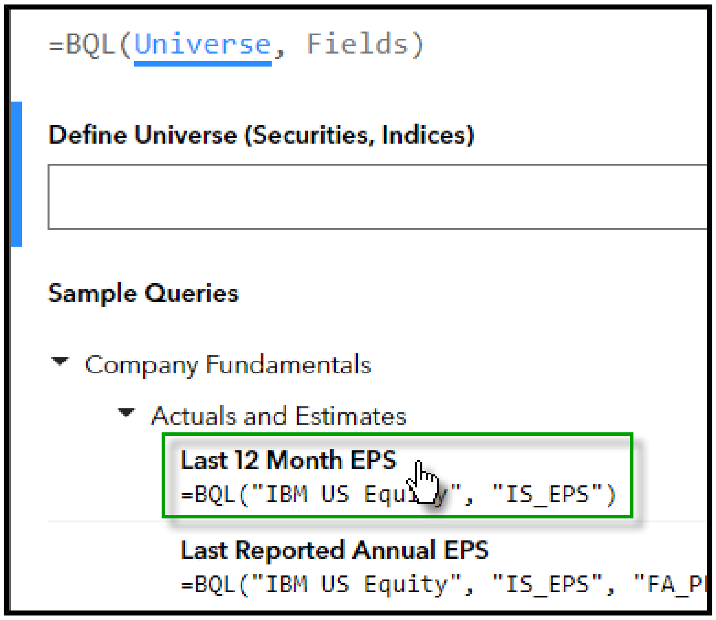


2. From the Bloomberg ribbon, click **BQL Builder**.



Note: The above image reflects the *Standard* format of the Bloomberg ribbon.

The BQL Builder window appears.
|Hint| *If you want to load a sample query on which to base your own query, under Sample Queries, you can expand a category and select the query* (e.g., *Last 12 Month EPS*). *Then, either continue following the steps to edit the example or proceed to Step 8*.



```
=BQL(Universe, Fields)
```

Define Universe (Securities, Indices)

Sample Queries

Company Fundamentals

Actuals and Estimates

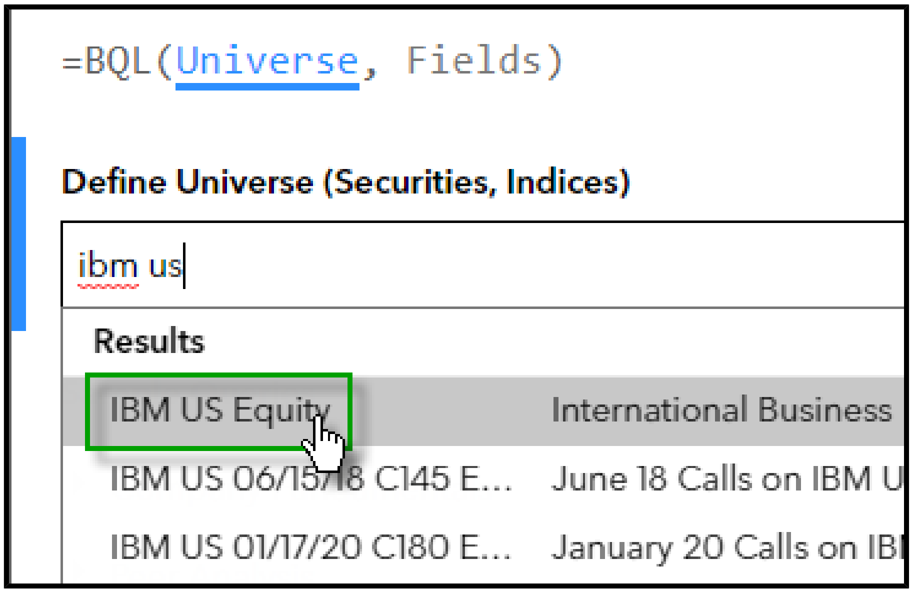
Last 12 Month EPS

```
=BQL("IBM US Equity", "IS_EPS")
```

Last Reported Annual EPS

```
=BQL("IBM US Equity", "IS_EPS", "FA_P
```

3. In the *Define Universe* (*Securities*, *Indices*) field, start typing the ticker or company name of the security from which you want to pull data, then select it from the list that appears (e.g., *IBM US Equity*).



```
=BQL(Universe, Fields)
```

Define Universe (Securities, Indices)

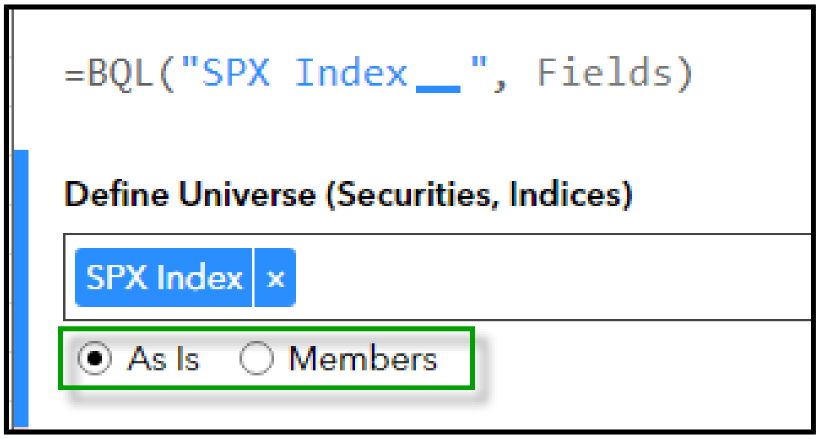
ibm us

Results

IBM US Equity	International Business
IBM US 06/15/18 C145 E...	June 18 Calls on IBM U
IBM US 01/17/20 C180 E...	January 20 Calls on IBI

The ticker appears in blue in the field and in the BQL() formula at the top of the window. TheAdd Fields field appears. If you selected an index (e.g., SPX Index), the As Is and Members options appear under the Define Universes field.

4. If you selected an index in Step 4, select **As Is** to retrieve data for the index itself or select **Members** to retrieve data for each member of the index.

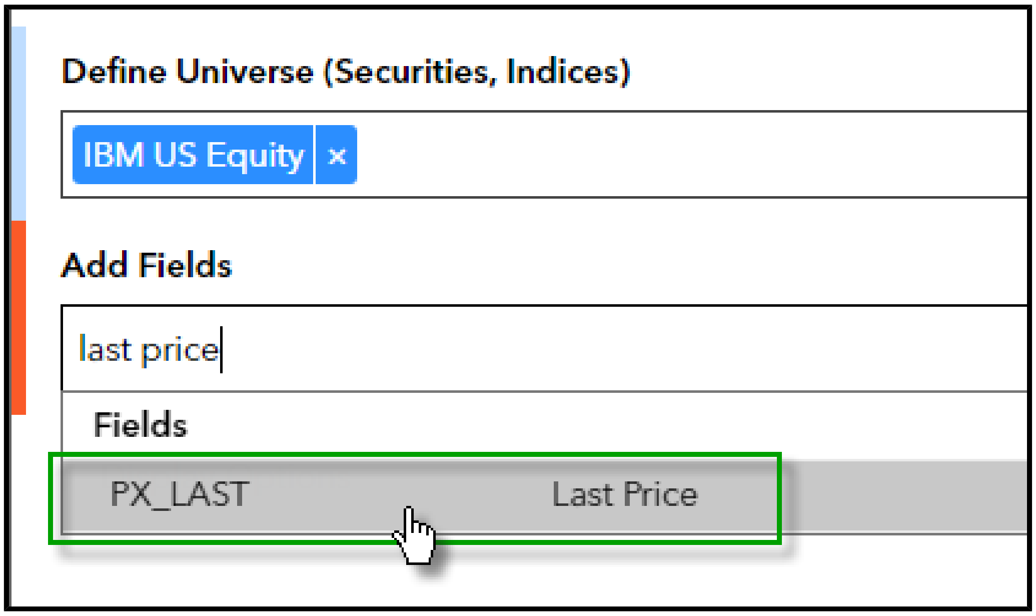


```
=BQL("SPX Index ", Fields)
```

Selected: [● As Is] ○ Members

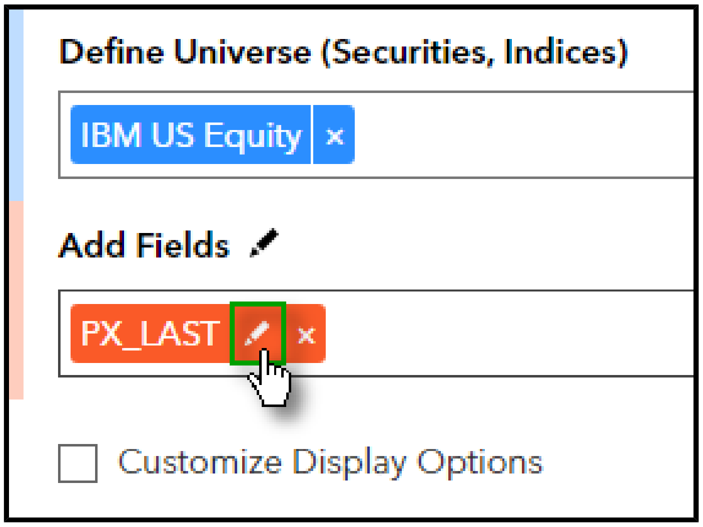
If you selected Members, the members() function is added to the BQL formula at the top of the window.

5. In the *Add Fields* field, start typing the data you want in plain English or the specific data item mnemonic you want to use, then select an option from the list that appears (e.g., *PX_LAST*).



The data item appears in orange in the field and in the BQL() formula at the top of the window.

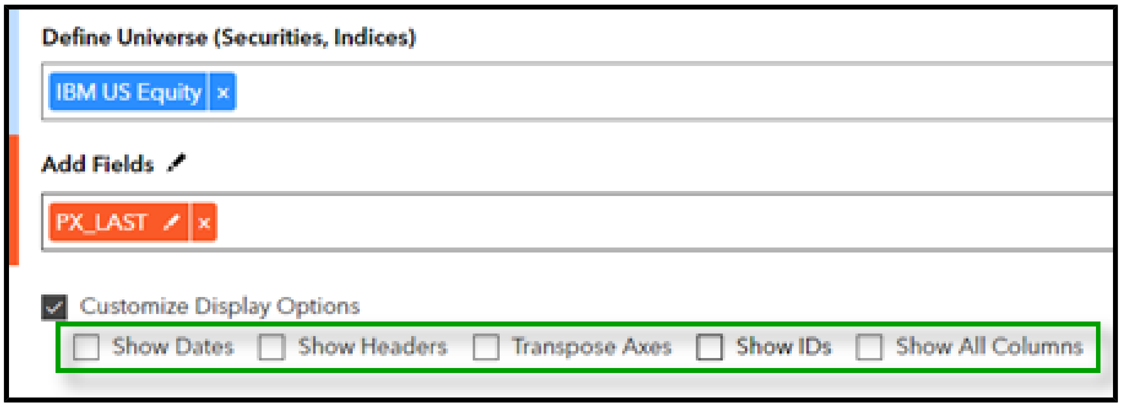
6. Repeat Steps 4-6 for as many securities, indices, and data items as you want to include in your formula.
7. If you want to set the values for parameters that control the format and calculation of the data (e.g., currency, fiscal period, and/or dates for historical data), click the pencil icon next to the data item to which you want to apply parameters.


Define Universe (Securities, Indices)
IBM US Equity
Add Fields
PX_LAST
Customize Display Options

For more on adding parameters and parameter definitions: *Updating BQL Parameters*.

The parameters you added appear in the BQL() formula at the top of the window.

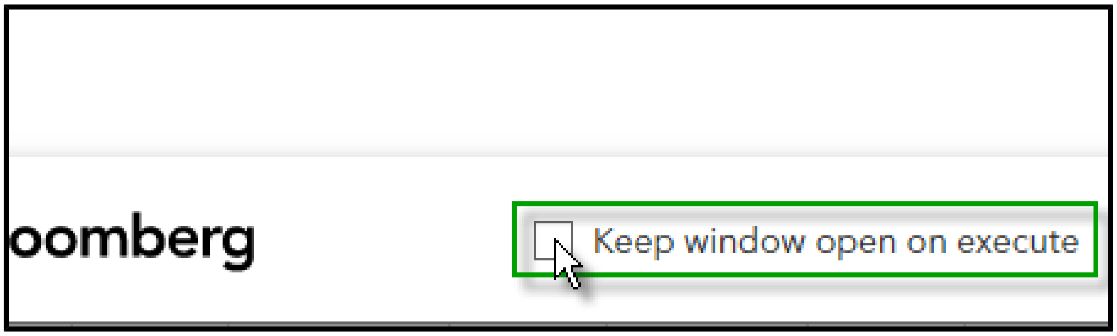
8. If you want to control the layout of the data in your spreadsheet, select **Customize Display Options**, then select the options you want to apply:



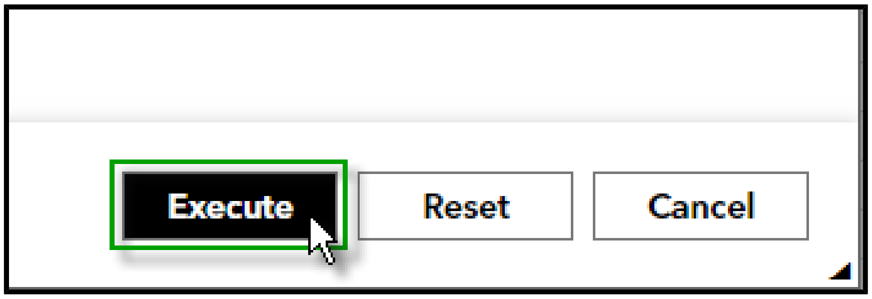
- **Show Dates**: Select this option to display the date(s) associated with each data point you retrieve.
- **Show Headers**: Select this option to display the column headers for the data you retrieve.
- **Transpose Axes**: If you are retrieving data that appears in multiple cells, select this option to change the layout of the data from vertical to horizontal.
- **Show IDs**: Select this option to display the IDs of the universe members in your formula.
- **Show All Columns**: Select this option to display all columns associated with the data items in your formula.

The selected options appear in the BQL() formula at the top of the window.

9. If you want to keep the *BQL Builder* open after running the formula you have created, at the bottom of the window, select **Keep Window Open on Execute**.



10. Click the **Execute** button.



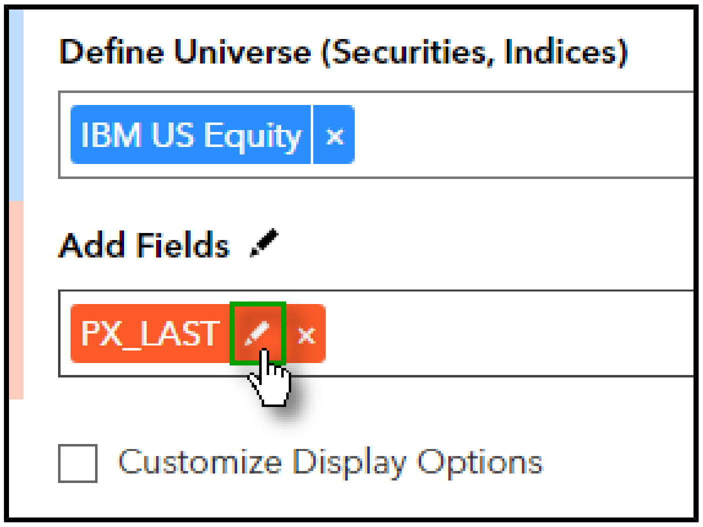
The BQL() formula is added to your spreadsheet in the selected cell and executes. The corresponding data is pulled from the Bloomberg data model and appears in your spreadsheet.

UPDATING BQL PARAMETERS

After adding one or more data items to your BQL() formula, you can add optional parameters that control the format and calculation of the data. For example, you can specify a date range if you want historical data or a currency if you want to convert a monetary value. You can set parameter values to specific data items or to all data items in your formula.

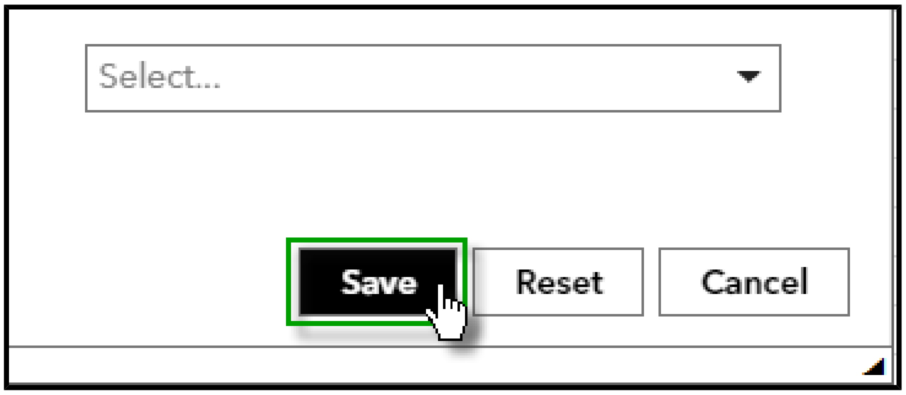
Steps:

1. Next to the data item, click the pencil icon.


Define Universe (Securities, Indices)
IBM US Equity ×
Add Fields
PX_LAST ×
Customize Display Options

The parameter side panel appears. The parameters in the panel vary based on the data item(s) you added to your formula. If there are default values for any parameter, the default value is selected.

2. If you want to set parameter values for all data items in your formula at once, from the *Edit Overrides For* drop-down menu, select (**Global Parameters**).
   The list of parameters updates based on your selection.
3. Enter or select values for any parameter you want to change.
   For definitions of some parameter and their possible values: *Fundamental Parameters*. For information on other parameters, see the documents in *Resources*, *Resources*, *Resources*, and *Resources*.
4. At the bottom of the panel, click the **Save** button.



The side panel closes. The BQL() formula at the top of the window updates to include the parameters whose default values you changed.

CURRENT/HISTORICAL DATA

Using a BQL() formula, you can easily download a current data point or a historical time series of data for a security.

- **Current**: To download the current last price for a security, you just need the ticker and the data item:

=BQL("IBM US Equity", "px_last")

- **Historical Data Point**: To download a single historical price for a security, you need the ticker, data item, and the *As Of Date* parameter. You can enter a relative date (e.g., -2D) or an absolute date (in the YYYY-MM-DD format):

  =BQL("IBM US Equity", "px_last(dates=-1Y)")

  =BQL("IBM US Equity", "px_last(dates=2014-01-07)")
- **Historical Data Series**: To download a historical time series of price for a security, use the *Dates* parameter with the *range*() function to enter the absolute or relative date range:

  =BQL("IBM US Equity", "px_last(dates=range(2014-01-07, 2014-01-15))")

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D))")

  Note: The relative date "0D" indicates today.

  Additional parameters let you control how historical data is downloaded. For example, *Periodicity* (per) and *Fill When Data is Missing* (fill) let you choose the sampling frequency of the data and fill in values for non-trading days:

  =BQL("IBM US Equity", "px_last(dates=range(2014-01-01, 2016-01-15), per=M, fill=PREV)")

INDEX MEMBERS

members() lets you create a universe in your BQL() formula by decomposing one or more indices into a list of their constituents.

If you specify a index ticker in your universe without using *members*(), you retrieve data for the index itself. For example, to retrieve the last price of the Euro Stoxx 50 Index:

=BQL("SX5E Index", "px_last")

However, if you want to retrieve the last price of each index member, just add *members*() to your universe:

=BQL("members(['SX5E Index'])", "px_last")

BQL.QUERY() FORMULAS

BQL.QUERY() SYNTAX

BQL.Query() formulas allow you to use raw BQL strings to retrieve data from Bloomberg.

The syntax for BQL.Query() formulas is unlike other Bloomberg API formulas. BQL.Query uses a *get*(), a *for*(), an optional *with*(), and an optional *let*() clause instead of the standard Bloomberg API "ticker", "field", "optional parameter" syntax.

Syntax: =BQL.Query("<let()> get() for() <with()>")

The *get*() clause specifies the type of data you want to download (aka the *Expression*) and the *for*() clause specifies the securities from which you want to download the data (aka the *Universe*). The *with*() clause lets you specify optional parameters that apply to all data items in the query (aka *Global Parameters*). The *let*() clause assigns variable names to expression values that you can reuse throughout your formula (aka *Local Variables*).

- **let**() (Optional) (*Local Variables*) can contain one or more variable name and value pairs. Each name and value pair begins with # and ends with a semi-colon (e.g., *#lastPrice=px_last*(*dates=range*(*-2M*, *0D*));). For more: *let*() *Clause*.
- **get**() (the *Expression*) can contain one or more comma-separated data items. Optional parameters that control the format or calculation of a specific data item are enclosed within parentheses adjacent to the data item and separated by commas (e.g., *px_last*(*dates=-1M*)). For more: *get*() *Clause*.
- **for**() (the *Universe*) can contain one or more comma-separated tickers within square brackets, each surrounded by a set of single quotation marks (e.g., *['IBM US Equity*', '*AAPL US Equity*']). It can also contain the *members*() function and an index ticker, which creates a universe that includes all members of the specified index (e.g., *members*([*'SPX Index*']). For more: *for*() *Clause*.
- **with**() (Optional) (*Global Parameters*) can contain parameters that control the format or calculation of all data items in your BQL.Query formula and are comma-separated (e.g., *dates=-1M*, *per=W*). For more: *with*() *Clause*.

For example:

- To retrieve the current last prices for IBM and Microsoft:

  =BQL.Query("get(px_last) for(['IBM US Equity', 'MSFT US Equity'])")
- To retrieve a historical time series of last prices for IBM over the last month, converted to EUR:

  =BQL.Query("get(px_last(dates=range(-1M, 0D), currency=EUR)) for(['IBM US Equity'])")
- To retrieve a historical time series of last prices and low prices for IBM over the last month, converted to EUR:

  =BQL.Query("get(px_last, px_low) for(['IBM US Equity']) with(dates=range(-1M, 0D), currency=EUR)")
- To retrieve a historical time series of last prices and low prices for IBM over the last month using #lastPrice as a variable for last price and #lowPrice as a variable for low price:

  =BQL.Query("let(#lastPrice=px_last; #lowPrice=px_low;) get(#lastPrice, #lowPrice) for(['IBM US Equity']) with(dates=range(-1M, 0D))")

Note: BQL.Query() formulas are case-insensitive.

GET() CLAUSE

The *get*() clause in a BQL.Query() formula lets you specify the type of data you want to download.

The minimum you can specify within the *get*() clause is a BQL data item. For example, px_last (*Last Price*), px_volume (*Volume*), and gics_sector_name (*GICS Sector Name*).

Within the clause, you can also specify different combinations of optional parameters for each data item to obtain customized data.

For example:

- **Data Items**: To download the last prices for IBM and Microsoft:

  =BQL.Query("**get(px_last**) for(['IBM US Equity', 'MSFT US Equity'])")
- **Parameters**: To download a time series of daily prices for IBM between January 7, 2014 and January 15, 2014:

  =BQL.Query(" **get**(**px_last**(**dates=range**(**2014-01-07**, **2014-01-15**))) for(['IBM US Equity'])")

Note: *get*() clauses can contain individual cell or cell range references, literal references (as in the examples above), or a combination of both. For more on cell referencing: *DAPI Help Page > Cell Referencing*.

LET() **CLAUSE**

The *let*() clause in a BQL.Query() formula lets you assign variable names to expression values, which you can use repeatedly throughout your formula and update all instances at once.

You can specify one or more variable and value pairs. Each variable and value pair must end with a semi-colon and each variable must begin with #.

For example:

- **Single Variable**:

  =BQL.Query("**let(#marketCap=eqy_sh_out*px_last;**) get(#marketCap/groupAvg(#marketCap, gics_sector_name)) for(members(['SPX Index']))")

- **Multiple Variables**:

  =BQL.Query("**let(#name=name; #price=px_last**().**value; #zChange= abs**(**groupZScore**(**day_to_day_total_return**(**fill=PREV**)).**value);**) get(dropNA(matches(#zChange, #zChange >= 1.5), remove_id=TRUE)) for(members(['SPX Index']))")

Note: *let*() clauses can contain individual cell or cell range references, literal references (as in the examples above), or a combination of both. For more on cell referencing local variables: *Cell Referencing Local Variables*.

FOR() **CLAUSE**

The *for*() clause in a BQL.Query() formula lets you specify the universe of tickers you want to analyze.

Each ticker you specify must be enclosed in single quotation marks, e.g., 'IBM US Equity'. The entire list of one or more tickers must be enclosed within a set of square brackets, e.g., ['IBM US Equity', 'MSFT US Equity', 'AAPL US Equity'].

For example:

- **Single Security**:

  =BQL.Query("get(px_last) **for**(**['IBM US Equity'**])")

- **Multiple Securities**:

  =BQL.Query("get(px_last) **for**(**['IBM US Equity'**, '**MSFT US Equity**', '**AAPL US Equity'**])")

- **Members of an Index**:

  =BQL.Query("get(px_last) **for**(**members**(**['SPX Index'**]))")

  For more on *members*(): *Index Members*.

Note: *for*() clauses can contain individual cell or cell range references, literal references (as in the examples above), or a combination of both. For more on cell referencing: *DAPI Help Page > Cell Referencing*.

WITH() **CLAUSE**

The *with*() clause in a BQL.Query() formula lets you specify global optional parameters that control the format or calculation of all data items in your formula at once.

You can specify one or more parameters. Each parameter and value set must be comma-separated.

For example:

- **Global Parameters**:

  =BQL.Query("get(px_last, px_low) for(['IBM US Equity']) **with**(**currency=EUR**, **dates=range**(-**1M**, **0D**))")

- **Global Parameters and Data Item-Specific Parameters**:

  =BQL.Query("get(**px_last**(**ca_adj=RAW**), px_low) for(['IBM US Equity']) **with**(**currency=EUR**, **dates=range**(-**1M**, **0D**))")

Note: *with*() clauses can contain individual cell or cell range references, literal references (as in the examples above), or a combination of both. For more on cell referencing: *DAPI Help Page > Cell Referencing*.

CURRENT/HISTORICAL DATA

Using a BQL.Query() formula, you can easily download a current data point or a historical time series of data for a security.

- **Current**: To download the current last price for a security, you just need the ticker and the data item:

  =BQL.Query("get(px_last) for(['IBM US Equity'])")

- **Historical Data Point**: To download a single historical price for a security, you need the ticker, data item, and the *As Of Date* parameter. You can enter a relative date (e.g., -2D) or an absolute date (in the YYYY-MM-DD format):

  =BQL.Query("get(px_last(dates=-1Y)) for(['IBM US Equity'])")

  =BQL.Query("get(px_last(dates=2014-01-07)) for(['IBM US Equity'])")

- **Historical Data Series**: To download a historical time series of price for a security, use the *Dates* parameter with the *range*() function to enter the absolute or relative date range:

  =BQL.Query("get(px_last(dates=range(2014-01-07, 2014-01-15))) for(['IBM US Equity'])")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for(['IBM US Equity'])")

  Note: The relative date "0D" indicates today.

  Additional parameters let you control how historical data is downloaded. For example, *Periodicity* (per) and *Fill When Data is Missing* (fill) let you choose the sampling frequency of the data and fill in values for non-trading days:

  =BQL.Query("get(px_last(dates=range(2014-01-01, 2016-01-15), per=M, fill=PREV)) for(['IBM US Equity'])")

INDEX MEMBERS

members() lets you create a universe in your BQL.Query() formula by decomposing one or more indices into a list of their constituents.

If you specify a index ticker in your universe without using *members*(), you retrieve data for the index itself. For example, to retrieve the last price of the Euro Stoxx 50 Index:

=BQL.Query("get(px_last) for(['SX5E Index'])")

However, if you want to retrieve the last price of each index member, just add *members*() to your universe:

=BQL.Query("get(px_last) for(members(['SX5E Index']))")

CELL REFERENCING

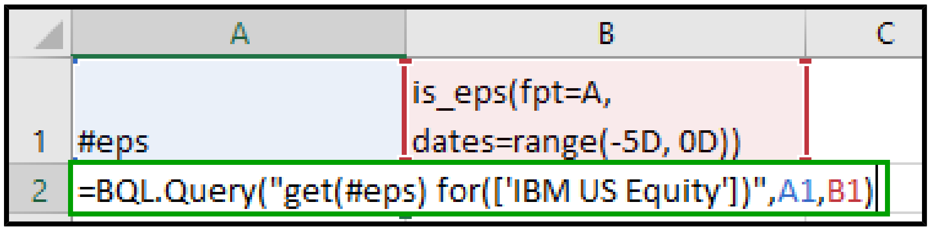
CELL REFERENCING LOCAL VARIABLES

When creating a BQL() or BQL.Query() formula that includes multiple instances of the same expression, you can save time by cell referencing a local variable. Referencing a local variable allows you to use an expression several times throughout your formula and update all the instances at once by changing the referenced cell value. You can include one or more local variables in your formula by entering a cell reference or a cell reference range.

To reference one or more local variables, type the variable name(s) and value(s) in adjacent cells and then reference the cells or cell range at the end of your formula.
|Hint| To add a cell reference to a formula, you can simply click the corresponding cell while typing the formula.

For example:

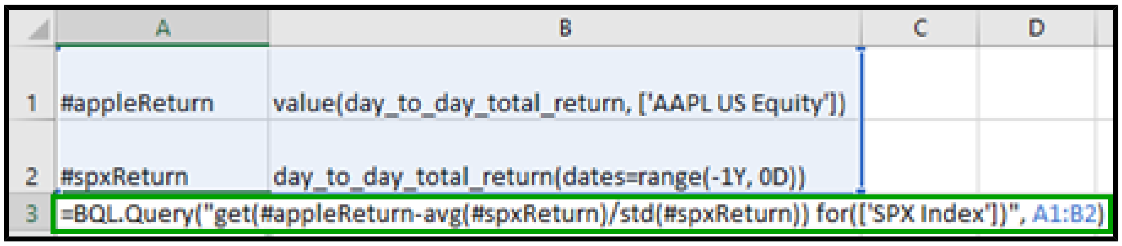
- Referencing individual cells in BQL.Query() formulas:
  =BQL.Query("get(#eps) for(['IBM US Equity'])", A1,B1)



	A	B	C
1	#eps	is_eps(fpt=A, dates=range(-5D, 0D))	
2	=BQL.Query("get(#eps) for(['IBM US Equity'])",A1,B1)		

  Cell A1 contains the name of the variable (#eps) and cell B1 contains the expression (is_eps, fpt=A, dates=range(-5D, 0D))).

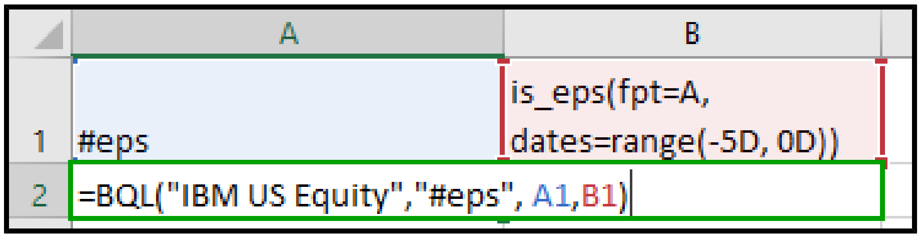
- Referencing a range of cells in BQL.Query() formulas:
  =BQL.Query("get(#appleReturn-avg(#spxReturn)/std(#spxReturn)) for(['SPX Index'])", A1:B2)



	A	B	C	D
1	#appleReturn	value(day_to_day_total_return, ['AAPL US Equity'])		
2	#spxReturn	day_to_day_total_return(dates=range(-1Y, 0D))		
3	=BQL.Query("get(#appleReturn-avg(#spxReturn)/std(#spxReturn)) for(['SPX Index'])", A1:B2)			

  Cell A1 contains the name of the first variable (#appleReturn), cell B1 contains the first expression (value(day_to_day_total_return, ['AAPL US Equity'])), cell A2 contains the name of the second variable (#spxReturn), and cell B2 contains the second expression (day_to_day_total_return(dates=range(-1Y, 0D))).

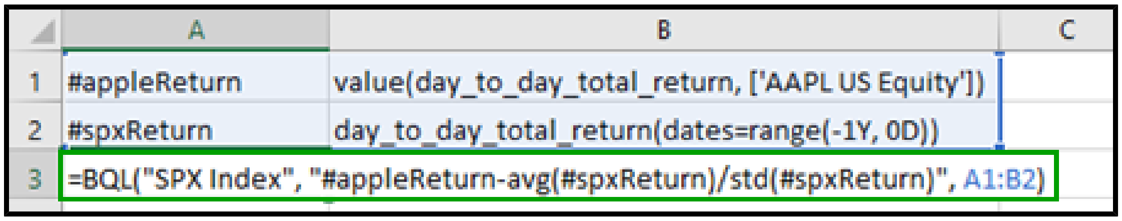
- Referencing individual cells in BQL() formulas:
  =BQL("IBM US Equity", "#eps", A1,B1)



	A	B
1	#eps	is_eps(fpt=A, dates=range(-5D, 0D))
2	=BQL("IBM US Equity","#eps", A1,B1)	

Cell A1 contains the name of the variable (#eps) and cell B1 contains the expression (is_eps(fpt=A, dates=range(-5D, 0D))).

- Referencing a range of cells in BQL() formulas:
  =BQL("SPX Index", "#appleReturn-avg(#spxReturn)/std(#spxReturn)", A1:B2)



	A	B	C
1	#appleReturn	value(day_to_day_total_return, ['AAPL US Equity'])	
2	#spxReturn	day_to_day_total_return(dates=range(-1Y, 0D))	
3	=BQL("SPX Index", "#appleReturn-avg(#spxReturn)/std(#spxReturn)", A1:B2)		

Cell A1 contains the name of the first variable (#appleReturn), cell B1 contains the first expression (value(day_to_day_total_return, ['AAPL US Equity'])), cell A2 contains the name of the second variable (#spxReturn), and cell B2 contains the second expression (day_to_day_total_return(dates=range(-1Y, 0D))).

For more: *Spreadsheet Tutorial: Cell Referencing in =BQL(), =BQL.Query()* and *DAPI Help Page > Cell Referencing*.

DISPLAY PARAMETERS

Display parameters let you change how the data returned by your BQL() or BQL.Query() formula is arranged in your spreadsheet:

Show Dates	*Show Query*	*Group By Fields*
Show Headers	*Transpose Axes*	
Show IDs	*Sort Dates*	

SHOW DATES

For BQL() and BQL.Query() formulas in Excel, the *Show Dates* (showdates) parameter lets you choose whether or not to return the dates associated with your data.

Parameter Mnemonic: showdates

Valid Values:

Value	Description
False or **N**	Dates are not shown. (**Default for one data point**)
True or **Y**	Dates are shown. (**Default for multiple data points**)

Example Formulas:

- Retrieve the last price of IBM and hide its as of date:

  =BQL("IBM US Equity", "px_last")

  =BQL.Query("get(px_last) for(['IBM US Equity'])")
- Retrieve the last price of IBM and show its as of date:

  =BQL("IBM US Equity", "px_last", "showdates=True")

  =BQL.Query("get(px_last) for(['IBM US Equity'])", "showdates=True")

SHOW HEADERS

For BQL() and BQL.Query() formulas in Excel, the *Show Headers* (showheaders) parameter lets you choose whether or not to return column headers associated with your data.

Parameter Mnemonic: showheaders

Valid Values:

Value	Description
False or **N**	Column headers are not shown.
True or **Y**	Column headers are shown. (**Default**)

Example Formulas:

- Retrieve the daily last price of IBM over the last month and show the column headers for the downloaded data:

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D))")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for(['IBM US Equity'])")
- Retrieve the daily last price of IBM over the last month and hide the column headers for the downloaded data:

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D))", "showheaders=False")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for (['IBM US Equity'])", "showheaders=False")

SHOW IDS

For BQL() and BQL.Query() formulas in Excel with more than one ID (e.g., ticker) in the universe, the *Show IDs* (showids) parameter allows you to show or hide the IDs of your universe members in the output of your formula.

Parameter Mnemonic: showids

Valid Values:

Value	Description
False or **N**	IDs are not shown.
True or **Y**	IDs are shown. (**Default**)

Example Formulas:

- Retrieve the daily last price of SPX Index members over the last month and show the security IDs in the output:

  =BQL("members('SPX Index')", "px_last(dates=range(-1M, 0D))")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for(members(['SPX Index']))")
- Retrieve the daily last price of IBM over the last month and hide the security IDs in the output:

  =BQL("members('SPX Index')", "px_last(dates=range(-1M, 0D))", "showids=True")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for(members(['SPX Index']))", "showids=False")

SHOW QUERY

For BQL() and BQL.Query() formulas in Excel, the *Show Query* (showquery) parameter allows you to show your formula as a raw BQL query string instead of downloading data to your spreadsheet.

Parameter Mnemonic: showquery

Valid Values:

Value	Description
False or **N**	Data is retrieved based on your BQL formula. (**Default**)
True or **Y**	Your BQL formula is displayed as a raw BQL query string. For example: get(px_last(dates=range(-1M, 0D))) for(['IBM US Equity'])

Example Formulas:

- Retrieve the daily last price of IBM over the last month:

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D))")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for(['IBM US Equity'])")
- Show the raw BQL query string that retrieves the daily last price of IBM over the last month:

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D))", "showquery=True")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for(['IBM US Equity'])", "showquery=True")

TRANSPOSE AXES

For BQL() and BQL.Query() formulas in Excel, the *Transpose Axes* (transpose) parameter lets you choose whether to return data tables vertically or horizontally.

Parameter Mnemonic: transpose

Valid Values:

Value	Description
False or **N**	Return data in a vertical table. (**Default**)
True or **Y**	Return data in a horizontal table.

Example Formulas:

- Retrieve the daily last price of IBM over the last month and arrange the data vertically:

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D))")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for(['IBM US Equity'])")
- Retrieve the daily last price of IBM over the last month and arrange the data horizontally:

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D))", "transpose=True")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for(['IBM US Equity'])", "transpose=True")

SORT DATES

For BQL() and BQL.Query() formulas in Excel, the *Sort Dates* (xlsort) parameter lets you choose whether to display dates in ascending or descending order when you are retrieving a time series of values.

Parameter Mnemonic: xlsort

Valid Values:

Value	Description
ASC	The list of values starts with the least recent date first.(**Default**)
DESC	The list of values starts with the most recent date first.

Example Formulas:

- Retrieve the last price of IBM over the last month, starting with the least recent date:

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D))")

  =BQL.Query("get(px_last(dates=range(-1M, 0D))) for('IBM US Equity')")
- Retrieve the last price of IBM, starting with the most recent date:

  =BQL("IBM US Equity", "px_last(dates=range(-1M, 0D))", "xlsort=DESC")

=BQL.Query("get(px_last(dates=range(-1M, 0D))) for('IBM US Equity')", "xlsort=DESC")

GROUP BY FIELDS

For BQL() and BQL.Query() formulas in Excel, the *Group By Fields* (groupbyfields) parameter lets you choose whether results are grouped by identifier first and then by fields, or by fields first and then by identifier.

Parameter Mnemonic: groupbyfields

Valid Values:

Value	Description
False or **N**	Results are arranged by identifier first. (**Default**)
True or **Y**	Results are arranged by field first.

Example Formulas:

- Retrieve the seven-day bid and ask prices of IBM and VodaFone, arranging results by ticker then field:

  =BQL("IBM US Equity, VOD LN Equity", "px_bid, px_ask", "dates=range(-7D, -0D)")

  =BQL.Query("get(px_bid,px_last) for(['IBM US Equity', 'VOD LN Equity']) with (dates=range(-7D,- 0D))")
- Retrieve the seven-day bid and ask prices of IBM and VodaFone, arranging results by field then ticker:

  =BQL("IBM US Equity, VOD LN Equity", "px_bid, px_ask", "dates=range(-7D, -0D)", "groupbyfields=True")

  =BQL.Query("get(px_bid,px_last) for(['IBM US Equity', 'VOD LN Equity']) with (dates=range(-7D, -0D))", "groupbyfields=true")

HELPER FORMULAS

Helper formulas in Microsoft® Excel allow you to easily convert inputs (e.g., dates or securities) into formats that can be ingested in a BQL() or BQL.Query() formula.

BQL.DATE()

The BQL.Date() formula allows you to convert Microsoft® Excel dates to the correct BQL format, so the output can be referenced in a BQL() or BQL.Query() formula. The only required value is an Excel date or the BToday() Bloomberg API formula.

For example, the following formula transforms the current date into the BQL format (YYYY-MM-DD):

=BQL.Date(BToday())

Note: The value within a BQL.Date() formula can be a cell references or values (as in the example above). For more on cell referencing: *DAPI Help Page > Cell Referencing*.

BQL.LIST()

The BQL.List() formula allows you to convert one or more securities into the correct BQL list format, so you can reference the list in a BQL.Query() formula. Note that a BQL.List() output cannot be referenced in a BQL() formula, which automatically creates a list from the tickers it contains.

For example, the following formula creates a list that contains IBM, Microsoft, and Apple:

=BQL.List("IBM US Equity, MSFT US Equity, AAPL US Equity")

Note: The value within a BQL.List() formula can be a cell reference or literal values. For more on cell referencing: *DAPI Help Page > Cell Referencing*.

BQL FOR EQUITIES

BQL is an efficient and flexible method for retrieving a wide range of equity data from the Bloomberg database, including fundamental, estimate, and pricing data.

KEY BENEFITS

Key benefits of BQL for Equities are:

- Point-in-time fundamental data that can take into account release and revision dates
- Easy retrieval of fundamentals across actuals and estimates
- Screening of an equities universe for stocks with certain characteristics
- Cross-asset signals for equities that reference bond yields, vol skew, and options volume
- Data sets spanning fundamentals, estimates, consensus, revisions, technical analysis, and environmental, social, and governmental (ESG) data

RESOURCES

The following videos and documents provide additional information about BQL for Equities in Microsoft® Excel.

Type	Title	Description
	Fact Sheet: BQL Fundamentals	An overview of the benefits of using BQL to get equity fundamentals in Excel and an overview of the calendarization and blending methodologies.
	Video Tutorial: Screen, Score, and Calculate Aggregate Stats from an Equity Universe	A video introduction to BQL, showcasing the new functionality released for equities in Excel that allows for custom analysis in seconds, with a focus on equities.
	Video Tutorial: BQL for Technical Analysis	A video introduction to BQL and how it can be used to request technical analysis data, create scores and screens, and perform relative valuation.
	Tutorial: BQL for Fundamentals	A spreadsheet tutorial that shows how to download equity fundamentals and estimates to Excel using BQL.
	Tutorial: BQL for Technical Analysis	A spreadsheet tutorial that shows how to perform technical analysis for scoring, screening, and aggregating data using BQL.
	Tutorial: Equities Scoring, Screening, and Aggregation	A spreadsheet tutorial that shows how to screen for, perform aggregated analysis of, and score equities in Excel using BQL.
	Tutorial: Key Benefits Overview for BQL Fundamentals	A spreadsheet tutorial that shows how to build fundamental formulas and perform financial analysis using BQL.

Date printed: 09/20/2019

Type	Title	Description
	Tutorial: BQL for Cash Dividends	A spreadsheet tutorial that shows how to download cash dividends to Excel using BQL.
	Guide: BQL Bloomberg Fundamentals: Data Parameters & Associative Columns	A reference guide to the parameters and associative columns you can use in BQL() and BQL.Query() formulas to download equity fundamentals to Excel.
	Guide: Data Items for Technical Analysis	A reference guide to data items and their inputs and outputs you can use to perform technical analysis with BQL.
	Guide: Function Reference	A reference guide to functions you can use to perform calculations, statistical analysis, and more with BQL.
	Workflow Solution: Monitoring Short-Term Warning Signals for Equities	A customizable spreadsheet that lets you monitor significant downward outlook revisions across a range of datasets using BQL.
	Workflow Solution: Screening for Equities by Piotroski Score	A customizable spreadsheet that lets you screen for equities with the highest Piotroski score using BQL.

PARAMETERS

The following topics provide information on parameters for equity data sets.

Fundamental Parameters
Fundamental Parameter Interactions
Pricing Data Parameters

FUNDAMENTAL PARAMETERS

Equity fundamental parameters let you fit company financials and estimates to your models:

Name & Mnemonics	Description	Default & Other Values
Accounting Standard fa_acct_std std	Choose the accounting standard for the fundamental data, where available.	**MIXED** (default), IFRS, US, LOCAL
Actuals or Estimates fa_act_est_data,	Choose whether the data you return is actual (reported) only, estimate only, or actual data for	**AE** (default), A, E

Name & Mnemonics	**Description**	**Default & Other Values**
ae	reported periods and estimate data for future periods.	
Adjusted for Abnormal Items fa_adjusted adj	Choose between adjusted or as reported data. Adjusting cleans one-off and recurring charges from the data.	Y, **N** (default)
Adjusted for Capital Changes capital_changes_adjust ca_adj	Adjust price history for capital changes , such as spin-offs, stock splits, stock dividends, and rights offerings.	**SPLITS** (default), RAW
As of Date as_of_date dates	Specify a historical date or date range to retrieve the publicly available data that was observed "as of" those date(s).	**0D** (default), absolute or relative date or date range
Consolidated Data fa_consolidated consl	Choose between consolidated or parent data.	**Y** (default), N
Currency currency	Convert values to a different currency.	**None** (default), any three-character ISO currency code
Currency Conversion Method currency_conversion_method cur_meth	Choose the method that the *Currency* parameter uses to convert values from one currency to another.	**ACC** (default), AOD
Estimate Source est_source	Choose the consensus type or broker source for estimate data. For example, the default source is the Bloomberg Standard Consensus (BST).	**BST** (default), BLI, BPE, CGD
Filing Status fa_filing_status fs	Get data from a specific report version, such as the least recent or most recent filing.	**MR** (default), MRXP, LR
Fill When Data is Missing	Fill missing values in a data series with either NA or the previous value.	**NA** (default), PREV

Name & Mnemonics	Description	Default & Other Values
fill		
Period Offset fa_period_offset fpo	Choose a relative fiscal period or period range for which you want data. The anchor period "0" is either today or the end of the period(s)/year(s) entered for the *Period Reference* parameter. For example, "-2, 2" gets data for the last three reported periods and next two estimates.	**0R** (default - no offset), see link for other possible values
Period Reference fa_period_reference fpr	Select a fixed date or fiscal period for the fundamental data you want to retrieve, based on the company's reporting calendar. You can choose a single date/period or a range of dates/periods. When no *Period Reference* is specified, the *Period Reference* is today.	**None** (default), see link for other possible values
Period Type fa_period_type fpt	Choose the periodicity of the fundamental data. You can select between Annual (A), Quarterly (Q), Semi-Annual (S), or synthetic periods calculated by Bloomberg, such as Blended Trailing (BT) and Blended Annual (BA).	**LTM** (default), Q, S, A, P, YTD, BA, BQ, BS, BT
Source for Calculated Period Types fa_period_type_source fpts	If you chose a synthetic fiscal periods for the *Period Type* parameter, such as the Last 12 Months (LTM) or Blended Trailing (BT) periods, select the reporting periodicity used to calculate the values. The default source is the company's primary reporting periodicity.	**P** (default), A, S, Q
Year End Alignment fa_period_end_year fye	Realign a company's fiscal period reported values to a custom calendar. You can enter "C" to realign the fiscal year with the calendar year or "C[MMDD]" (e.g., "C0925") to end the fiscal year on a custom date.	**F** (default), C, C + <MMDD>

For more on how the parameters interact: *Fundamental Parameter Interactions*.

ACCOUNTING STANDARD

For BQL() and BQL.Query() formulas in Excel, the *Accounting Standard* (fa_acct_std) parameter lets you choose the accounting standard for the company financial data you are retrieving, when that accounting standard is available.

Parameter Mnemonics:

- fa_acct_std
- std

Valid Values:

Value	Description
MIXED	Mixed Accounting Standards (depending on data availability) (**Default**) Retrieves data in the most widely used accounting standard, depending on data availability for the company. If data sets in more than one accounting standard are available for a fiscal period, the priority waterfall of the retrieved accounting standard is IFRS, then US GAAP, then Local GAAP.
IFRS	IFRS Retrieves the data set reported by the company in the International Financial Reporting Standards (IFRS) accounting standard, if available. This standard is set by the International Accounting Standards Board (IASB) and provides a reliable and transparent framework for financial reporting to support economic decisions by investors, lenders, and other users of general purpose financial statements.
US	US GAAP Retrieves the data set reported by the company in the US Generally Accepted Accounting Principles (GAAP) accounting standard, if available. GAAP procedures and rules govern accepted accounting practices as defined and supervised by the Financial Accounting Standards Board (FASB), a self-regulatory organization.
LOCAL	Local GAAP Retrieves the data set reported by company in the Local GAAP accounting standard, if available.

Example Formulas:

- Retrieve Toyota's earnings per share (EPS) for the latest Last 12 Months (LTM) period, using data based on the IFRS accounting standard (if not available, the formula attempts to download data based on US GAAP, then local GAAP):

  =BQL("TOYOF US Equity", "is_eps")

  =BQL.Query("get(is_eps) for(['TOYOF US Equity'])")

- Retrieve Toyota's EPS for the latest Last 12 Months (LTM) period, using data based on the local GAAP accounting standard, if available:

  =BQL("TOYOF Equity", "is_eps(fa_acct_std=LOCAL)")

  =BQL.Query("get(is_eps(std=LOCAL)) for(['TOYOF US Equity'])")

- Retrieve Toyota's EPS for the latest Last 12 Months (LTM) period, using data based on the IFRS accounting standard (which is not available for Toyota):

  =BQL("TOYOF US Equity", "is_eps(std=IFRS)")

  =BQL.Query("get(is_eps(std=IFRS)) for(['TOYOF US Equity'])")

ACTUALS OR ESTIMATES

For BQL() and BQL.Query() formulas in Excel, the *Actual or Estimate* (fa_act_est_data) parameter lets you choose whether to retrieve actual or estimated company financials data. For example, you can use the parameter to retrieve actual data for historical periods and estimate data for future periods, or estimate data for both historical and future periods.

Parameter Mnemonics:

- fa_act_est_data
- ae

Valid Values:

Value	Description
AE	Actuals for Reported Periods; Estimates for Future Periods (**Default**) Retrieves actual data reported by the company for historical fiscal periods and estimates data for fiscal periods not yet reported by the company.
A	Actuals Retrieves actual data reported by the company for historical fiscal periods.
E	Estimates Retrieves estimates data both for historical fiscal periods reported by the company and for fiscal periods not yet reported by the company.

Example Formulas:

- Retrieve the 2017 fiscal year estimate revisions for IBM's earnings per share (EPS), as forecast every day between May 1, 2017 and May 5, 2017:

  =BQL("IBM US Equity", "is_eps(dates=range(2017-05-01, 2017-05-05), fpr='2017', fa_act_est_data=E)")

  =BQL.Query("get(is_eps(dates=range(2017-05-15, 2017-05-19), fpr='2017', fa_act_est_data=E)) for(['IBM US Equity'])")
- Retrieve IBM's actual EPS for the last five quarters and estimate EPS for the next two quarters:

  =BQL("IBM US Equity", "is_eps(fpt=Q, fpo=range('-4', '2'), ae=AE)")

  =BQL.Query("get(is_eps(fpt=Q, fpo=range('-4', '2'), ae=AE)) for(['IBM US Equity'])")

ADJUSTED FOR ABNORMAL ITEMS

For BQL() and BQL.Query() formulas in Excel, the *Adjusted for Abnormal Items* (fa_adjusted) parameter lets you choose between adjusted or as-reported data for company financials data items. Adjusting cleans the data for one off and non-recurring charges.

Note: This parameter only impacts data for companies where the value for the data item adjusted_financials_indicator is Y.

Parameter Mnemonics:

- fa_adjusted

- adj

Valid Values:

Value	Description
Y	Adjusted Retrieves data that is adjusted for abnormal items.
N	Not Adjusted (**Default**) Retrieves data that is not adjusted for abnormal items, where available.

Example Formulas:

- Retrieve IBM's EPS for the latest LTM period, not adjusted for abnormal items:

  =BQL("IBM US Equity", "is_eps")

  =BQL.Query("get(is_eps) for(['IBM US Equity'])")
- Retrieve IBM's earnings per share (EPS) for the latest Last 12 Months (LTM) period, adjusted for abnormal items:

  =BQL("IBM US Equity", "is_eps(adj=Y)")

  =BQL.Query("get(is_eps(adj=Y)) for(['IBM US Equity'])")

ADJUSTED FOR CAPITAL CHANGES

For BQL() and BQL.Query() formulas in Excel, when using company financials data items that take into account the company's stock price, the *Adjusted for Capital Changes* (capital_changes_adjust) parameter lets you adjust the stock's price history for capital changes, such as spin-offs, stock splits, stock dividends, and rights offerings.

Parameter Mnemonics:

- capital_changes_adjust
- ca_adj

Valid Values:

Value	Description
SPLITS	Adjusted (**Default**) Retrieves data adjusted for capital changes, such as spin-offs, stock splits, stock dividends, and rights offerings.
RAW	Not Adjusted Retrieves data not adjusted for any capital changes.

Example Formulas:

- Retrieve Argus Therapeutics' earnings per share (EPS) for the latest Last 12 Months (LTM) period, adjusted for capital changes:

=BQL("ARGS US Equity", "is_eps")

=BQL.Query("get(is_eps) for(['ARGS US Equity'])")

- Retrieve Argus Therapeutics' EPS for the latest Last 12 Months (LTM) period, not adjusted for capital changes:

  =BQL("IBM US Equity", "is_eps(ca_adj=RAW)")

  =BQL.Query("get(is_eps(ca_adj=RAW)) for(['ARGS US Equity'])")

AS OF DATE

For BQL() and BQL.Query() formulas in Excel, the *As Of Date* (as_of_date) parameter lets you perform point-in-time analysis as of a specific date or date range. For example, for company financials data, you can use the parameter to request publically available data that was observed "as of" a given date in the past.

Parameter Mnemonics:
- as_of_date
- dates

Default Value: 0D (today's date)

Valid Values:

Value	Description
<Date>	Absolute As Of Date Retrieves the data that would have been known to an observer having perfect access to public information on a fixed date. The date must be in the format <YYYY-MM-DD>. For example, "dates=2016-09-15" retrieves the data that was publicly available as of September 15, 2016.
<Date range>	Range of Absolute As Of Dates Retrieves the data that would have been known to an observer having perfect access to public information between two fixed observation dates. The dates must be specified as an argument to the *range*() function and in the <YYYY-MM-DD> format. For example, "dates=range(2016-01-01, 2017-12-31)" retrieves the series of data that was publicly available as of every date from January 1, 2016 to December 31, 2017.
<Non-positive integer> + D, W, M, Q, S, or Y	Relative As Of Date Retrieves the data that would have been known to an observer having perfect access to public information on a relative, rolling observation date. For example, "dates=-1Y" retrieves the data that was publicly available exactly one year ago, measured from today.
<Non-positive integer range> +	Range of Relative As Of Dates

Value	Description
D, W, M, Q, S, or Y	Retrieves the data that would have been known to an observer having perfect access to public information between two relative, rolling observation dates. The relative date range must be specified as an argument to the *range*() function. For example, "dates=range(-10D, 0D)" retrieves the stream of data that was publicly available as of every day between ten days ago and today.

Note: The value you assign to *As Of Date* interacts with the *Period Offset*, *Period Reference*, and *Period Type* parameters. For more information, see *Fundamental Parameter Interactions*.

Example Formulas:

- Retrieve IBM's earnings per share (EPS) for the Last 12 Months (LTM) period reported as of September 15, 2016:

  =BQL("IBM US Equity", "is_eps(dates=2016-09-15)")

  =BQL.Query("get(is_eps(dates=2016-09-15)) for(['IBM US Equity'])")
- Retrieve IBM's EPS for the LTM period on the first date in the range, then the new value on each day it was reported between January 1, 2016 to December 31, 2017. Output is blank when no new value was reported:

  =BQL("IBM US Equity", "is_eps(dates=range(2016-01-01, 2017-12-31))")

  =BQL.Query("get(is_eps(dates=range(2016-01-01, 2017-12-31))) for(['IBM US Equity'])")
- Retrieve IBM's EPS for the Last 12 Months (LTM) period reported as of ten days ago:

  =BQL("IBM US Equity", "is_eps(dates=-10D)")

  =BQL.Query("get(is_eps(dates=-10D)) for(['IBM US Equity'])")
- Retrieve IBM's EPS for the last reported quarter every day between ten days ago and today. The last reported EPS is retrieved for the first date in the range, then on every day a new value was reported. Output is NA when no new value was reported:

  =BQL("IBM US Equity", "is_eps(dates=range(-10D, 0D), fpt=Q)")

  =BQL.Query("get(is_eps(dates=range(-10D, 0D), fpt=Q)) for(['IBM US Equity'])")

CONSOLIDATED DATA

For BQL() and BQL.Query() formulas in Excel, the *Consolidated Data* (fa_consolidated) parameter allows you to choose between consolidated or unconsolidated data.

A consolidated financial statement covers the activities of a parent company and its subsidiaries in a single report, as if they were a single company operating under one roof. Unconsolidated financial statements, treat each entity as if it were entirely separate - the parent unrelated to the subsidiaries, and the subsidiaries unrelated to one another. For example, if a subsidiary earned $1 in income, that $1 would show up on the parent's consolidated statement and the subsidiary's unconsolidated statement, but not the parent's unconsolidated statement.

Parameter Mnemonics:

- fa_consolidated
- consl

Valid Values:

Value	Description
Y	Consolidated (**Default**) Retrieves consolidated data.
N	Unconsolidated Retrieves unconsolidated data, when available. Unconsolidated data is available only for companies where the value for the data item eqy_consol_fund_avail is Y.

Example Formulas:

- Retrieve Toyota's parent sales figures for Q2 2016:

  =BQL("TOYOF US Equity", "sales_rev_turn(fpt=Q, fpr='2016Q2', consl=N)")

  =BQL.Query("get(sales_rev_turn(fpt=Q, fpr='2016Q2', consl=N)) for(['TOYOF US Equity'])")
- Retrieve sales figures for Q2 2016 for the parent company (Toyota) and all its subsidiaries:

  =BQL("TOYOF US Equity", "sales_rev_turn(fpt=Q, 'fpr=2016Q2')")

  =BQL.Query("get(sales_rev_turn(fpt=Q, fpr='2016Q2')) for(['TOYOF US Equity'])")

CURRENCY

For BQL() and BQL.Query() formulas in Excel, the *Currency* (currency) parameter allows you to convert monetary values into another currency. For example, you can use the parameter to compare company financials between companies that report in different currencies.

Parameter mnemonic: currency

Default Value: None (local currency)

Valid Values: Three-character ISO currency codes. For a list of codes: *CURR Help Page > Currency Codes*.

Example Formulas:

- Retrieve IBM's earnings per share (EPS) for the latest Last 12 Months (LTM) period converted from USD to EUR:

  =BQL("IBM US Equity", "is_eps(currency=EUR)")

  =BQL.Query("get(is_eps(currency=EUR)) for(['IBM US Equity'])")
- Retrieve IBM's EPS for the latest Last 12 Months (LTM) period, as reported on February 15, 2018, converted from USD to EUR using the conversion rate on that date:

  =BQL("IBM US Equity", "is_eps(currency=EUR, cur_meth=AOD, dates=2018-02-15)")

  =BQL.Query("get(is_eps(currency=EUR, cur_meth=AOD, dates=2018-02-15)) for(['IBM US Equity'])")

CURRENCY CONVERSION METHOD

For BQL() and BQL.Query() formulas in Excel, the *Currency Conversion Method* (currency_conversion_method) parameter lets you choose the method that the *Currency* (currency) parameter uses to convert monetary values from one currency to another.

Parameter Mnemonics:
- currency_conversion_method
- cur_meth

Valid Values:

Value	Description
ACC	Account Method (**Default**) Converts data to another currency using the average conversion rate during the fiscal period for flow accounting concepts (such as Revenue) and the currency conversion rate at the end of the fiscal period for stock accounting concepts (such as Assets).
AOD	As Of Date Method Converts fundamental data to another currency using the currency conversion rate on a date specified via the *As of Date* parameter.

Example Formulas:
- Retrieve IBM's earnings per share (EPS) for the Last 12 Months (LTM) period as reported on February 15, 2018, converted from USD to EUR using the average conversion rate during the period:

  =BQL("IBM US Equity", "is_eps(currency=EUR, dates=2018-02-15)")

  =BQL.Query("get(is_eps(currency=EUR, dates=2018-02-15)) for(['IBM US Equity'])")
- Retrieve IBM's EPS for the Last 12 Months (LTM) period as reported on February 15, 2018, converted from USD to EUR using the conversion rate on that date:

  =BQL("IBM US Equity", "is_eps(currency=EUR, cur_meth=AOD, dates=2018-02-15)")

  =BQL.Query("get(is_eps(currency=EUR, cur_meth=AOD, dates=2018-02-15)) for(['IBM US Equity'])")

ESTIMATE SOURCE

For BQL() and BQL.Query() formulas in Excel, the *Estimate Source* (est_source) parameter lets you specify your preferred consensus type or broker source for estimates data.

Parameter Mnemonic: est_source

Valid Values:

Value	Description
BST	Bloomberg Standard Consensus (**Default**) Retrieves estimates data from the Bloomberg Standard Consensus.
BLI	Bloomberg Leading Indicator Consensus

Value	Description
	Retrieves estimates data from the Bloomberg Leading Indicator Consensus, which includes only estimates updated within the last four weeks.
BPE	Bloomberg Post Event Consensus Retrieves estimates data from the Bloomberg Post Event Consensus, which includes only estimates updated after the last earnings announcement date.
CGD	Company Guidance Retrieves guidance data provided by the company.

Example Formulas:

- Retrieve Toyota's estimate earnings per share (EPS) from the Bloomberg Standard consensus for the next fiscal period:

  =BQL("TOYOF US Equity", "is_eps(fpo=1)")

  =BQL.Query("get(is_eps(fpo=1)) for(['TOYOF US Equity'])")
- Retrieve Toyota's estimate EPS from company guidance for the next fiscal period:

  =BQL("TOYOF US Equity", "is_eps(fpo='1', est_source=CGD)")

  =BQL.Query("get(is_eps(fpo='1', est_source=CGD)) for(['TOYOF US Equity'])")
- Retrieve Toyota's estimate EPS from the Bloomberg Leading Indicator consensus for the next fiscal period:

  =BQL("TOYOF US Equity", "is_eps(fpo='1', est_source=BLI)")

  =BQL.Query("get(is_eps(fpo='1', est_source=BLI)) for(['TOYOF US Equity'])")

FILING STATUS

For BQL() and BQL.Query() formulas in Excel, the *Filing Status* (fa_filing_status) parameter lets you get company financials data from a specific report version, such as the least recent or most recent filing. You can also filter out filings with less complete data, such as preliminary filings.

Parameter Mnemonics:

- fa_filing_status
- fs

Valid Values:

Value	Description
MR	Most Recent Filing (**Default**) Retrieves the most recent data for a fiscal period from the different versions of filings available for that period.
MRXP	Most Recent Filing, Excluding Earnings or Preliminary Filings

Value	Description
	Retrieves the most recent data for a fiscal period from the different versions of filings available for that period, excluding earnings or preliminary records.
LR	Least Recent (First) Filing Retrieves the least recent data for a fiscal period from the different versions of filings available for that period (i.e., the first version of data that a company reported for a fiscal period).

Example Formulas:

- Retrieve IBM's earning per share (EPS) for the latest Last 12 Months (LTM) period, using reported data from the most recent filing:

  =BQL("IBM US Equity", "is_eps")

  =BQL.Query("get(is_eps) for(['IBM US Equity'])")
- Retrieve IBM's EPS for the latest Last 12 Months (LTM) period, using reported data from the least recent filing:

  =BQL("IBM US Equity", "is_eps(fs=LR)")

  =BQL.Query("get(is_eps(fs=LR)) for(['IBM US Equity'])")

FILL WHEN DATA IS MISSING

For BQL() and BQL.Query() formulas in Excel, the *Fill When Data is Missing* (fill) parameter lets you fill missing values in a data series of company financials with the last available data, so you can get a continuous stream of data even when there is no data available for certain observations.

Parameter Mnemonic: fill

Valid Values:

Value	Description
NA	No Data (**Default**) Returns NA.
PREV	Previous Data Returns the last available value.

Example Formulas:

- Retrieve a time series of estimates for IBM's LTM price to earnings (P/E) ratio, as reported between January 1, 2014 and December 31, 2014; for non-trading days, "NaN" appears:

  =BQL("IBM US Equity", "pe_ratio(dates=range(2014-01-01, 2014-12-31), ae=E)")

  =BQL.Query("get(pe_ratio(dates=range(2014-01-01, 2014-12-31), ae=E)) for(['IBM US Equity'])")
- Retrieve a time series of estimates for IBM's LTM price to earnings (P/E) ratio, as reported between January 1, 2014 and December 31, 2014; for non-trading days, the value from the previous day appears:

=BQL("IBM US Equity", "pe_ratio(dates=range(2014-01-01, 2014-12-31), ae=E, fill=PREV)")

=BQL.Query("get(pe_ratio(dates=range(2014-01-01, 2014-12-31), ae=E, fill=PREV)) for(['IBM US Equity'])")

PERIOD OFFSET

For BQL() and BQL.Query() formulas in Excel, the *Period Offset* (fa_period_offset) parameter lets you choose relative fiscal period(s) for which you want data.

The data starts either from today or from the end of the period or year you specify with the *Period Reference* (fa_period_reference) parameter. Offsets are entered as positive and/or negative integers, with the reference date (today or another date) defined as period "0".

For example, "fpo='1'" retrieves data for the next period forward. "fpo=range('-2', '2')" retrieves data for the last three reported periods and the next two estimates.

Parameter Mnemonics:

- fa_period_offset
- fpo

Default Value: 0R (no offset)

Valid Values:

Value	Description
<Integer>	Integer A negative integer retrieves data for a fiscal period in the past, while a positive integer pulls data for a fiscal period in the future. For example, "fpo='-1'" pulls data from one fiscal period prior to period 0, while "fpo='1'" pulls from one fiscal period after period 0. **Note:** When using this syntax, the length of one offset period is determined by the *Period Type* parameter.
<Integer> + R or <Integer> + G	Rolling Integer Retrieves data on a rolling basis when you do not specify a reference date with the *Period Reference* parameter. For example, "fpo='-1R'" always pulls data from the fiscal period prior to the last reported period before the observation date (when you specify a date or date range with the *As of Date* parameter or today [when a value for dates is not specified]).
<Integer> + F	Fixed Integer Retrieves data on a fixed basis. For example, "fpo='-1F'" retrieves data from the fiscal period prior to the last reported fiscal period on the specified date. **Note:** When using this syntax, you must use the *As of Date* parameter to specify an observation date or date range.
<Integer> +	Rolling Integer with Step Length

Value	Description
A, S, or Q < Integer> + RA, RS, or RQ	Retrieves data for a fiscal period in the past or future. The letter indicates the length of the step: annual (A), semi-annual (S), or quarterly (Q) periods. For example: • "fpo='1A'" retrieves data looking forward one annual period. • "fpo='-1S'" retrieves data looking back one semi-annual period prior to the last reported period. • "fpo='-3Q'" retrieves data looking back three quarters prior to the last reported period. **Note:** If you do not specify a value for the *Period Type* parameter, this syntax also determines the periodicity for the data. For example, "fpo='1A'" specifies that you retrieve data for an annual period.
< Integer> + FA, FS, or FQ	Fixed Integer with Step Length Retrieves data for annual, semi-annual, or quarterly periods on a fixed basis. For example: • "fpo='-1FA'" retrieves data looking back one annual period prior to the last reported period on the specified date. • "fpo='-1FS'" retrieves data looking back one semi-annual period prior to the last reported period on the specified date. • "fpo='-1FQ'" retrieves data looking back one quarter prior to the last reported period on the specified date. **Note:** When using this syntax, you must use the *As of Date* parameter to specify an observation date or date range.
<Integer range>	Integer Range Retrieves data for a range of periods relative to the specified date. The integer range must be specified as an argument to the *range*() function. For example, "fpo=range('-3', '3')" retrieves seven points of data from fiscal periods -3, -2, -1, 0, 1, 2 and 3. **Note:** When using this syntax, the length of one relative period is determined by the *Period Type* parameter.
<Integer range> + A, S, or Q	Integer Range with Step Length Retrieves data for a range of periods of a specified length. The letter indicates the length of the steps: annual (A), semi-annual (S), or quarterly (Q) periods. The range must be specified as an argument to the *range*() function. For example: • "fpo=range('-3A', '3A')" retrieves seven points of data from fiscal periods -3A, -2A, -1A, 0, 1A, 2A and 3A, where the length of the offset between the fiscal periods is one annual period. • "fpo=range('-2S', '3S')" retrieves six points of data from fiscal periods -2S, -1S, 0, 1S, 2S and 3S, where the length of the offset between the fiscal periods is one semi-annual period. • "fpo=range('-2Q', '2Q')" retrieves five points of data from fiscal periods -2Q, -1Q, 0, 1Q, and 2Q, where the length of the offset between the fiscal periods is one quarter.

Note: The value you assign to *Period Offset* interacts with the *Period Reference*, *Period Type*, and *As of Date* parameters. For more: *Fundamental Parameter Interactions*.

Example Formulas:

- Retrieve IBM's estimate earnings per share (EPS) for the next twelve-month period:

  =BQL("IBM US Equity", "is_eps(fpo='1')")

  =BQL.Query("get(is_eps(fpo='1')) for(['IBM US Equity'])")

  Note: Because the *Period Type* (FPT) parameter is not included in the formula, the default period Last Twelve Months (LTM) is used.
- Retrieve IBM's actual EPS for the last four Last 12 Month (LTM) periods and estimate EPS for the next three LTM periods:

  =BQL("IBM US Equity", "is_eps(fpo=range('-3', '3'))")

  =BQL.Query("get(is_eps(fpo=range('-3', '3'))) for(['IBM US Equity'])")
- Retrieve IBM's actual EPS for the last five quarters and estimate EPS for the next two quarters:

  =BQL("IBM US Equity", "is_eps(fpt=Q, fpo=range('-4', '2'))")

  =BQL.Query("get(is_eps(fpt=Q, fpo=range('-4', '2'))) for(['IBM US Equity'])")
- Retrieve IBM's next LTM estimate EPS:

  =BQL("IBM US Equity", "is_eps(fpt=BT, fpo='1')")

  =BQL.Query("get(is_eps(fpt=BT, fpo='1')) for(['IBM US Equity'])")
- Retrieve IBM's estimate EPS for the LTM period that ends at the same time as the next reported quarter:

  =BQL("IBM US Equity", "is_eps(fpo='1Q', fpt=LTM)")

  =BQL.Query("get(is_eps(fpo='1Q', fpt=LTM)) for(['IBM US Equity'])")
- Retrieve IBM's LTM EPS, looking back from the end of Q2 2016 and Q2 2017:

  =BQL("IBM US Equity", "is_eps(fpr=range('2015Q4', '2016Q4'), fpo='2Q', fpt=LTM)")

  =BQL.Query("get(is_eps(fpr=range('2015Q4', '2016Q4'), fpo='2Q', fpt=LTM)) for(['IBM US Equity'])")

PERIOD REFERENCE

For BQL() and BQL.Query() formulas in Excel, the *Period Reference* (fa_period_reference) parameter lets you choose a fixed fiscal period for company financials data based on the company's reporting calendar. You can choose a single period or a range of periods. For example, "2015 " retrieves data for the year 2015.

If you also use the *Period Offset* (fa_period_offset) parameter in your query, the offset periods for which you want data are relative to the reference period.

Parameter Mnemonics:

- fa_period_reference
- fpr

Default Value: None (the reference period is the last reported period prior to today or the observation date you specify with the *As of Date* (dates) parameter)

Valid Values:

Value	Description
<Date>	Sets the reference to the fiscal period that ends on or directly before the date you specify. The date must be in the YYYY-MM-DD format. For example, "fpr=2015-12-31" sets the reference to the fiscal period ending on December 31, 2015.
<Date range>	Sets the reference to a range of fiscal periods with one year jumps, where the first reference is the fiscal period that ends on or directly before the start date and the last reference is the fiscal period that ends on or directly before the end date. The dates must be specified as an argument to the *range*() function and in the YYYY-MM-DD format. For example, "fpr=range(2013-12-31, 2017-12-31)" sets the first reference to the fiscal period with an end date on or directly before 12/31/2013 and the last reference to the fiscal period with an end date on or directly before 12/31/2017.
<YYYY> or <YYYY> + A	Sets the reference to the fiscal period that ends at the same time as the year you specify. For example, "fpr='2015'" or "fpr='2015A'" sets the reference to the fiscal period with the same end date as 2015A.
<YYYY> + S1, S2, Q1, Q2, Q3, or Q4	Sets the reference to the fiscal period that ends at the same time as the semi-annual or quarterly period you specify. For example: • "fpr='2015S2'" sets the reference to the fiscal period that ends at the same time as S2 2015. • "fpr='2015Q1'" sets the reference to the fiscal period with an end date on or directly before the end of Q1 2015.
<YYYY Range> + A, S1, S2, Q1, Q2, Q3, or Q4	Sets the reference to a range of periods, with the letter representing the length of the steps between reference periods in the range. The range must be specified as an argument to the *range*() function. For example: • "fpr=range('2015A', '2016A')" specifies a range of reference periods, with one year jumps between the reference periods. • "fpr=range('2015S1', '2016S2')" specifies a range of reference periods, with one semi-annual period jumps between the reference periods: S1 2015, S2 2015, S1 2016, and S2 2016. • "fpr=range('2015Q1', '2016Q2')" specifies a range of reference periods, with one quarter jumps between the reference periods: Q1 2015, Q2 2015, Q3 2015, Q4 2015, Q1 2016, Q2 2016.

Note: The value you assign to *Period Reference* interacts with the *Period Offset*, *Period Type*, and *As of Date* parameters. For more: *Fundamental Parameter Interactions*.

Example Formulas:

- Retrieve IBM's earnings per share (EPS) for the Last 12 Months (LTM) period ending at the same time as Q3 2015:

  =BQL("IBM US Equity", "is_eps(fpr='2015Q3')")

  =BQL.Query("get(is_eps(fpr='2015Q3')) for(['IBM US Equity'])")
- Retrieve IBM's EPS for the Last 12 Months (LTM) period ending at the same time as the fiscal period that ends on or directly before September 30, 2015 (i.e., Q3 2015):

  =BQL("IBM US Equity", "is_eps(fpr=2015-09-30)")

  =BQL.Query("get(is_eps(fpr=2015-09-30)) for(['IBM US Equity'])")
- Retrieve IBM's EPS at the end of each quarter between Q3 2015 and Q3 2017:

  =BQL("IBM US Equity", "is_eps(fpr=range('2015Q3', '2017Q3'))")

  =BQL.Query("get(is_eps(fpr=range('2015Q3', '2017Q3'))) for(['IBM US Equity'])")
- Retrieve IBM's EPS for each fiscal period that ends on or between September 30, 2015 and September 30, 2017 (the fiscal periods for data depend on the primary periodicity that the company uses for reporting):

  =BQL("IBM US Equity", "is_eps", "fpr=range(2015-09-30, 2017-09-30)")

  =BQL.Query("get(is_eps(fpr=range(2015-09-30, 2017-09-30))) for(['IBM US Equity'])")
- Retrieve IBM's EPS for each trailing twelve-month period ending at the same time as each quarter between Q2 2016 to Q4 2017:

  =BQL("IBM US Equity", "is_eps", "fpr=range('2016Q2', '2017Q4')", "fpt=BA")

  =BQL.Query("get(is_eps(fpr=range('2016Q2', '2017Q4'), fpt=BA)) for(['IBM US Equity'])")

PERIOD TYPE

For BQL() and BQL.Query() formulas in Excel, the *Period Type* (fa_period_type) parameter lets you choose the periodicity of company financials data. You can specify Annual (A), Quarterly (Q), Semi-Annual (S), or synthetic periods calculated by Bloomberg, such as Blended Trailing (BT) and Blended Annual (BA). For 12-month data, you can choose from multiple calculation methods.

The source for reported data can be the company (from periodic financial filings) or sell-side research (from research reports).

Parameter Mnemonics:

- fa_period_type
- fpt

Valid Values:

Value	Description
A	Annual Retrieves data for annual fiscal periods, as reported by the source.
Q	Quarter

Value	Description
	Retrieves data for quarterly fiscal periods, as reported by the source.
S	Semi-Annual Retrieves data for semi-annual fiscal periods, as reported by the source.
P	Primary Retrieves data for the primary periodicity used by the company to report its financials (e.g., quarters, annuals, etc.). **Note:** For the primary reporting periodicity of the company, see the company's value for the data item primary_periodicity.
LTM	Last Twelve Months (**Default**) Retrieves data for a synthetic, Bloomberg-calculated fiscal period that covers the previous 12 months. The calculation may use quarterly, semi-annual, or annual data reported by the company, depending on the company's primary reporting periodicity. LTM can be calculated as either: • The sum of four quarterly or two semi-annual periods (depending on company disclosure) for flow accounting concepts (such as Revenue) • The average of four quarterly or two semi-annual periods (depending on company disclosure) for average accounting concepts (such as Average Shares Outstanding) • The data reported in the latest fiscal period for stock accounting concepts (such as Total Assets) • The data reported at the beginning of the period for any beginning-of-period accounting concepts (such as Cash at the beginning of the period) For any last-12-month fiscal period coinciding with the annual fiscal period, the annual data is returned without any additional calculations.
YTD	Year to Date Retrieves data for a synthetic, Bloomberg-calculated fiscal period that covers the last three, six, nine, or 12 months of the current calendar year. The calculation may use quarterly, semi-annual, or annual data reported by a company, depending on the company's primary reporting periodicity.
BA	Blended Annual Retrieves data for a synthetic, 12-month fiscal period calculated by Bloomberg that ends on a custom date and is independent of any observation date you specify with the *As of Date* parameter. The calculation blends data from quarterly, semi-annual, or annual data reported by a company (depending on the company's primary reporting periodicity), using a time-weighted average.

Value	Description
	The default fiscal year end date is the latest December 31 date that precedes the end date of the last fiscal period reported by the company. However, you can customize the end date with the *Year End Alignment* parameter.
BQ	Blended Quarter Retrieves data for a synthetic, three-month fiscal period calculated by Bloomberg that ends on a custom date and is independent of any observation date you specify with the *As of Date* parameter. The calculation blends data from quarterly, semi-annual, or annual data reported by a company (depending on the company's primary reporting periodicity), using a time-weighted average. The default fiscal year end date is the latest December 31 date that precedes the end date of the last fiscal period reported by the company. However, you can customize the end date with the *Year End Alignment* parameter.
BS	Blended Semi-Annual Retrieves data for a synthetic, six-month fiscal period calculated by Bloomberg that ends on a custom date and is independent of any observation date you specify with the *As of Date* parameter. The calculation blends data from quarterly, semi-annual, or annual data reported by a company (depending on the company's primary reporting periodicity), using a time-weighted average. The default fiscal year end date is the latest December 31 date that precedes the end date of the last fiscal period reported by the company. However, you can customize the end date with the *Year End Alignment* parameter.
BT	Blended Trailing Retrieves data for a synthetic 12-month fiscal period calculated by Bloomberg that ends on a custom date. The calculation blends data from quarterly, semi-annual, or annual data reported by a company (depending on the company's primary reporting periodicity), using a time weighted average. The end date is determined by the value for the *As of Date* parameter.

Note: The value you assign to *Period Type* interacts with the *Period Offset*, *Period Reference*, and *As of Date* parameters. For more: *Fundamental Parameter Interactions*.

Example Formulas:

- Retrieve IBM's earnings per share (EPS) for the last reported quarter:

  =BQL("IBM US Equity", "is_eps(fpt=Q)")

  =BQL.Query("get(is_eps(fpt=Q)) for(['IBM US Equity'])")
- Retrieve IBM's EPS for a trailing 12-month period that ends on February 1, 2018:

  =BQL("IBM US Equity", "is_eps(fpt=BT, dates=2018-02-01)")

  =BQL.Query("get(is_eps(fpt=BT, dates=2018-02-01)) for(['IBM US Equity'])")
- Retrieve IBM's estimate EPS for the next quarter:

=BQL("IBM US Equity", "is_eps(fpo='1', fpt=Q)")

=BQL.Query("get(is_eps(fpo='1', fpt=Q)) for(['IBM US Equity'])")

- Retrieve IBM's estimate EPS for the trailing 12-month period ending on October 20, 2018, as forecast each day from February 01, 2018 to April 15, 2018. Output is blank when no new estimate was made:

  =BQL("IBM US Equity", "is_eps(fpt=BA, dates=range(2018-02-01, 2018-04-15), fye=C1020)")

  =BQL.Query("get(is_eps(fpt=BA, dates=range(2018-02-01, 2018-04-15), fye=C1020)) for(['IBM US Equity'])")

SOURCE FOR CALCULATED PERIOD TYPES

For BQL() and BQL.Query() formulas in Excel, the *Source for Calculated Period Types* (fa_period_type_source) parameter lets you choose the fiscal periodicity used to calculate values for synthetic periods you select with the *Period Type* (fa_period_type) parameter, such as the Last 12 Months (LTM) and Blended Trailing (BT) periods. By default, the source is the Primary (P) periodicity.

Parameter Mnemonics:

- fa_period_type_source
- fpts

Valid Values:

Value	Description
P	Primary (**Default**) Calculates the data for synthetic periods using the primary fiscal periodicity used by the company for reporting. Companies may report data annually, semi-annually, and/or quarterly. **Note:** For the primary reporting periodicity of the company, see the company's value for the data item primary_periodicity.
A	Annual Calculates the data for synthetic fiscal periods from annual periods reported by the company.
S	Semi-Annual Calculates the data for synthetic fiscal periods from semi-annual periods reported by the company.
Q	Quarter Calculates the data for synthetic fiscal periods from quarterly periods reported by the company.

Example Formulas:

- Retrieve IBM's earnings per share (EPS) for the latest Last 12 months (LTM) period, calculated from the annual reported EPS from the company:

  =BQL("IBM US Equity", "is_eps(fpts=A)")

=BQL.Query("get(is_eps(fpts=A)) for(['IBM US Equity'])")

- Retrieve IBM's EPS for the trailing 12-month period ending on September 15, 2017, calculated from the quarterly reported EPS from the company:

  =BQL("IBM US Equity", "is_eps(fpts=Q, fpt=BT, dates=2017-09-15)")

  =BQL.Query("get(is_eps(fpts=Q, fpt=BT, dates=2017-09-15)) for(['IBM US Equity'])")

YEAR END ALIGNMENT

For BQL() and BQL.Query() formulas, the *Year End Alignment* (fa_period_year_end) parameter allows you to realign a company's fiscal periods to a custom calendar for reported values. You can specify 'C' to align the fiscal year with the calendar year or C + MMDD (e.g., 'C0925') to end the fiscal year on a custom date.

Note: If you do not use a blended period for the *Period Type* (fa_period_type) parameter, such as Blended Annual (BA), you are just shifting values to a different date and not recalculating them.

Parameter Mnemonics:

- fa_period_end_year
- fye

Valid Values:

Value	Description
F	Fiscal Year End (**Default**) Defines the fiscal periods of a company according to the standard fiscal calendar of that company. **Note:** Bloomberg defines the standard fiscal calendar for a company based on the period end date of a company's annual report. If the period end date of a company's annual report falls between 01/15/<YYYY> to 01/14/<YYYY>+1, Bloomberg defines that annual report as Fiscal Year <YYYY>. Interim (semiannual or quarterly) reports are then defined based on the definition of the annual report, which includes the time frames covered by those interim reports.
C	Calendar Year End Realigns the fiscal periods of a company based on a calendar year (ending on December 31).
C + <MMDD>	Custom Year End Realigns the fiscal periods of a company based on a fiscal year ending on a custom month and date.

For example, Company X has a fiscal year ending on September 30 and Company Y has a fiscal year ending on June 30. Both companies report Annual and Quarterly filings. The fye parameters of "C" or "C0531" make fiscal period names consistent for both companies:

Period end	Company	fye='F'	fye='C'	fye='C0531'
March 31	Company X	Q2	Q1	Q3

Period end	Company	fye='F'	fye='C'	fye='C0531'
	Company Y	Q3	Q1	Q3
June 30	Company X	Q3	Q2	Q4/A
	Company Y	Q4/A	Q2	Q4/A
September 30	Company X	Q4/A	Q3	Q1
	Company Y	Q1	Q3	Q4
December 31	Company X	Q1	Q4/A	Q2
	Company Y	Q2	Q4/A	Q2

Example Formulas:

- Retrieve IBM's earnings per share (EPS) for the last reported fiscal year, realigning the fiscal year to end at the end of the calendar year:

  =BQL("IBM US Equity", "is_eps(fye='C', fpt=A)")

  =BQL.Query("get(is_eps(fye='C', fpt=A)) for(['IBM US Equity'])")
- Retrieve IBM's EPS for the last reported fiscal year, realigning the fiscal year to end on September 15:

  =BQL("IBM US Equity", "is_eps(fye='C0915', fpt=A)")

  =BQL.Query("get(is_eps(fye='C0915', fpt=A)) for(['IBM US Equity'])")

FUNDAMENTAL PARAMETER INTERACTIONS

When using equity fundamental parameters to retrieve company financials and estimates for specific and/or relative time periods, it is important to understand how the parameters interact and what data you will get:

Type and Offset
Type and Reference
Type, Offset, and Reference
Type, Offset, Reference, and Dates

For more on each of the parameters, see *Fundamental Parameters*.

Date printed: 09/20/2019

TYPE AND OFFSET

Period Type (fa_period_type; abbrev: fpt) and *Period Offset* (fa_period_offset; abbrev: fpo):

Example	Explanation
Periodicity Only =BQL("IBM US Equity", "is_eps(fpt=Q)") =BQL.Query("get(is_eps(fpt=Q)) for(['IBM US Equity'])")	The value for *fpt* is "Q", so the query retrieves quarterly data. Because you are not using *fpo* to specify an offset, the query gets data for the last reported period. Therefore, the query retrieves the EPS for the last reported quarter.
Offset Only =BQL("IBM US Equity", "is_eps(fpo='1')") =BQL.Query("get(is_eps(fpo='1')) for(['IBM US Equity'])")	Because you are not using *fpt* to specify a periodicity, the default Last 12 Months ("LTM") period is implied. The value for *fpo* is a positive integer ("1"), so the query looks forward one period. However, *fpo* does not include a length for the "1" offset, so the *fpt* value - "LTM" - is also used as the offset length. Therefore, the query retrieves the estimate EPS for the next LTM period after the last reported LTM period.
Periodicity and Offset (Integer Only) =BQL("IBM US Equity", "is_eps(fpo='1', fpt=Q)") =BQL.Query("get(is_eps(fpo='1', fpt=Q)) for(['IBM US Equity'])")	The value for *fpo* is a positive integer ("1"), so the query looks forward one period. However, *fpo* ("1") does not specify the length of the offset, so the value for *fpt* determines the offset length in addition to determining the periodicity. Since *fpt*=Q, the query retrieves the estimated EPS for the next quarter.
Periodicity and Offset (Integer + Period) =BQL("IBM US Equity", "is_eps(fpo='1Q', fpt=LTM)") =BQL.Query("get(is_eps(fpo='1Q', fpt=LTM)) for(['IBM US Equity'])")	The value for *fpo* specifies the length of the offset step as one quarter ("1Q") forward. The value for *fpt* specifies the periodicity as Last 12 Months ("LTM"). Therefore, the query retrieves the estimated EPS for the LTM period that ends at the same time as the next reported quarter.
Periodicity and Offset (Integer Range + Period) =BQL("IBM US Equity", "is_eps(fpo=range('-1Q', '1Q'), fpt=LTM)") =BQL.Query("get(is_eps(fpo=range('-1Q', '1Q'), fpt=LTM)) for(['IBM US Equity'])")	The value for *fpo* ("range(-1Q, 1Q)") indicates an offset range with steps of one quarter (the quarter before the last reported quarter, the last reported quarter, and the next reported quarter). The value for *fpt* is Last 12 Months ("LTM"). Therefore, the query retrieves the actual EPS for the LTM period that ended at the same time as the quarter before the last reported quarter; the actual EPS for the LTM period that ended at the same time as the last reported quarter; and the estimate EPS for the LTM period that ends at the same time as the next reported quarter.

For additional information: *BQL Bloomberg Fundamentals: Data Parameters & Associative Columns* .

TYPE AND REFERENCE

Period Type (fa_period_type; abbrev: fpt) and *Period Reference* (fa_period_reference; abbrev: fpr):

Example	Explanation
Periodicity Only =BQL("IBM US Equity", "is_eps(fpt=Q)") =BQL.Query("get(is_eps(fpt=Q)) for(['IBM US Equity'])")	Because no value is specified for *fpr*, the default value (today) is implied, so data is retrieved for the last fiscal period that ended on or directly before today. The value for *fpt* indicates the periodicity is quarterly ("Q"). Therefore, the query retrieves the EPS for the last reported quarter.
Reference Only =BQL("IBM US Equity", "is_eps(fpr='2016Q4')") =BQL.Query("get(is_eps(fpr='2016Q4')) for(['IBM US Equity'])")	Because no value is specified for *fpt*, the default Last 12 Months ("LTM") period is implied. The value for *fpr* defines the end of the fiscal period for which you want data as the end of Q4 2016. Therefore, the query retrieves the EPS for the LTM period that ended at the same time as Q4 2016.
Periodicity and Reference =BQL("IBM US Equity", "is_eps(fpr='2016Q4', fpt=Q)") =BQL.Query("get(is_eps(fpr='2016Q4', fpt=Q)) for(['IBM US Equity'])")	The value for *fpr* is "2016Q4", which indicates that the end of the fiscal period for which you want data is the end of Q4 2016. The value for *fpt* is quarterly ("Q"). Therefore, the query retrieves the Q4 2016 EPS.
Periodicity and Reference Range (Years) =BQL("IBM US Equity", "is_eps(fpr=range('2015', '2016'), fpt=Q)") =BQL.Query("get(is_eps(fpr=range('2015', '2016'), fpt=Q)) for(['IBM US Equity'])")	The value for *fpr* is a range ("range('2015', '2016')"), so the query retrieves data between the end of 2015 and the end of 2016. By default, when you specify only a year for *fpr* (like "2015"), an annual fiscal period is implied. That means each step in the *fpr* range is one year. The value for *fpt* is quarterly ("Q"). Therefore, the query retrieves the Q4 2015 EPS and Q4 2016 EPS.
Periodicity and Reference Range (Years + Period) =BQL("IBM US Equity", "is_eps(fpr=range('2016Q2', '2016Q4'), fpt=LTM)") =BQL.Query("get(is_eps(fpr=range('2016Q2', '2016Q4'), fpt=LTM)) for(['IBM US Equity'])")	The value for *fpr* is a range ("range('2016Q2', '2016Q4')"), so the query retrieves data between the end of Q2 2016 and the end of Q4 2016. By including a periodicity in the *fpr* values ("Q"), you specify the length of each step (one quarter) in the *fpr* range. The value for *fpt* is Last 12 Months ("LTM"). Therefore, the query retrieves the LTM EPS values for periods ending at the same time as the end of Q2 2016, Q3 2016, and Q4 2016.

Example	Explanation
Periodicity and Reference Range (Dates) =BQL("IBM US Equity", "is_eps(fpr=range(2015-12-31, 2016-12-31), fpt=Q)") =BQL.Query("get(is_eps(fpr=range(2015-12-31, 2016-12-31), fpt=Q)) for(['IBM US Equity'])")	The value for *fpr* is "range(2015-12-31, 2016-12-31)", which specifies the range for which you want data begins with the fiscal period ending on or directly before December 31, 2015 and ends with the period that ends on or directly before December 31, 2016. Because the *fpr* range only specifies fiscal period end dates and no fiscal period type (e.g., "2015Q2" or "2016"), the length of the steps is determined by the value for *fpt*, which is quarterly ("Q"). *fpt*=Q also indicates that you want quarterly data. Therefore, the query retrieves the Q4 2015, Q1 2016, Q2 2016, Q3 2016, and Q4 2016 EPS.

For additional information: *BQL Bloomberg Fundamentals: Data Parameters & Associative Columns* .

PRICING DATA PARAMETERS

Pricing parameters let you fit data such as bids, offers, VWAP, and turnover to your models:

Corporate Action Adjustment
Currency
As Of Date
Fill When Data is Missing
Periodicity

CORPORATE ACTION ADJUSTMENT

For BQL() and BQL.Query() formulas in Excel, the *Corporate Action Adjustment* (ca_adj) parameter lets you choose whether equity data is adjusted for corporate actions that impact dividends and the number of shares outstanding.

Parameter Mnemonic: ca_adj

Valid Values:

Value	Description
SPLITS	Adjusted for Splits (**Default**) Adjusts data for stock splits only.
FULL	Full Adjustment Adjusts data for corporate actions that impact pricing and the number of shares outstanding, including splits, stock dividends, spin-offs, rights offerings, etc.

Date printed: 09/20/2019

Value	Description
RAW	Not Adjusted Leaves data unadjusted for corporate actions.

Example Formulas:

- Retrieve a time series of daily last prices for PG&E Corp from April 1, 2013 to June 1, 2013, unadjusted for corporate actions:

  =BQL("PCG US Equity", "px_last(dates=range(2013-04-01, 2013-06-01), ca_adj=RAW)")

  =BQL.Query("get(px_last(dates=range(2013-04-01,2013-06-01), ca_adj=RAW)) for(['PCG US Equity'])")

- Retrieve a time series of daily last prices for PG&E Corp from April 1, 2013 to June 1, 2013, adjusted for all corporate actions:

  =BQL("PCG US Equity", "px_last(dates=range(2013-04-01, 2013-06-01), ca_adj=FULL)")

  =BQL.Query("get(px_last(dates=range(2013-04-01,2013-06-01), ca_adj=FULL)) for(['PCG US Equity'])")

CURRENCY

For BQL() and BQL.Query() formulas in Excel, the *Currency* (currency) parameter allows you to convert currency-based data into another currency.

Parameter Mnemonic: currency

Default Value: None (local currency is used)

Valid Values: Three-character ISO currency codes. For a list of codes: *CURR Help Page > Currency Codes*.

Example Formulas:

- Retrieve IBM's last price converted from USD to EUR:

  =BQL("IBM US Equity", "px_last(currency=EUR)")

  =BQL.Query("get(px_last(currency=EUR)) for(['IBM US Equity'])")
- Retrieve a time series of IBM's daily prices from February 1, 2016 to February 28, 2016, converted from USD to JPY:

  =BQL("IBM US Equity", "px_last(dates=range(2016-02-01, 2016-02-28), currency=JPY)")

  =BQL.Query("get(px_last(dates=range(2016-02-01, 2016-02-28), currency=JPY)) for(['IBM US Equity'])")

AS OF DATE

For BQL() and BQL.Query() formulas in Excel, the *As Of Date* (dates) parameter lets you perform historical analysis on a single date or a time series of data. You can use the parameter to request data that was observed 'as of' a given date in the past.

When retrieving data for a historical time series, add the *range*() function to specify the date range.

Parameter Mnemonic: dates

Default Value: 0D (today's date)

Valid Values:

Value	Description
<Date>	Absolute as of date Retrieves data as of a specific, fixed date. The date must be in the YYYY-MM-DD format. For example, "dates=2016-09-15" retrieves the data from September 15, 2016.
<Date range>	Range of Absolute As Of Dates Retrieves a historical time series of data as of specific, fixed date range. Each of the two dates must be in the YYYY-MM-DD format, with the start date first and the dates separated by commas. Additionally, the dates must be nested in the *range*() function. For example, "dates=range(2016-01-01, 2017-12-31)" retrieves a series of data from January 1, 2016 to December 31, 2017.
<Integer> + D, W, M, Q, S, or Y	Relative As Of Date Retrieves data as of a relative, rolling date. For example, "dates=-1M" retrieves the data from one month ago, looking back from today.
<Non-positive integer range> + D, W, M, Q, S, or Y	Range of Relative As Of Dates, expressed in days, weeks, months, quarters, semi-annuals, or years Retrieves a time series of data between two specified relative, rolling observation dates. The start date must be first and the two dates must be separated by commas. Additionally, the dates must be nested in the *range*() function. For example, "dates=range(-10D,0D)" retrieves a stream of data between ten days ago and today.

Example Formulas:

- Retrieve IBM's last price on February 10, 2017:

  =BQL("IBM US Equity", "px_last(dates=2017-02-10)")

  =BQL.Query("get(px_last(dates=2017-02-10)) for(['IBM US Equity'])")

- Retrieve a time series of daily prices for IBM between January 7, 2014 and January 15, 2014:

  =BQL("IBM US Equity", "px_last(dates=range(2014-01-07, 2014-01-15))")

  =BQL.Query("get(px_last(dates=range(2014-01-07, 2014-01-15))) for(['IBM US Equity'])")

FILL WHEN DATA IS MISSING

For BQL() and BQL.Query() formulas in Excel, the *Fill When Data is Missing* (fill) parameter specifies the filler value for non-trading days when downloading a historical time series of data. The parameter gives you the ability to get a continuous stream of data even when there is no data available for certain observation dates.

Parameter Mnemonic: fill

Valid Values:

Value	Description
NA	No Data (**Default**) Shows "NA" in the cell.
PREV	Previous Data Returns the last available value.
NEXT	Next Data Returns the next available value.

Example Formulas:

- Retrieve a time series of IBM's daily prices between January 1, 2014 and December 31, 2014; for non-trading days, "NA" appears:

  =BQL("IBM US Equity", "px_last(dates=range(2014-01-01, 2014-12-31))")

  =BQL.Query("get(px_last(dates=range(2014-01-01, 2014-12-31))) for(['IBM US Equity'])")

- Retrieve a time series of IBM's daily prices between January 1, 2014 and December 31, 2014; for non-trading days, the value from the previous day appears:

  =BQL("IBM US Equity", "px_last(dates=range(2014-01-01, 2014-12-31), fill=PREV)")

  =BQL.Query("get(px_last(dates=range(2014-01-01, 2014-12-31), fill=PREV)) for(['IBM US Equity'])")

PERIODICITY

For BQL() and BQL.Query() formulas in Excel, the *Periodicity* (per) parameter specifies the sampling periodicity of data when downloading a historical times series.

Parameter Mnemonic: per

Valid Values:

Value	Description
D	Daily (**Default**)

Value	Description
W	Weekly
M	Monthly
Q	Quarterly
S	Semi-Annually
Y	Yearly

Example Formulas:

- Retrieve a time series of IBM's weekly prices between January 1, 2014 and December 31, 2014:

  =BQL("IBM US Equity", "px_last(dates=range(2014-01-01, 2014-12-31), per=W)")

  =BQL.Query("get(px_last(dates=range(2014-01-01, 2014-12-31), per=W)) for(['IBM US Equity'])")
- Retrieve a time series of IBM's yearly prices between December 31, 2005 and December 31, 2015:

  =BQL("IBM US Equity", "px_last(dates=range(2005-12-31, 2015-12-31), per=y)")

  =BQL.Query("get(px_last(dates=range(2005-12-31, 2015-12-31), per=y)) for(['IBM US Equity'])")

CONSENSUS STATISTICS

For data items that return company financials, you can retrieve specific consensus or revision statistics, based on the contributions included in the consensus estimate for the data item. For example, you can retrieve the count of all contributors included in the consensus, the highest estimate, or the number of contributors who dropped coverage within a particular time frame.

contributor_count()
contributor_revisions()
contributor_stats()

CONTRIBUTOR_COUNT()

In BQL() and BQL.Query() formulas in Excel, *contributor_count*() allows you to specify a company financials data item and return the number of contributors included in the Bloomberg Consensus who are providing an estimate for that data item. The number of contributors helps you assess whether the consensus estimate is robust.

Signature: contributor_count(*data_item*)

where: *data_item* is a company financials data item (e.g., is_eps or net_income) and its named parameters. Note that data items calculated through BQL expressions (e.g., pe_ratio) are not supported at this time.

Date printed: 09/20/2019

Note: The data item must include the parameter *Period Type* (fa_period_type or fpt) with the parameter value A (Annual), S (Semi-Annual), or Q (Quarterly). For example, "is_eps(fpt=Q)".

Example Formulas:

- Count the number of contributors whose estimates are included in the calculation of the Bloomberg Consensus estimate for earnings per share (EPS) for the current annual fiscal period:

  =BQL("IBM US Equity", "contributor_count(is_eps(adj=Y, fpo='1', fpt=A))")

  =BQL.Query("get(contributor_count(is_eps(adj=Y, fpo='1', fpt=A))) for(['IBM US Equity'])")
- Count the number of contributors whose estimates are included in the calculation of the Bloomberg Consensus estimate for operating income for the current quarter:

  =BQL("IBM US Equity", "contributor_count(is_oper_inc(adj=Y, fpo='1', fpt=Q))")

  =BQL.Query("get(contributor_count(is_oper_inc(adj=Y, fpo='1', fpt=Q))) for(['IBM US Equity'])")

CONTRIBUTOR_REVISIONS()

In BQL() and BQL.Query() formulas in Excel, *contributor_revisions*() allows you to specify a company financials data item and count the number of estimate revisions or coverage changes for all contributors to the Bloomberg Consensus estimate for that data item. For example, you can retrieve the number of upgrades to IBM's net income estimates or the number of brokers who dropped coverage within a specified time frame. Revision statistics provide insight into sentiment around a particular measure.

Signature: contributor_revisions(*data_item*, *revision_type*, *revision_window*)

where:

- *data_item* is a company financials data item (e.g., is_eps or net_income) and its named parameters. Note that data items calculated through BQL expressions (e.g., pe_ratio) are not supported at this time.
- *revision_type* is the type of estimate revision data you want to retrieve.
- *revision_window* is the time frame for counting the revisions.

Note: The data item must include the parameter *Period Type* (fa_period_type or fpt) with the parameter value A (Annual), S (Semi-Annual), or Q (Quarterly). For example, "is_eps(fpt=Q)".

Valid Parameter Values for revision_type:

Value	Description
NUMUP	Number of Upgrades Retrieves the number of upgrade events from contributors included in the consensus estimate during the window.
NUMDN	Number of Downgrades Retrieves the number of downgrade events from contributors included in the consensus estimate during the window.
NUMCONF	Number of Confirmations

Value	Description
	Retrieves the number of confirmation events from contributors included in the consensus estimate during the window.
NUMADD	Number of Additions Retrieves the number of coverage initiation events from contributors included in the consensus estimate during the window.
NUMDROP	Number of Drops Retrieves the number of coverage drop events from contributors included in the consensus estimate during the window.
NUMCHG	Number of Changes Retrieves the number of revision events changing a value from contributors included in the consensus estimate during the window.
NUMUNCHG	Number of No Changes Retrieves the number of days for which each contributor included in the consensus estimate did not change any values during the window.
NETUP	Number of Net Upgrades Retrieves the number of contributors included in the consensus whose estimate at the end of the the window was higher than the estimate at the beginning of the window.
NETDN	Number of Net Downgrades Retrieves the number of contributors included in the consensus whose estimate at the end of the window was lower than the estimate at the beginning of the window.
NETCONF	Number of Net Confirmations Retrieves the number of contributors included in the consensus who confirmed that their estimate at the end of the window was the same as the estimate at the beginning of the window.
NETADD	Number of Net Additions Retrieves the number of contributors included in the consensus at the end of the window who were not covering the security at the beginning of the window.
NETDROP	Number of Net Drops Retrieves the number of contributors included in the consensus at the beginning of the window who were not covering the security at the end of the window.

Value	Description
NETCHG	Number of Net Changes Retrieves the number of contributors included in the consensus whose estimate at the beginning of the window was different from the estimate at the end of the window.
NETUNCHG	Number of Net No Changes Retrieves the number of contributors included in the consensus whose estimate at the beginning of the window was the same as the estimate at the end of the window.

Valid Parameter Values for revision_window:

Value	Description
<Positive integer> + D	Number of Days Allows you to set the time period for the calculation to a specific number of days.
<Positive integer> + W	Number of Weeks Allows you to set the time period for the calculation to a specific number of weeks.

Example Formulas:

- Retrieve the number of upgrades made over the last 13 weeks to estimates for IBM's earnings per share (EPS) for the current annual fiscal period by contributors to the Bloomberg Consensus estimate:

  =BQL("IBM US Equity", "contributor_revisions(is_eps(adj=Y, fpo='1', fpt=A), NETUP, 13W)")

  =BQL.Query("get(contributor_revisions(is_eps(adj=Y, fpo='1', fpt=A), NETUP, '13W')) for(['IBM US Equity'])")
- Retrieve the number of contributors to the Bloomberg Consensus estimate who added coverage of IBM's current quarter EPS over the last ten weeks:

  =BQL("IBM US Equity", "contributor_revisions(is_eps(adj=Y, fpt=Q, fpo='1'), NUMADD, '10W')")

  =BQL.Query("get(contributor_revisions(is_eps(adj=Y, fpt=Q, fpo='1'), NUMADD, '10W')) for(['IBM US Equity'])")
- Retrieve the number of upgrade events in the last eight days for the current quarter's estimates for Toyota's net income, based on all contributors to the Bloomberg Consensus estimate:

  =BQL("TOYOF US Equity", "contributor_revisions(net_income(adj=Y, fpt=Q, fpo='1'), NUMUP, '8D')")

  =BQL.Query("get(contributor_revisions(net_income(adj=Y, fpt=Q, fpo='1'), NUMUP, '8D')) for(['TOYOF US Equity'])")

CONTRIBUTOR_STATS()

In BQL() and BQL.Query() formulas in Excel, *contributor_stats*() allows you to specify a company financials data item and return a statistic based on contributions to the Bloomberg Consensus estimate for that data item. You can retrieve the average, median, maximum, or minimum estimate, or the standard deviation between estimates.

Signature: contributor_stats(*data_item*, *stat_type*) where:

- *data_item* is a company financials data item (e.g., is_eps or net_income) and its named parameters. Note that data items calculated through BQL expressions (e.g., pe_ratio) are not supported at this time.
- *stat_type* is the consensus statistic you want to retrieve.

Note: The data item must include the parameter *Period Type* (fa_period_type or fpt) with the parameter value A (Annual), S (Semi-Annual), or Q (Quarterly). For example, "is_eps(fpt=Q)".

Valid Parameter Values for stat_type:

Value	Description
AVG	Average (**Default**) Retrieves the average of the values from contributors included in the consensus estimate.
MEDIAN	Median Retrieves the median of the values from contributors included in the consensus estimate.
MAX	Maximum Retrieves the highest value among contributors included in the consensus estimate.
MIN	Minimum Retrieves the lowest value among contributors included in the consensus estimate.
STD	Standard Deviation Retrieves the standard deviation of values from contributors included in the consensus estimate.

Example Formulas:

- Retrieve the average of operating income estimates for IBM for the current quarter from all contributors to the Bloomberg Consensus estimate:

  =BQL("IBM US Equity", "contributor_stats(is_oper_inc(adj=Y, fpo='1', fpt=Q), AVG)")

  =BQL.Query("get(contributor_stats(is_oper_inc(adj=Y, fpo='1', fpt=Q), AVG)) for(['IBM US Equity'])")
- Retrieve the median estimate for earnings per share for the current annual period, based on all contributors to the Bloomberg Consensus estimate:

  =BQL("IBM US Equity", "contributor_stats(adj=Y, is_eps(fpo='1', fpt=A), MEDIAN)")

  =BQL.Query("get(contributor_stats(is_eps(adj=Y, fpo='1', fpt=A), MEDIAN)) for(['IBM US Equity'])")
- Retrieve the standard deviation in net income estimates for IBM for the current quarter, based on contributors to the Bloomberg Consensus estimate:

  =BQL("IBM US Equity", "contributor_stats(net_income(adj=Y, fpo='1', fpt=Q), STD)")

```
=BQL.Query("get(contributor_stats(net_income(adj=Y, fpo='1', fpt=Q), STD)) for (['IBM US Equity'])")
```

BQL FOR FUNDS

BQL is an efficient and flexible method for retrieving a wide range of fund data from the Bloomberg database, including screening, risk return metrics, and fund flow data.

KEY BENEFITS

Key benefits of BQL for Funds are:

- Customized risk-return metrics, such as Sharpe Ratio, Sortino Ratio, and Downside Volatility—current, as of a given date, and on a rolling basis
- Filtering of funds based on specified criteria
- Aggregation of fund flows across dates/fund total assets

RESOURCES

The following videos and documents provide additional information about BQL for Funds in Microsoft® Excel.

Type	Title	Description
🖥	*Video Tutorial: Analyze Fund Flows, AUM, Peers, and Risk/Return Metrics using BQL*	A video introduction to BQL, showcasing new functionality released for funds in Excel that allows for custom analysis in seconds.
🗎	*Tutorial: BQL for Funds*	A spreadsheet tutorial that shows how to download funds data to Excel using BQL.
🗎	*Tutorial: Risk Return Metrics*	A spreadsheet tutorial that shows how to download risk return metrics for funds to Excel using BQL. new functionality released for equities in Excel that allows for CUSTOM analysis in seconds, with a focus on equities.
🗎	*Guide: Fund Data Items*	A reference guide of data items and data item values you can use for analyzing fund data and screening for funds with BQL in Excel.
🗎	*Guide: Function Reference*	A reference guide to functions you can use to perform calculations, statistical analysis, and more with BQL.
🗎	*Workflow Solution: Tracking Changes in ETF Flows*	A customizable spreadsheet that lets you monitor changes in ETF flows between asset classes and fund classifications using BQL.

BQL FOR FIXED INCOME

BQL is an efficient and flexible method for retrieving a wide range of fixed income data from the Bloomberg database, including data on bonds, loans, mortgages, munis, and fixed income indices.

KEY BENEFITS

Key benefits of BQL for Fixed Income are:

- Easier, more efficient data retrieval—reference multiple securities and fields in one formula
- Quick retrieval of bond and loan chains for capital structure or debt distribution analysis
- Customized bond screening at the issuer or index level, as well as across the entire Bloomberg fixed income database
- Ability to write a single query to aggregate data on an index, returning information like median credit spread by sector and rating

RESOURCES

The following videos and documents provide additional information about BQL for Fixed Income in Microsoft® Excel.

Type	Title	Description
📄	*Fact Sheet: BQL for Fixed Income*	A fact sheet highlighting capabilities of the Bloomberg Query language for fixed income.
🖥	*Video Tutorial: BQL for Fixed Income (Analysts & PMs)*	A video introduction to BQL and how it can be applied to build customized Bond and Loan screening and index analytics using the Bloomberg Desktop API in Excel. The use cases are intended for analysts and portfolio managers on the buy side.
🖥	*Video Tutorial: BQL for Fixed Income (Sell Side)*	A video introduction to BQL and how it can be applied to build customized Bond and Loan screening and index analytics using the Bloomberg Desktop API in Excel. The use cases are intended for the sell side.
🖥	*Video Tutorial: Mortgage, Preferred, and Muni Bond Screening and Analytics*	A video introduction to BQL and how it can be applied to build customized screening and analytical queries for mortgages, preferreds, and municipals in Excel.
📊	*Tutorial: BQL for Fixed Income*	A spreadsheet tutorial that shows how to download fixed income data to Excel using BQL.
📊	*Tutorial: Issuer and Parent Data*	A spreadsheet tutorial that shows how to download issuer and parent data to Excel using BQL.

Type	Title	Description
📊	*Guide: Function Reference*	A reference guide to functions you can use to perform calculations, statistical analysis, and more with BQL.
📊	*Example: Issuer Curve Analysis*	Example spreadsheet that analyses an issuers Corporate bond curve using BQL in Excel.
📊	*Example: Bond Index Analysis*	Example spreadsheet that performs a customized analysis of a bond index using BQL.
📊	*Example: Bond and Loan Issuance*	Example spreadsheet that analyses loan and bond issuance using BQL in Excel.
📊	*Example: Mortgage Origination Analysis*	Example spreadsheet application that shows how to use BQL for analysis Mortgage Origination.

BQL FOR ECONOMICS

BQL is an efficient and flexible method for retrieving a wide range of economic data from Bloomberg, including national accounts indicators, labor market statistics, and monetary sector data.

KEY BENEFITS

Key benefits of BQL for Economics are:

- Improved discoverability of economic concepts across all countries. In BQL for Economics, countries are now tickers (e.g., "US Country") and economic concepts are fields (e.g., "GDP").
- Aggregation of economic data for a country time series
- Screening of countries that meet specific economic criteria by geographical region (e.g., "Asia")

RESOURCES

The following document and video provide additional information about BQL for Economics in Microsoft® Excel.

Type	Title	Description
📊	*Tutorial: BQL for Economics*	A spreadsheet tutorial that shows how to download economic data to Excel using BQL.
🖥	*Video Tutorial: Quantifying the Economic Trade War*	A video introduction to BQL for Economics that illustrates how you can create country risk scores and quantify exposure to trade war shocks.

BQL FOR PORTFOLIOS

BQL is an efficient and flexible method for retrieving a wide range of data about a portfolio, including portfolio members and positions/weights (depending on the portfolio type).

For example, you can create the following simple query to return the company names of equities in your portfolio (where "U17911388-100" is the unique ID assigned to your portfolio):

```
=BQL.Query("get(name) for(members('U17911388-100', type=PORT))")
```

For more details and examples, see the spreadsheet tutorial below.

RESOURCES

The following document provides additional information about BQL for Portfolios in Microsoft® Excel.

Type	Title	Description
	Tutorial: BQL for Portfolios	A spreadsheet tutorial that shows how to access portfolio data using BQL.

BQL FOR ESG

BQL is an efficient and flexible method for retrieving a wide range of environmental, social, and governance (ESG) data from Bloomberg.

KEY BENEFITS

Key benefits of BQL for Environmental, Social, and Governance (ESG) include:

- Data coverage across Environmental, Social, and Governance categories
- ESG scoring of companies using your own scoring methodology and weights
- Screening of an investment universe based on your ESG screening criteria

RESOURCES

The following document provides information about BQL for ESG in Microsoft® Excel.

Type	Title	Description
	Tutorial: BQL for ESG	A spreadsheet tutorial that shows how to download ESG data to Excel using BQL.

Bloomberg

>> >

BEIJING +86 10 6649 7500	**FRANKFURT** +49 69 9204 1210	**LONDON** +44 20 7330 7500	**NEW YORK** +1 212 318 2000	**SÃO PAULO** +55 11 2395 9000	**SYDNEY** +61 2 9777 8600
DUBAI +971 4 364 1000	**HONG KONG** +852 2977 6000	**MUMBAI** +91 22 6120 3600	**SAN FRANCISCO** +1 415 912 2960	**SINGAPORE** +65 6212 1000	**TOKYO** +81 3 3201 8900

bloomberg.com