

# BQL Quick Start for BQuant

## Bloomberg Query Language (BQL)

Bloomberg Query Language (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 query. 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.

## BQL Package in BQuant

```
# Import package of BQL to connect the database
import bql

# Create the database connection
bq = bql.Service()
```

The key package to use is “bql” and bql.Service() will create the database connection. Database connection will take time and best practice is to keep the global database connection for reuse.

## Two Ways to Query with BQL in BQuant

### Query String

Query strings contains keywords: “let”, “get”, “for”, “with”

```
# Specify the query string
# The query pulling out the 20-day average price for SP500 members
q_str = """
let(#avg_px = avg(px_last(dates=range(-20d, 0d))));

get(#avg_px)

for(members('SPX Index'))

with(currency=USD)
"""

# Execute the query string with database connection
response = bq.execute(q_str)

# combined_df is a utility function to convert response object into panda's DataFrame
data = bql.combined_df(response)
```

Let: defines the customized data items

Get: defines the data items to retrieve

For: defines the universe and any filtering with defined conditions

With: defines the parameters to apply to all data items above if applicable

The query in the example above pulls out the 20-day average price in USD for the members of S&P 500 index.

Database connection executes the query string and returns the response object. The utility function under bql package “combined\_df” then takes the response object and returns the pandas DataFrame.

|                | DATE       | CURRENCY | #avg_px    |
|----------------|------------|----------|------------|
| ID             |            |          |            |
| LYB UN Equity  | 2019-05-10 | USD      | 86.884000  |
| AXP UN Equity  | 2019-05-10 | USD      | 116.696667 |
| VZ UN Equity   | 2019-05-10 | USD      | 56.843333  |
| AVGO UW Equity | 2019-05-10 | USD      | 311.477334 |

The DataFrame returned from BQL query contains the associated columns such as “DATE” and “CURRENCY” to describe the data returned, such as “#avg\_px”.

Users can customize the query with BQL data items such as “PX\_LAST” with its parameters, BQL functions such as “avg” and universe functions such as “members”. To search for the BQL data items, functions and universes, users can use the following documentations and utility functions:

BQL Help Function on the Bloomberg Terminal {BQLX <GO>}

Generic 1st 'ES' Future Index | BQLX | Related Functions Menu | @xPRO Data S

Live Help | Generate PDF | Help on BQL for Excel

Search BQLX Content

What is BQL?  
Getting Started  
Excel Formula Reference  
BQL for Equities  
Key Benefits  
Resources  
BQL for Funds  
Key Benefits  
Resources  
BQL for Fixed Income  
Key Benefits  
Resources  
BQL for Economics  
Key Benefits  
Resources  
BQL for Portfolios  
Resources

### 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.

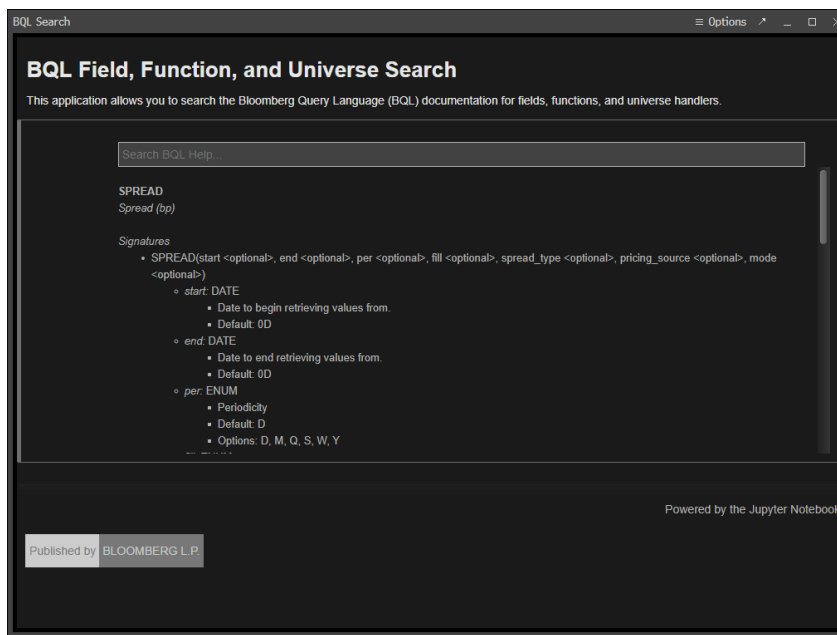
In a single BQL query, you construct a universe of tickers (such as a list of peers) and define the data you want (such as earnings per share). You can apply analysis and calculations on the data directly on the server side, such as calculating an average or multiplying two values. You can also add optional 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:

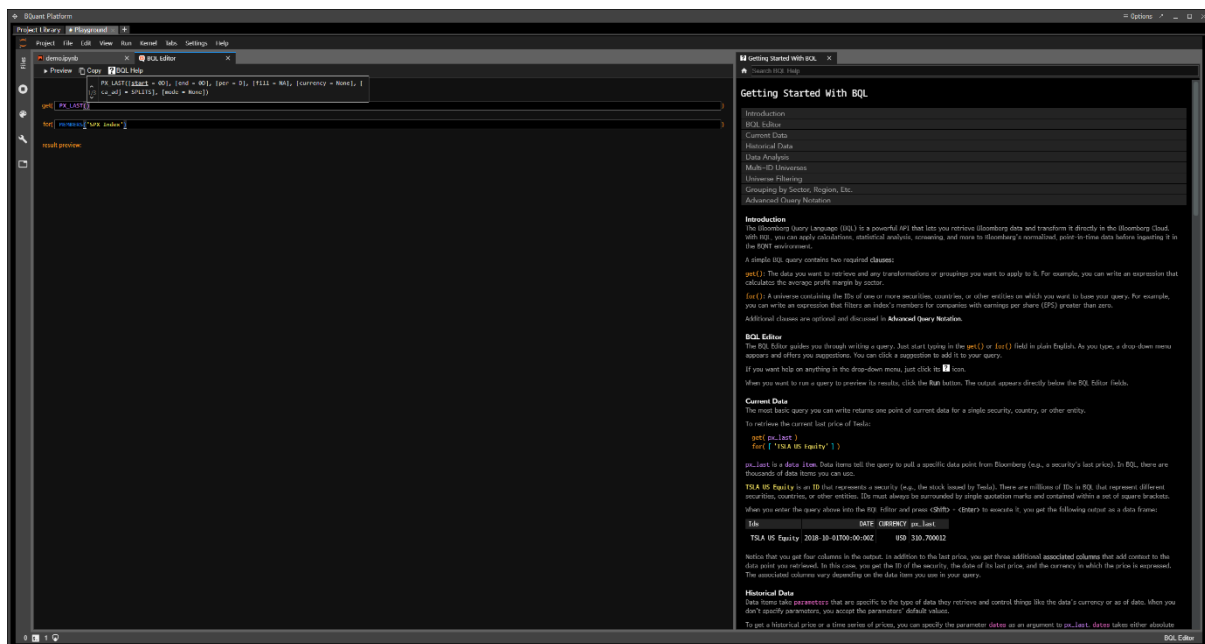
Stats | BQLX <GO> | Last Published 3/22/19 - Version History

Suggested Functions | WB Compare sovereign debt data on one screen | GLCO Track global commodity prices

## BQL Data Item, Functions and Universes Search Tool {BQNT BQLSEARCH <GO>}



## BQL Editor in BQuant Platform with Getting Started with BQL help tool on the side pane



The users also get 24/7 support from Bloomberg Help Desk (double press F1/Help Key).

## Python Object Model

Query string is easier to use when the query is relatively short. When the query gets complicated, we recommend using Python Object Model. The database connection “bql.Service()” provides three key modules (data, func and univ) to construct the query.

```

# The query pulling out the 20-day average price for SP500 members

# Define px as price series for last 20 days
px = bq.data.px_last(dates=bq.func.range('-20d', '0d'))
# Define the avg_px by applying function avg to the time series px
avg_px = bq.func.avg(px)

# Define the universe as index members of SP500
index_memb = bq.univ.members('SPX Index')

# Create the request
request = bql.Request(index_memb, {'avg_px': avg_px}, with_params={'currency': 'USD'})

# Execute the request with database connection
response = bq.execute(request)

# combined_df is a utility function to convert response object into panda's DataFrame
data = bql.combined_df(response)

```

The query string in the first example can be converted into Python Object Model with three key modules under the database connection.

[bq.data](#) provides the data items

[bq.func](#) provides the functions to apply to the data items

[bq.univ](#) provides the universes such as members of index and filtering function ([bq.univ.filter](#))

Instead of execute the query string, in Python Object Model, the database connection executes the request object created with the function “`bql.Request`”, which takes the universe, data items and with parameters.

Python Object Model can be converted into query string by print out the request object.

```

[18]: print(request)

get(AVG(PX_LAST(dates=RANGE(-20D,0D)))) for(MEMBERS('SPX Index')) with(currency=CURRENCY.USD)

```

## Additional BQuant Packages

`bqviz`: easy to use charting library with pre-built charting functions for common charts and interactions. Tutorials are within BQuant Example Projects.

| Project Library +      |     |                                  |             |
|------------------------|-----|----------------------------------|-------------|
| + Create New Project ▾ |     | ⊗ Delete                         | 📄 Duplicate |
|                        |     | ➤ Send                           | 🔄 Refresh   |
| My Projects            | 104 | Name                             |             |
| 👤 Sent Projects        | 826 | 1) Intro to BQuant Visualization |             |
| 👤 Received Projects    | 198 | 2) bqviz Basics                  |             |
| ▼ Example Projects 28  |     | 3) bqviz Plot Customization      |             |
| Getting Started        | 7   | 4) bqplot Basics                 |             |
| Visualization          | 4   |                                  |             |
| App Creation           | 3   |                                  |             |
| Factor Scoring         | 3   |                                  |             |
| Equity Screening       | 1   |                                  |             |
| Quant Lab              | 9   |                                  |             |
| Spotlight Webinars     | 1   |                                  |             |

The screenshot displays the JupyterLab interface with a Python 3 environment. The notebook, titled 'CDE Upload.ipynb', is open and shows three code cells. The first cell, 'Uploading Numeric CDE Data', imports 'Inpd' from 'Inpd' and defines a list of data points. The second cell, 'Uploading Text/Deined List CDE Data', imports 'Inpd' from 'Inpd' and defines a list of data points. The third cell, 'Retrieve CDE Data', imports 'Inpd' from 'Inpd' and defines a list of data points. The output of the third cell is a table with three columns: 'ID', 'CDE', and 'mg'.

| ID           | CDE        | mg  |
|--------------|------------|-----|
| MC PP Equity | 2018-05-15 | 6.0 |
| PP PP Equity | 2018-05-01 | 7.0 |

bqwidgets: UI components specific to Bloomberg such as ticker auto-complete for the application building.

Project Library

CDE Upload

+



+ Create New Project

Delete

Duplicate

Send

Refresh

|   |     | Name                               |
|---|-----|------------------------------------|
| My Projects   | 104 | 1) Factor Scoring App Template     |
|  Sent Projects     | 826 | 2) BQuant App Development Tutorial |
|  Received Projects | 198 | 3) bqwidgets Reference             |
| Example Projects  | 28  |                                    |
| Getting Started   | 7   |                                    |
| Visualization   | 4   |                                    |
| App Creation  | 3   |                                    |
| Factor Scoring  | 3   |                                    |
| Equity Screening  | 1   |                                    |
| Quant Lab   | 9   |                                    |
| Spotlight Webinars  | 1   |                                    |

bqplot: Bloomberg open source advanced charting library

(<https://bqplot.readthedocs.io/en/latest/>)

Most of the common packages are pre-installed:

## Data Analysis

| Library      | Version | Description   |
|--------------|---------|---|
| algopy       | 0.5.3   | Algorithmic differentiation and Taylor polynomial approximations                              |
| bqfunc       | 0.4.5   | Bloomberg library for financial closed-form expressions                                       |
| bqpde1d      | 0.5.1   | Bloomberg library for a one-dimensional generic PDE solver                                    |
| bsplines     | 0.1.1   | Bloomberg library for bsplines, or basis-splines, in regression and interpolation             |
| mpmath       | 0.19    | Library for arbitrary-precision floating-point arithmetic                                     |
| networkx     | 1.11    | Creation, manipulation and study of the structure, dynamics and functions of complex networks |
| nltk         | 3.2.1   | Natural Language Toolkit  |
| numdifftools | 0.9.2   | Solves automatic numerical differentiation problems   |
| numexpr      | 2.6.0   | Fast numerical expression evaluator for NumPy   |
| numpy        | 1.11.1  | Array processing for numbers, strings, records and objects                                    |
| openpyxl     | 2.3.2   | Read/write Excel 2010 xlsx/xlsm files   |
| pandas       | 0.19.0  | Data structures and data analysis   |
| patsy        | 0.4.1   | Describe statistical models and for building design matrices                                  |
| pytables     | 3.2.2   | Manage hierarchical datasets  |
| scikit-image | 0.12.3  | Image processing routines for SciPy   |
| scikit-learn | 0.18.1  | Machine learning library  |
| scipy        | 0.18.1  | Scientific library  |
| sqlalchemy   | 1.0.13  | Python SQL toolkit and object relational mapper   |
| statsmodels  | 0.6.1   | Statistical modeling and econometrics   |
| sympy        | 1.0     | Library for symbolic mathematics  |
| xlrd         | 1.0.0   | Extract data from Microsoft Excel spreadsheet files   |
| xlwt         | 1.1.2   | Create spreadsheet files compatible with MS Excel 97/2000/XP/2003 XLS files                   |

## Visualization

| Library      | Version | Description   |
|--------------|---------|---|
| bokeh        | 0.12.13 | Interactive visualization library for web browsers                          |
| bqplot       | 0.9.1   | Bloomberg library for plotting in the Jupyter notebook                      |
| bqplot-extra | 0.5.0   | Bloomberg library for add-ons to bqplot                                     |
| bqviz        | 0.6.0   | Bloomberg visualization wrapper for bqplot                                  |
| bqwidgets    | 0.10.11 | Bloomberg library for interactive HTML widgets in the Jupyter notebook      |
| cycler       | 0.10.0  | Cycles for Matplotlib   |
| ipyleaflet   | 0.3.0   | Jupyter / Leaflet bridge enabling interactive maps in the notebook          |
| ipywidgets   | 6.0.0   | Interactive HTML widgets for Jupyter Notebook                               |
| matplotlib   | 2.0.0   | 2D plotting library   |
| pillow       | 2.9.0   | Imaging library   |
| seaborn      | 0.7.1   | High-level interface for drawing statistical graphics. Based on matplotlib. |

## Install Package with Specific Version

“%install package version” can be used to install the packages with a specific version.

```
[19]: %install pandas 0.24.2
      Pypi package pandas-0.24.2 activated
```

\* Only the packages with python wheel compiled for Python 3 can be installed and subject to the internal firewall set up to run pip.

## Community Publishing Application and Send Project

For all BQuant platform helps, please use the {HELP BQNT} page on the Bloomberg Terminal.