oduct is in a pre-release state and might change or have limited support. For more information, see the product laun
s (/products/#product-launch-stages).

BigQuery is a versatile tool that solves the problem of storing and querying massive datasets without having to worry about data formats, underlying resources, and other things that distract you from your analysis.

You should review the following resources:

- Learn more about Cloud PowerShell cmdlets (http://googlecloudplatform.github.io/google-cloud-powershell/).
- Understand BigQuery Access Controls (/bigquery/docs/access-control) as some tasks require additional permissions to run.
- Learn more about BigQuery (/bigquery).

To use BigQuery in a Cloud project, first create a `Dataset` using the `New-BqDataset` cmdlet. This will take in basic information and create the resource server-side. Locally, a `Dataset` *reference object* is returned.

To get a reference object for an existing dataset, use `Get-BqDataset`.

This object `$dataset` can be modified and passed into further cmdlets such as `Set-BqDataset` to manipulate cloud resources. This cmdlet also handles adding and removing *labels* with `-SetLabel` and `-ClearLabel`.

Labels are used to tag datasets with keywords and/or values so they can be filtered and searched later. The `Get-BqDataset` cmdlet has a built in `-Filter` flag that allows fine grained control when listing datasets for processing with other cmdlets.

`Datasets` can be deleted by the `Remove-BqDataset` cmdlet. This cmdlet supports `ShouldProcess` (the `-WhatIf` parameter) and will prompt for user confirmation before deleting a non-empty `Dataset`. This safeguard can be bypassed with the `-Force` parameter when scripting.

Each `Dataset` has a number of `Tables` to hold data. `Tables` are created with the `New-BqTable` cmdlet by passing in a *TableId* and the `Dataset` where the table will reside. The `Dataset` can be passed in by object or with the `-DatasetId` parameter. `Get-BqTable` and `Set-BqTable` work the same way as the `Get-` and `Set-` dataset cmdlets above.

`Tables` can be deleted by the `Remove-BqTable` cmdlet. This cmdlet supports *ShouldProcess* (the `-WhatIf` parameter) and will prompt for user confirmation before deleting a `Table` that contains data. This safeguard can be bypassed with the `-Force` parameter.

`Tables` need `Schemas` to describe the format of the data they contain. Schemas are created with the `New-BqSchema` and `Set-BqSchema` cmdlets. `New-BqSchema` can take the formats for rows as parameters directly or as a JSON array of row descriptions. The results of `New-BqSchema` are always passed into `Set-BqSchema` which can either output a `Schema` object or assign the schema to an existing `Table`.

`Schema` objects can be passed as parameters in `Table` creation if they are created ahead of time.

Data is added and removed from `Tables` in *Rows*. These rows are accessible using the `Add-BqTableRow` and `Get-BqTableRow` cmdlets. `Add-BqTableRow` takes CSV, JSON, and AVRO files to import into BigQuery.

There are four types of `Jobs`: Query, Load, Extract, and Copy. *Query* jobs run SQL style queries and output results to tables.

Load jobs import Cloud Storage files into BigQuery.

Extract jobs export BigQuery tables to Cloud Storage.

Copy jobs copy an existing table to another new or existing table.

`Start-BqJob` starts any of these kinds of jobs as an asynchronous operation. Use the `-PollUntilComplete` flag to have the cmdlet block until the job is done. `Receive-BqJob` will return the results of a query job once it is finished. `Get-BqJob` will return a reference object detailing the current state and statistics on the job. `Stop-BqJob` will send a request to the server to stop a certain job, and then returns immediately.

Note on formatting table names within query strings: BigQuery format specifies that table names should be surrounded by backticks, but backticks are also PowerShell's escape operators. Because of this, backticks must be escaped by adding a second backtick. See the *Query* jobs sample code for an example.