GET TO KNOW

AZURE DATA STUDIO

BY MARGARITA NAUMOVA

About the e-book

This e-book introduces Azure Data Studio and helps you discover the main ADS features you can use for your daily job. It starts with installing and using the basic querying features in ADS and goes through more advanced concepts in an easy and exciting way by using examples and visuals. It goes beyond the basics explaining your next steps of developing modules and using notebooks. You will understand how to organize your files in workspace and even customize your ADS environment. The book is a good starting companion of your ADS journey.

About the author

Margarita Naumova is a well-known Bulgarian SQL Expert. Magi has achieved the highest possible SQL Server Technical Certification in the field – Microsoft Certified Master, making her one of the best SQL Server Experts Worldwide, among the top one percent of IT professionals in the World.

She has almost 20 years of SQL Server and BI technologies experience delivering consulting and training in Bulgaria and abroad (UK, Finland, Germany, Sweden, Cyprus, Norway) and is a trusted advisor for major Bulgarian customers in SQL Server Platform Area.

She works in the area of Designing and Architecting SQL Server and BI Platform Solutions, SQL Server Performance Tuning and Optimization, High Availability and Disaster Recovery SQL Server Solutions, and Azure Data Platform solution design.

Magi Naumova is a leader and founder of Bulgarian SQL & BI User Group “Let’s SQL Together!” and is a regular presenter of the largest IT events in Bulgaria and in Europe.

She is a CEO of two companies - Inspirit LTD located in Sofia and Inspir-IT AS located in Oslo. Both are Data platform companies delivering services in Bulgaria and Norway. She is an author and founder of SQL Master Academy – a SQL Training Program that has helped hundreds of specialists to feel knowledgeable in their daily work or to find an inspirational career path in the world of SQL Server.

Table of contents

Introduction to ADS

When it comes to tools it is easy to say that we are in the age of developers' heaven. There are so many tools it is hard to choose the right one to work with. When you work with SQL Server and Azure, the thinks are not different. We have SSMS, VS, Visual Studio Code, ML Studio and Azure Data Studio. It's time to differentiate Azure Data Studio among the others.

Azure Data Studio goes further then all those tools and can be a great choice for those who work with data, develop code in T-SQL but also use Python, Scala and R. It works with many data sources and on different platforms, it comes with Intellisense, code snippets, integration for source control and an integrated terminal.

Azure Data Studio is a cross platform and cross database tool which offers a modern editor experience for managing data across multiple sources with fast intellisense, code snippets, source control integration, and an integrated terminal. Azure Data Studio is engineered with the data platform user in mind, with built-in charting of query result-sets and customizable dashboards. (https://azure.microsoft.com/en-ca/updates/azure-data-studio-is-now-available/)

The Azure Data Studio is not an entirely new tool, it has its predecessor called SQL Operations Studio, which was developed with an idea of having a cross platform tool to work with SQL Server not only under OS Windows like SSMS or VS do, but on Linux and MAC as well. As part of the SQL Server Linux age, starting SQL Server 2017 and above the releasing of the tool back in November 2017, which covers many aspects of this multi-platform work with monthly releases since then.

This was how the tool officially released back in the late 2017. Many features have been added and improved since its first release, but a primary focus of the tool remains the same as a best tools developer can use on Mac and Linux (even Windows) to develop and manage their SQL data stored on premises and in Azure.

You probably have already gotten the main difference this tool has comparing to the one we already know but let me emphasize the most important.

ADS run not only on Windows but currently installs on macOS and Linux. You can choose the platform when you download the ADS from the official download page: https://docs.microsoft.com/en-us/sql/azure-data-studio/download?view=sql-server-ver15

In Azure Data Studio you can connect to multiple data systems, not just SQL Server. These are: All SQL Server data sources on premise and in Azure cloud including Azure Managed Instance, Azure SQL database and Azure Synapse (Azure SQL Datawarehouse), as well as Apache Hadoop HDFS, Apache Spark™, Postgre SQL Servers. You can create more sources, because the tool is extendable, which is yet another big advantage of it.

Development environment includes many options a data professional or a database developer would need in his daily job. Some of them are IntelliSense keyword completion, code snippets, code navigation for creating and deploying T-SQL code, as well as many options for tuning the environment, staring from choosing color schema, going through organizing connections in groups and ending up with workspaces, which are very similar to project experience we have in VS, but is more flexible in including many files located in different places in a single workspace. The editor experience includes working with extended object and code search, replacing all occurrences of a code, and multiple integrated terminals running at the same time, which help us avoid switching in out of the editor each time you want to do some command line work. By using terminals, we can have PowerShell and CMD running, or even an Azure Command-line interface (CLI) shell quick command.

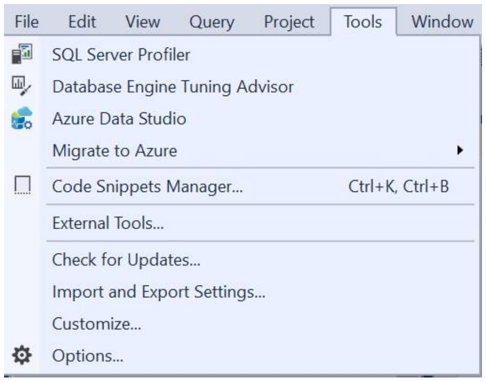
There's support for Git in Azure Data Studio inherited from VS Code. ADS ships with a Git source control manager (SCM). We still need to install Git (version 2.0.0 or later) before these features are available in ADS.

Extensibility is probably one of the great benefits of ADS. You use JSON for tuning configuration settings and customizing the environment including workspaces or adding custom snippets, and creating widgets to show information about your system. But there is more than that, the tool is built with the extensibility in mind and adding extensions comes as a new and exciting task.

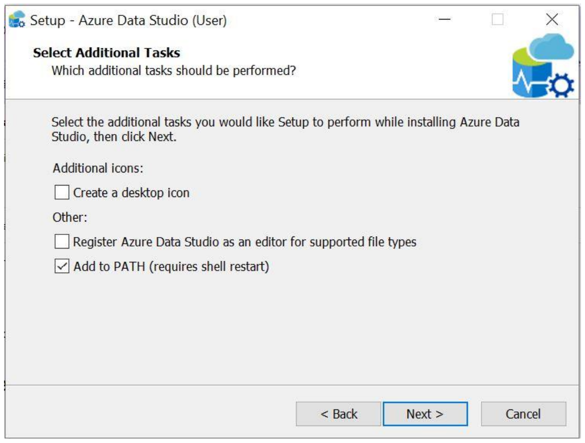
This allows the tool to work as a platform, which we can extend and customize according to our needs. Extensions come as additional installations we perform from inside the tool by downloading them from Git, or we can add extensions from VS Code, we can even develop our own extension. Some extensions examples coming in the product are SQL Server Agent for managing jobs, SQL Server 2019 (Preview) from connecting to Big Data Cluster, Admin Pack for SQL Server containing SQL Server Agent, Import from text files, SQL Server Profiler and SQL Server dacpac, SQL Server Schema Compare and many more.

First steps in ADS – installing and defining connections

Installation of ADS is rather simple. We need to download the latest product version first. We can find the link in SSMS 18. Once the ADS is installed on the machine clicking on the command runs it.



After downloading the product from the official download link, we need to just specify the default folder and the default choice of creating the Azure Data Studio Start menu folder. It is good to perform an installation of both Git and Tortoise Git first so that we have this support later in the tool.



Setup - Azure Data Studio (User)

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Azure Data Studio, then click Next.

Additional icons:

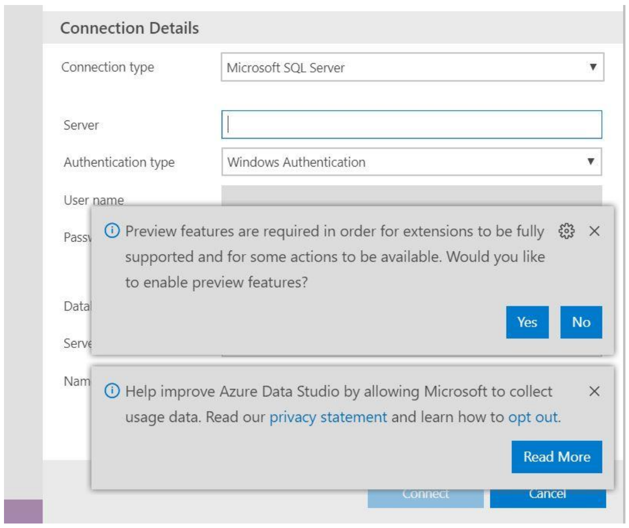
- [ ] Create a desktop icon

Other:

- [ ] Register Azure Data Studio as an editor for supported file types
- [x] Add to PATH (requires shell restart)

The default installation directory is located under the local user's profile. Also, you will need about 400 MB of free space to complete the install.

When we first start the product, we are asked to enable the preview features. It is a very good idea to click **Yes**, because we will have more support on new features and extensions.



Connection Details

Connection type: Microsoft SQL Server

Server:

Authentication type: Windows Authentication

User name

Preview features are required in order for extensions to be fully supported and for some actions to be available. Would you like to enable preview features?

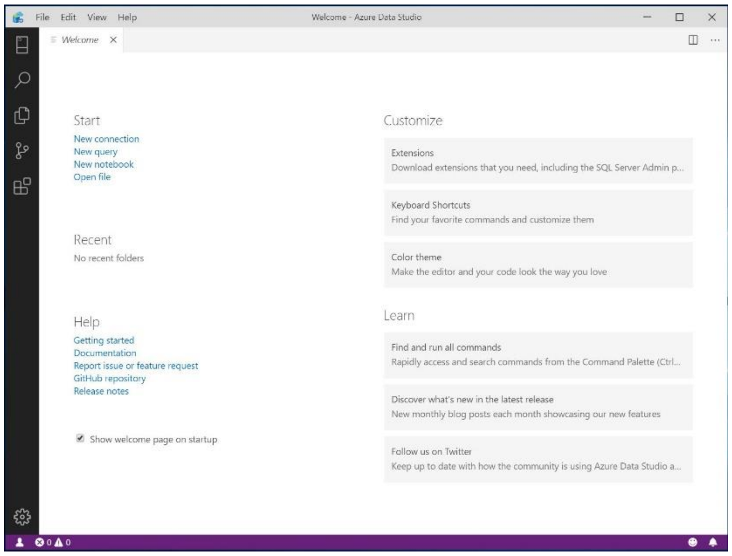
Yes No

Help improve Azure Data Studio by allowing Microsoft to collect usage data. Read our privacy statement and learn how to opt out.

Read More

Connect Cancel

Then we are welcomed in the tool. From its first screen we can start using the ADS by choosing to customize the appearance, learn how to find and run commands, getting help or just starting a new connection or a new query.



File Edit View Help — Welcome - Azure Data Studio

Welcome

Start
- New connection
- New query
- New notebook
- Open file

Recent
No recent folders

Help
- Getting started
- Documentation
- Report issue or feature request
- GitHub repository
- Release notes

☑ Show welcome page on startup

Customize
- Extensions — Download extensions that you need, including the SQL Server Admin p...
- Keyboard Shortcuts — Find your favorite commands and customize them
- Color theme — Make the editor and your code look the way you love

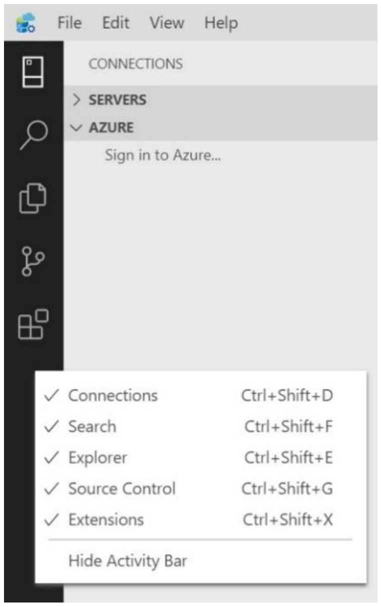
Learn
- Find and run all commands — Rapidly access and search commands from the Command Palette (Ctrl...
- Discover what's new in the latest release — New monthly blog posts each month showcasing our new features
- Follow us on Twitter — Keep up to date with how the community is using Azure Data Studio a...

The left side of the window is the Activity bar, which contains five important features:

- Connections
- Search
- Explorer

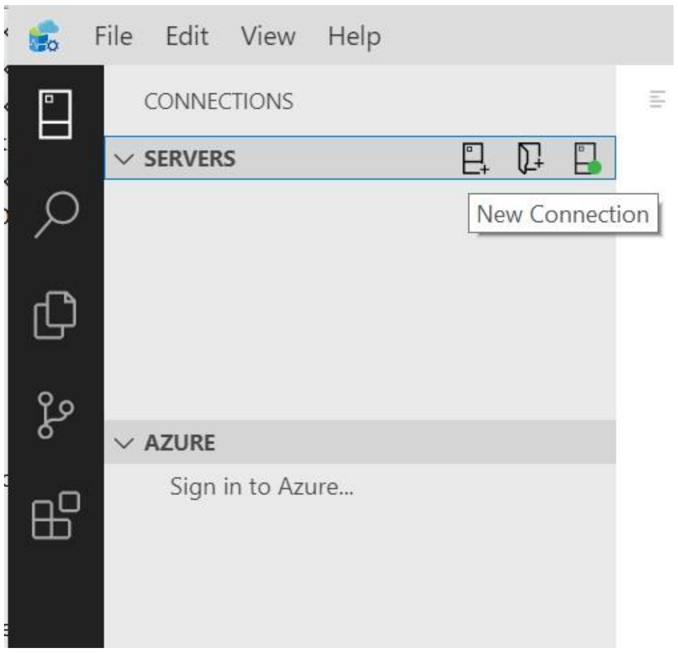
- Source Control
- Extensions

All of them have short keys. Right clicking on the activity bar can show or hide them.

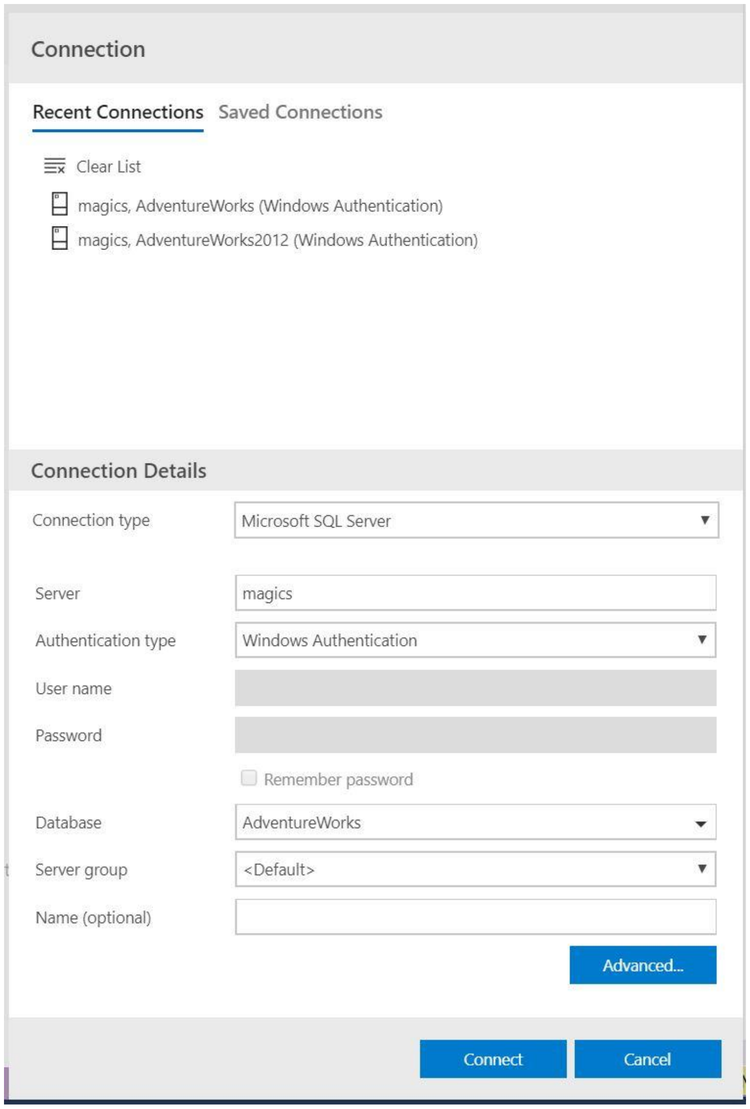


Let us start by discovering connections first and then let’s run a simple query in SQL Server database and in Azure SQL Database.

We can start creating a connection from multiple places in ADS. The most common is by clicking New Connection button next to SERVERs group in Connections menu



Not surprisingly creating a connection requires providing the server name and the credentials. We can optionally select a specific database. In this case the connection will always be associated with this database, and we will have it as a default once we need to create a query.


Connection

Recent Connections | Saved Connections

Clear List

magics, AdventureWorks (Windows Authentication)

magics, AdventureWorks2012 (Windows Authentication)

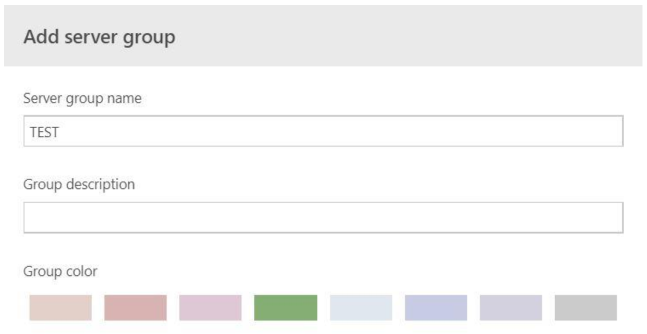
Connection Details

Connection type	Microsoft SQL Server
Server	magics
Authentication type	Windows Authentication
User name	
Password	
	Remember password
Database	AdventureWorks
Server group	<Default>
Name (optional)	

A connection can optionally have a name which will appear in the server list. The tool shows a list of all recent connections performed.

When start working with one server, we define a connection and it appears in the SERVERS group, then we will be able to find it easy and reuse it. But what if we have many servers, databases from different servers, and Azure database connections. It could be useful to be able to group them. This is what Server Groups are in ADS. Let's add two server groups with names TEST and PROD and then move our newly created connection to the TEST server group.

To add a new server group, we use the New Group command button next to SERVERs group in Connections menu, we choose the color and press OK.


Add server group

Server group name

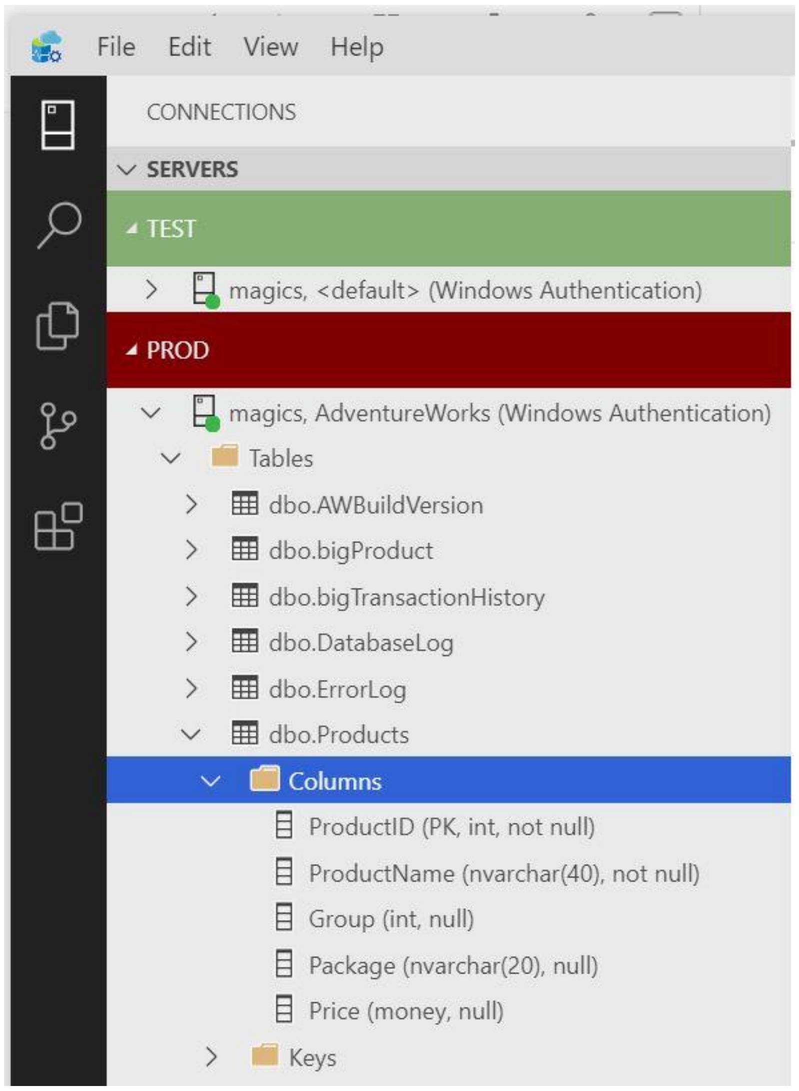
TEST

Group description

Group color

We create the second server Group PROD with a red color. Then in order to move our existing connection to newly created PRDO Group we drag the connection and drop it over the target server group name bar.

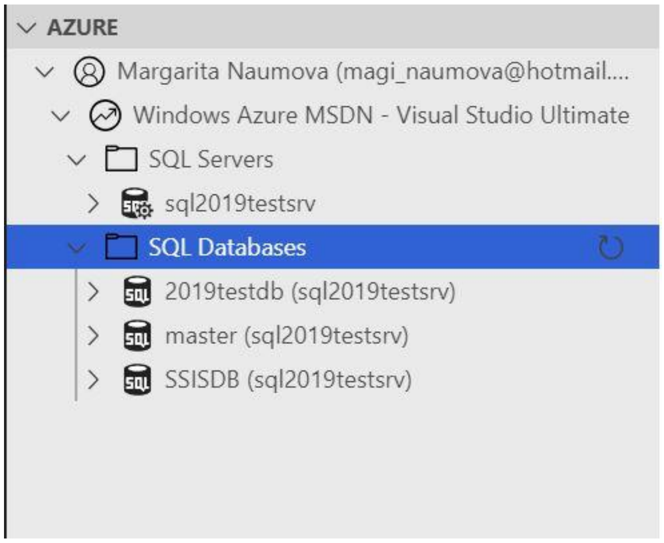
We create another connection in the TEST Server Group by right clicking on the header of the Server group. An alternative is to specify the server group in connection properties. Now we have our connections ordered and in groups. And as expected exploring a connection shows all objects just like SSMS does, so I can see tables and columns.



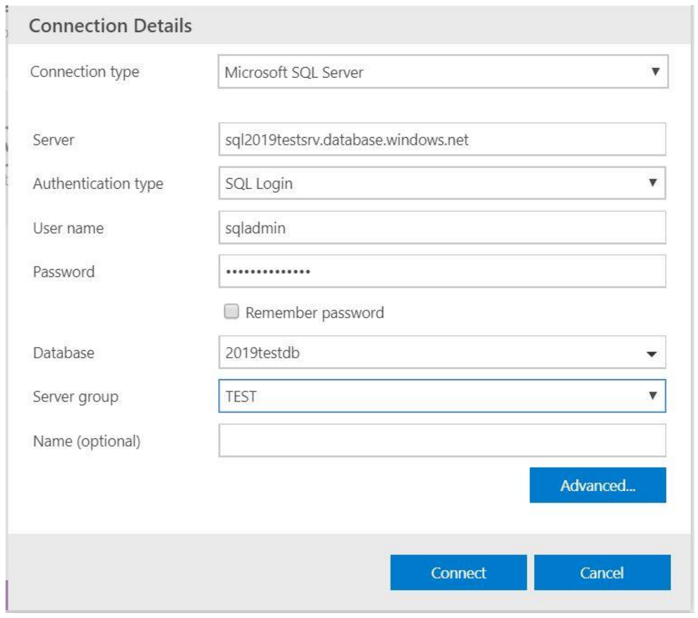
```
CONNECTIONS
SERVERS
  TEST
    magics, <default> (Windows Authentication)
  PROD
    magics, AdventureWorks (Windows Authentication)
      Tables
        dbo.AWBuildVersion
        dbo.bigProduct
        dbo.bigTransactionHistory
        dbo.DatabaseLog
        dbo.ErrorLog
        dbo.Products
          Columns
            ProductID (PK, int, not null)
            ProductName (nvarchar(40), not null)
            Group (int, null)
            Package (nvarchar(20), null)
            Price (money, null)
          Keys
```

What is still missing is Azure connection. The azure connection group allows the ADS to scan the available servers and databases in a subscription. But before doing so a signing is required and a registration of the device, which means I need to authenticate my computer using the device login URL and a user code provided by ADS. Instructions are easy to follow by clicking Copy&Open button.

Once I register my computer, ADS can scan my subscriptions and lists all possible SQL Connections. This doesn’t mean we can connect yet. I still need to be sure I have my IP address added to Azure SQL Server which I need to connect to. Security configuration of Azure connections is outside of ADS scope, but you need to make sure the IP address is added to the SQL Server firewall list, you can do it in azure portal. The first time I click on the Azure SQL Server or any database it contains, the ADS Opens the connection menu for me asking for the account password. It's better to save it for having smooth experience with it later.

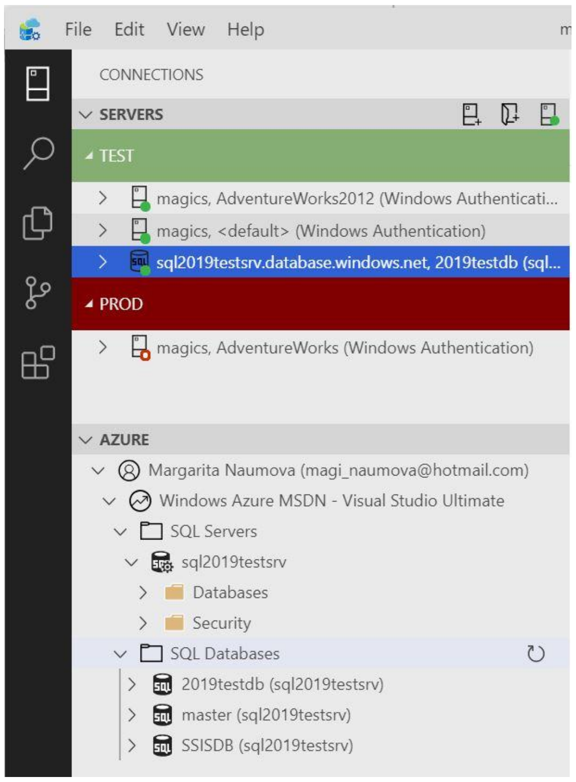

AZURE
Margarita Naumova (magi_naumova@hotmail....
Windows Azure MSDN - Visual Studio Ultimate
SQL Servers
sql2019testsrv
SQL Databases
2019testdb (sql2019testsrv)
master (sql2019testsrv)
SSISDB (sql2019testsrv)

Azure connection can be moved to SERVERS group or any other custom created server group. We click on Add to Servers button on the right of Azure Database Connection name, which opens the connection, then we type the password and we specify the server group which we want to have that connection in.


Connection Details

Connection type	Microsoft SQL Server
Server	sql2019testsrv.database.windows.net
Authentication type	SQL Login
User name	sqladmin
Password	••••••••••••••
	☐ Remember password
Database	2019testdb
Server group	TEST
Name (optional)	

We have all connections grouped and places properly now:

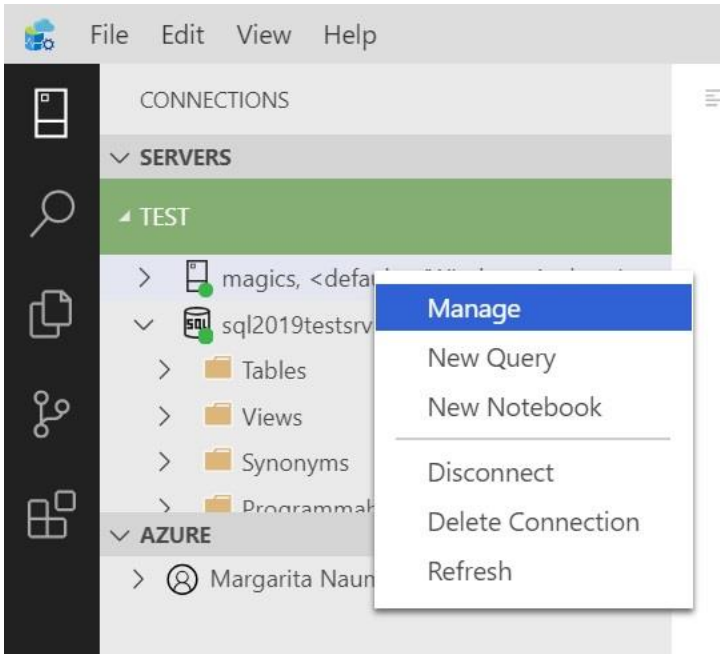


magics, AdventureWorks2012 (Windows Authenticati...
magics, <default> (Windows Authentication)
sql2019testsrv.database.windows.net, 2019testdb (sql...
magics, AdventureWorks (Windows Authentication)
Margarita Naumova (magi_naumova@hotmail.com)
Windows Azure MSDN - Visual Studio Ultimate
SQL Servers
sql2019testsrv
Databases
Security
SQL Databases
2019testdb (sql2019testsrv)
master (sql2019testsrv)
SSISDB (sql2019testsrv)

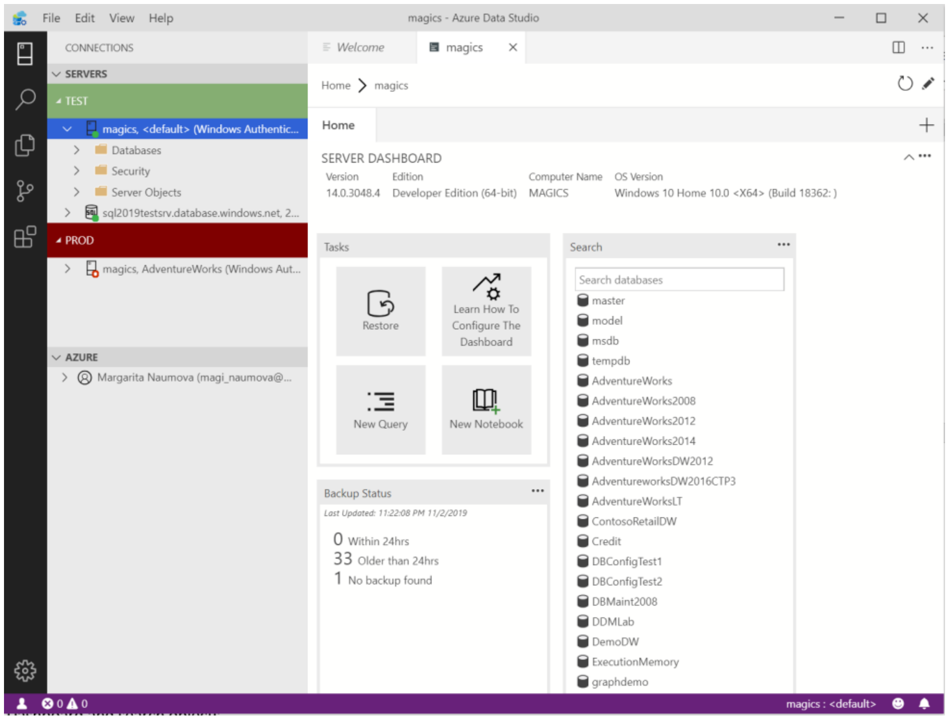
Now its time to see how we can use these connections to query data and write code.

Getting further – writing queries in ADS Query Editor

Let us continue our journey by performing queries and data analysis by using ADS Query Editor. We need to understand the concept of server and database dashboards first. Right clicking on the connection or a database provides us with two commands – Manage and New Query. The Manage command opens a dashboard, either server or database dashboard depending on what we have right clicked.

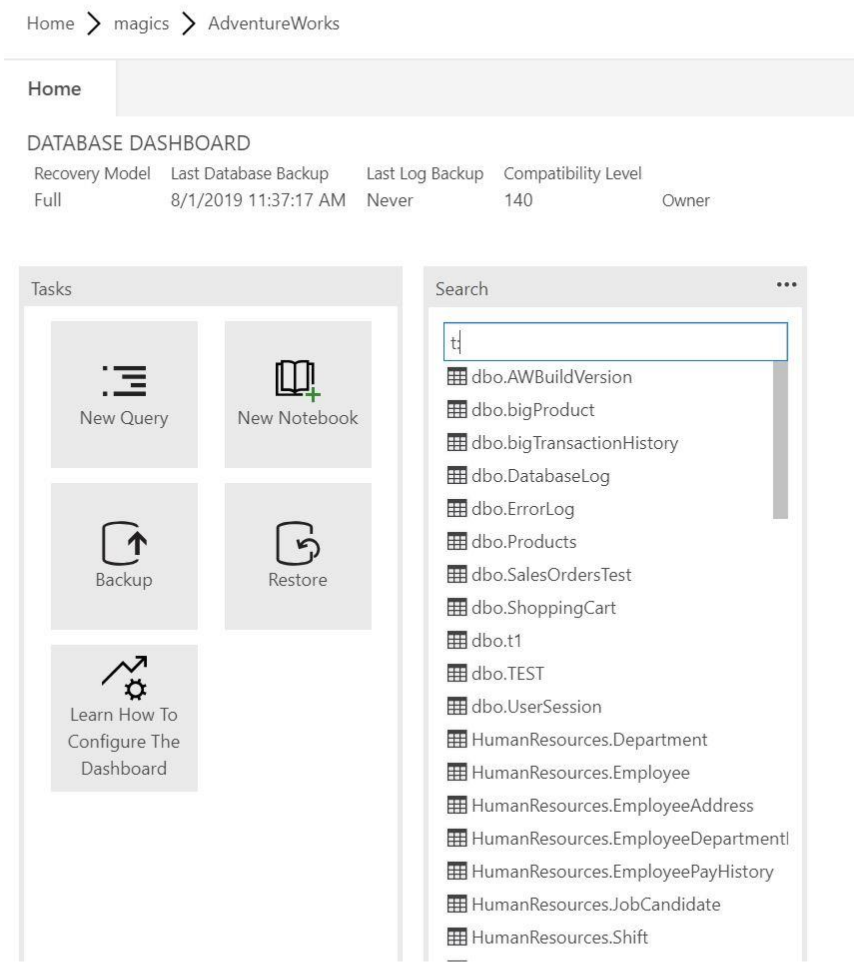


Dashboard is a combination of widgets for a database or a server. The widgets we have in the dashboard by default are Tasks and Search, and a header information.



We can customize the dashboard by including widgets coming from the queries we type or using sample insight widget we are provided with like *the five slowest queries* sample widget or *table space usage* widget. We can also customize SERVER dashboard by adding metrics to our server to measure resource utilization or workload intensity. Let us do the thinks one step at a time.

Lets assume we need to perform analysis. The purpose of our analysis is to analyze product prices so we can decide how to optimize the price or the profit of some of our products. We need to have this analysis as a persistent module in our database, so we can reuse it. Let's find tables we will work with first. Double clicking on the AdventureWorks database in the SEARCH widget will get us to the database dashboard. We can search by object type there or by object name or by both. To search for tables we need to type t:, we type v: for views, sp: for stored procedures and f: for functions.

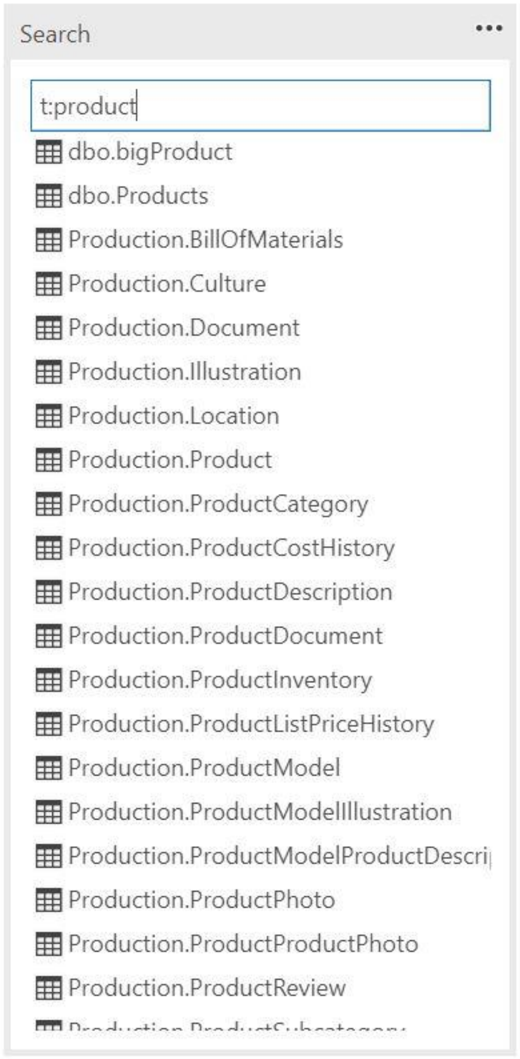

Home > magics > AdventureWorks

DATABASE DASHBOARD

Recovery Model	Last Database Backup	Last Log Backup	Compatibility Level	
Full	8/1/2019 11:37:17 AM	Never	140	Owner

dbo.AWBuildVersion
dbo.bigProduct
dbo.bigTransactionHistory
dbo.DatabaseLog
dbo.ErrorLog
dbo.Products
dbo.SalesOrdersTest
dbo.ShoppingCart
dbo.t1
dbo.TEST
dbo.UserSession
HumanResources.Department
HumanResources.Employee
HumanResources.EmployeeAddress
HumanResources.EmployeeDepartmentI
HumanResources.EmployeePayHistory
HumanResources.JobCandidate
HumanResources.Shift

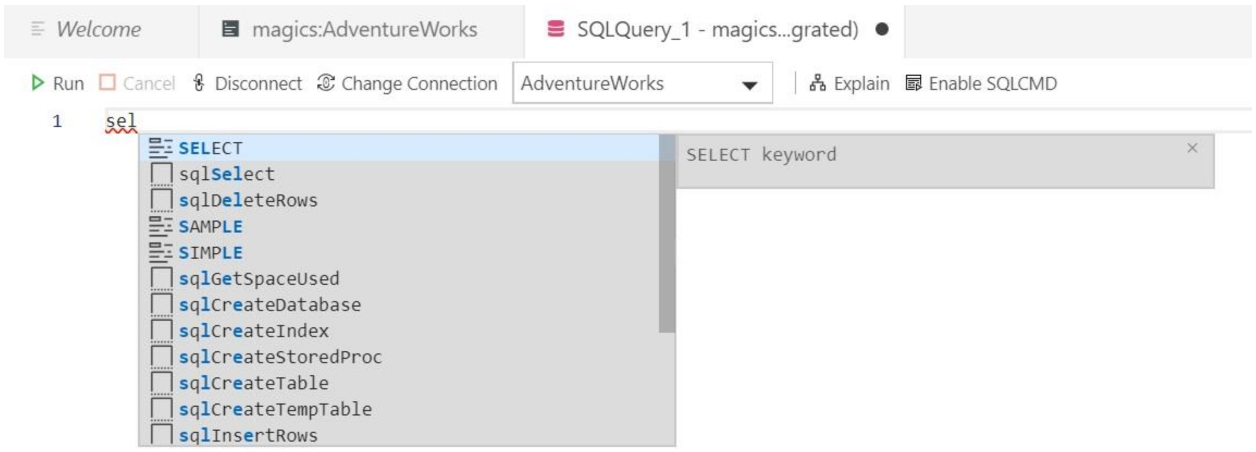
The database has many objects, so we need to filter our search by name. Typing a name of the object, we are looking for immediately after the object type will get us there. Now we got all the tables which contain product in their name or schema:



dbo.bigProduct
dbo.Products
Production.BillOfMaterials
Production.Culture
Production.Document
Production.Illustration
Production.Location
Production.Product
Production.ProductCategory
Production.ProductCostHistory
Production.ProductDescription
Production.ProductDocument
Production.ProductInventory
Production.ProductListPriceHistory
Production.ProductModel
Production.ProductModelIllustration
Production.ProductModelProductDescri
Production.ProductPhoto
Production.ProductProductPhoto
Production.ProductReview

Search widget provide very good experience when we need to find an object fast. Table Production.Product looks like the one we need for our analysis, so we use it. The first step is to build the query and then based on the result we will decide to enclose it into a stored procedure in our database.

By selecting New Query or pressing Ctrl+N we get our T-SQL Query editor. Starting code typing already feels the difference in the intelisence.



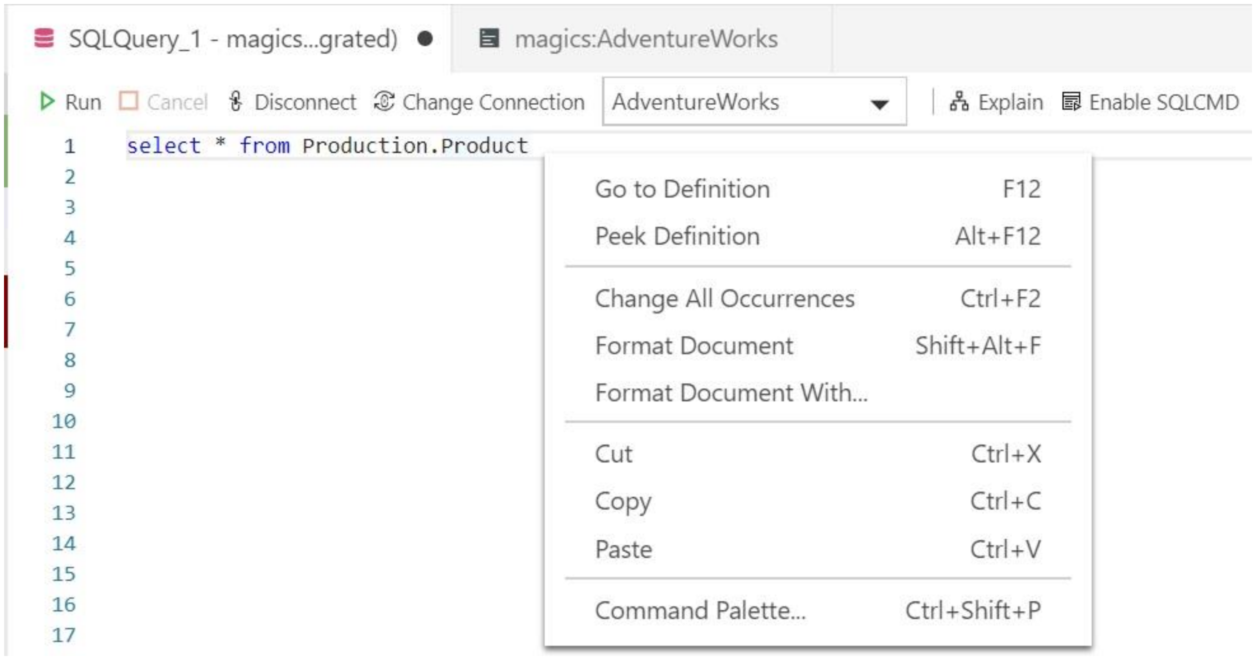
```
sel
```

```
SELECT
sqlSelect
sqlDeleteRows
SAMPLE
SIMPLE
sqlGetSpaceUsed
sqlCreateDatabase
sqlCreateIndex
sqlCreateStoredProc
sqlCreateTable
sqlCreateTempTable
sqlInsertRows
```

SELECT keyword

It lists key words we can choose from, recognizes schemas, object names and provides a great help on typing, we can build a simple query to get all the data from Production.Product table.

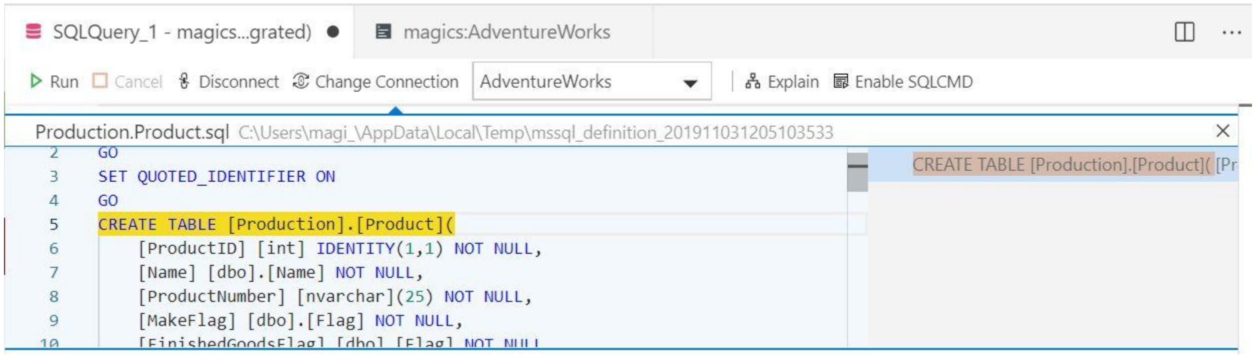
Select * is probably not the best query we can run, so we need to be sure the table doesn't have hundred of columns and to specify just what we are looking for – price, quantity or any other columns which will be of potential help in ours analysis.



```
select * from Production.Product
```

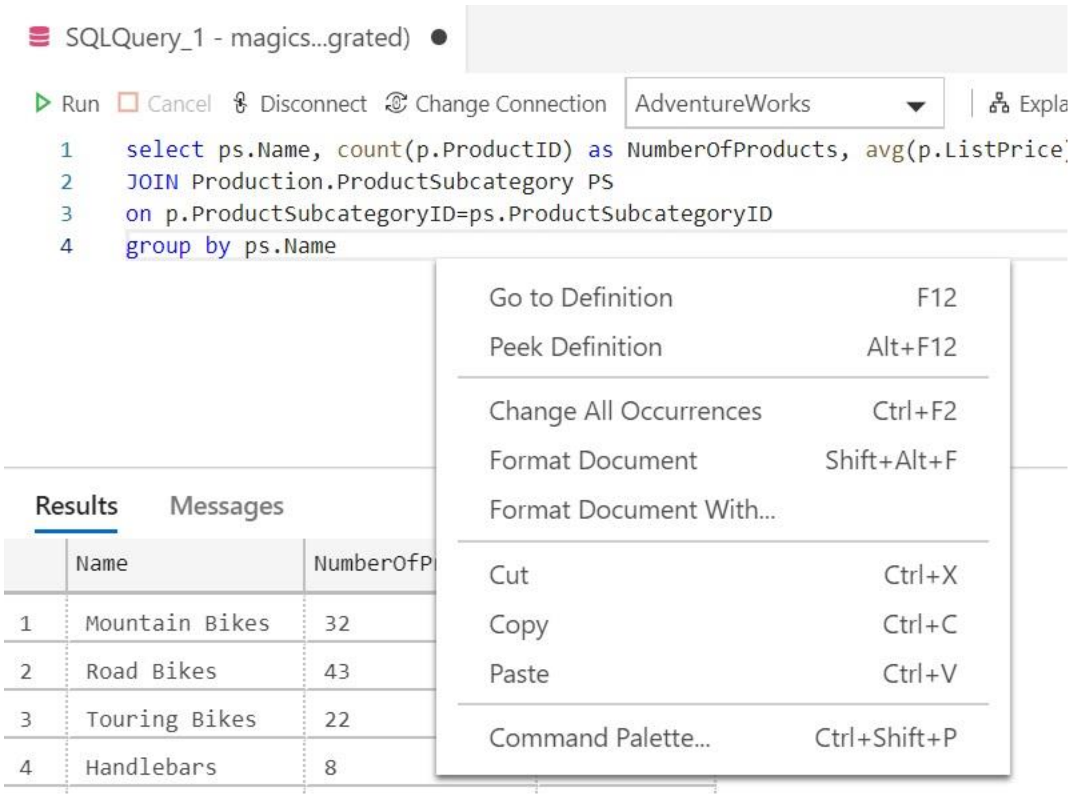
We have a very good help in exploring object definitions. GO TO Definition (or pressing F12 key) will open the CREATE table script in a new query window without using the same connection meaning it is in a disconnected state.

Peek definition on the other hand will provide us with an in-query editor window of the create table command, from which we can explore the list of columns, and once we finish our check we can close the window and we can continue to work in our query editor. I would say this is very convenient and give a great experience in constructing queries, and we have short key combination for every command.


SQLQuery_1 - magics...grated)
magics:AdventureWorks
Run
Cancel
Disconnect
Change Connection
AdventureWorks
Explain
Enable SQLCMD
Production.Product.sql C:\Users\magi_\AppData\Local\Temp\mssql_definition_201911031205103533
CREATE TABLE [Production].[Product]([Pr

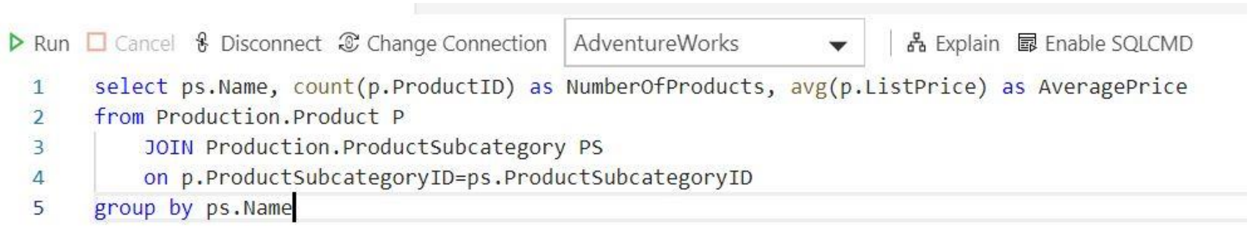
```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [Production].[Product](
    [ProductID] [int] IDENTITY(1,1) NOT NULL,
    [Name] [dbo].[Name] NOT NULL,
    [ProductNumber] [nvarchar](25) NOT NULL,
    [MakeFlag] [dbo].[Flag] NOT NULL,
    [FinishedGoodsFlag] [dbo].[Flag] NOT NULL
```

We can safely run our query and seems we have columns we can use for our analysis. Let us run the query to explore them. We will continue with constructing the query by selecting just Subcategory, List Price and quantity. We will make a join with Production.SubCategories table to include Subcategory name. Our final query looks ready to run. We can press Shift+Alt+F to apply code formatting or we can choose Format Document from the context menu:


SQLQuery_1 - magics...grated)

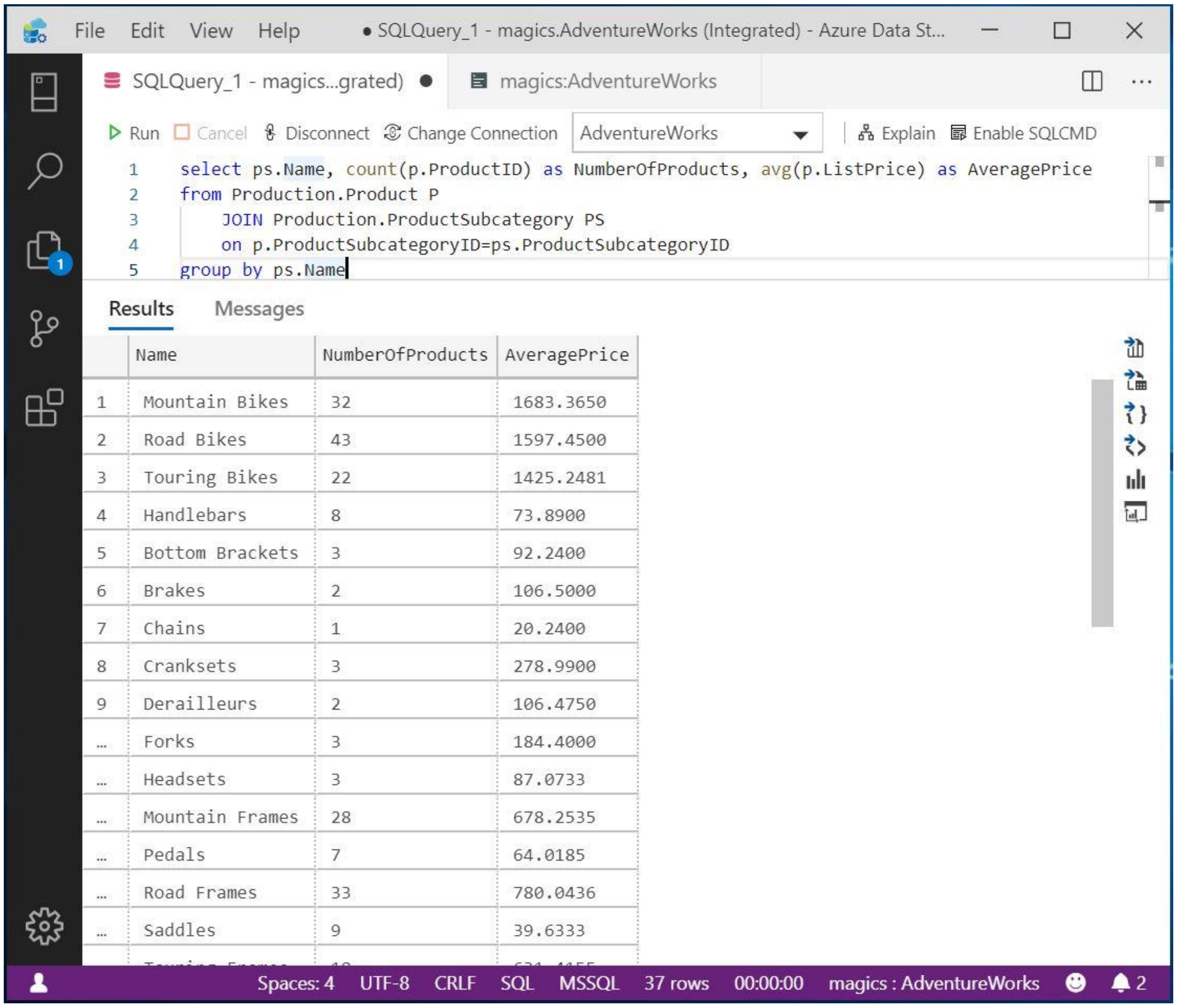
```
select ps.Name, count(p.ProductID) as NumberOfProducts, avg(p.ListPrice)
JOIN Production.ProductSubcategory PS
on p.ProductSubcategoryID=ps.ProductSubcategoryID
group by ps.Name
```

	Name	NumberOfP
1	Mountain Bikes	32
2	Road Bikes	43
3	Touring Bikes	22
4	Handlebars	8



```
select ps.Name, count(p.ProductID) as NumberOfProducts, avg(p.ListPrice) as AveragePrice
from Production.Product P
    JOIN Production.ProductSubcategory PS
    on p.ProductSubcategoryID=ps.ProductSubcategoryID
group by ps.Name
```

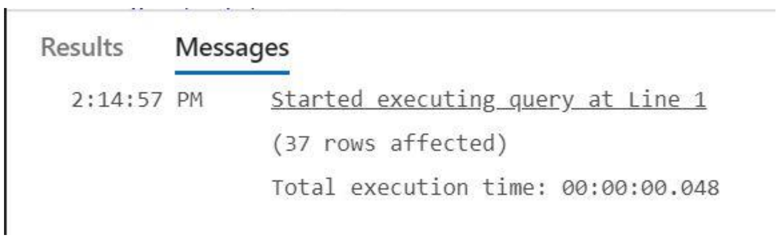
Let us explore the results to find out the data and if the query needs some corrections. The results pane does not have anything surprising at a first look, we have data in grid with execution time statistics. Pretty much like SSMS experience.



File Edit View Help
● SQLQuery_1 - magics.AdventureWorks (Integrated) - Azure Data St...
SQLQuery_1 - magics...grated)
magics:AdventureWorks
Run Cancel Disconnect Change Connection AdventureWorks Explain Enable SQLCMD
Results Messages
Spaces: 4 UTF-8 CRLF SQL MSSQL 37 rows 00:00:00 magics : AdventureWorks

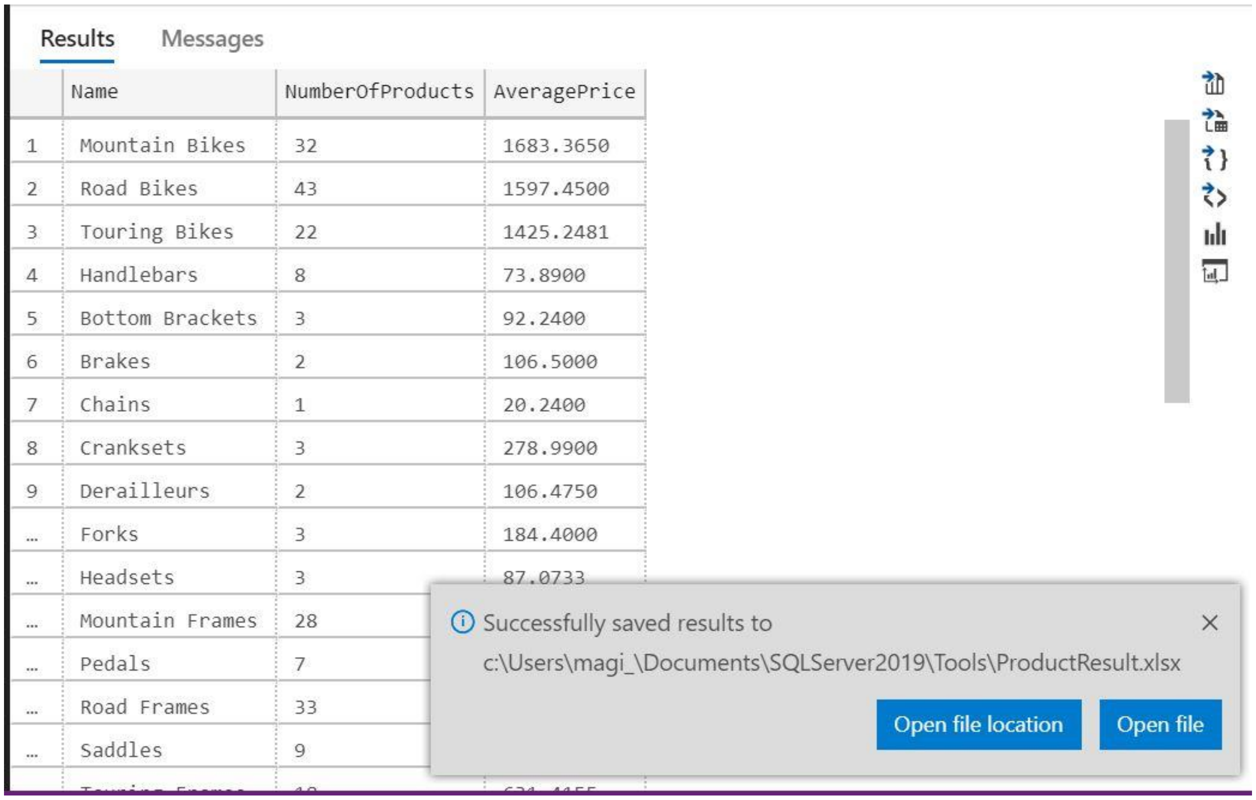
```sql
select ps.Name, count(p.ProductID) as NumberOfProducts, avg(p.ListPrice) as AveragePrice
from Production.Product P
    JOIN Production.ProductSubcategory PS
    on p.ProductSubcategoryID=ps.ProductSubcategoryID
group by ps.Name
```

	Name	NumberOfProducts	AveragePrice
1	Mountain Bikes	32	1683.3650
2	Road Bikes	43	1597.4500
3	Touring Bikes	22	1425.2481
4	Handlebars	8	73.8900
5	Bottom Brackets	3	92.2400
6	Brakes	2	106.5000
7	Chains	1	20.2400
8	Cranksets	3	278.9900
9	Derailleurs	2	106.4750
...	Forks	3	184.4000
...	Headsets	3	87.0733
...	Mountain Frames	28	678.2535
...	Pedals	7	64.0185
...	Road Frames	33	780.0436
...	Saddles	9	39.6333



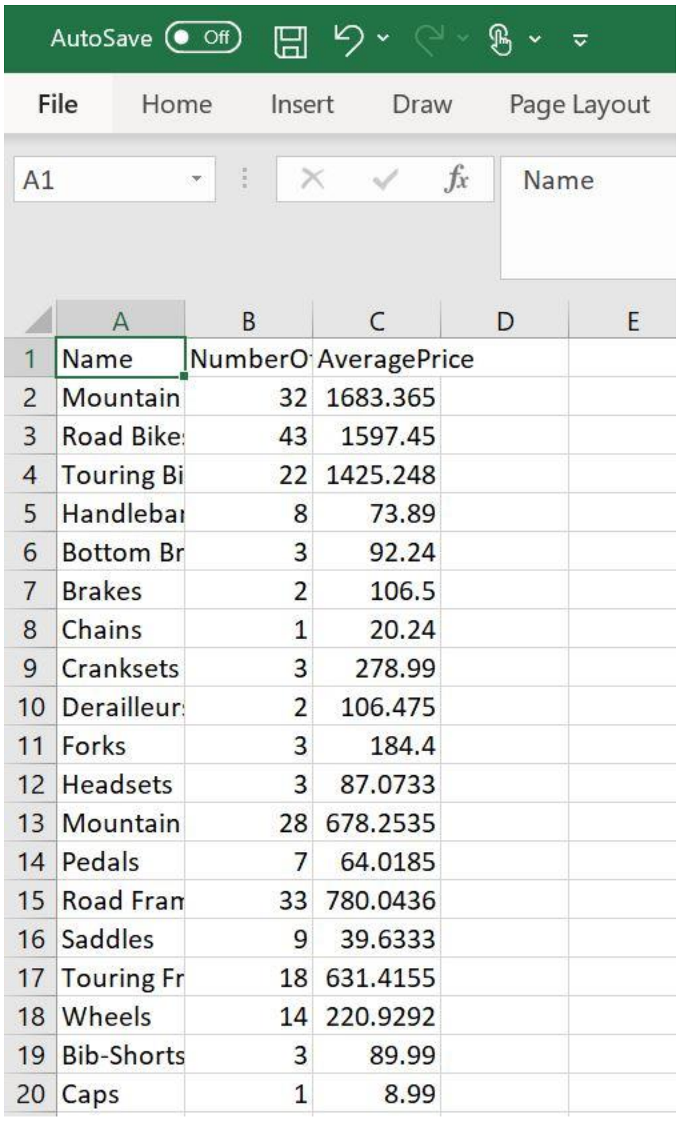
```
2:14:57 PM    Started executing query at Line 1
              (37 rows affected)
              Total execution time: 00:00:00.048
```

BUT let’s look at the buttons on the right side of the results pane. We have several options for exporting and working with the result set. We have Export to excel, CSV and JSON, everyone of each creates a corresponding file type file and for xlsx and csv it includes column headers in file! Something we don't have in SSMS.



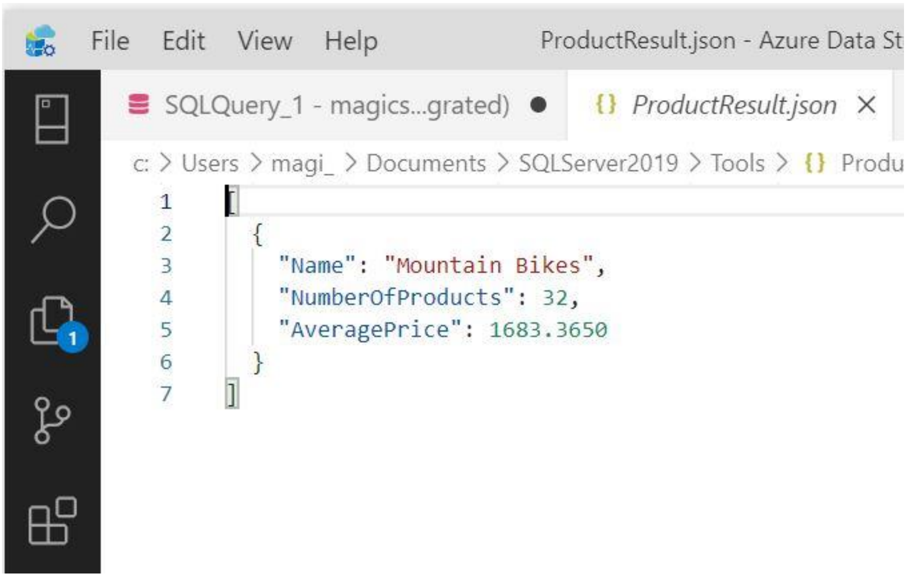
	Name	NumberOfProducts	AveragePrice
1	Mountain Bikes	32	1683.3650
2	Road Bikes	43	1597.4500
3	Touring Bikes	22	1425.2481
4	Handlebars	8	73.8900
5	Bottom Brackets	3	92.2400
6	Brakes	2	106.5000
7	Chains	1	20.2400
8	Cranksets	3	278.9900
9	Derailleurs	2	106.4750
...	Forks	3	184.4000
...	Headsets	3	87.0733
...	Mountain Frames	28	
...	Pedals	7	
...	Road Frames	33	
...	Saddles	9	

Successfully saved results to c:\Users\magi_\Documents\SQLServer2019\Tools\ProductResult.xlsx



	A	B	C	D	E
1	Name	NumberO	AveragePrice		
2	Mountain	32	1683.365		
3	Road Bike:	43	1597.45		
4	Touring Bi	22	1425.248		
5	Handlebar	8	73.89		
6	Bottom Br	3	92.24		
7	Brakes	2	106.5		
8	Chains	1	20.24		
9	Cranksets	3	278.99		
10	Derailleur:	2	106.475		
11	Forks	3	184.4		
12	Headsets	3	87.0733		
13	Mountain	28	678.2535		
14	Pedals	7	64.0185		
15	Road Fram	33	780.0436		
16	Saddles	9	39.6333		
17	Touring Fr	18	631.4155		
18	Wheels	14	220.9292		
19	Bib-Shorts	3	89.99		
20	Caps	1	8.99		

We can select just one row by clicking on the row header and we can export it to JSON, this will help a lot in having a correct JSON template of just one row we can use in developing further modules.


File Edit View Help ProductResult.json - Azure Data St

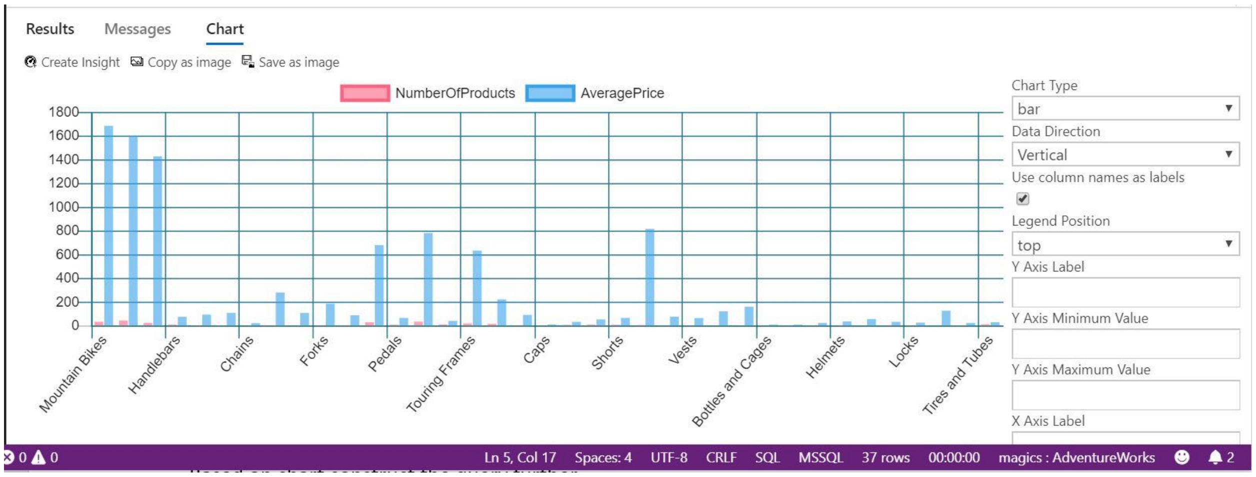
SQLQuery_1 - magics...grated) ● {} ProductResult.json ×

c: > Users > magi_ > Documents > SQLServer2019 > Tools > {} Produ

```
[
  {
    "Name": "Mountain Bikes",
    "NumberOfProducts": 32,
    "AveragePrice": 1683.3650
  }
]
```

We have chart button as well; it provides us with a valuable option of exploring the data by using simple data visualizations. Clicking on chart adds a tab in the results pane showing the result in a visual.

We tune the data direction to vertical and we got a chart showing our number of products and average of their price by categories. Easy and fast!

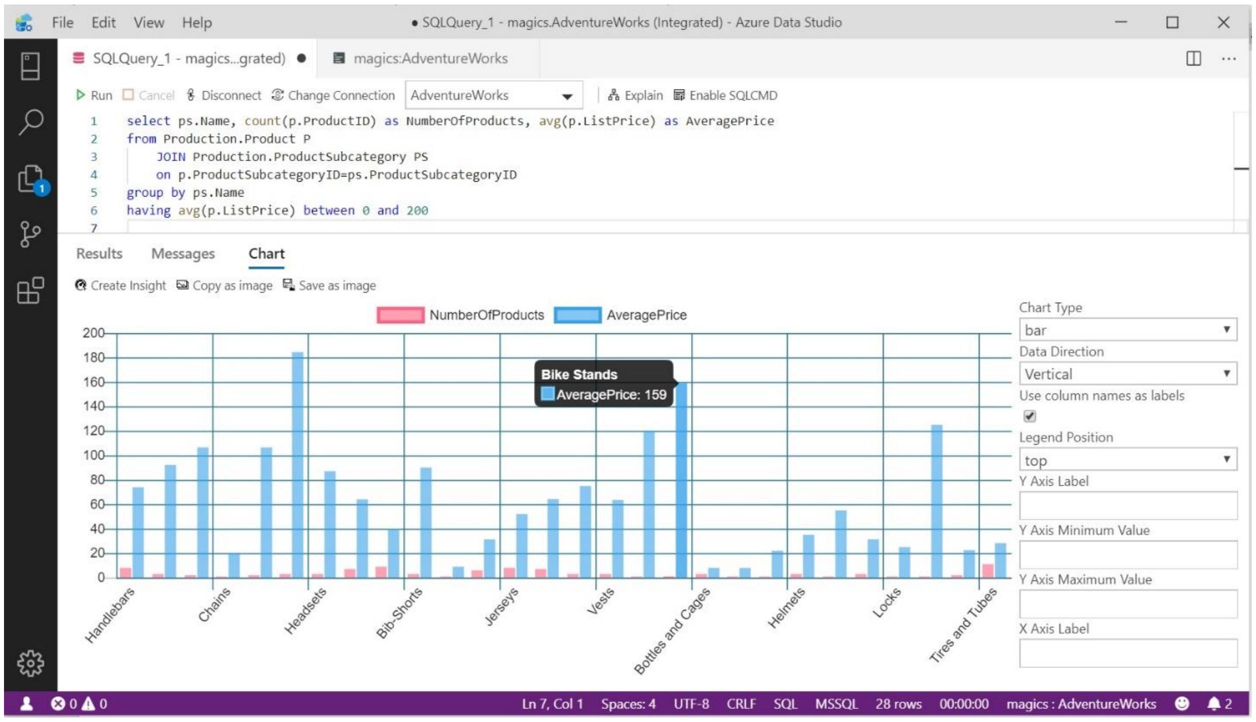

Results Messages Chart

Create Insight Copy as image Save as image

Chart Type: bar
Data Direction: Vertical
Use column names as labels: ✓
Legend Position: top
Y Axis Label
Y Axis Minimum Value
Y Axis Maximum Value
X Axis Label

⊗ 0 ⚠ 0 Ln 5, Col 17 Spaces: 4 UTF-8 CRLF SQL MSSQL 37 rows 00:00:00 magics : AdventureWorks 🔔 2

Discovering the data shown in the chart lead us to probably defining tree groups of average pricing levels, which is worthy to perform analysis on. These are the most expensive categories having average price level between 800 and 1800, middle range price level between 200 and 800, and low-level price range of categories with product price levels up to 200. We can test this and how the data look like by putting additional HAVING Clause in our query.



```
select ps.Name, count(p.ProductID) as NumberOfProducts, avg(p.ListPrice) as AveragePrice
from Production.Product P
    JOIN Production.ProductSubcategory PS
    on p.ProductSubcategoryID=ps.ProductSubcategoryID
group by ps.Name
having avg(p.ListPrice) between 0 and 200
```

Chart Type: bar
Data Direction: Vertical
Use column names as labels: ✓
Legend Position: top
Y Axis Label
Y Axis Minimum Value
Y Axis Maximum Value
X Axis Label

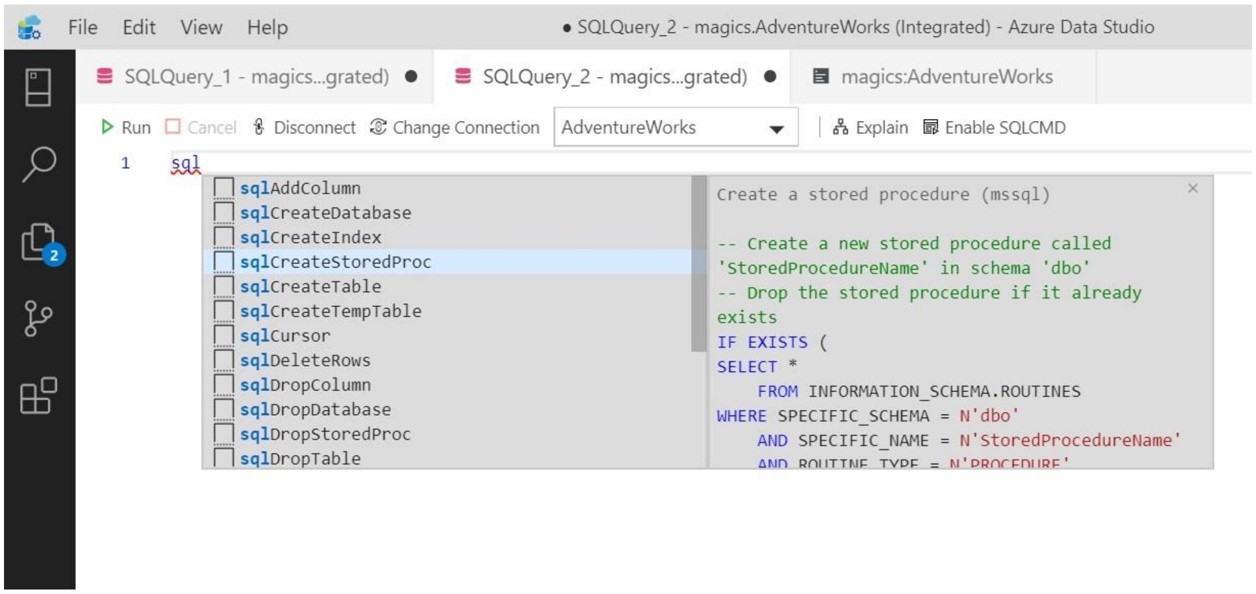
The chart shows the quantity and average price of products by categories fitting in the low-level price range. It gives interesting view of the data and probably a starting point of discovering is it a good idea to increase the quantity of some products fitting in this range, this could lead to increasing the profit. We can Copy the image or even save it a png image file by clicking on Save as image button.

We are ready to include our query into a stored procedure and provide it as a module a user can use for source of data analysis. Query Editor and its result pane helped us to get a proper query construct fast and easy.

Mastering Development – using snippets and notebooks

We will use code snippets to create a stored procedure, this will help us re-use our fancy query. Code snippets are block of reusable code which we can insert or load in our code window by using a command or hot key combination. ADS makes no exception in providing this experience to us. To load a snippet, we open a new query window (Ctrl+N) and type sql. This opens a list which we can navigate in and locate a snippet we want to use.

We Select the item to load, and press ENTER, and the code snippet loads into the window:

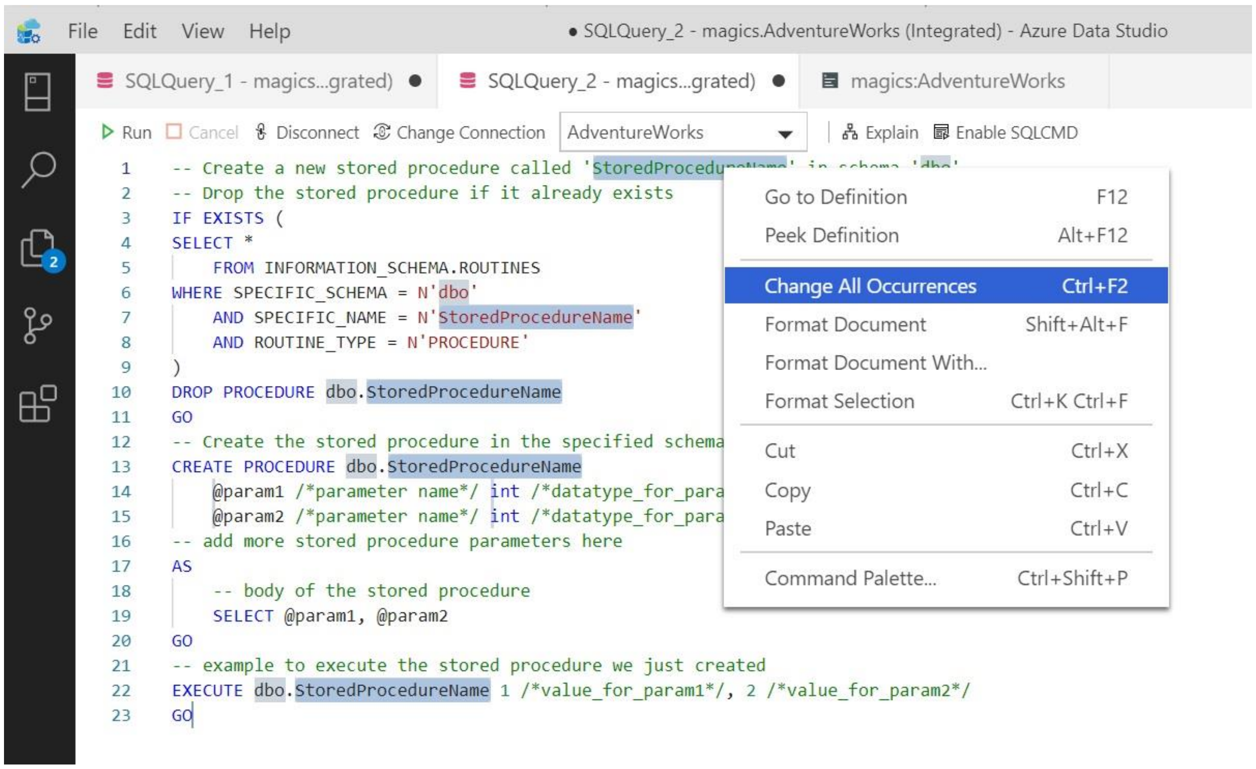

File
Edit
View
Help
● SQLQuery_2 - magics.AdventureWorks (Integrated) - Azure Data Studio
SQLQuery_1 - magics...grated)
SQLQuery_2 - magics...grated)
magics:AdventureWorks
Run
Cancel
Disconnect
Change Connection
AdventureWorks
Explain
Enable SQLCMD

```
sql
```

```
sqlAddColumn
sqlCreateDatabase
sqlCreateIndex
sqlCreateStoredProc
sqlCreateTable
sqlCreateTempTable
sqlCursor
sqlDeleteRows
sqlDropColumn
sqlDropDatabase
sqlDropStoredProc
sqlDropTable
```

```
Create a stored procedure (mssql)

-- Create a new stored procedure called
'StoredProcedureName' in schema 'dbo'
-- Drop the stored procedure if it already
exists
IF EXISTS (
SELECT *
    FROM INFORMATION_SCHEMA.ROUTINES
WHERE SPECIFIC_SCHEMA = N'dbo'
    AND SPECIFIC_NAME = N'StoredProcedureName'
    AND ROUTINE_TYPE = N'PROCEDURE'
```

We choose sqlCreateStoredProc and the snippet inserts into our query window.

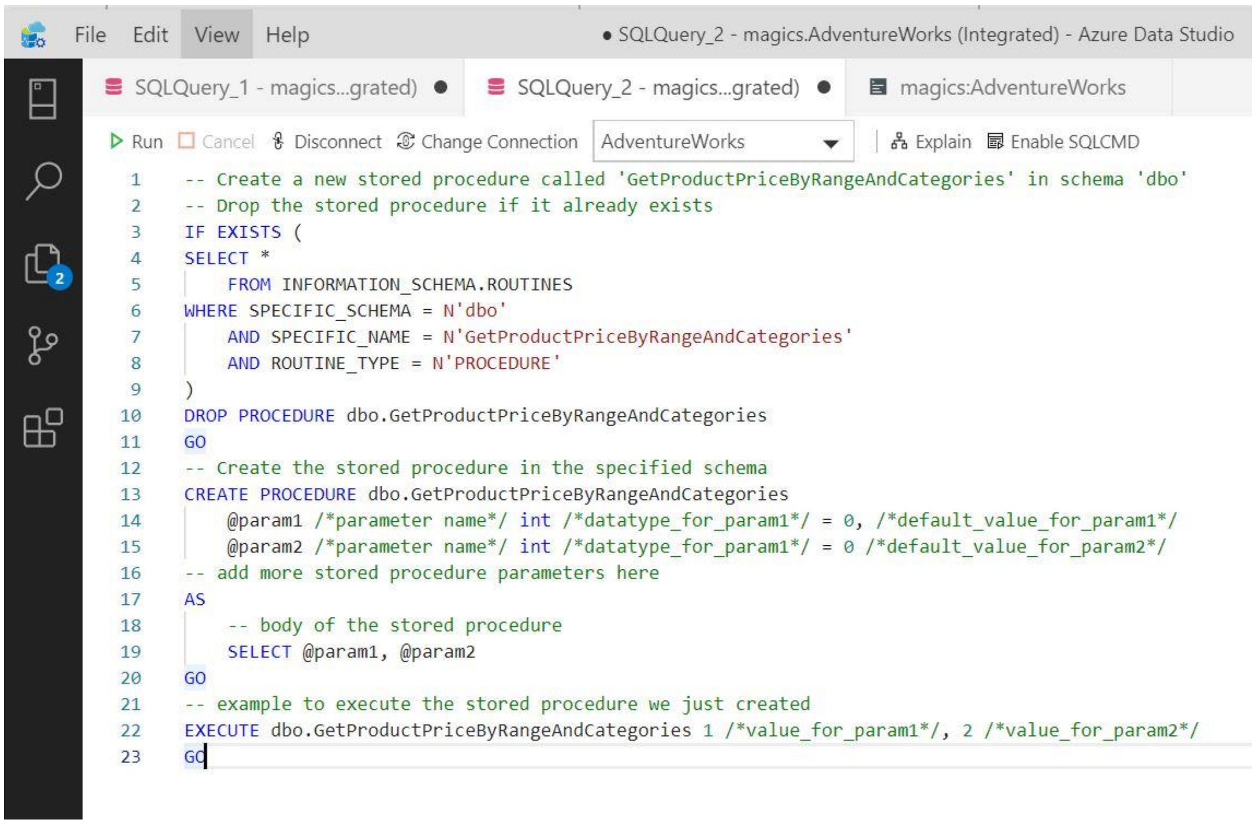

File
Edit
View
Help
● SQLQuery_2 - magics.AdventureWorks (Integrated) - Azure Data Studio
SQLQuery_1 - magics...grated) ●
SQLQuery_2 - magics...grated) ●
magics:AdventureWorks
Run
Cancel
Disconnect
Change Connection
AdventureWorks
Explain
Enable SQLCMD
Go to Definition F12
Peek Definition Alt+F12
Change All Occurrences Ctrl+F2
Format Document Shift+Alt+F
Format Document With...
Format Selection Ctrl+K Ctrl+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Command Palette... Ctrl+Shift+P

```
-- Create a new stored procedure called 'StoredProcedu...' in schema 'dbo'
-- Drop the stored procedure if it already exists
IF EXISTS (
SELECT *
    FROM INFORMATION_SCHEMA.ROUTINES
WHERE SPECIFIC_SCHEMA = N'dbo'
    AND SPECIFIC_NAME = N'StoredProcedureName'
    AND ROUTINE_TYPE = N'PROCEDURE'
)
DROP PROCEDURE dbo.StoredProcedureName
GO
-- Create the stored procedure in the specified schema
CREATE PROCEDURE dbo.StoredProcedureName
    @param1 /*parameter name*/ int /*datatype_for_para
    @param2 /*parameter name*/ int /*datatype_for_para
-- add more stored procedure parameters here
AS
    -- body of the stored procedure
    SELECT @param1, @param2
GO
-- example to execute the stored procedure we just created
EXECUTE dbo.StoredProcedureName 1 /*value_for_param1*/, 2 /*value_for_param2*/
GO
```

We need to fill the *StoredProcedureName* and *SchemaName. To make it easy s*elect *StoredProcedureName*, right-click, and select **Change All Occurrences or type Ctrl+F2**. We type the proc name as GetProductPriceByRangeAndCategories and all *StoredProcedureName* entries change to GetProductPriceByRangeAndCategories.

We keep the schema as dbo, so we end up with Dbo.GetProductPriceByRangeAndCategories as

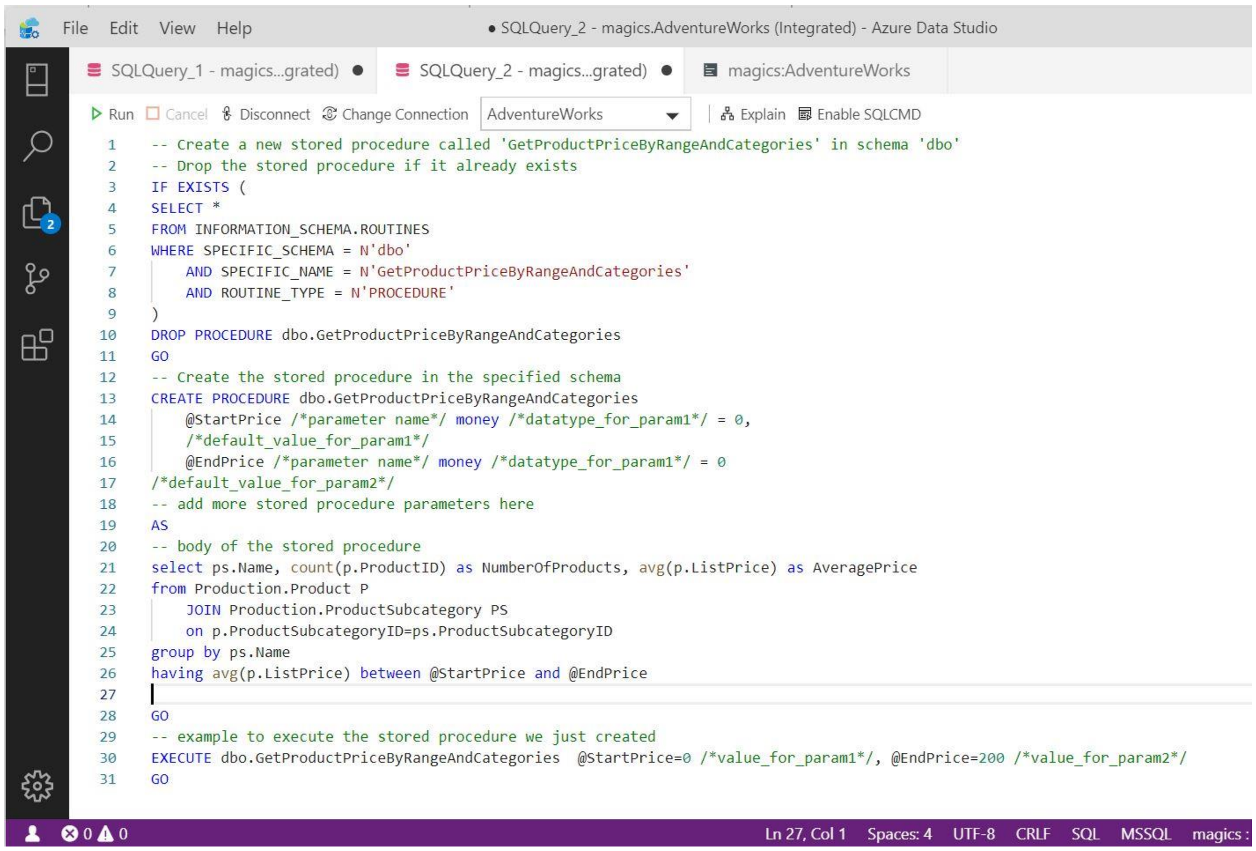
a consistent name used in the entire script


File
Edit
View
Help
● SQLQuery_2 - magics.AdventureWorks (Integrated) - Azure Data Studio
SQLQuery_1 - magics...grated)
SQLQuery_2 - magics...grated)
magics:AdventureWorks
Run
Cancel
Disconnect
Change Connection
AdventureWorks
Explain
Enable SQLCMD

```sql
-- Create a new stored procedure called 'GetProductPriceByRangeAndCategories' in schema 'dbo'
-- Drop the stored procedure if it already exists
IF EXISTS (
SELECT *
    FROM INFORMATION_SCHEMA.ROUTINES
WHERE SPECIFIC_SCHEMA = N'dbo'
    AND SPECIFIC_NAME = N'GetProductPriceByRangeAndCategories'
    AND ROUTINE_TYPE = N'PROCEDURE'
)
DROP PROCEDURE dbo.GetProductPriceByRangeAndCategories
GO
-- Create the stored procedure in the specified schema
CREATE PROCEDURE dbo.GetProductPriceByRangeAndCategories
    @param1 /*parameter name*/ int /*datatype_for_param1*/ = 0, /*default_value_for_param1*/
    @param2 /*parameter name*/ int /*datatype_for_param1*/ = 0 /*default_value_for_param2*/
-- add more stored procedure parameters here
AS
    -- body of the stored procedure
    SELECT @param1, @param2
GO
-- example to execute the stored procedure we just created
EXECUTE dbo.GetProductPriceByRangeAndCategories 1 /*value_for_param1*/, 2 /*value_for_param2*/
GO
```

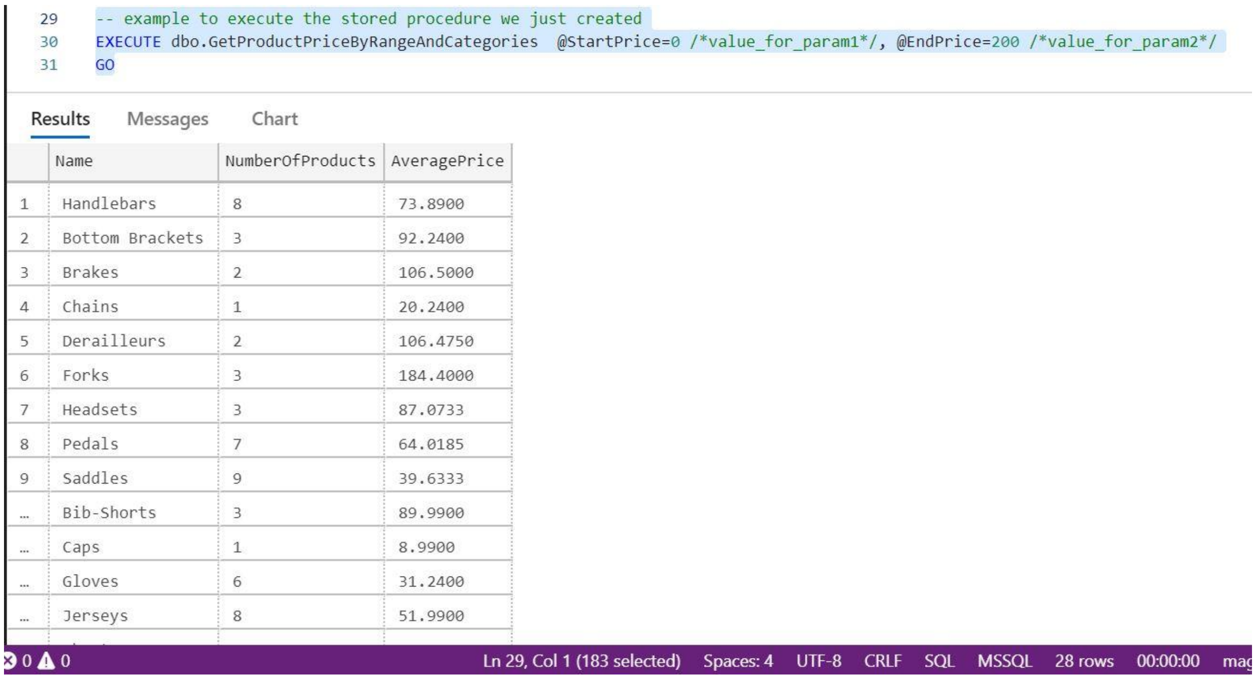
The snippet contains placeholder parameters and body text that needs updating. The *EXECUTE* statement also contains placeholder text because it does not know how many parameters the procedure will have. We will parameterize our HAVING Clause such as user could run the procedure for a price range. Using the same hot key combination of Change All Occurrences we replace parameters with their new names. We copy our query and paste it in the body of the stored procedure. A useful feature of query editor is it shows the type of the column once you hover over it. In ours case the p.ListPrice is of type money, so we need to align the parameters and change the int to money type.

We change the EXECUTE stored proc row and we are ready for creating and testing our stored proc.



```
1   -- Create a new stored procedure called 'GetProductPriceByRangeAndCategories' in schema 'dbo'
2   -- Drop the stored procedure if it already exists
3   IF EXISTS (
4   SELECT *
5   FROM INFORMATION_SCHEMA.ROUTINES
6   WHERE SPECIFIC_SCHEMA = N'dbo'
7       AND SPECIFIC_NAME = N'GetProductPriceByRangeAndCategories'
8       AND ROUTINE_TYPE = N'PROCEDURE'
9   )
10  DROP PROCEDURE dbo.GetProductPriceByRangeAndCategories
11  GO
12  -- Create the stored procedure in the specified schema
13  CREATE PROCEDURE dbo.GetProductPriceByRangeAndCategories
14      @StartPrice /*parameter name*/ money /*datatype_for_param1*/ = 0,
15      /*default_value_for_param1*/
16      @EndPrice /*parameter name*/ money /*datatype_for_param1*/ = 0
17  /*default_value_for_param2*/
18  -- add more stored procedure parameters here
19  AS
20  -- body of the stored procedure
21  select ps.Name, count(p.ProductID) as NumberOfProducts, avg(p.ListPrice) as AveragePrice
22  from Production.Product P
23      JOIN Production.ProductSubcategory PS
24      on p.ProductSubcategoryID=ps.ProductSubcategoryID
25  group by ps.Name
26  having avg(p.ListPrice) between @StartPrice and @EndPrice
27  
28  GO
29  -- example to execute the stored procedure we just created
30  EXECUTE dbo.GetProductPriceByRangeAndCategories  @StartPrice=0 /*value_for_param1*/, @EndPrice=200 /*value_for_param2*/
31  GO
```

We run the code up to line 28 to create our stored proc and then we test it by running the EXECUTE command



```
-- example to execute the stored procedure we just created
EXECUTE dbo.GetProductPriceByRangeAndCategories  @StartPrice=0 /*value_for_param1*/, @EndPrice=200 /*value_for_param2*/
GO
```

	Name	NumberOfProducts	AveragePrice
1	Handlebars	8	73.8900
2	Bottom Brackets	3	92.2400
3	Brakes	2	106.5000
4	Chains	1	20.2400
5	Derailleurs	2	106.4750
6	Forks	3	184.4000
7	Headsets	3	87.0733
8	Pedals	7	64.0185
9	Saddles	9	39.6333
...	Bib-Shorts	3	89.9900
...	Caps	1	8.9900
...	Gloves	6	31.2400
...	Jerseys	8	51.9900

0 0
Ln 29, Col 1 (183 selected) Spaces: 4 UTF-8 CRLF SQL MSSQL 28 rows 00:00:00

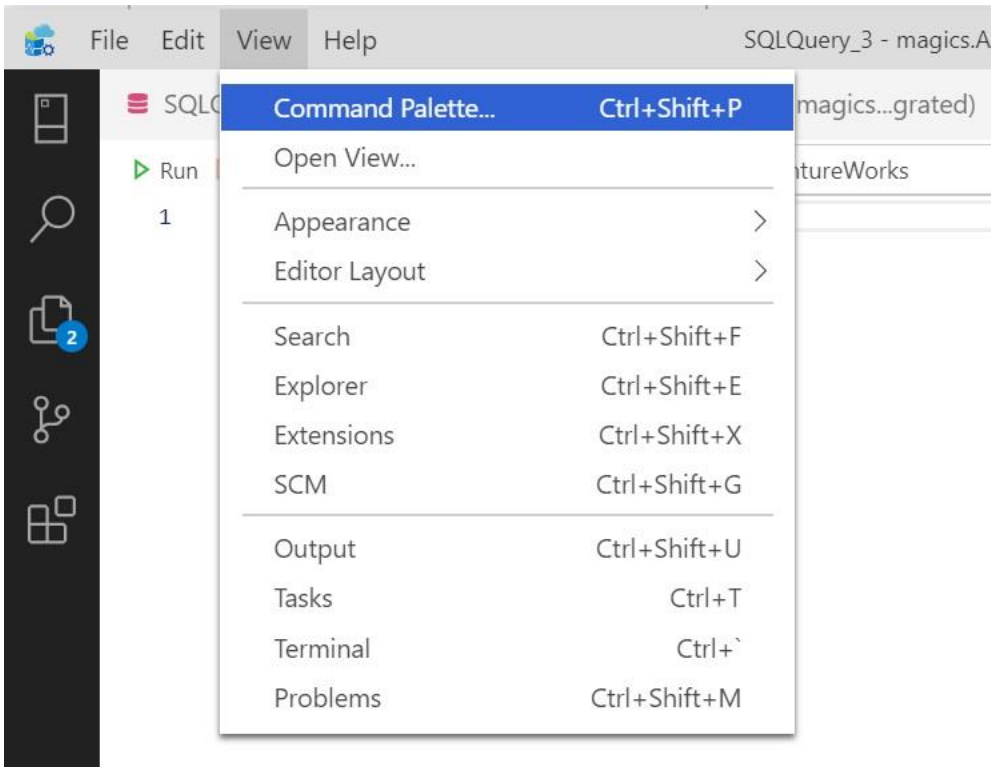
Great experience up until now and seems that we have completed our task of creating a module in the database for product price analysis. Let’s get one step further and create a SQL Notebook to tell the story of our data discovery and module creation.

A notebook is a new term, coming in an open source initiative called Project Jupiter. Notebooks are created for sharing text code, images, data insights in the same document.

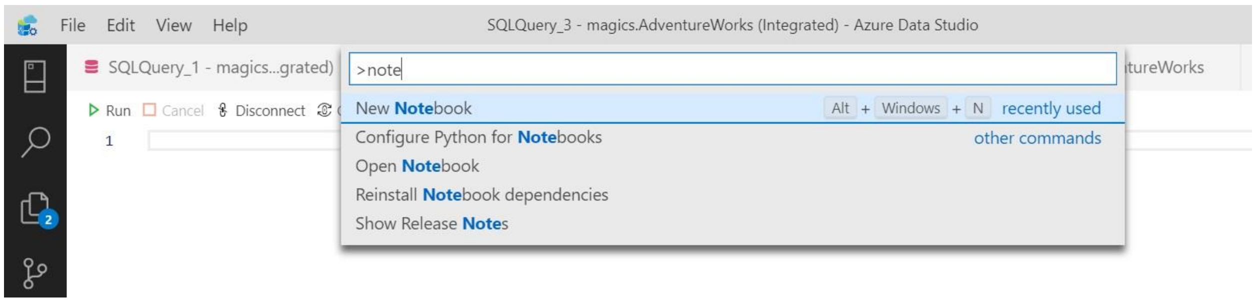
The text in notebook is formatted using MD Language (MarkDown language), which is also used for producing .md files in GitHub. MD language is a cross-platform language.

Notebooks are featured in Azure Data Studio, which makes them available with T-SQL kernel. This allows us to create and share documents which contain text, T-SQL code, query results, and even images and charts to share database insights and to tell a story or to create runbooks.

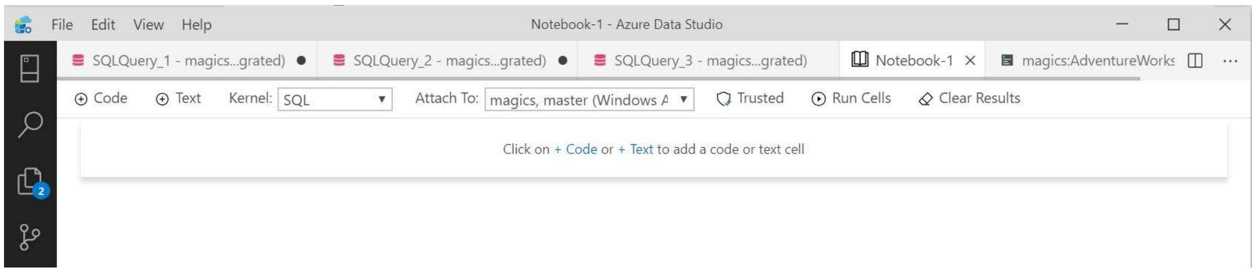
To create a new notebook we can use the Command Palette. We can run it from the View Menu or by pressing its hot key Ctrl+Shift+P. The Command palette is very useful in code typing when you need to find a command.



You start typing and the tool provides you with the list of possible commands to choose from. Trying to find how to create a new notebook leads us to the New Notebook command which we can choose from the list or its shortkey combination Alt_Windows+N


File
Edit
View
Help
SQLQuery_3 - magics.AdventureWorks (Integrated) - Azure Data Studio
SQLQuery_1 - magics...grated)
Run
Cancel
Disconnect
>note
New Notebook Alt + Windows + N recently used
Configure Python for Notebooks other commands
Open Notebook
Reinstall Notebook dependencies
Show Release Notes

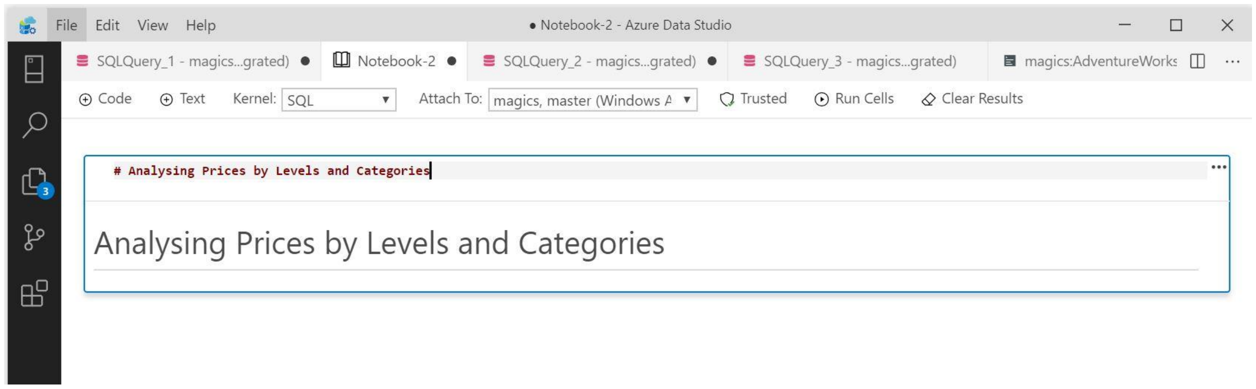
A new Notebook opens and we are ready to put our story here.


File Edit View Help
Notebook-1 - Azure Data Studio
SQLQuery_1 - magics...grated)
SQLQuery_2 - magics...grated)
SQLQuery_3 - magics...grated)
Notebook-1
magics:AdventureWorks
Code
Text
Kernel: SQL
Attach To: magics, master (Windows A
Trusted
Run Cells
Clear Results
Click on + Code or + Text to add a code or text cell

Let's take a look at notebook components first. We have *Kernel*, which specifies the language we will write the code in this notebook and forces the engine that will execute it. It will be SQL in our new notebook. *Attach to* sets the connection from the list of connections we work with in this notebook. *Code and Text* add snippets for typing the corresponding content in our notebook. We can use *Run Cells* to run the code and *Clear Results* to delete all the table results which we can potentially have stored in the notebook.

Our notebook can be stored as an .ipynb file. We can share it to other users, so we need to be sure it is trusted, meaning there is no malicious code inside.

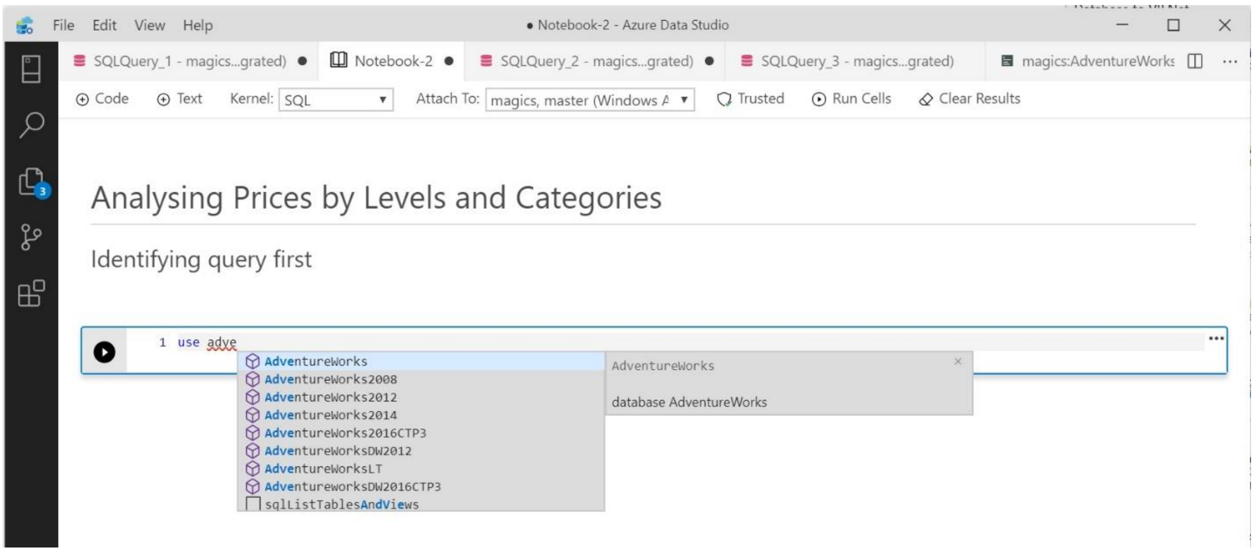
Creating content in a notebook can be an exciting experience, but it requires getting to know the MD language or at least some basic concepts of its syntax. Even if it is not that complex it still has a long list of elements. As diving into them is far beyond the scope of this book, I am going to use just # element for formatting Headings of the new notebook.


File Edit View Help
● Notebook-2 - Azure Data Studio
SQLQuery_1 - magics...grated)
Notebook-2
SQLQuery_2 - magics...grated)
SQLQuery_3 - magics...grated)
magics:AdventureWorks
Code
Text
Kernel: SQL
Attach To: magics, master (Windows A
Trusted
Run Cells
Clear Results

```
# Analysing Prices by Levels and Categories
```

Analysing Prices by Levels and Categories

Code snippets allow to paste or type a code, and this comes with a fully functional code editor and intelisense.


File Edit View Help
• Notebook-2 - Azure Data Studio
SQLQuery_1 - magics...grated)
Notebook-2
SQLQuery_2 - magics...grated)
SQLQuery_3 - magics...grated)
magics:AdventureWorks
Code
Text
Kernel: SQL
Attach To: magics, master (Windows A
Trusted
Run Cells
Clear Results

Analysing Prices by Levels and Categories

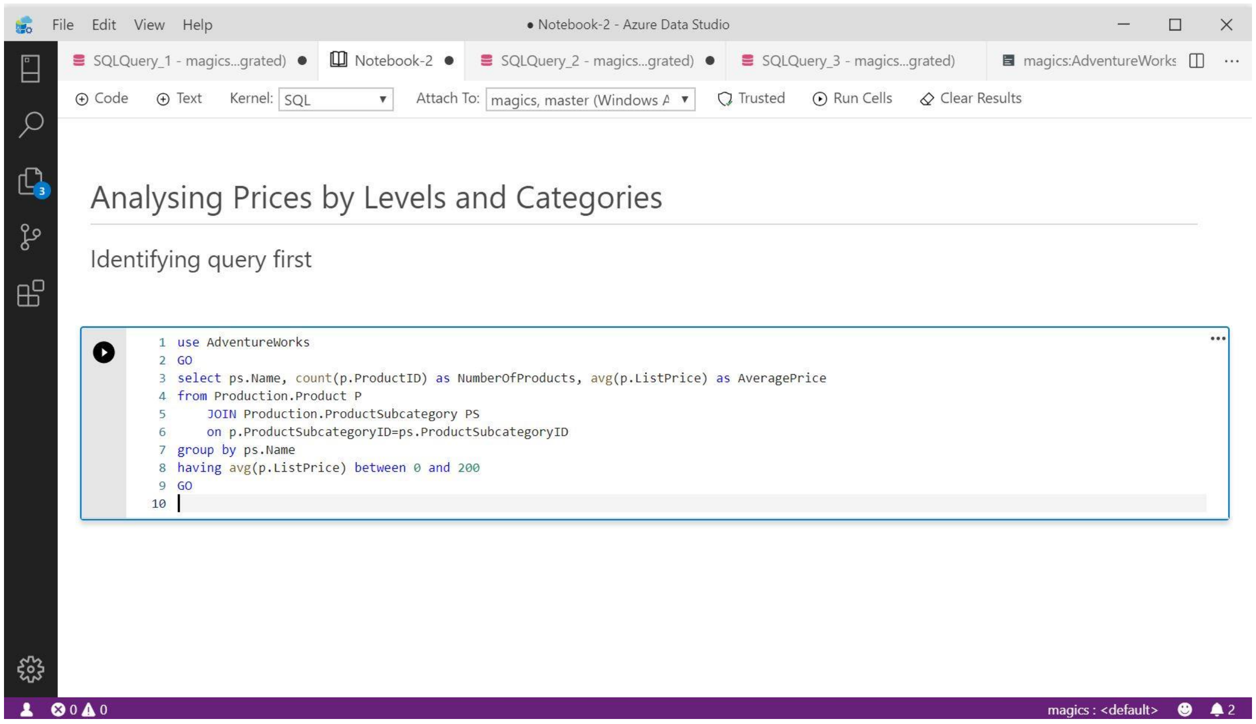
Identifying query first

```
use adve
```

```
AdventureWorks
AdventureWorks2008
AdventureWorks2012
AdventureWorks2014
AdventureWorks2016CTP3
AdventureWorksDW2012
AdventureWorksLT
AdventureworksDW2016CTP3
sqlListTablesAndViews
```

AdventureWorks

database AdventureWorks

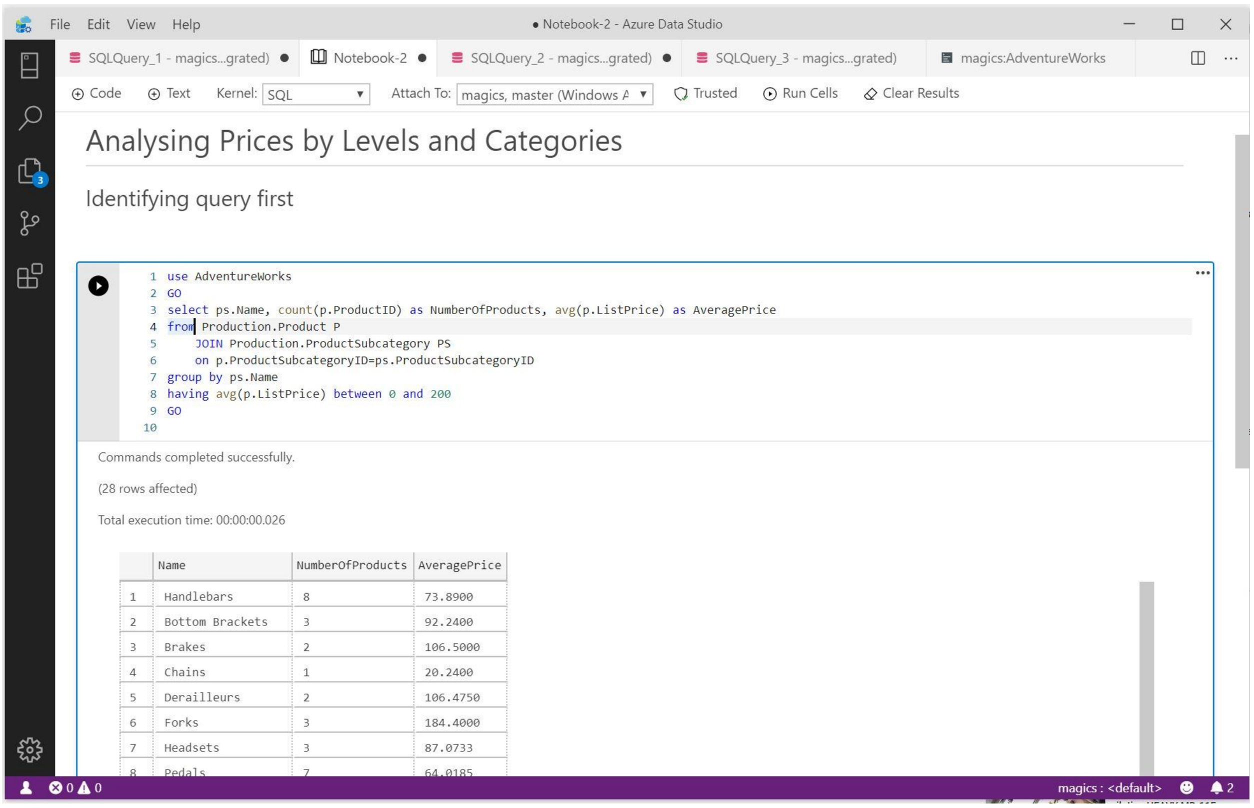

File Edit View Help
• Notebook-2 - Azure Data Studio
SQLQuery_1 - magics...grated)
Notebook-2
SQLQuery_2 - magics...grated)
SQLQuery_3 - magics...grated)
magics:AdventureWorks
Code
Text
Kernel: SQL
Attach To: magics, master (Windows A
Trusted
Run Cells
Clear Results
0 0
magics : <default>
2

Analysing Prices by Levels and Categories

Identifying query first

```
use AdventureWorks
GO
select ps.Name, count(p.ProductID) as NumberOfProducts, avg(p.ListPrice) as AveragePrice
from Production.Product P
    JOIN Production.ProductSubcategory PS
    on p.ProductSubcategoryID=ps.ProductSubcategoryID
group by ps.Name
having avg(p.ListPrice) between 0 and 200
GO
```

A code snippet has a run button on the top left corner, which allows us to run the code of this snippet only, and even to include the results in the notebook. If we don’t want the results to appear or be preserved in the code snippets then we can click the *Clear Results* button in the command line of the notebook.



Analysing Prices by Levels and Categories

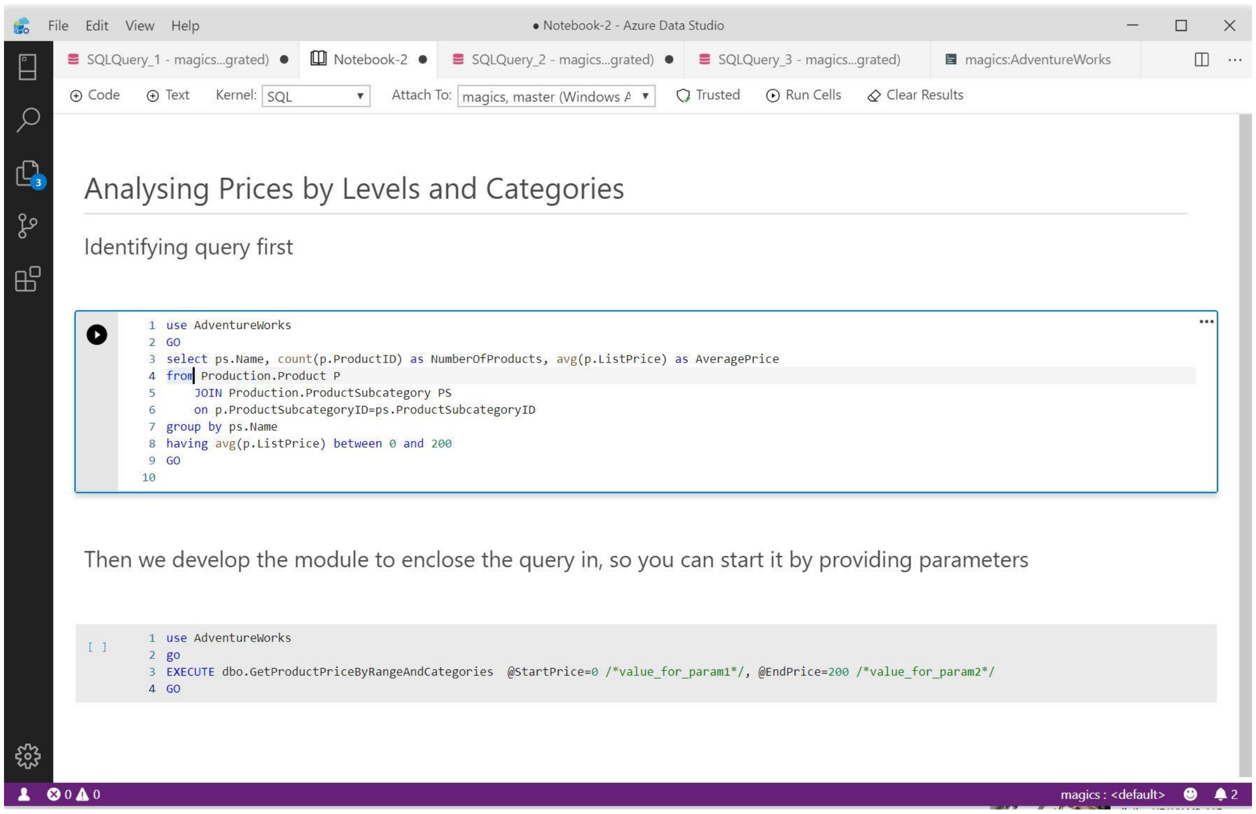
Identifying query first

```
use AdventureWorks
GO
select ps.Name, count(p.ProductID) as NumberOfProducts, avg(p.ListPrice) as AveragePrice
from Production.Product P
    JOIN Production.ProductSubcategory PS
    on p.ProductSubcategoryID=ps.ProductSubcategoryID
group by ps.Name
having avg(p.ListPrice) between 0 and 200
GO
```

```
Commands completed successfully.

(28 rows affected)

Total execution time: 00:00:00.026
```

	Name	NumberOfProducts	AveragePrice
1	Handlebars	8	73.8900
2	Bottom Brackets	3	92.2400
3	Brakes	2	106.5000
4	Chains	1	20.2400
5	Derailleurs	2	106.4750
6	Forks	3	184.4000
7	Headsets	3	87.0733
8	Pedals	7	64.0185

Combining text and code snippets allow us to create a very simple but still good looking T-SQL notebook


File Edit View Help
● Notebook-2 - Azure Data Studio
SQLQuery_1 - magics...grated)
Notebook-2
SQLQuery_2 - magics...grated)
SQLQuery_3 - magics...grated)
magics:AdventureWorks
Code
Text
Kernel: SQL
Attach To: magics, master (Windows A
Trusted
Run Cells
Clear Results
magics : <default>

Analysing Prices by Levels and Categories

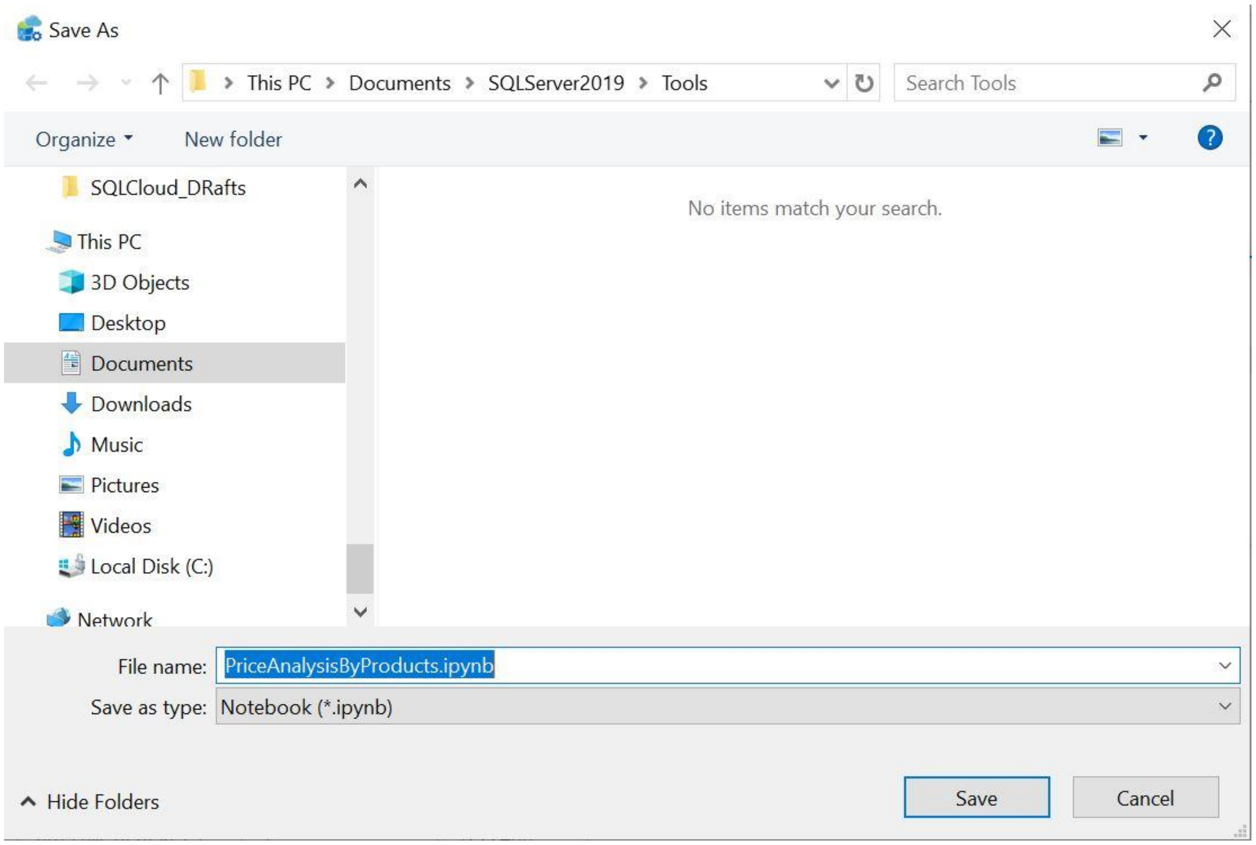
Identifying query first

```
use AdventureWorks
GO
select ps.Name, count(p.ProductID) as NumberOfProducts, avg(p.ListPrice) as AveragePrice
from Production.Product P
    JOIN Production.ProductSubcategory PS
    on p.ProductSubcategoryID=ps.ProductSubcategoryID
group by ps.Name
having avg(p.ListPrice) between 0 and 200
GO
```

Then we develop the module to enclose the query in, so you can start it by providing parameters

```
use AdventureWorks
go
EXECUTE dbo.GetProductPriceByRangeAndCategories  @StartPrice=0 /*value_for_param1*/, @EndPrice=200 /*value_for_param2*/
GO
```

As soon as we are ready we can save our notebook with a name Price AnalysisByProduct.ipynb

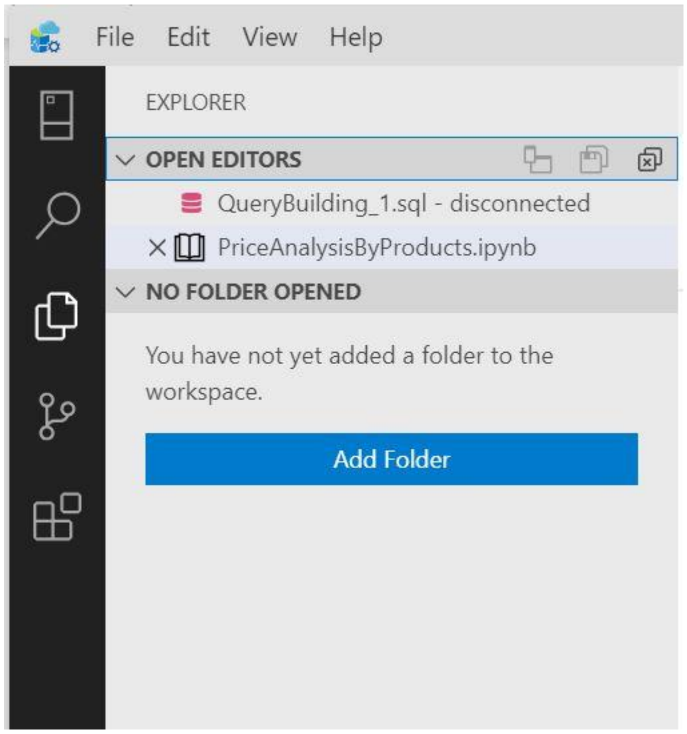


File name: PriceAnalysisByProducts.ipynb

Save as type: Notebook (*.ipynb)

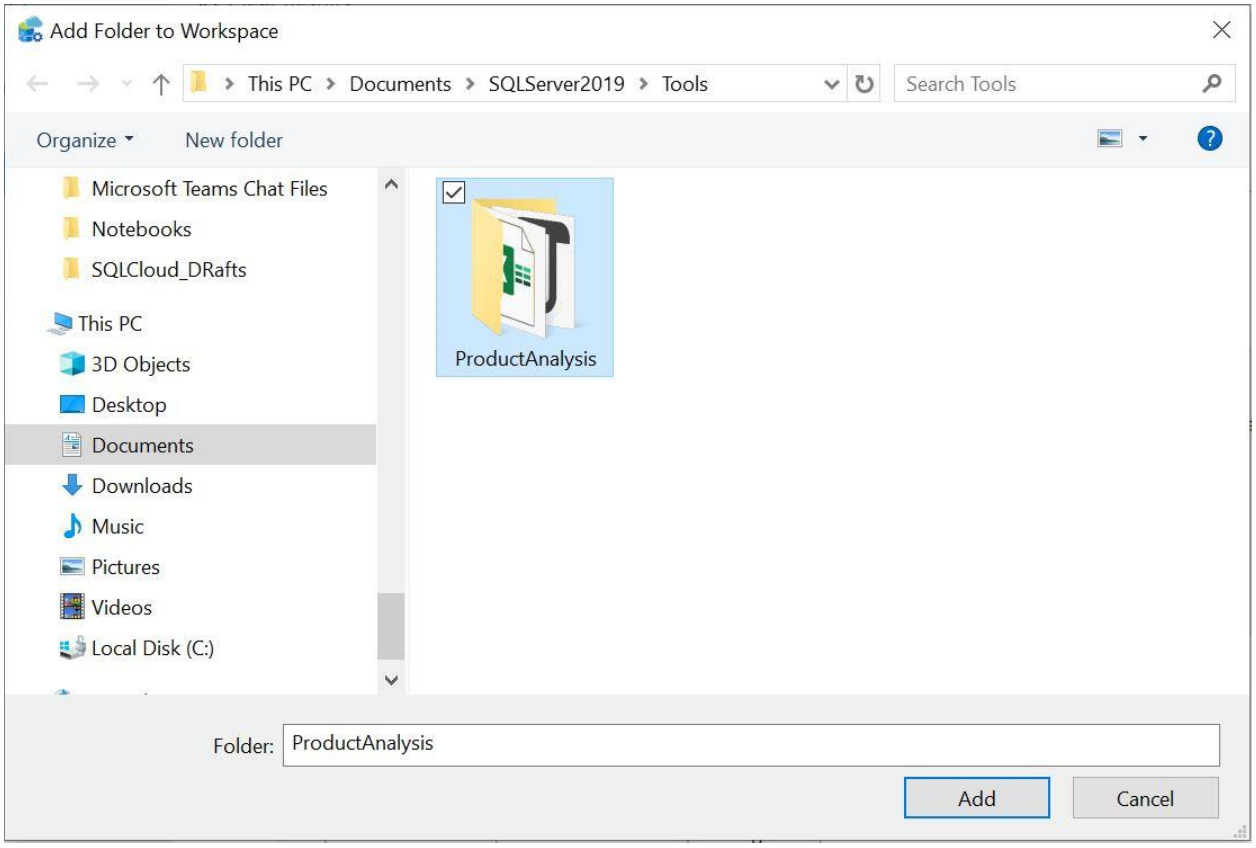
Getting organized – using workspaces

Workspace is a great way of storing all the files and queries we work with as a single store group. We can add more files later, or we can add even a folder to a workspace. This will make the task of working with files in it easy and we can open any of them by just clicking on it. We have Explorer menu on the left side of the tool under the Connections and Search menu. We work with *Explorer* Menu to organize files and folder in our workspace. We put all the files including JSON and CSV in a folder ProductAnalysis, so I can add it to the workspace by clicking on the *Add Folder* button.


File
Edit
View
Help
EXPLORER
OPEN EDITORS
QueryBuilding_1.sql - disconnected
PriceAnalysisByProducts.ipynb
NO FOLDER OPENED
Add Folder

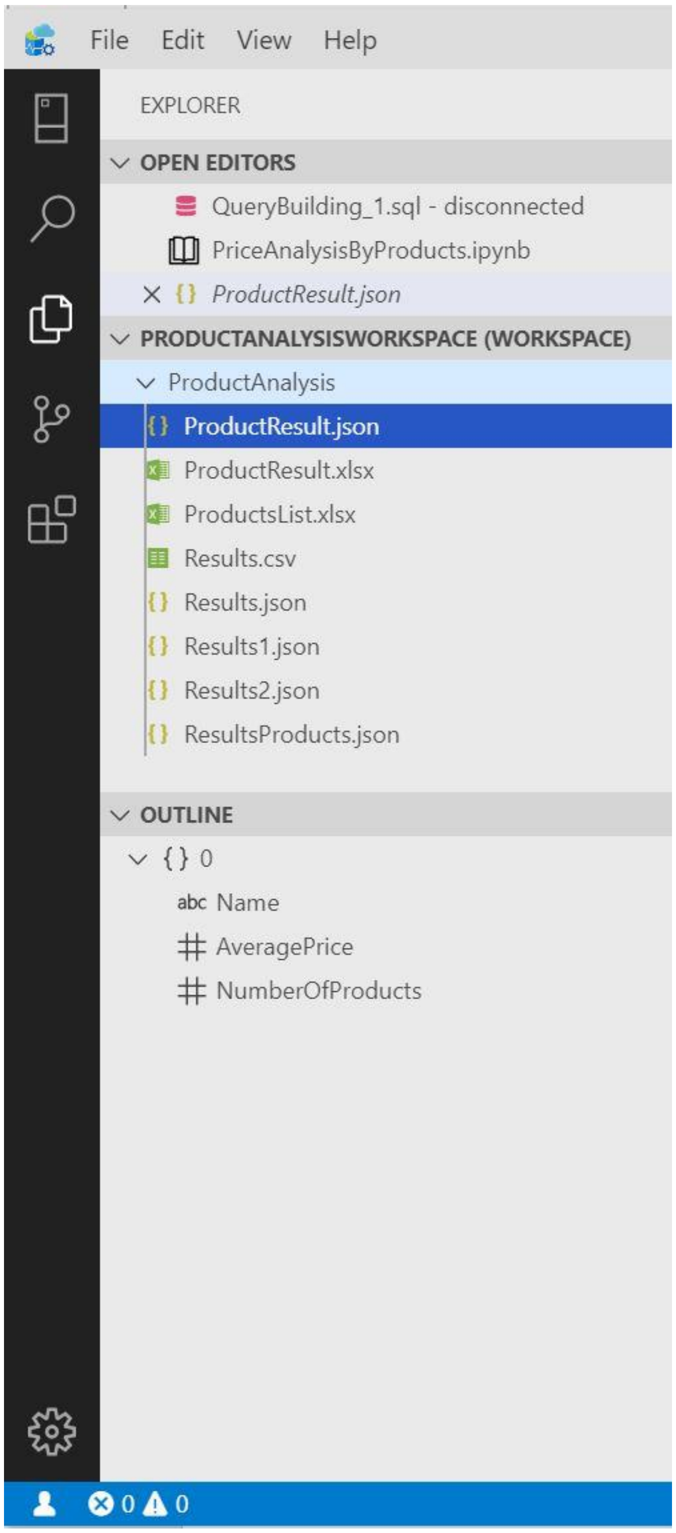
You have not yet added a folder to the workspace.

We add the folder and then it appears in the list of workspace folders



Folder: ProductAnalysis

We can click on files and see outline


File
Edit
View
Help
EXPLORER
OPEN EDITORS
QueryBuilding_1.sql - disconnected
PriceAnalysisByProducts.ipynb
ProductResult.json
PRODUCTANALYSISWORKSPACE (WORKSPACE)
ProductAnalysis
ProductResult.json
ProductResult.xlsx
ProductsList.xlsx
Results.csv
Results.json
Results1.json
Results2.json
ResultsProducts.json
OUTLINE
{ } 0
abc Name
AveragePrice
NumberOfProducts
0
0

It is not necessary all the files of a workspace to be in the same folder. Saving the workspace is simple, we Select Save Workspace from the File menu and provide name and location.

Going extra mile - ADS Extensions and customizations

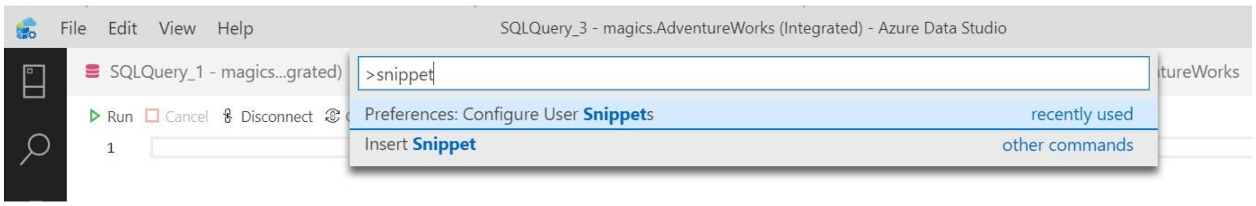
Customizing Azure Data Services is one of its greatest benefits making it rather development platform for working with SQL Databases rather than just a T-SQL tool. We have many options to customize and extent the product, all of them intuitive and easy to perform. Customizations usually contain editing or typing JSON configurations not only using T-SQL Scripts. Let's get through some examples.

Custom code snippet

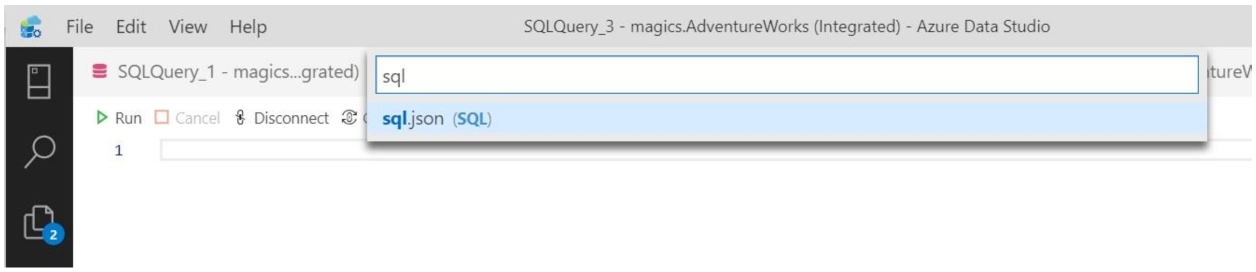
We discovered code snippets earlier when we created our stored procedure for product price analysis. We found that using code snippets is practical and convenient in ADS. We can create our own code snippets. Let us create a snippet for updating statistics of a table.

Adding a simple custom T-SQL Snippet is possible at it is very easy to perform. Let's create a custom snippet

By calling the Command Palette (Ctrl+Shift+P) and searching for Configuring User Snippets. We click on the Preferences: Configure User Snippets command from the list which lead us to type of snippet we want to create

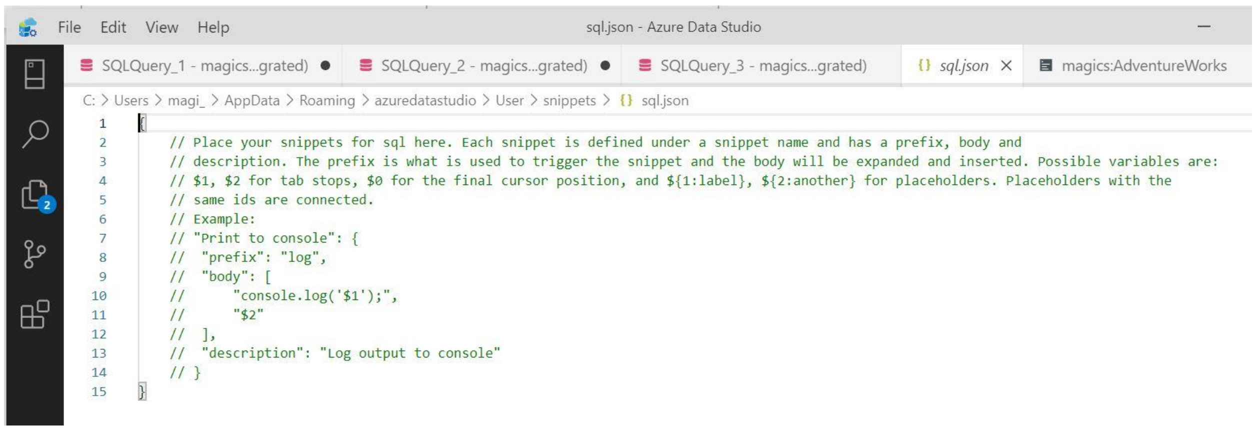

File
Edit
View
Help
SQLQuery_3 - magics.AdventureWorks (Integrated) - Azure Data Studio
SQLQuery_1 - magics...grated)
>snippet
Run
Cancel
Disconnect
Preferences: Configure User Snippets
recently used
Insert Snippet
other commands

We type sql which lists the only option to configure a custom sql snippet – to create our snippet code in sql.json file. We choose the sql.json option and we are ready to type the create statistics snippet code.


File Edit View Help
SQLQuery_3 - magics.AdventureWorks (Integrated) - Azure Data Studio
SQLQuery_1 - magics...grated)
Run Cancel Disconnect
sql
sql.json (SQL)

Snippet is defined in a JSON format and it has the following main elements:

- Prefix – what triggers the snippet and what we see as a sql. Command when we search for sql snippets
- body – this is the t-sql code we type and we can use variables and placeholders in the way the instruction suggests
- description – short explanation which user will see when it hovers over the snippet

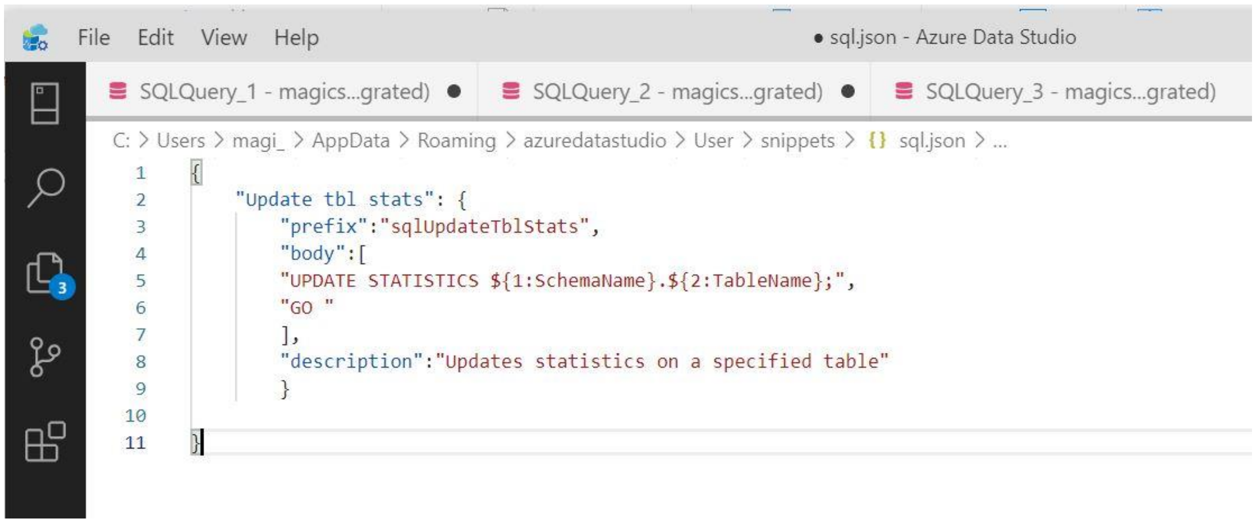

File Edit View Help
sql.json - Azure Data Studio
SQLQuery_1 - magics...grated)
SQLQuery_2 - magics...grated)
SQLQuery_3 - magics...grated)
sql.json
magics:AdventureWorks
C: > Users > magi_ > AppData > Roaming > azuredatastudio > User > snippets > sql.json

```
{
    // Place your snippets for sql here. Each snippet is defined under a snippet name and has a prefix, body and
    // description. The prefix is what is used to trigger the snippet and the body will be expanded and inserted. Possible variables are:
    // $1, $2 for tab stops, $0 for the final cursor position, and ${1:label}, ${2:another} for placeholders. Placeholders with the
    // same ids are connected.
    // Example:
    // "Print to console": {
    //  "prefix": "log",
    //  "body": [
    //      "console.log('$1');",
    //      "$2"
    //  ],
    //  "description": "Log output to console"
    // }
}
```

According to the instructions and the elements needed, the JSON code of our Update statistics on a user table snippet will look like this:

```
{
"Update tbl stats": {
"prefix":"sqlUpdateTblStats",
"body":[
"UPDATE STATISTICS ${1:SchemaName}.${2:TableName};",
"GO "
],
"description":"Updates statistics on a specified table"
}
}
```

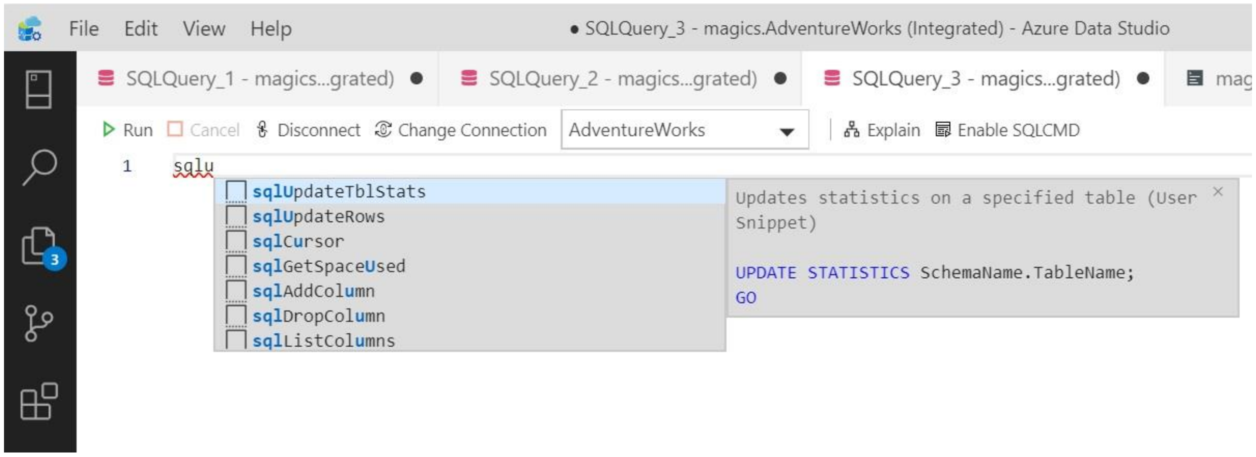
We type the elements in the sql.json file. And all we need to do is to save the file.



File Edit View Help
● sql.json - Azure Data Studio
SQLQuery_1 - magics...grated) ●
SQLQuery_2 - magics...grated) ●
SQLQuery_3 - magics...grated)
C: > Users > magi_ > AppData > Roaming > azuredatastudio > User > snippets > {} sql.json > ...

```
{
    "Update tbl stats": {
        "prefix":"sqlUpdateTblStats",
        "body":[
        "UPDATE STATISTICS ${1:SchemaName}.${2:TableName};",
        "GO "
        ],
        "description":"Updates statistics on a specified table"
        }

}
```

We have the snippet listed in our sql snippets list. We can find it by typing sqlu (coming from UDPATE), it appears first in the list and we can choose it, fill the SchemaNAme and TableName and run the command to update stats



File Edit View Help
● SQLQuery_3 - magics.AdventureWorks (Integrated) - Azure Data Studio
SQLQuery_1 - magics...grated) ●
SQLQuery_2 - magics...grated) ●
SQLQuery_3 - magics...grated) ●
mag
Run
Cancel
Disconnect
Change Connection
AdventureWorks
Explain
Enable SQLCMD

```
sqlu
sqlUpdateTblStats
sqlUpdateRows
sqlCursor
sqlGetSpaceUsed
sqlAddColumn
sqlDropColumn
sqlListColumns
```

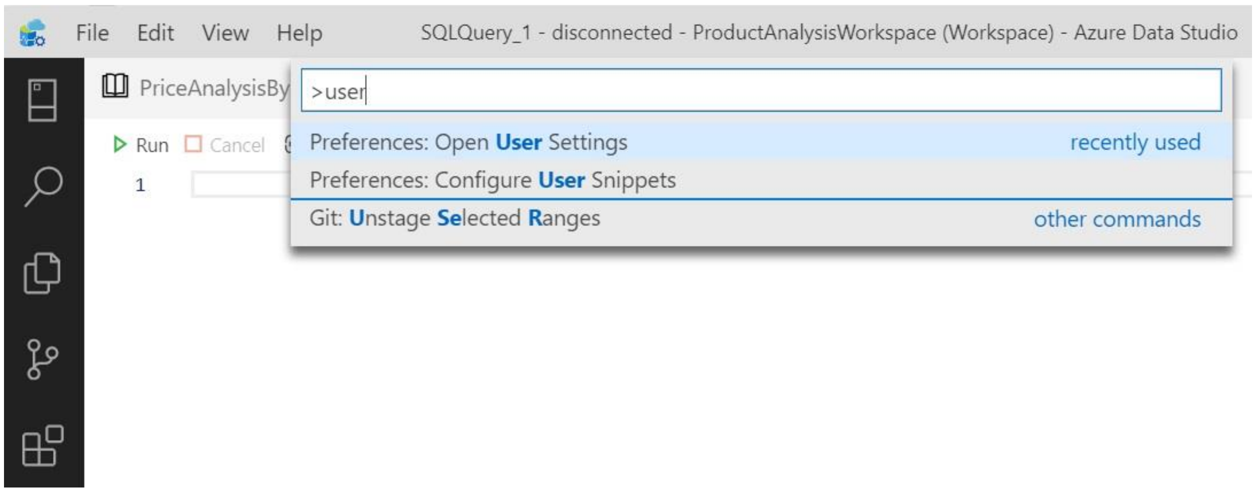
```
Updates statistics on a specified table (User Snippet)

UPDATE STATISTICS SchemaName.TableName;
GO
```

Adding widget

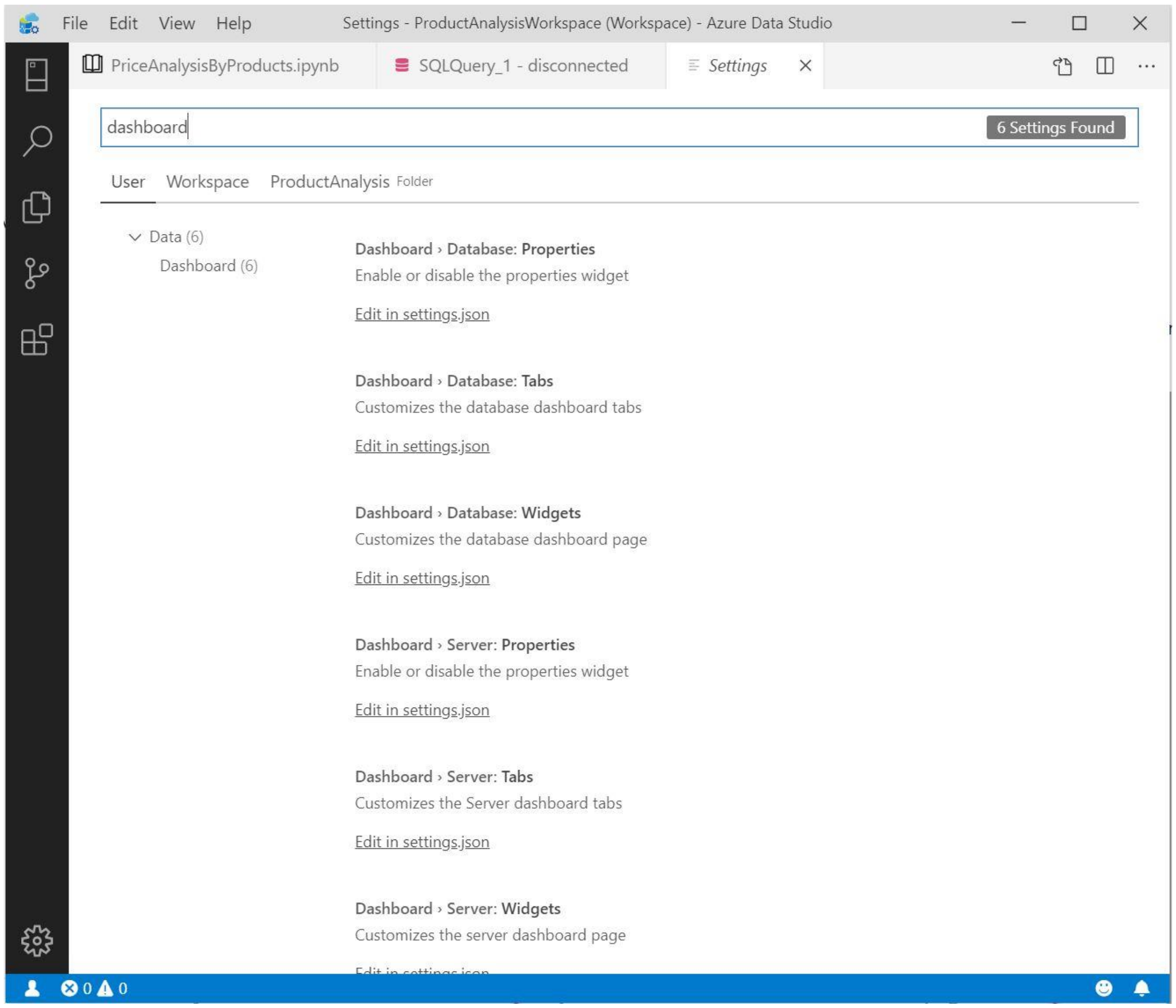
Widgets are yet another customizable element in ADS. They can appear in dashboards as data insights, providing an at-a-glance view about object status, resource usage, workload intensity or any other important measure, including those coming from queries on data in tables.

Azure Data Studio has a built-in sample widget to monitor the space used by tables in a database. We need to just add it to the user configuration settings JSON file.

To do that we start Command Palette and search for User settings command


File
Edit
View
Help
SQLQuery_1 - disconnected - ProductAnalysisWorkspace (Workspace) - Azure Data Studio
PriceAnalysisBy
Run
Cancel
>user
Preferences: Open User Settings
recently used
Preferences: Configure User Snippets
Git: Unstage Selected Ranges
other commands

We open User Settings and search for dashboard until we find Dashboard>Database Widgets.


File Edit View Help Settings - ProductAnalysisWorkspace (Workspace) - Azure Data Studio

PriceAnalysisByProducts.ipynb | SQLQuery_1 - disconnected | Settings

dashboard — 6 Settings Found

User | Workspace | ProductAnalysis Folder

Data (6)
Dashboard (6)

Dashboard › Database: Properties
Enable or disable the properties widget
Edit in settings.json

Dashboard › Database: Tabs
Customizes the database dashboard tabs
Edit in settings.json

Dashboard › Database: Widgets
Customizes the database dashboard page
Edit in settings.json

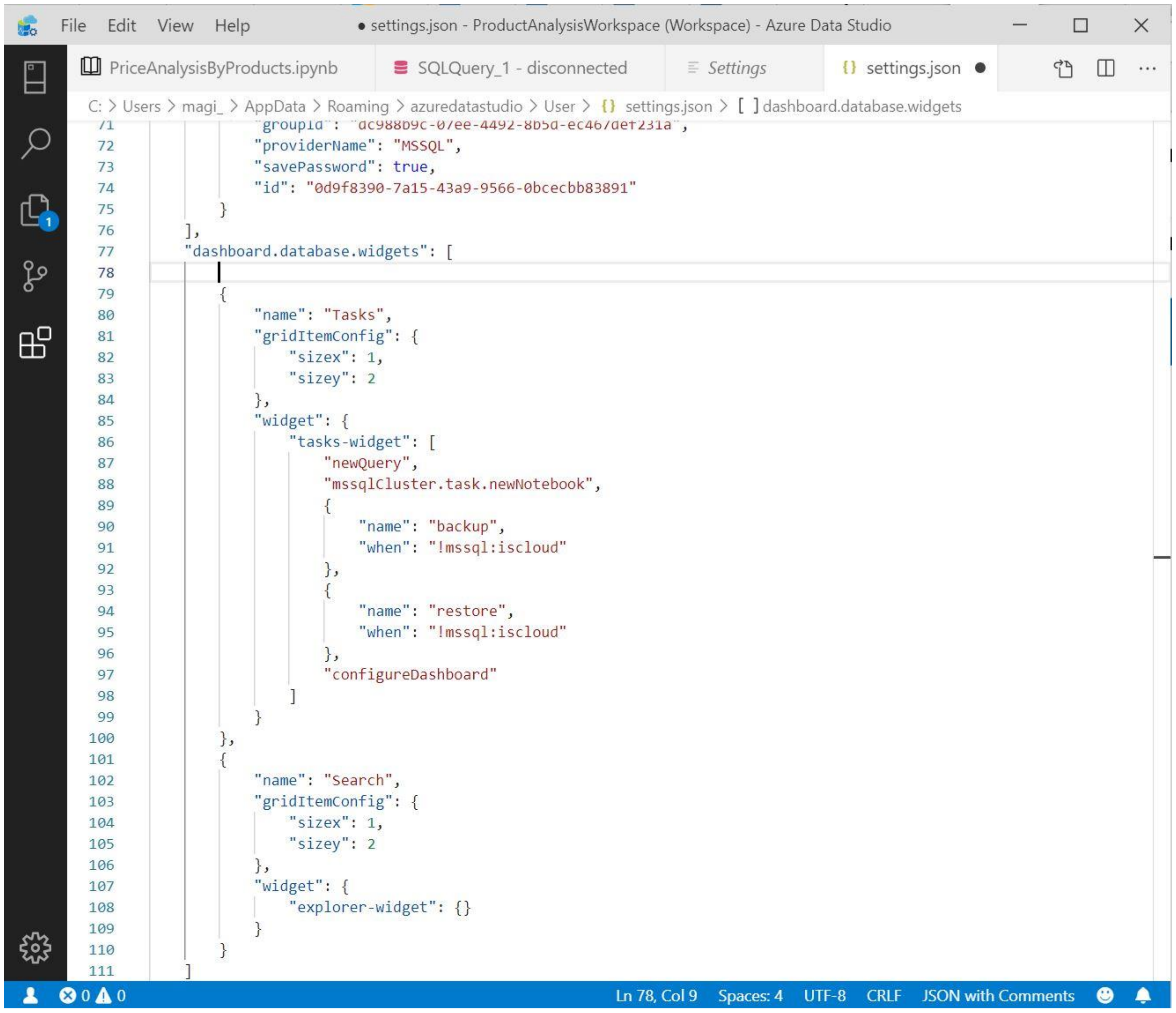
Dashboard › Server: Properties
Enable or disable the properties widget
Edit in settings.json

Dashboard › Server: Tabs
Customizes the Server dashboard tabs
Edit in settings.json

Dashboard › Server: Widgets
Customizes the server dashboard page

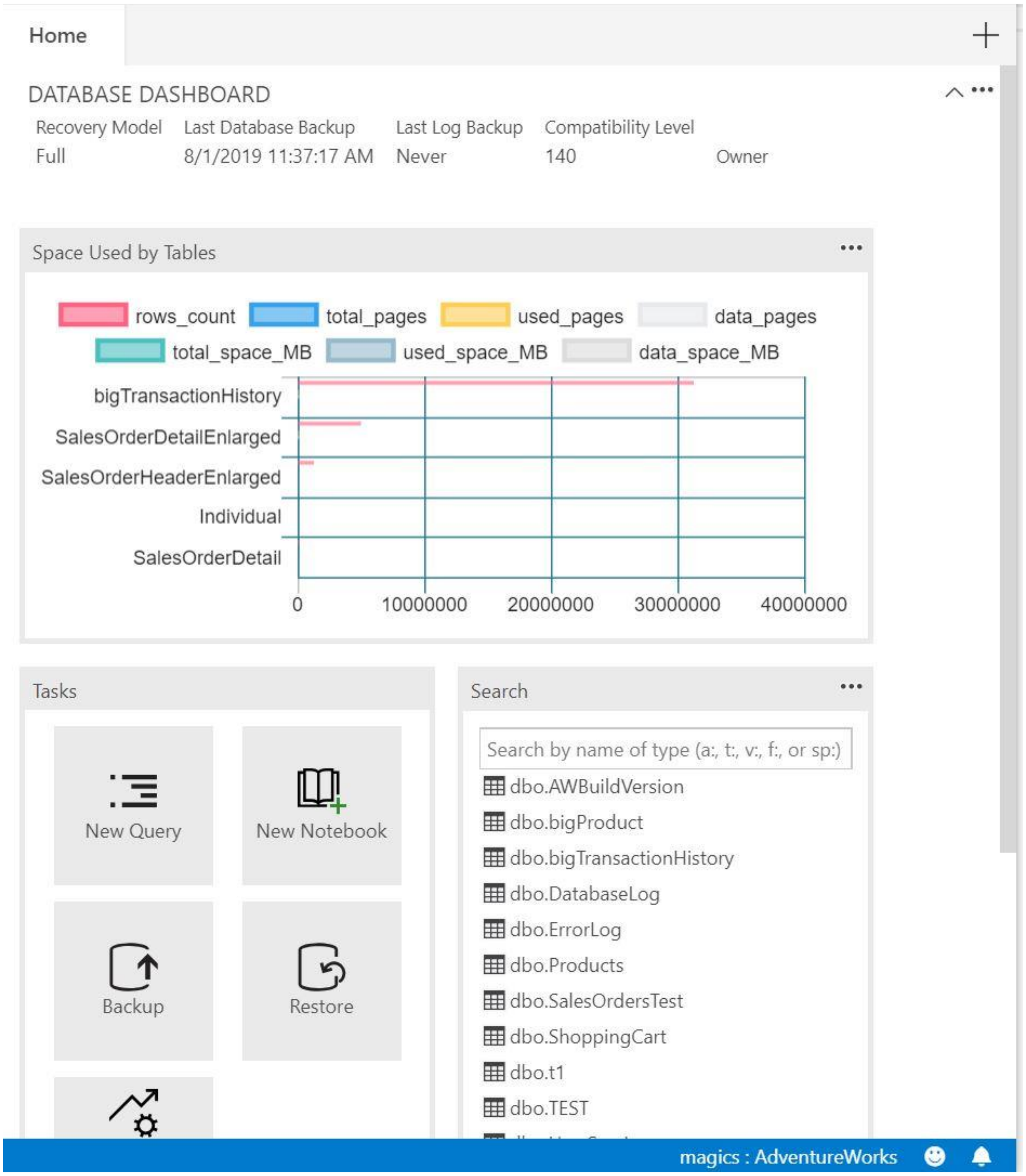
We click on Edits in settings.json to add the widget description. Typing “dashboard.database.widgets” will help us to find the place in the file where we need to insert the widget elements. We insert the description and save the settings.json file

```
{
    "name": "Space Used by Tables",
    "gridItemConfig": {
      "sizex": 2,
      "sizey": 1
    },
    "widget": {
      "table-space-db-insight": null
    }
},
```


File Edit View Help
● settings.json - ProductAnalysisWorkspace (Workspace) - Azure Data Studio
PriceAnalysisByProducts.ipynb
SQLQuery_1 - disconnected
Settings
settings.json
C: > Users > magi_ > AppData > Roaming > azuredatastudio > User > settings.json > [ ] dashboard.database.widgets
Ln 78, Col 9 Spaces: 4 UTF-8 CRLF JSON with Comments

```
                "groupId": "dc988b9c-07ee-4492-8b5d-ec467def231a",
                "providerName": "MSSQL",
                "savePassword": true,
                "id": "0d9f8390-7a15-43a9-9566-0bcecbb83891"
            }
        ],
        "dashboard.database.widgets": [
            
            {
                "name": "Tasks",
                "gridItemConfig": {
                    "sizex": 1,
                    "sizey": 2
                },
                "widget": {
                    "tasks-widget": [
                        "newQuery",
                        "mssqlCluster.task.newNotebook",
                        {
                            "name": "backup",
                            "when": "!mssql:iscloud"
                        },
                        {
                            "name": "restore",
                            "when": "!mssql:iscloud"
                        },
                        "configureDashboard"
                    ]
                }
            },
            {
                "name": "Search",
                "gridItemConfig": {
                    "sizex": 1,
                    "sizey": 2
                },
                "widget": {
                    "explorer-widget": {}
                }
            }
        ]
```

Then it is just a matter of showing the dashboard of our AdventureWorks database to find the new table space widget there



Recovery Model	Last Database Backup	Last Log Backup	Compatibility Level	
Full	8/1/2019 11:37:17 AM	Never	140	Owner

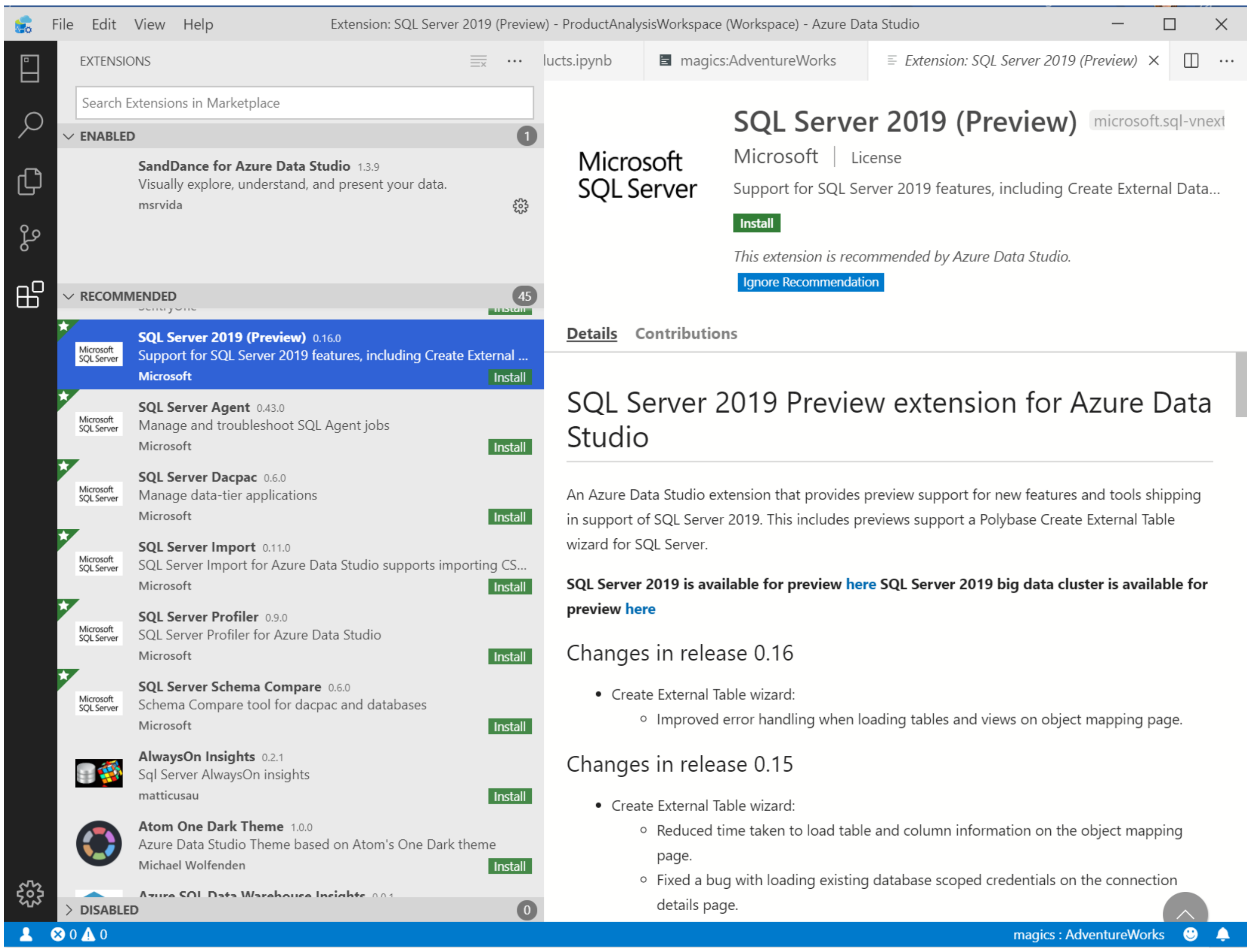
dbo.AWBuildVersion
dbo.bigProduct
dbo.bigTransactionHistory
dbo.DatabaseLog
dbo.ErrorLog
dbo.Products
dbo.SalesOrdersTest
dbo.ShoppingCart
dbo.t1
dbo.TEST

At the time of writing there is one more out of the box widget for analyzing the top five slowest queries based on Query Store execution statistics. We can still develop our own custom widgets.

Extensions

Extensions is yet another way of customizing ADS. They provide additional features to the product and are listed in Extensions command on the left side of the screen. While some of the extensions are listed within Azure Data Studio. Others you'll need to go to the Visual Studio Code marketplace to get. For example, here's the PowerShell extension: https://marketplace.visualstudio.com/items?itemName=ms-vscode.PowerShell

Just download the *.vsix file and install that.



SandDance for Azure Data Studio 1.3.9
Visually explore, understand, and present your data.
msrvida

SQL Server 2019 (Preview) 0.16.0
Support for SQL Server 2019 features, including Create External ...
Microsoft

SQL Server Agent 0.43.0
Manage and troubleshoot SQL Agent jobs
Microsoft

SQL Server Dacpac 0.6.0
Manage data-tier applications
Microsoft

SQL Server Import 0.11.0
SQL Server Import for Azure Data Studio supports importing CS...
Microsoft

SQL Server Profiler 0.9.0
SQL Server Profiler for Azure Data Studio
Microsoft

SQL Server Schema Compare 0.6.0
Schema Compare tool for dacpac and databases
Microsoft

AlwaysOn Insights 0.2.1
Sql Server AlwaysOn insights
matticusau

Atom One Dark Theme 1.0.0
Azure Data Studio Theme based on Atom's One Dark theme
Michael Wolfenden

Azure SQL Data Warehouse Insights 0.0.1

SQL Server 2019 (Preview) microsoft.sql-vnext
Microsoft | License
Support for SQL Server 2019 features, including Create External Data...

This extension is recommended by Azure Data Studio.

SQL Server 2019 Preview extension for Azure Data Studio

An Azure Data Studio extension that provides preview support for new features and tools shipping in support of SQL Server 2019. This includes previews support a Polybase Create External Table wizard for SQL Server.

SQL Server 2019 is available for preview here SQL Server 2019 big data cluster is available for preview here

Changes in release 0.16

- Create External Table wizard:
  - Improved error handling when loading tables and views on object mapping page.

Changes in release 0.15

- Create External Table wizard:
  - Reduced time taken to load table and column information on the object mapping page.
  - Fixed a bug with loading existing database scoped credentials on the connection details page.

Every extension contains a detailed description, recommendations and an installation button. In the description we can find a detailed information on what the extension does and how to use it. We can even develop extension

Conclusion

Based on everything we saw Azure Data Studio is a great tool providing data management and exciting coding experience for database developers. It is best suited for those who Query SQL databases like SQL Server, Azure SQL Databases including Managed Instance and PostgreSQL. ADS is the tool when you work on Linux or MAC, why not on Windows also. Its highly extensible, probably not very mature yet, but still a great option for those seeking light and flexible coding tool.