

Data Warehouse

A traditional sql database would have a bunch of tables usually with primary keys and foreign keys defining the relationships between them. The database is also normalized so that there isn't duplicate data in the table. This would usually be the structure for transactional purposes where there needs to be high write capacity in the database.

For Analytical purposes we need a Data Warehouse. The tables would be in the form of fact tables and dimension tables.

Fact Tables

- These are measurements or metrics that correspond to facts
- For example, a sales table would have records of all the sales that have been made
- The sales data are facts that sales have actually been made

Dimension Tables

- These help provide some sort of context to the facts that are being presented
- For example, what are the products that were sold, who were the customers who bought the product, etc. is context about facts presented in the fact table

For the dimension tables, it is not necessary that the data would only come from the sql database. It could come from various sources.

So, let's say if there are two or more sources for a particular dimension table, then we would make use of something called a surrogate key. It helps uniquely identify each row in the table. So even if there are matching ids from two different sources, the surrogate key would never be the same for the two and hence every row would get its own unique identifier that way.

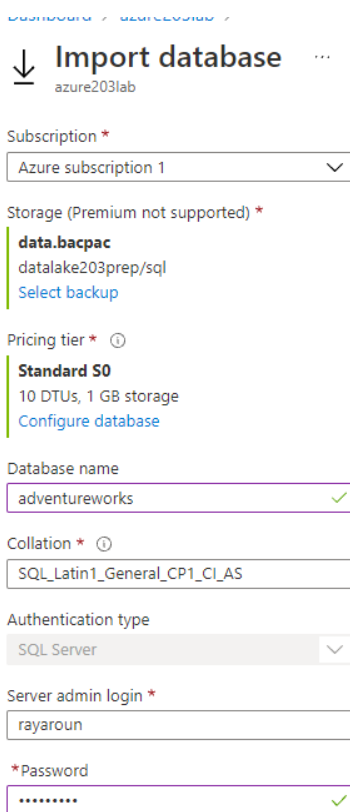
We can make use of the identity column feature in Azure Synapse for tables to generate the unique id.

Also, its best practice to not have null values in your dimension tables.

Creating a new Data Warehouse

Download [this](#) bacpac file so that we can use the data to create a warehouse. Upload this file to the same data lake where we have our other files that we have been using (Log.csv , parquet files). Create a subdirectory in the raw folder by the name of sql and upload the bacpac file there.

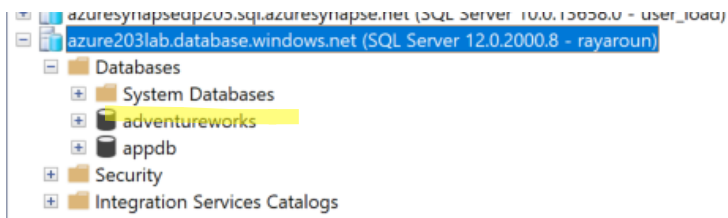
Begin by heading over to your SQL database server. In my case it is named azure203lab. Click on the import database button. Click on select backup. It would load up all the places from where it can load a backup, select your bacpac file from the datalake. Configure the database according to your needs, give a database name finally enter your password for the server admin log in.



The screenshot shows the 'Import database' configuration page in the Azure portal for the resource 'azure203lab'. The page includes the following fields and options:

- Subscription ***: A dropdown menu showing 'Azure subscription 1'.
- Storage (Premium not supported) ***: A section showing the selected backup file 'data.bacpac' located at 'datalake203prep/sql'. A 'Select backup' link is visible.
- Pricing tier ***: A section showing the selected tier 'Standard S0' with details '10 DTUs, 1 GB storage' and a 'Configure database' link.
- Database name**: A text input field containing 'adventureworks' with a green checkmark icon.
- Collation ***: A dropdown menu showing 'SQL_Latin1_General_CP1_CI_AS'.
- Authentication type**: A dropdown menu showing 'SQL Server'.
- Server admin login ***: A text input field containing 'rayaroun'.
- *Password**: A password input field with masked characters and a green checkmark icon.

Now we can go to the SSMS check out our database in the sql server.



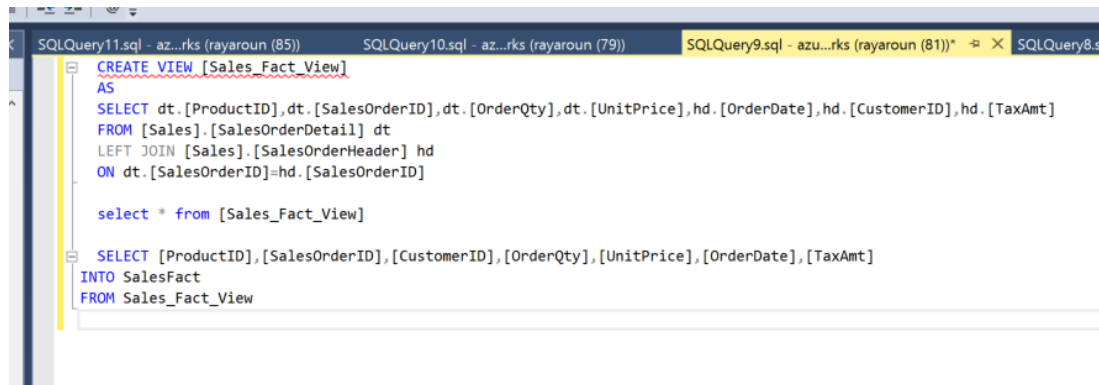
Building a fact table

These are usually large in size and contain measurable facts. These would contain the primary keys used in the dimension tables.

We would be using [this](#) scrip to create our fact table. Right click on adventureworks and query it. We are creating a fact table based on the SaledOrderDetail table and the SaledOrderHeader table.

We are creating the Fact table in the SQL database itself but the right approach is to create the table in Azure Synapse Directly.

We are first creating a view and then creating a table based on that view.



This is how it looks finally –

	ProductID	SalesOrderID	CustomerID	OrderQty	UnitPrice	OrderDate	TaxAmt
1	776	43659	29825	1	2024.994	2011-05-31 00:00:00.000	1971.5149
2	777	43659	29825	3	2024.994	2011-05-31 00:00:00.000	1971.5149
3	778	43659	29825	1	2024.994	2011-05-31 00:00:00.000	1971.5149
4	771	43659	29825	1	2039.994	2011-05-31 00:00:00.000	1971.5149
5	772	43659	29825	1	2039.994	2011-05-31 00:00:00.000	1971.5149

So it has facts as to what product was ordered, what was the sales id, how many products were ordered etc. Now we can have corresponding dimension tables that would have information about products based on product ids, costumers based on customer id, etc.

Building a Dimension table

For this part we would be using [this](#) script. Like previously mentioned, the dimension tables would play the role of supporting the information in the fact tables so like there should be a dimension table that would have customer information based on customer id, we create a customer dimension table.

We create this customer table from the Customer table joined with the Store table.

```
SQLQuery11.sql - az...rks (rayaroun (85))  SQLQuery10.sql - az...rks (rayaroun (79))  SQLQuery9.sql - azu...rks (rayaroun (81))
CREATE VIEW Customer_view
AS
SELECT ct.[CustomerID],ct.[StoreID],st.[BusinessEntityID],st.[Name] as StoreName
FROM [Sales].[Customer] as ct
LEFT JOIN [Sales].[Store] as st
ON ct.[StoreID]=st.[BusinessEntityID]
WHERE st.[BusinessEntityID] IS NOT NULL

SELECT [CustomerID],[StoreID],[BusinessEntityID],StoreName
INTO DimCustomer
FROM Customer_view
```

Looks like –

	CustomerID	StoreID	BusinessEntityID	StoreName
1	1	934	934	A Bike Store
2	2	1028	1028	Progressive Sports
3	3	642	642	Advanced Bike Components
4	4	932	932	Modular Cycle Systems
5	5	1026	1026	Metropolitan Sports Supply
6	6	644	644	Aerobic Exercise Companv

Next we create a product dimension table.

```
SQLQuery11.sql - az...rks (rayaroun (85))  SQLQuery10.sql - az...rks (rayaroun (79))  SQLQuery9.sql - azu...rks (rayaroun (81))*  SQLQuery8.sql - azu...rks (rayaroun (67))*
CREATE VIEW Product_view
AS
SELECT prod.[ProductID],prod.[Name] as ProductName,prod.[SafetyStockLevel],model.[ProductModelID],model.[Name] as ProductModelName,category.[ProductSubcategoryID],category.[N
FROM [Production].[Product] prod
LEFT JOIN [Production].[ProductModel] model ON prod.[ProductModelID] = model.[ProductModelID]
LEFT JOIN [Production].[ProductSubcategory] category ON prod.[ProductSubcategoryID]=category.[ProductSubcategoryID]
WHERE prod.[ProductModelID] IS NOT NULL

SELECT [ProductID],[ProductModelID],[ProductSubcategoryID],ProductName,[SafetyStockLevel],ProductModelName,ProductSubCategoryName
INTO DimProduct
FROM Product_view
```

Which looks like –

	ProductID	ProductModelID	ProductSubcategoryID	ProductName	SafetyStockLevel	ProductModelName	ProductSubCategoryName
1	680	6	14	HL Road Frame - Black, 58	500	HL Road Frame	Road Frames
2	706	6	14	HL Road Frame - Red, 58	500	HL Road Frame	Road Frames
3	707	33	31	Sport-100 Helmet, Red	4	Sport-100	Helmets
4	708	33	31	Sport-100 Helmet, Black	4	Sport-100	Helmets
5	709	18	23	Mountain Bike Socks, M	4	Mountain Bike Socks	Socks

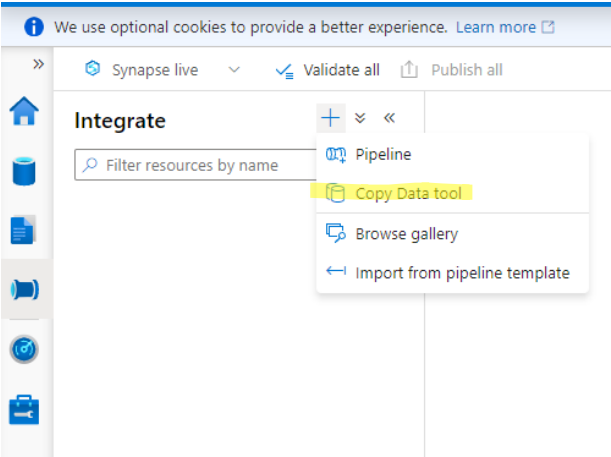
Transferring these table to the Dedicated SQL Pool

We can perform the load operation using the integrate feature that is present in the Synapse Studio. This is done using creating pipelines which are based on Azure Data Factory. For now we would be creating a pipeline that would copy our data from the sql tables as it is onto our dedicated sql pool.

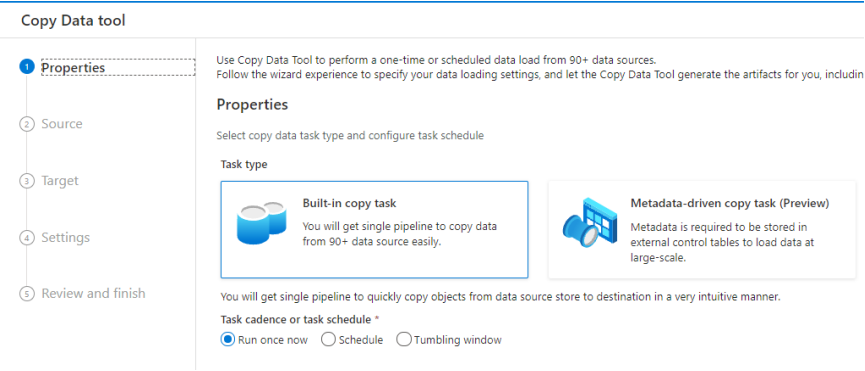
For this part we would be using [this](#) script.

Connect to your dedicated sql pool on your SSMS and Create the SalesFact table followed by DimCustomer followed by DimProduct.

Head over to the Synapse Studio, go to the integrate section from the left-hand menu. Click on the plus sign and select Copy Data tool.



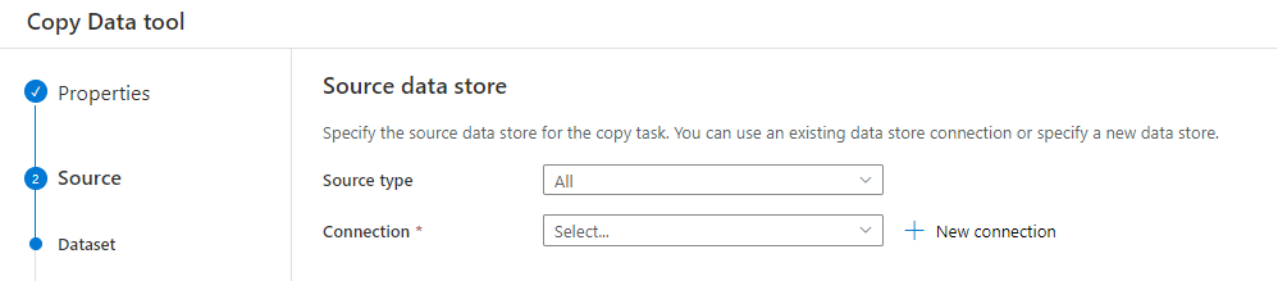
We see the following screen – we select Run once now



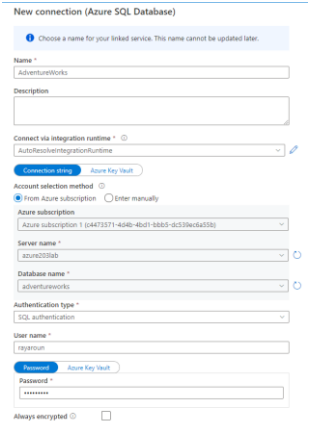
Note - Whenever we are building a pipeline where the target is on Azure Synapse, then there needs to be a storage account set up to be used as a staging area. This storage account would take data from the source and perform any sort of work required on the data and then send the data to Synapse.

For this purpose, we create a new container in our Gen2 storage account and name it synapse.

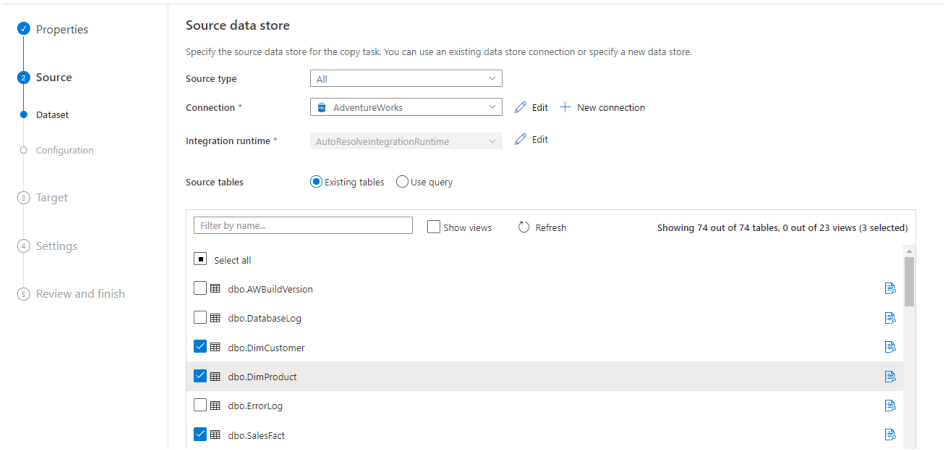
In Synapse studio after click next we get – we select new connection here.



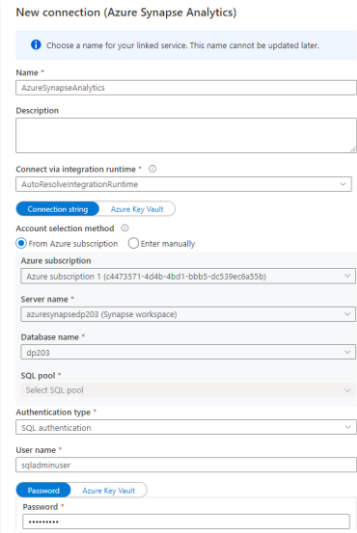
In the new connection screen we go to the Azure tab, select Azure SQL Database. Then we name the connection, select our subscription, server name, database name, authentication as sql authentication and finally enter our credential. We test the connection and then hit create.



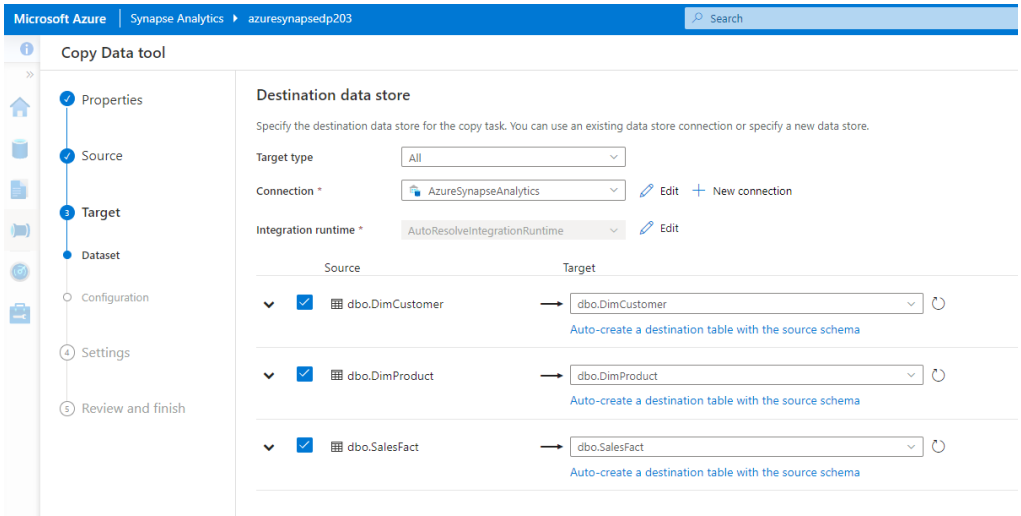
On the next screen after creating the connection, you would be asked the tables that you want to use. We select our fact table and the two dimension tables.



For the next screen we have the again a new connection for our target which is Azure Synapse. We click new connection and select Azure Synapse Analytics from the Azure tab. Enter a name for the connection and select other details just like before.



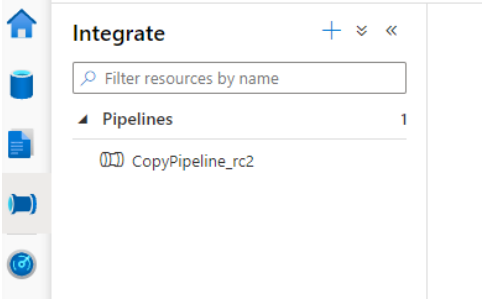
On the next screen you can see it has automatically mapped the tables in the source to the tables in the target because the table names are the same.



Click next. On the next screen we can even do a column based mapping. We again click next.

On the next screen under Staging setting, we would have to like a staging area like previously mentioned. Click on New.

Enter a name for the service, select Azure Data Lake Storage Gen2 in the type, select the Data lake in storage name and click create. On the next screen in storage path browse to the synapse container and select it. In advanced we can see it is using the Polybase technique to copy it. We can select bulk insert as we are copying everything. Click next and finish. It would show up as –



If you go to the monitor tab, you can also see that the pipeline has also succeeded in running.

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Error	Run	Parameters	Annotations	Run ID
CopyPipeline_rc2	10/20/21, 2:17:31 PM	10/20/21, 2:18:19 PM	00:00:48	Manual trigger	Succeeded		Original			9b96e1e1-190a-47c

Meaning the data has been loaded. If we go to the SSMS and select * on the fact or dimension tables, the data would show up.

The fact table shows up as –

ProductID	SalesOrderID	CustomerID	OrderQty	UnitPrice	OrderDate	TaxAmt
725	47415	29864	2	202.332	2012-07-31 00:00:00.000	266.7193
776	43659	29825	1	2024.994	2011-05-31 00:00:00.000	1971.5149
771	43875	29624	6	2039.994	2011-07-01 00:00:00.000	11871.5033
756	44093	29708	1	874.794	2011-08-01 00:00:00.000	1021.8926
757	44298	29824	1	874.794	2011-08-31 00:00:00.000	2656.7147
774	44514	30107	2	2039.994	2011-10-01 00:00:00.000	4095.822
750	44622	13771	1	3578.27	2011-10-09 00:00:00.000	286.2616
776	44794	29915	2	2024.994	2011-10-31 00:00:00.000	388.7988
749	45143	29278	1	3578.27	2011-12-12 00:00:00.000	286.2616
745	45329	29507	1	809.76	2012-01-01 00:00:00.000	5940.8882
729	45556	29746	2	183.9382	2012-01-29 00:00:00.000	2199.474