Power Query VS SQL

First Scenario: Accessing raw data directly using only Power Query:

Here Power Query does the entire job. We are going to access the data as it is. Once in the editor we will add or remove columns, split or concatenate fields, replace values and/or add new ones, and so on, until we have a dataset shaped as we desire.

Power Query doesn't have all the transformations that we would like with a couple of clicks, however with some coding and a few more steps, we can get virtually any transformation. This option can be applied with low resources in infrastructure, technology and techy knowledge.

Advantages:

1. **Quick access to multiple types of sources:** Power Query can access directly to a lot of different types of sources. We don't need any additional integration process
2. **Easy to develop:** Power query is intuitive and straightforward. Anyone with few databases knowledge can make almost any transformation
3. **No need for other tools:** Power Query is capable of completing the whole process

Disadvantages:

1. **Many steps:** In a lot of cases we need to add many steps
2. **High refreshing times:** When the report refreshes, all the historical data is transformed. This can cause long refreshing times if there is a lot of data
3. **Several reports maintenance:** If we can't share the data model among several reports, we need to maintain each file
4. **Failures due to source structure changes:** Any changes on the structure of the sources may cause errors during the transformation process

Second scenario: Accessing raw data in databases through queries:

Power query will not be the do everything. The transformations here will be minimal, or even nonexistent. Here we will use the strength of SQL Server. From now on it will be necessary to have the data integrated into a database.

The idea with this scenario is to develop a single query for each dataset. This query will access the raw data and transform it into the desired result set. In many cases, one simple query may be enough, in some others we will need to create a couple of subqueries.

When the report is quite big, we will improve the refreshing times and even the report size thanks to the database engine.

Advantages:

1. **Few steps:** We can do almost all the transformations in a single query
2. **Better performance:** We'll reduce the refresh time and server impact
3. **Changes on the source structure will not affect the process:** The integration process will take care of the changes on the source's structure

Disadvantages:

1. **Required database knowledge**: Database queries are not so intuitive nor the interface so friendly
2. **Medium refreshing times:** We are yet transforming all the data every time. This can increase the refresh time
3. **Several reports maintenance:** This issue is not solved yet. We must maintain every report individually if we can't share the data model

Third scenario: Stored procedures or views that manage the transformations:

Now we want the same query as scenario 2 for the raw data, but we put the query inside a stored procedure or a view. We will isolate the transformations from the application layer. The main goal here is to use the SPs or views among all the reports even when we can't share the data model. The maintenance is now centralized in the data layer, while the application only receive transformed data.

It's good to mention that we may not have the required permission on the databases and we'll need to go to other team inside our organization to create the SPs or views.

Advantages:

1. **Same from the previous scenario:** The process is quite similar, thus, similar advantages
2. **Shared dataset and transformations:** We can share SPs or views and the maintain only one process even if we can't share the data model

Disadvantages:

1. **Required database knowledge:** Like before, we will need some database knowledge
2. **Medium refreshing times:** We may not see a lot of improvement on the refreshing times compared to the previous scenario
3. **Need for more DB permissions:** Now we need to have CREATE and EXECUTE permission

Forth and last scenario: Accessing production tables with the data already transformed:

Now we will use a small ETL process to feed our data model. The idea is to have production tables separated from the raw data in different locations, servers, instances or databases. They will be ready to feed our reports with limited historical data and optimized characteristics for the querying. We will update this production tables with the same frequency that we refresh the reports.

To do this we'll use stored procedures and scheduled jobs run by the SQL Server Agent. The stored procedure will remove all the rows that surpass the historical parameters from the production tables. Then, the process will take the last added or updated rows from the raw data and transform them. If no errors are raised it will make an incremental load into the production tables.

With this process we transform few rows at a time, depending on the required reprocess defined by the business. On the other hand, any issues on the sources or the access to it will not impact the production tables nor the report itself. Power BI will then access only transformed data.

Advantages:

1. **Shared dataset:** We can share production tables among the reports
2. **Few rows to process each time:** You will only need to process a few days every time
3. **Less refreshing time:** Refreshing times are improved since a we are only doing a simple select with no transformations
4. **Isolation of the data:** All the production data is isolated. Not only the process from the visualization tool, but also from the raw data itself

Disadvantages:

1. **Required DB knowledge:** It will be even more important now since there are more tasks to accomplish
2. **More DB permissions required:** Now we also need to create, scheduled and execute jobs. We will also need inserting and deleting permissions for the production tables
3. **The need for a more robust infrastructure:** The infrastructure resources must be robust and may even need the use of other ETL common tools like SSIS in some cases