Azure SQL Data Warehouse
PoC Playbook
Get ready for a PoC

To make an informed business decision for hosting your cloud-based data warehouse on Azure SQL Data Warehouse a proof of concept (PoC) project may be called for. A PoC project will identify your key goals and business drivers that your data warehousing solution must support and will test key metrics and prove key behaviors that are critical to your data warehousing success. A proof of concept is a quickly executed project that focuses on key questions and is not designed to be deployed to a production environment but is designed to execute quick tests and then be discarded.
Preparing for your PoC

Before you begin planning your PoC project, do the following:

- Identify any restrictions or guidelines your organization has about moving data to the cloud
- Identify executive/business sponsorship for a cloud-based data warehouse project
- Review Azure SQL Data Warehouse Workload Patterns and Anti-Patterns to verify that your data workload is appropriate for Azure SQL Data Warehouse

Now that you have determined that there are no immediate blockers, start preparing for your PoC.

If you are new to Azure SQL Data Warehouse, review the Azure SQL Data Warehouse - massively parallel processing (MPP) architecture where you can get an overview of the architecture.

Develop an understanding of these key concepts:
- Separation of compute and storage
- Data distribution
- Clustered columnstore indexing
Key concepts

**Compute**
Azure SQL Data Warehouse decouples compute resources from storage so that you can better manage your data warehouse and control costs. You can grow and shrink your computing power independent of your storage. You can pause compute entirely so that you only pay for storage. You can scale up your compute for busy times to support more users or large loads and then scale it down during less intense processing times. Scaling and pausing can be used effectively to reduce costs. Your PoC tests should include query and load tests at different scales to compare performance at a variety of scale. Learn more here: Manage Compute in Azure SQL Data Warehouse.

**Data distributions**
Azure SQL Data Warehouse offers three data distributions; Hash, round robin, and replicate. Your PoC should analyze the appropriate distribution for your tables. Dependent upon your specific use cases the PoC may be the perfect time to test different distributions for some tables and compare the behavior using different distributions. For tips on distributed tables review Guidance for designing distributed tables in Azure SQL Data Warehouse and Design guidance for using replicated tables in Azure SQL Data Warehouse.

**Indexing**
Azure SQL Data Warehouse offers several indexing options for your tables; Clustered columnstore Index (CCI), Clustered Index, Nonclustered Index, and Heap (no index). CCI is the default indexing structure for tables in Azure SQL Data Warehouse. CCI provides excellent data compression and facilitates query performance over large datasets. Prior to your PoC read Indexing tables in SQL Data Warehouse to understand your indexing options and Columnstore indexes: Overview to learn more about CCI in particular. During your PoC testing tables with different indexing options may give you better insight into achieving the best query and loading performance for your implementation.

**Partitioned tables**
Partitioned Tables can be very useful for managing data and improving query performance. Please read Partitioning tables in SQL Data Warehouse that covers table partitioning implementation considerations in Azure SQL Data Warehouse.
Set the goals for your PoC

A successful PoC project requires planning. Identify why you are doing a PoC. Write out clear goals for your PoC. What do you want as the outputs of your PoC and what will you do with those outputs? Who will utilize the outputs? What will define a successful PoC?

Keep in mind that a PoC should be a short and focused effort to quickly prove or test a limited set of concepts. If you have a long list of items to proof you may want more than one PoC with gates between them where you determine if you need the next PoC.

As you consider your PoC keep some of the following questions in mind to help you shape your goals:

Are you migrating from an existing data warehouse environment?
- Are you migrating and want to make as few changes to existing ETL and data warehouse consumption as possible?
- Are you migrating but want to do some extensive improvements along the way?

Are you building an entirely new data warehouse environment (greenfield)?

What are your current pain points?
What new business needs are you being asked to support?
What are the SLAs you are required to meet?
What will be the workloads - ETL, batch queries, analytics, reporting queries, interactive queries?
Example PoC goal setting

Our PoC will be successful if we have the data needed and have completed the testing identified to determine if Azure SQL Data Warehouse will support our Enterprise Data Warehousing.

We will have determined if we can move to the next phase or if additional PoC testing is needed to finalize our decision. We will be able to make a sound business decision backed by data.

Why are we doing a PoC?
We need to know that the query performance for our big complex reporting queries will meet our new SLAs.
We need to know the query performance for our interactive users.
We need to know if our existing ETL processes are a good fit and where improvements need to be made.
We need to know if we can shorten our ETL run times and by how much.
We need to know if our existing user reports and tools can work with ADW.

At the conclusion of this PoC:
We will have the data to determine if our query performance requirements can be met.
We will have tested some of our existing reporting and can identify the work that will need to be completed to migrate our portfolio of reports to the new environment.
We will have tested data loading and will have the data to estimate the effort our initial migration load will require.
We will have tested loading and can determine if our ETL processing requirements can be met.
We will have gained insight to better estimate the effort required to complete the implementation project.
We will have tested scale and scaling options and have the data to better configure our platform.
We will have a list of items that may need more testing.
Plan your PoC project

Using your goals identify specific tests to execute to support those goals and provide the outputs you identified.

It is important to make sure that you have at least one test to support each goal and expected output. Identify specific queries, reports, ELT, and other processes that will be executed so that a very specific dataset can be identified.

Examples of the needed level of specificity in planning

<table>
<thead>
<tr>
<th>GOAL</th>
<th>OUTPUT</th>
<th>TESTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>We need to know the query performance for our interactive users</td>
<td>We will have the data to determine if our query performance requirements can be met</td>
<td>Queries A, B and C are identified as a good performance test as they are commonly executed by our interactive users. Queries X, Y and Z is identified as a good performance tests as they contain known complex relationships and processing and are executed interactively. Execute interactive tests with concurrency of up to nn users using queries A, C and X</td>
</tr>
<tr>
<td>We need to know if our existing user reports and tools can work with ADW</td>
<td>We will have tested some of our existing reporting and can identify the work that will need to be completed to migrate our portfolio of reports to the new environment</td>
<td>These 20 reports (…) will be tested as part of the PoC</td>
</tr>
<tr>
<td>We will have tested data loading and will have the data to estimate the effort our initial migration load will require.</td>
<td>We will have tested loading and can determine if our ETL processing requirements can be met</td>
<td>Test the daily update load process for xxxxxxx To test the queries and reports identified above and the daily update load process will load these nn tables (…) as our PoC dataset Execute daily load update process concurrently with interactive queries of concurrency of nn.</td>
</tr>
</tbody>
</table>

Refine your tests by adding multiple testing scenarios

The flexibility of SQL DW makes it easy to test different scale, different distribution, different indexing, and different resource groups compare performance and behavior.

| TESTS | We will execute tests n, m, and p using scale of 500DWU, 1000DWU, 2000DWU | Test load scenarios A and B using different dynamic and static resource groups Test all interactive scenarios using different static resource groups to gather concurrency behavior data |
Evaluate your PoC dataset

From the specific queries, processes and user interactions that you have identified the dataset required to support these tests should also have been specifically identified. Now take some time to review this dataset.

You now need to verify that the dataset will adequately represent your future processing on Azure SQL Data Warehouse in both content, complexity and scale. Do not use a dataset that is too small (< 1TB) - you will not see representative performance. Do not use a dataset that is too big - the PoC is not the time to complete the full data migration.

Identify the distribution pattern and indexing option for each table. If there are any questions about distribution or indexing or partitioning, add tests to you PoC to answer your questions. Remember that you may want to test more than one distribution option or indexing option for certain tables.

Make sure you have checked with business owners for any blockers for moving this data to the cloud. Identify any security or privacy concerns.

Refine your tests by adding multiple table testing scenarios to clarify any table structure questions that have arisen.

---

**Tests**

- We will test FactTableB distributed by Key1 and then by Key2
- We will test DimCustomer (4B rows) as a replicate table and as round robin distribution and as a hash distribution by CustomerID
- We will test DimProduct (1M) rows as a CCI and as a Clustered Index table
Create high-level architecture for your PoC

Based upon the high-level architecture of your future architecture, identify the components that will be a part of your PoC. Your high-level future architecture likely contains many data sources, numerous data consumers, Big Data components and possibly Machine Learning and AI data consumers. Create an architecture for your PoC that specifically identifies the components that will be part of the PoC and clearly identifies which components will not be part of the PoC testing.

If you are already using Azure, identify any resources you already have in Azure (AAD, ExpressRoute, etc.) that can be used during the PoC. Also identify what Azure Regions your organization prefers. Now is a great time to identify the throughput of your ExpressRoute connection and check with other business users that your PoC can consume some of that throughput without adverse effect on production solutions.

Identify PoC Resources

- Specifically identify the resources and commitment that will be required to support your PoC.
- A business representative to oversee requirements and results
- An application data expert, to source the data for the PoC
- An Azure SQL Data Warehouse specialist
- An expert advisor, to optimize the PoC tests
- Resources that will be required for specific components of your PoC project, but not necessarily required for the duration of the PoC. These resources could include network admin resources, Azure Admin Resources, Active Directory Admins, etc.

Since you are evaluating a new platform, we recommend engaging an expert advisor to assist with your PoC. Microsoft’s partner community has global availability of expert consultants, able to demonstrate the features and performance of Azure SQL Data Warehouse. You can find local partners at Solution Providers Home.

Set PoC Timeline

Review the details of your PoC planning and business needs to identify a time constraint for your PoC. Make realistic estimates of the time that will be required to compete the tasks in your plan. The time to complete your PoC will be influenced on the size of your PoC dataset, the number of database objects (tables, views, stored procedures, etc.), and the complexity of your database objects, the number of interfaces you are testing, etc. If you find your PoC is estimated to run longer than 4 weeks, consider reducing the PoC to keep focus on the highest priority goals. Get buy in from all the lead resources and sponsors for the timeline before continuing.
Run your PoC Project

Execute your PoC project with the discipline and rigor of any production project. Execute according to plan and have a change request process in place to prevent your PoC from growing and changing out of control.

Example high level tasks

- Provision Azure SQL Data Warehouse, Storage Accounts, and all Azure resources identified in the PoC plan.
- Deploy Schema (DDL)
- Deploy Objects (DML, views, stored procedures)

Load PoC dataset

Make data available in Azure by extracting from source or creating sample data in Azure as needed.

Best practices for loading data into Azure SQL Data Warehouse >

Land data to Azure storage >

Set up monitoring

Monitoring resource utilization and query activity in Azure SQL Data Warehouse >

Azure Advisor SQL Data Warehouse Recommendations >

Extract management insights from SQL Data Warehouse with SQL Operations Studio >

Execute tests

Many tests can be executed in parallel (i.e. the ETL can be tested on an independent track from Report Tool connection testing) Record your results in a consumable and readily understandable format
Evaluate results

When all the PoC tests are completed you can evaluate the results.

Begin by evaluating if the PoC goals have been met and the desired outputs collected.

Note where additional testing is warranted or where additional questions were raised.
Next Steps

To further your discovery, there are several Best Practices documents available within the SQL Data Warehouse Documentation:

Cheat sheet for Azure SQL Data Warehouse >
Best practices for Azure SQL Data Warehouse >
Best practices for loading data into Azure SQL Data Warehouse >
Development best practices for Azure SQL Data Warehouse >

Get started today

Start your Azure SQL Data Warehouse Free Trial >
See TPC-H and TPC-DS benchmark results for cloud data warehouse providers >