

2.1 Introduction to project:

In the modern era of data-driven decision-making, the ability to efficiently manage and analyze large volumes of data is crucial for businesses aiming to maintain a competitive edge. This project focuses on the development and implementation of an automated data engineering pipeline that seamlessly manages and analyzes orders data, leveraging the advanced capabilities of Amazon S3 and Microsoft Azure.

The project starts with raw orders data stored in Amazon S3, a scalable storage solution that provides high durability and availability. To streamline the data processing workflow, the pipeline employs an event-driven architecture. This means that as soon as new files are added to the S3 bucket, the system automatically triggers the data extraction process. This automation significantly reduces the need for manual intervention, ensuring that data processing is timely and efficient.

Once the data extraction is initiated, the pipeline securely transfers the data from Amazon S3 to Azure Data Lake Storage (ADLS). ADLS offers a highly scalable and secure data storage solution, optimized for big data analytics. This transfer process is designed to be efficient and reliable, maintaining data integrity throughout the process.

In Azure Data Lake Storage, the raw data undergoes a series of transformations using Azure Data Factory. Azure Data Factory is a cloud-based data integration service that orchestrates and automates the movement and transformation of data. The transformation logic applied in this stage includes data cleaning, aggregation, and structuring, which prepares the data for comprehensive analysis. This step ensures that the data is in a consistent and usable format, enhancing its value for analytical purposes.

After the data is transformed, it is loaded into an Azure SQL Database. Azure SQL Database is a fully managed relational database service that provides high availability,

performance, and security. By loading the data into this database, the project enables advanced querying and analysis capabilities. This setup allows for real-time insights into the orders data, facilitating data-driven decision-making and operational efficiency.

The project demonstrates the seamless integration of diverse cloud services, showcasing best practices in data engineering such as ensuring data integrity, optimizing performance, and maintaining scalability. It not only highlights the technical prowess required to integrate AWS and Azure platforms but also underscores the importance of automation in reducing manual workloads and improving processing times.

Furthermore, this project lays the foundation for future enhancements. Potential upgrades include incorporating real-time data processing to handle streaming data and integrating additional data sources to enrich the analysis. Advanced analytics using machine learning techniques can also be added to derive deeper insights and predictive capabilities, driving even greater business value.

In summary, this automated data engineering pipeline project exemplifies the effective use of cloud technologies to manage and analyze orders data efficiently. By automating data extraction, transfer, and transformation processes, it provides a robust solution for real-time data analysis, empowering businesses with the insights needed to make informed decisions and optimize their operations.

2.2 Background of study:

The rapid advancement of technology and the exponential growth of data have transformed the landscape of business operations and decision-making. In this context, efficient data management and analysis have become pivotal for organizations striving to harness the power

of big data. Companies generate vast amounts of data daily, which, if processed and analyzed effectively, can provide valuable insights and drive strategic decisions. However, managing such large volumes of data presents significant challenges, including data integration from multiple sources, ensuring data quality, and achieving real-time data processing.

Cloud computing platforms like Amazon Web Services (AWS) and Microsoft Azure have emerged as key enablers in addressing these challenges. AWS offers a suite of services for data storage, processing, and analytics, with Amazon S3 being one of the most widely used storage solutions due to its scalability, durability, and cost-effectiveness. Microsoft Azure, on the other hand, provides a robust set of tools for data engineering and analytics, including Azure Data Lake Storage (ADLS), Azure Data Factory, and Azure SQL Database. These platforms offer comprehensive solutions for building scalable and efficient data pipelines.

The integration of AWS and Azure services presents a powerful approach to creating an end-to-end data engineering pipeline. By leveraging the strengths of both platforms, organizations can achieve seamless data migration, transformation, and analysis.

In this project, we aim to design and implement an automated data engineering pipeline that extracts raw orders data from Amazon S3, transfers it to Microsoft Azure, and transforms it for comprehensive analysis in SQL. This pipeline will demonstrate best practices in data engineering, including data extraction, secure transfer, transformation, and loading into a SQL database. By focusing on automation and real-time processing, the project seeks to provide a scalable and efficient solution for managing and analyzing large volumes of orders data.

2.3 Novel features of Project:

This project introduces several innovative features that set it apart from conventional data engineering solutions. Leveraging the strengths of Amazon Web Services (AWS) and Microsoft Azure, the project integrates state-of-the-art technologies to create a robust, automated, and efficient data pipeline. The pipeline features an event-driven architecture,

which automatically triggers data processing upon detecting new files in Amazon S3, ensuring timely data handling with minimal manual intervention. It showcases seamless integration between AWS and Azure, utilizing AWS for scalable storage and Azure for powerful data processing and analytics. Automated data extraction and secure transfer to Azure Data Lake Storage maintain data integrity and privacy, while advanced data transformations using Azure Data Factory prepare the data for meaningful analysis. The system supports real-time data processing, providing up-to-date insights crucial for immediate decision-making. Scalability and performance optimization are achieved through best practices in data engineering, ensuring the pipeline can handle increasing data volumes and complexity. By loading transformed data into Azure SQL Database, the project enables comprehensive data analysis through advanced SQL queries. Designed with future enhancements in mind, the architecture can evolve to incorporate additional data sources, real-time streaming, and machine learning analytics. These features collectively ensure the system delivers timely, accurate insights, driving better decision-making and operational efficiency.

2.4 Problem Statement

In the current data-driven business environment, companies face significant challenges in efficiently managing and analyzing large volumes of data from diverse sources. Specifically, the process of extracting, transforming, and loading (ETL) orders data from storage systems like Amazon S3 into analytical platforms often involves manual intervention, which can lead to delays, errors, and inefficiencies. Additionally, the lack of seamless integration between different cloud services hampers the ability to perform real-time data processing and comprehensive analysis, limiting the potential for timely insights and informed decision-making.

Existing solutions often fall short in providing a fully automated, scalable, and secure pipeline

that can handle the dynamic nature of orders data. This inefficiency affects operational efficiency and hinders the ability to respond quickly to market changes and customer demands. Therefore, there is a critical need for a robust data engineering pipeline that can automate the ETL process, ensure seamless cloud integration, and support real-time data processing and advanced analytics.

This project aims to address these challenges by developing an automated data engineering pipeline that extracts raw orders data from Amazon S3, transfers it to Microsoft Azure, and transforms it for comprehensive analysis in SQL. The pipeline's event-driven architecture will automatically trigger data processing upon detecting new files, ensuring timely and efficient data handling. By leveraging the robust cloud capabilities of AWS and Azure, the project seeks to provide a reliable, scalable, and secure solution for real-time data analysis, ultimately enabling data-driven decision-making and enhancing operational efficiency.

2.5 Project Objective

The objective of this project is to develop and implement a fully automated data engineering pipeline that efficiently manages and analyzes orders data. The project aims to design an event-driven architecture that automatically triggers data extraction from Amazon S3 upon detecting new files, minimizing manual intervention and ensuring timely processing. It seeks to implement a seamless transfer of raw orders data from Amazon S3 to Azure Data Lake Storage, leveraging the strengths of both AWS and Microsoft Azure platforms. Utilizing Azure Data Factory, the project will apply sophisticated data transformation logic, including data cleaning, aggregation, and structuring, to prepare the data for comprehensive analysis. The transformed data will be loaded into Azure SQL Database, enabling advanced querying and analysis to derive meaningful business insights. The project will ensure that the data transfer and transformation processes are secure, maintaining data integrity and compliance with relevant data protection regulations. Additionally, the pipeline will be optimized for scalability

and performance, capable of handling increasing volumes of data and complex queries efficiently. Designed to support real-time data processing, the system will provide up-to-date insights for timely decision-making. Finally, the project aims to build a future-ready architecture that can be easily enhanced with additional data sources, real-time streaming capabilities, and advanced analytics using machine learning techniques. By achieving these objectives, the project will provide a robust, scalable, and efficient solution for managing and analyzing orders data, driving better decision-making and operational efficiency for businesses.

2.6 Scope of Study

The scope of this study encompasses the development and implementation of an automated data engineering pipeline designed to manage and analyze orders data effectively. It includes the process of automatically detecting and extracting new orders data files from Amazon S3 using an event-driven architecture, ensuring timely and efficient data handling. The study also covers the secure and efficient transfer of extracted data from Amazon S3 to Microsoft Azure Data Lake Storage, maintaining data integrity and security throughout the transfer process. Utilizing Azure Data Factory, the pipeline applies automated data transformation logic, including cleaning, aggregating, and structuring the raw data to prepare it for comprehensive analysis. The transformed data is then loaded into Azure SQL Database, optimized for performance and scalability to handle large volumes of data and complex queries efficiently. Additionally, the study involves implementing real-time data processing capabilities, providing up-to-date insights and minimizing latency. Ensuring data security and compliance with relevant data protection regulations is a critical aspect, including data encryption and access controls. The study also focuses on performance optimization techniques, scalable architecture design, and integration with analytical tools for advanced data analysis. Lastly, potential areas for future enhancements are identified, such as incorporating additional data

sources, integrating machine learning techniques for predictive analytics, and expanding real-time data processing capabilities. By addressing these areas, the study aims to create a comprehensive, efficient, and scalable data engineering pipeline that supports data-driven decision-making within organizations.

2.7 Feasibility Study

The feasibility study evaluates the technical, operational, and economic viability of developing an automated data engineering pipeline for managing and analyzing orders data.

Technical Feasibility

The project leverages reliable and scalable cloud platforms: Amazon Web Services (AWS) for data storage (Amazon S3) and Microsoft Azure for data processing and analysis (Azure Data Lake Storage, Azure Data Factory, and Azure SQL Database). The event-driven architecture ensures efficient and automated data processing, leveraging well-established technologies and best practices.

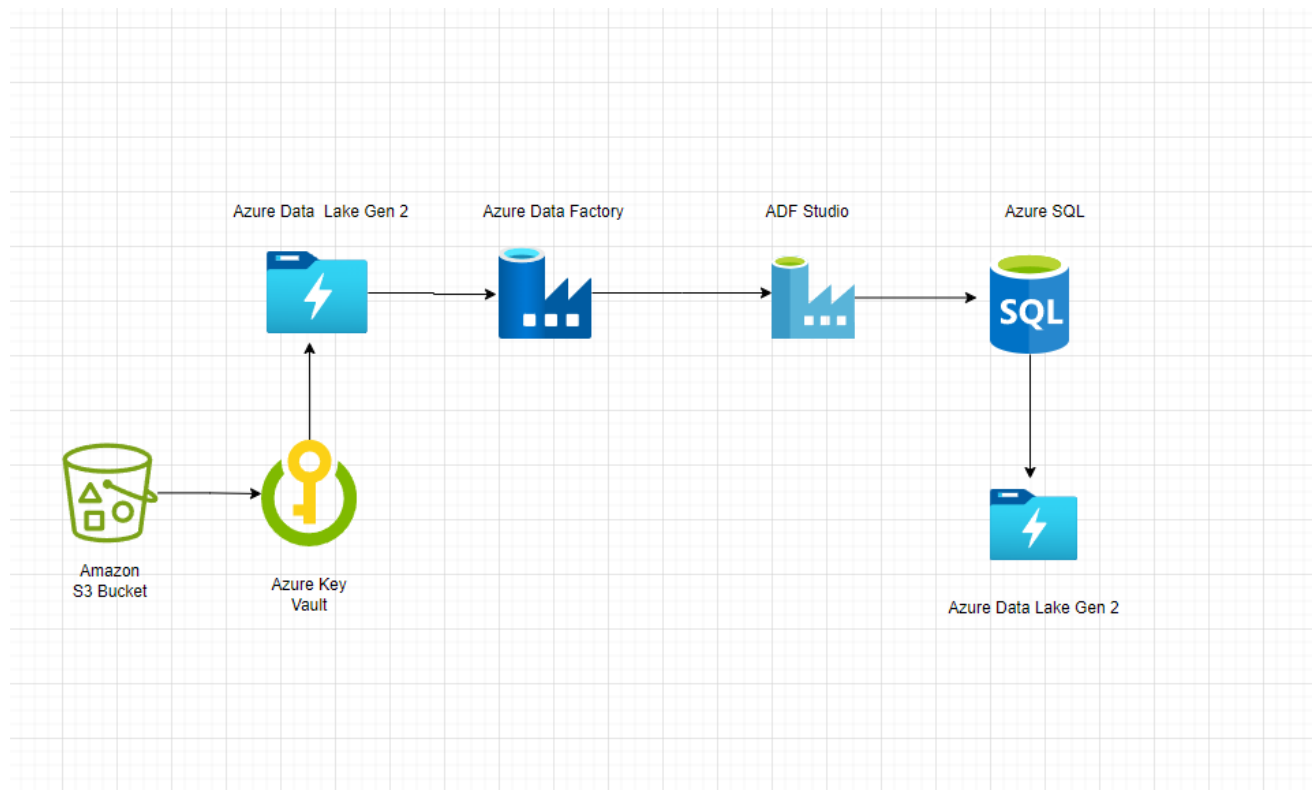
Operational Feasibility

Operationally, the project aims to reduce manual intervention through automation, enhancing efficiency and accuracy. The integration of AWS and Azure services can be managed with proper training and documentation. The scalable nature of cloud services ensures the pipeline can handle increasing data volumes, with comprehensive monitoring and logging for performance tracking.

Economic Feasibility

Economically, the pay-as-you-go pricing models of cloud services reduce upfront infrastructure investments. Automation lowers labor costs and minimizes downtime, leading to efficient resource use. The potential for real-time data insights can drive business growth and

provide a substantial return on investment (ROI). Long-term cost savings are expected from

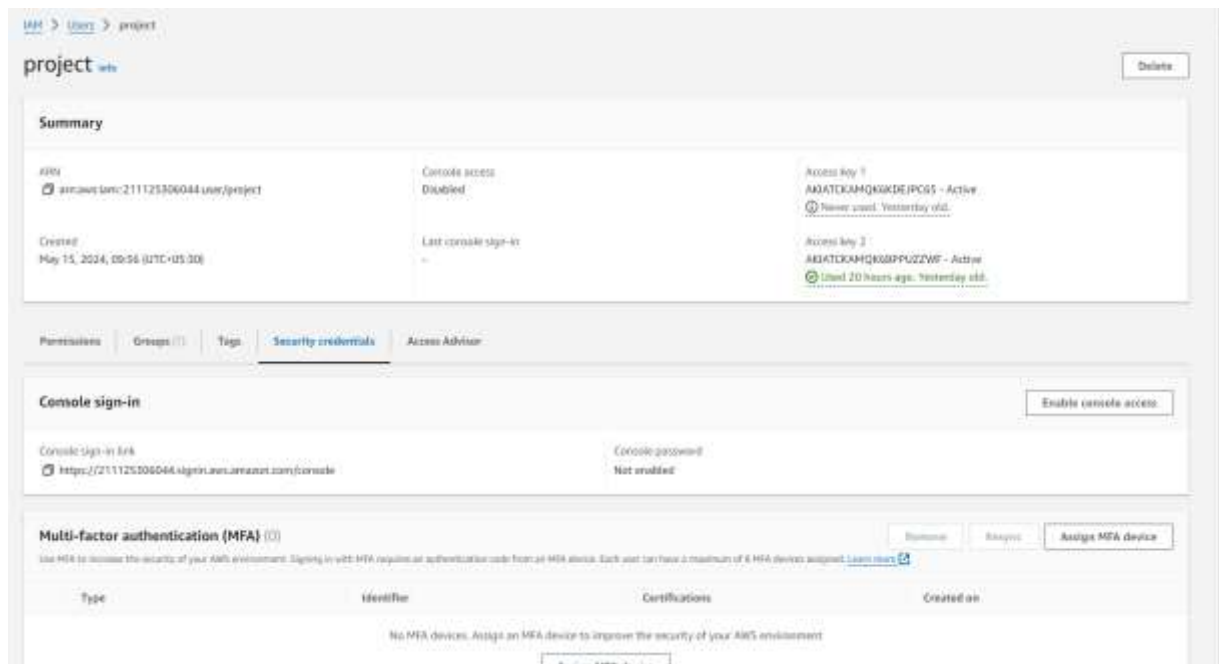


optimized data processing and storage.

5.1 Flow chart of the Project:

5.2 Amazon s3 Bucket Injection

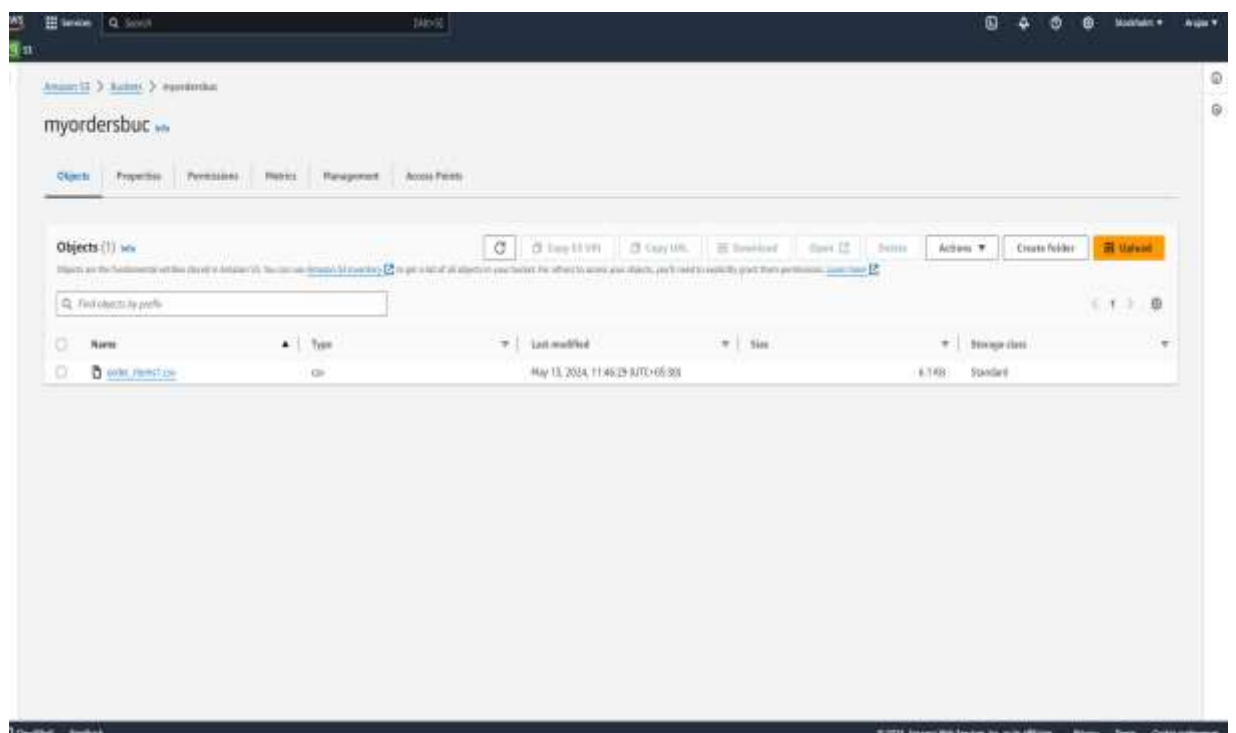
First step is to create a s3 bucket in aws where the Dataset is stored in



the Bucket

Second Step is to create a user in the IAM menu

Third Step is to create an Access Key



Home > project01 Resource group

Search

+ Create Manage view Delete resource group Refresh Export to CSV Open survey Assign tags More Tags Report trouble Open in mobile

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Events

Settings

Cost Management

Monitoring

Automation

Help

Essentials

Subscription: [view](#) | [link](#) | [edit](#)

Subscription ID: T162515-4548-4110-44C9-00226448196

Region: [view](#) | [add](#) | [type](#)

Location: Central India

ADPS view

Resources Recommendations

Filter for any field...

Type equals all location equals all Add filter

Showing 1 to 6 of 6 records

Show hidden types

Name	Type	Location
keyvaultproj01	Key vault	Central India
project01storage	Storage account	Central India
project01databricks	Data factory (V2)	Central India
project01server	SQL server	Central India
project01database	SQL database	Central India
project01storage	Storage account	Central India

Page 1 of 1

Access keys (2)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Create access key

AKIATCKAMQK6KDEJPC65

Actions

Description

Status

-

Active

Last used

Created

None

Yesterday

Last used region

Last used service

N/A

N/A

5.2 Creation of Resource Group

Under Resorce Group Creating the necessary Resources for the project

Resource1: Azure Key Vault creation and Giving the access to Azure

Step1:creating the key vault

Resource1: Azure Key Vault creation and Giving the access to Azure

Home > keyvaultproject135

Overview

- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Access policies
- Events
- Objects
 - Keys
 - Secrets
 - Certificates
- Settings
- Monitoring
- Automation
- Help

Events

Resource group: [keyvaultproject135](#)

Location: Central India

Subscription: [keyvaultproject135](#)

Subscription ID: 5558555-4865-414-8439-06222846919e

Vault URI: <https://keyvaultproject135.vault.azure.net/>

SKU (Pricing tier): Standard

Directory ID: a809c143-6c27-4479-8706-5e05857e420f

Directory Name: Default Directory

Soft delete: [Enabled](#)

Purge protection: [Enabled](#)

Tags: [Add tags](#)

Get started Properties Monitoring Tools + SDKs Tutorials

Manage keys and secrets used by apps and services

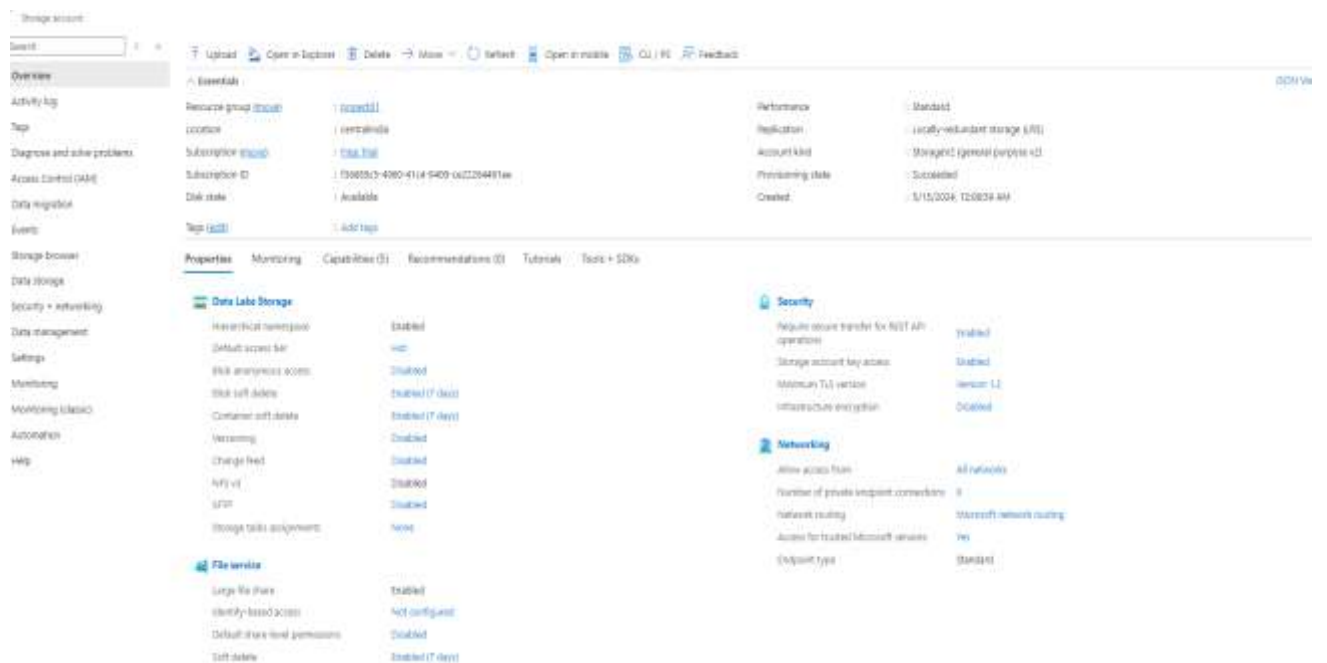
Our recommendation is to use a vault per application per environment (Development, Pre-Production and Production). This helps you to not share secrets across environments and also reduces the threat in case of a breach.

- Control access to key vault**
Assign access policy and determine whether a given service principal, identity, application or user group, can perform different operations on key vault keys, secrets or certificates.
[Access configuration](#)
Access policies
Access control (IAM)
- Enable logging and set up alerts**
Enable logging to monitor how, when and by whom your key vaults are accessed. Monitor performance and configure alerts for key vault metrics e.g., service API latency, error code, throttling.
[View](#)
- Turn on recovery options**
For protection against accidental or malicious deletion, soft-delete is enabled. Turn on purge protection to guard against manual purging of deleted key vaults and items. [Learn more](#)

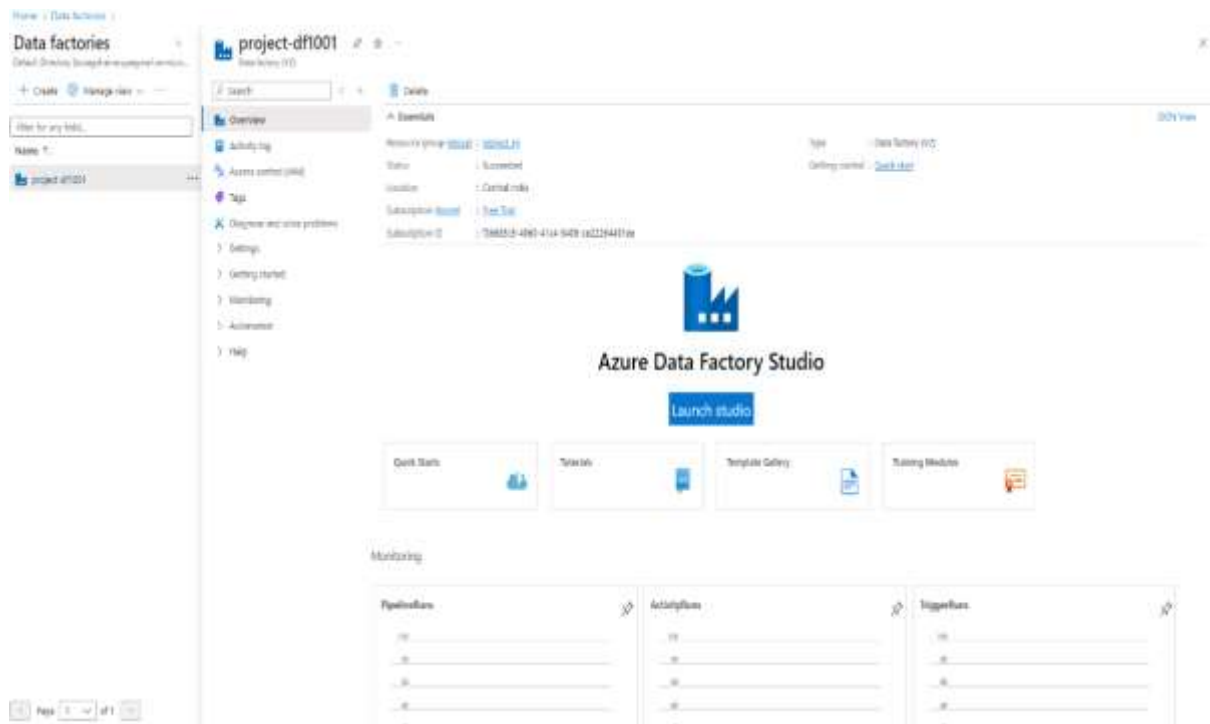
Step1:creating the secret keys



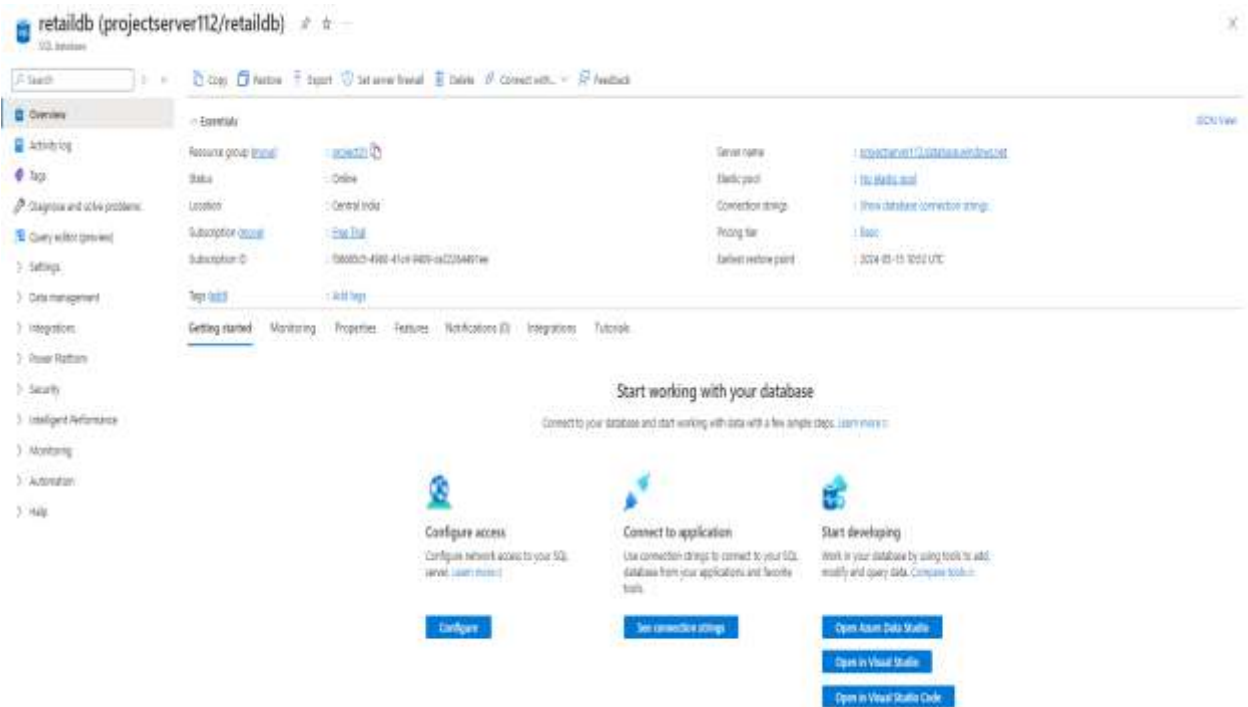
Resource 2: Creating Azure Data Lake Gen 2:



Resource 3:Creating Azure Data Factory



Resource 4: Creation of Server for SQL



Resource 5:SQL Storage

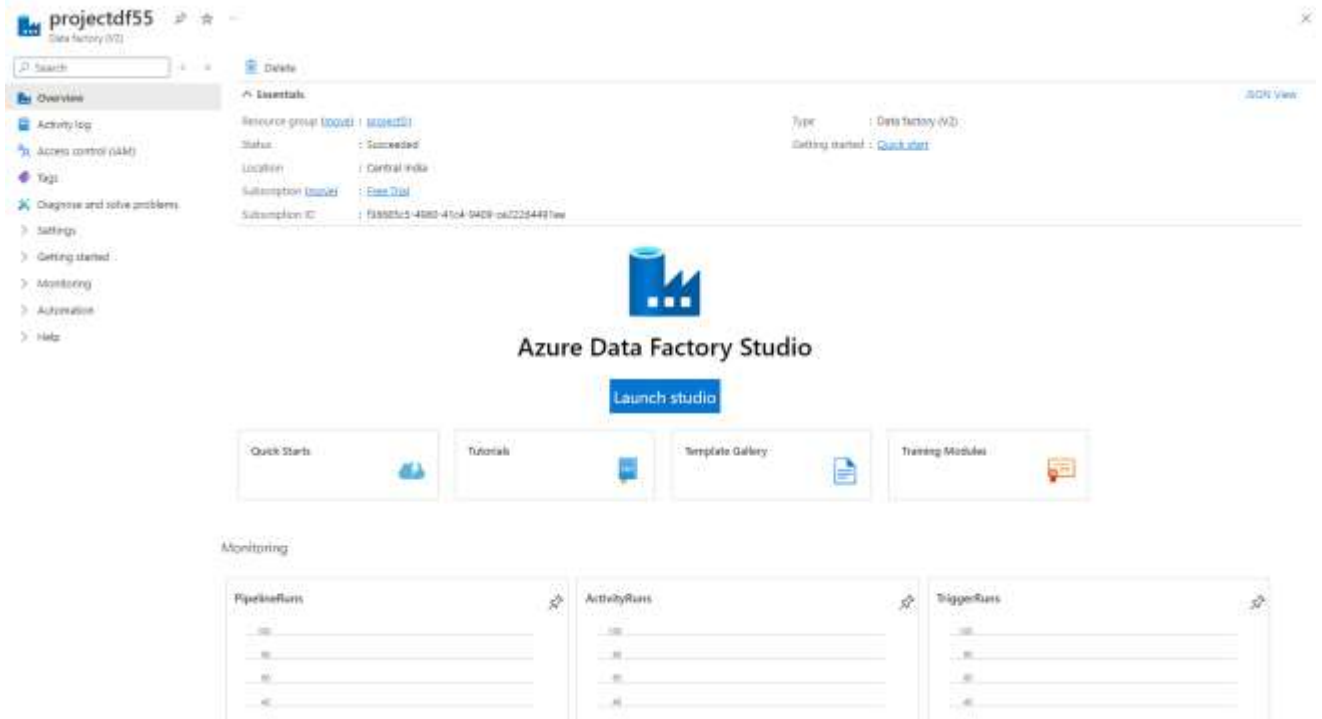
The screenshot displays the Azure portal interface for a resource named 'projectserver12'. The left sidebar shows navigation options like Overview, Activity log, Access control (IAM), Tags, Quickstart, Diagnose and solve problems, Settings, Data management, Security, Intelligent performance, Monitoring, and Automation. The main content area shows the 'Essentials' tab with details such as Resource group (project12), Location (centralindia), Subscription (15882513-4900-4704-9409-0A2226643756), and Subscription ID. Below this, there are several configuration cards: Microsoft Entra admin (NOT CONFIGURED), Microsoft Defender for SQL (NOT CONFIGURED), Automatic tuning (CONFIGURED), Auditing (NOT CONFIGURED), Firewall groups (NOT CONFIGURED), and Transparent data encryption (SERVICE MANAGED KEY). At the bottom, there is a table for 'Available resources' with columns for Name, Type, Status, and Pricing tier. The table shows one entry: 'SQL database' with status 'Online' and pricing tier 'Basic'.

Resource 6:Storage account

The screenshot displays the Azure portal interface for a resource named 'storageaccountproject1'. The left sidebar shows navigation options like Overview, Activity log, Tags, Diagnose and solve problems, Access control (IAM), Data migration, Events, Storage browser, Storage Explorer, Data storage, Security + networking, Data management, Settings, Monitoring (DataD), Monitoring (Kusto), Automation, and HW. The main content area shows the 'Essentials' tab with details such as Resource group (project12), Location (centralindia), Subscription (15882513-4900-4704-9409-0A2226643756), and Subscription ID. Below this, there are several configuration cards: Blob service (Disabled), File service (Enabled), Security (Configured), and Networking (Standard). The 'Blob service' card shows settings like Hierarchical namespace (Disabled), Default access tier (HOT), Blob anonymous access (Disabled), Blob soft delete (Enabled (7 days)), Container soft delete (Enabled (7 days)), Versioning (Disabled), Change feed (Disabled), LFT v3 (Disabled), Allow cross-tenant replication (Disabled), and Storage quota assignments (None). The 'File service' card shows settings like Large file share (Enabled), Identity-based access (Not configured), and Default share-level permissions (Disabled). The 'Security' card shows settings like Require secure transfer for REST API operations (Configured), Storage account key access (Enabled), Minimum TLS version (Version 1.2), and Inheritance exemption (Disabled). The 'Networking' card shows settings like Allow access from (All networks), Number of private endpoint connections (0), Network routing (Microsoft network routing), Access for trusted Microsoft services (Yes), and Endpoint type (Standard).

5.3 Go to Azure data Factory studio in Azure data factory

STEP1:OPEN ADLS

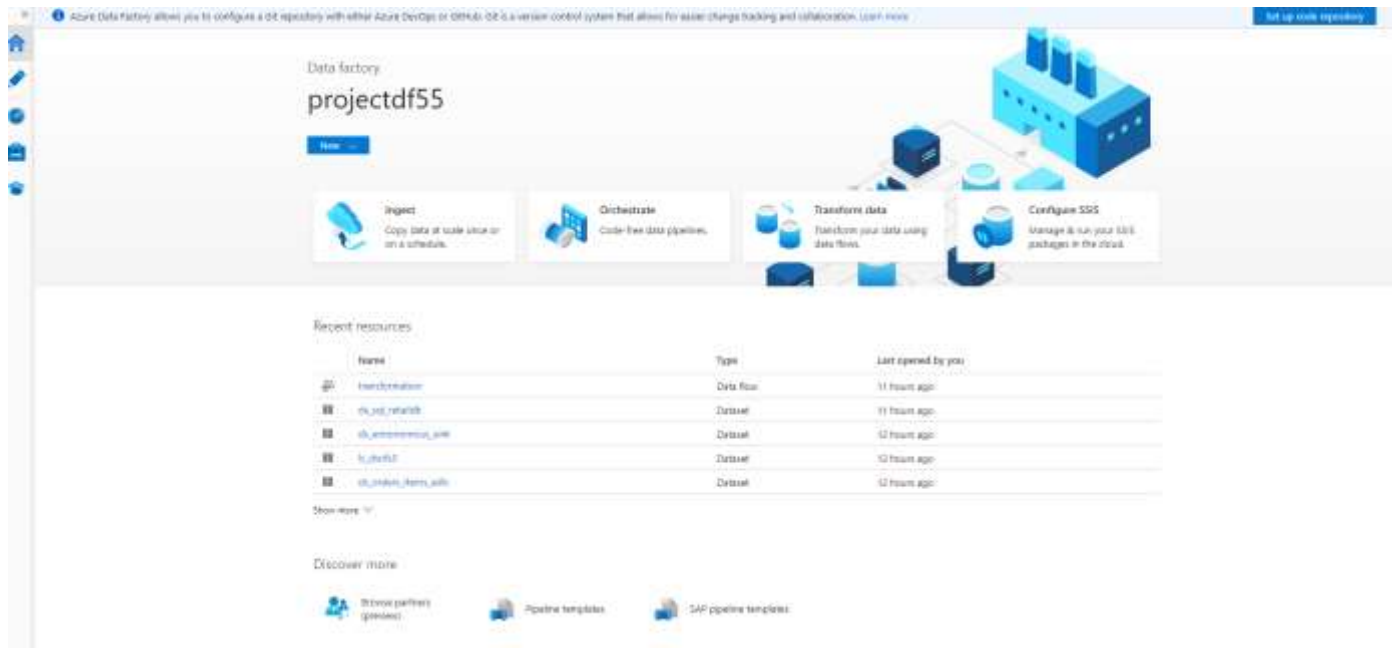


The screenshot displays the Azure Data Factory Studio interface for a resource named 'projectdf55'. The left sidebar contains a navigation menu with options: Overview (selected), Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Getting started, Monitoring, Automation, and Help. The main content area shows the 'Essentials' tab with the following details:

- Resource group: 000001 | [View](#)
- Status: Succeeded
- Location: Central India
- Subscription: [View](#) | [Edit](#)
- Subscription ID: f388b5c5-4880-41c4-9409-0622244917ee

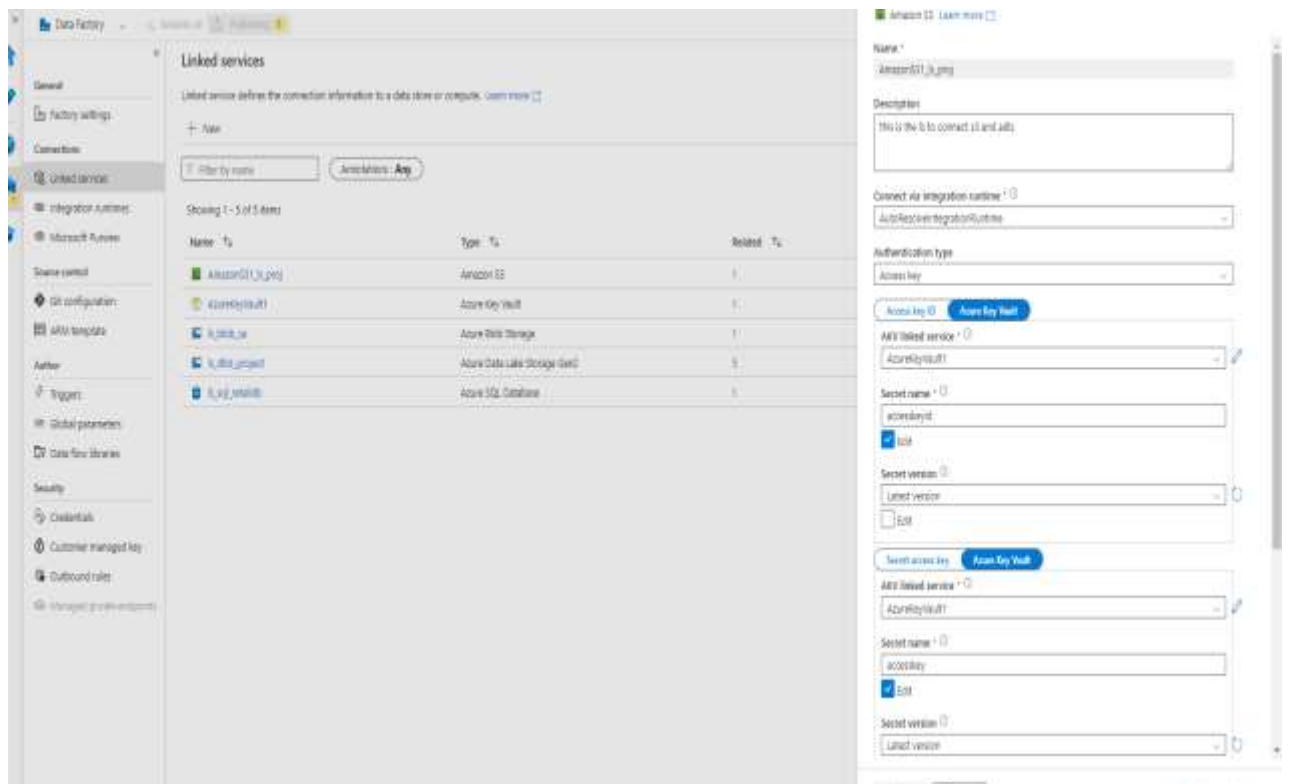
Below the essentials, the 'Azure Data Factory Studio' logo is centered, followed by a blue 'Launch studio' button. Underneath are four tiles: 'Quick Starts', 'Tutorials', 'Template Gallery', and 'Training Modules'. At the bottom, the 'Monitoring' section features three expandable panels: 'PipelineRuns', 'ActivityRuns', and 'TriggerRuns', each with a star icon and a list of items.

STEP2: LAUNCH THE ADLS STUDIO



STEP 3: ADD THE LINK SERVICES

Linked service to connect amazon s3 bucket



Link Service to connect to adls

Link service to connect key vault

The screenshot displays the 'Linked services' section of the Microsoft Fabric Data Factory interface. The main area shows a table of existing linked services, and the right sidebar shows the configuration for the 'AzureKeyVault' service.

Linked services table:

Name	Type	Related
AmazonS3Link	Amazon S3	1
AzureKeyVault	Azure Key Vault	1
ADLSLink	Azure Blob Storage	1
ADLSLink2	Azure Data Lake Storage Gen2	1
ADLSLink3	Azure SQL Database	1

Edit linked service sidebar (AzureKeyVault):

- Name:** AzureKeyVault
- Description:** (Empty text box)
- Azure key vault selection method:** ☒ From Azure subscription, ☐ Enter manually
- Base URL:** <https://myvault.azure.net/>
- Authentication method:** System assigned managed identity
- Managed identity name:** propmadf15
- Managed identity object ID:** 273058c-ba26-41e6-8d6d-7528ac738e4
- Test connection:** ☒ To linked service, ☐ To parent
- Annotations:** + New
- Parameters:** > Parameters
- Advanced:** > Advanced

Microsoft recently announced the public preview of Microsoft Fabric, a brand new and exciting way to build cloud data solutions. Click here to get started with Fabric Data Factory.

Data Factory

Windows 10

Windows 10

General

Factory settings

Connections

Linked services

Integration runtime

Microsoft Azure

Secure vault

API configuration

Alert templates

Author

Triggers

Global parameters

Data flow libraries

Security

Credentials

Customized managed log

Default rules

Use managed private endpoints

Linked services

Linked services allow the connection information for a data store or compute. Learn more >

Filter by name

Actions: Any

Showing 1 / 1 of 1 items

Name	Type	Related
Account123456789	Account ID	1
Account123456789	Account ID	1
Account123456789	Account ID	1
Account123456789	Account ID	1
Account123456789	Account ID	1
Account123456789	Account ID	1
Account123456789	Account ID	1
Account123456789	Account ID	1
Account123456789	Account ID	1
Account123456789	Account ID	1

Linked service

Account Data Lake Storage Gen2 Learn more >

Name

Account123456789

Description

Connect via integration runtime

AutoSelecting the runtime

Authentication type

Account key

Account selection method

☐ From Azure subscription ☒ Enter manually

URL

https://myaccount.blob.core.windows.net/

Storage account key

Account key

Storage account key

Account key

Test connection

☒ To linked service ☐ To file path

Permissions

+ New

> Permissions

> Advanced

Test connection

Link Service to connect blob storage

The screenshot shows the 'Edit linked service' window for an 'Azure SQL Database' in the Microsoft Fabric Data Factory console. The left sidebar contains navigation options like 'General', 'Factory settings', 'Connections', 'Linked services', 'Integration runtime', 'Microsoft Purview', 'Secure enclaves', 'On configuration', 'ARIS template', 'Author', 'Triggers', 'Global parameters', 'Data flow libraries', 'Security', 'Credentials', 'Customer managed key', and 'Outbound rules'. The main pane displays a table of linked services:

Name	Type	Related
amazonr11_b_pjw	Amazon S3	1
Amazonr11b1	Azure Key Vault	1
A_bldc_je	Azure Blob Storage	1
A_bldc_project	Azure Data Lake Storage Gen2	1
A_bldc_retaildb	Azure SQL Database	1

The right pane shows the configuration for the 'Azure SQL Database' linked service:

- Name:** A_bldc_retaildb
- Description:** (empty)
- Connect via integration runtime:** AutoResolveIntegrationRuntime
- Account selection method:** From Azure subscription (selected), Order manually
- Fully qualified domain name:** projectw111.blob.core.windows.net
- Database name:** retaildb
- Authentication type:** SQL authentication
- Username:** retailuser
- Password:** (masked)
- Always encrypted:** (unchecked)
- Additional connection properties:** (empty)

Buttons at the bottom include 'Save', 'Cancel', and 'Test connection'.

Link service to connect sql

The screenshot shows the 'Edit linked service' window for an 'Azure Blob Storage' in the Microsoft Fabric Data Factory console. The left sidebar is identical to the previous screenshot. The main pane displays the same table of linked services:

Name	Type	Related
amazonr11_b_pjw	Amazon S3	1
Amazonr11b1	Azure Key Vault	1
A_bldc_je	Azure Blob Storage	1
A_bldc_project	Azure Data Lake Storage Gen2	1
A_bldc_retaildb	Azure SQL Database	1

The right pane shows the configuration for the 'Azure Blob Storage' linked service:

- Name:** A_bldc_je
- Description:** (empty)
- Connect via integration runtime:** AutoResolveIntegrationRuntime
- Authentication type:** Account key
- Account selection method:** From Azure subscription (selected), Order manually
- Storage account name:** storageaccountproject
- Storage account key:** (masked)
- Partitioned DMS enabled:** (unchecked)
- Endpoint suffix:** core.windows.net
- Additional connection properties:** (empty)

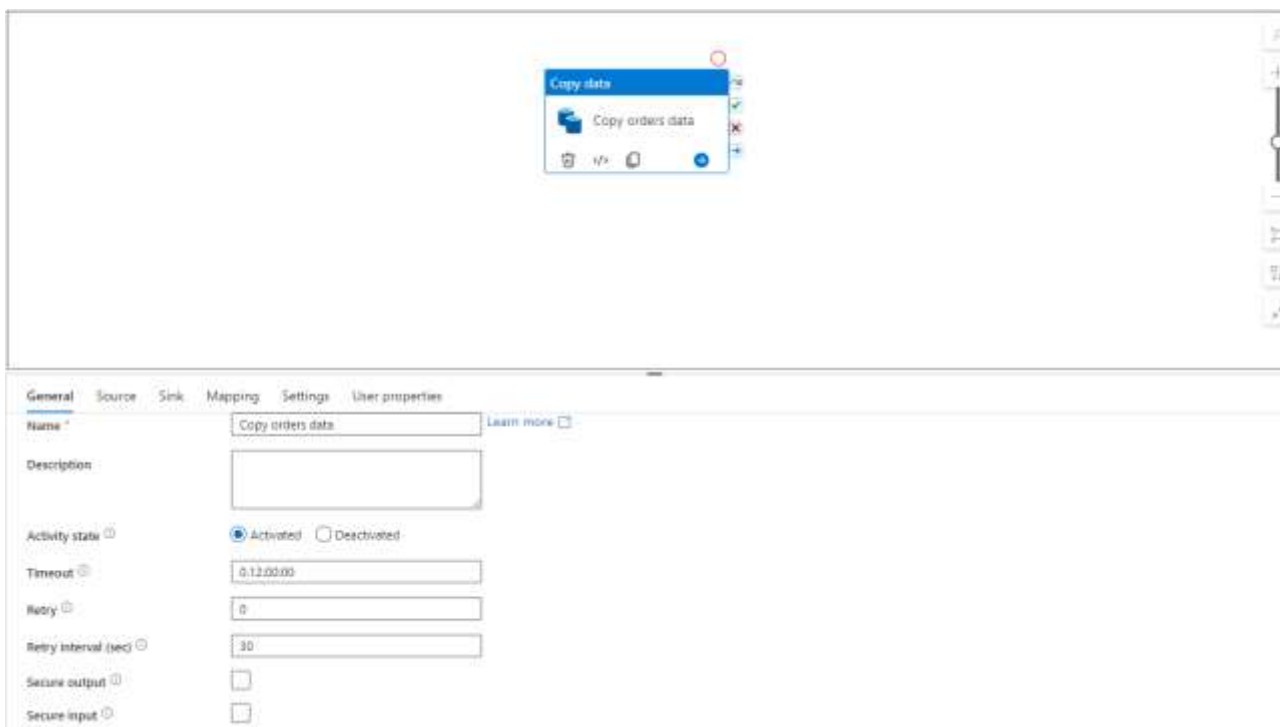
Buttons at the bottom include 'Save', 'Cancel', and 'Test connection'.

Step4: load the data set format

▲ Datasets	8
▲ folder order items dataset	2
ds_orders_items_adls	
ls_dsofs3	
▲ folder orders datasets	5
ds_ennonomous_sink	
ds_highvalue_orders_sink	
ds_lowvalueorder_sink	
ds_orders_adls_sink	
ds_orders_raw	
▲ folder sql	1
ds_sql_retaildb	

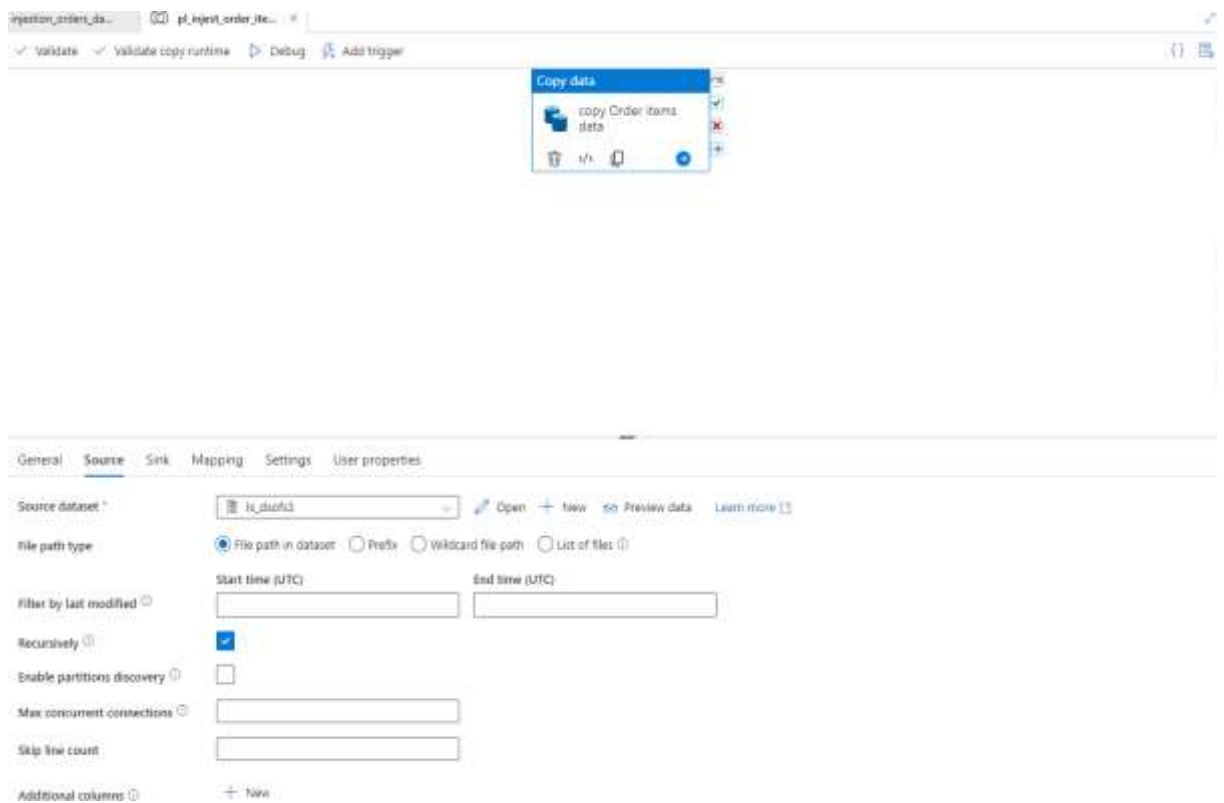
STEP4: Creating the Pipeline to ingest the data

Pipeline for ingest the data from s3 to adls



The screenshot displays the Azure Data Factory (ADF) interface. At the top, a 'Copy data' activity is shown in a preview window, titled 'Copy orders data'. Below this, the configuration pane for the activity is visible, showing the following settings:

- Name:** Copy orders data
- Description:** (Empty text box)
- Activity state:** ☒ Activated ☐ Deactivated
- Timeout:** 0.12.00.00
- Retry:** 0
- Retry interval (sec):** 30
- Secure output:** ☐
- Secure input:** ☐

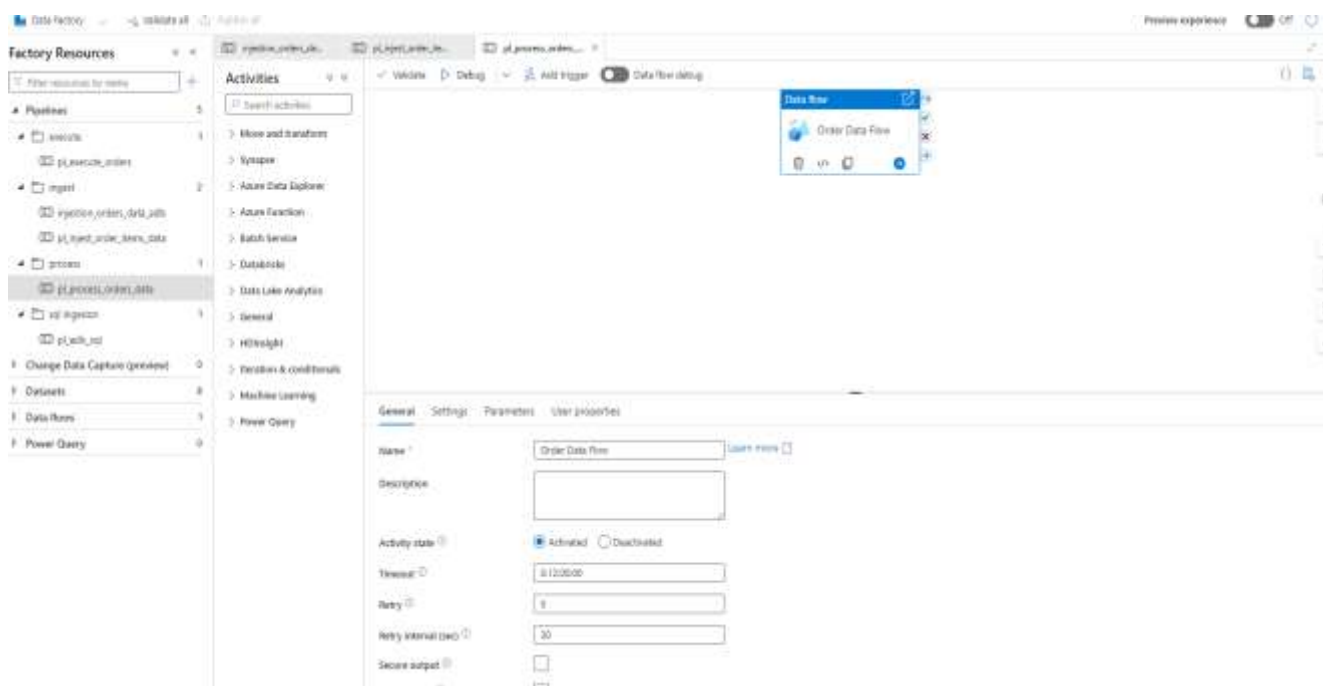


Pipeline to inject data from blob storage to adls

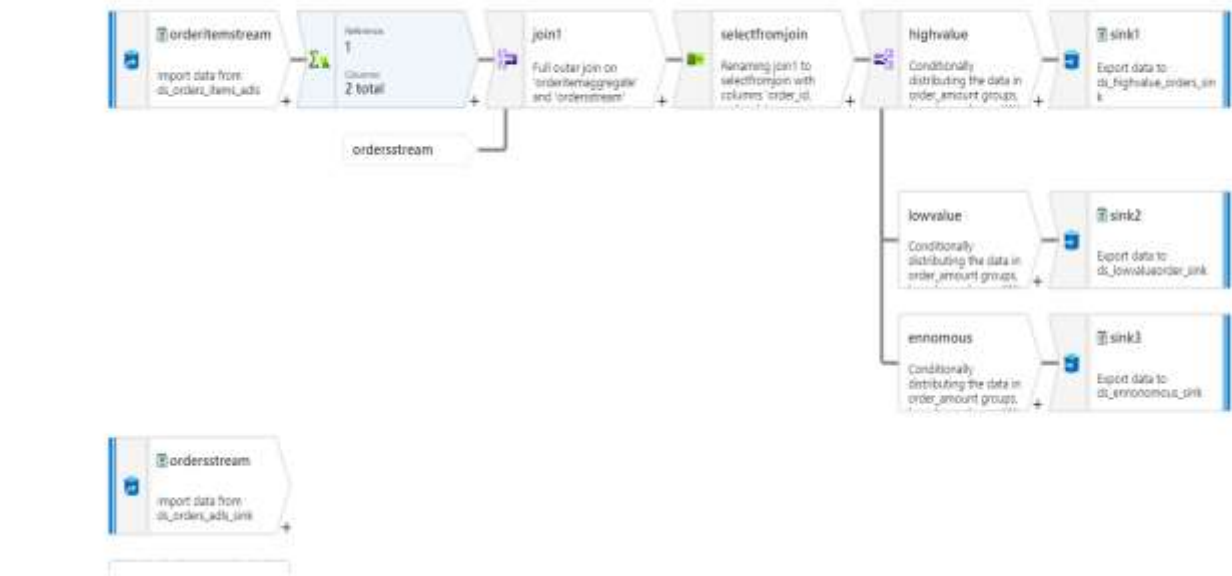
Pipeline to process orders data

Step 5: Transform the data using data flow

Data Flow Diagram



Load the data from the adls



Source settings Source options Projection Optimize Inspect **Data preview**

Number of rows: 1000 100 UPDATES 0 DELETES 0 APPEND 0 CUDROP 0 SINK 0 TOTAL 233

Refresh Ignored Invalid Map output Statistics View Export to CSV

order_item_id	order_item_order_id	order_item_product_id	order_item_quantity	order_item_subtotal	order_item_product_id
1	1	957	1.0	299.98	299.98
2	2	1073	1.0	188.88	188.88
3	2	302	3.0	259.0	55.0
4	2	403	1.0	128.09	128.09
5	4	857	2.0	49.99	34.99
6	4	365	3.0	298.95	59.98
7	6	302	3.0	159.0	50.0
8	4	1074	4.0	188.92	40.98
9	5	957	1.0	299.98	299.98
10	5	855	3.0	298.99	59.99
11	5	1074	2.0	95.95	49.98
12	5	957	1.0	299.98	299.98



Aggregate settings Optimize Inspect **Data preview**

Number of rows: **INSERT** 83 **UPDATE** 0 **DELETE** 0 **UPSERT** 0 **LOOKUP** 0 **ERROR** 0 **TOTAL**

Refresh Refresh New City Skip diff diff Statistics Remove Export to CSV

order_item_order_id	order_item_subtotal
1	299.9600109863261
2	579.9600109863261
4	889.8500099162129
5	1129.8600387573242
7	379.9200134277344
8	729.8400115966787
9	589.9600067138672

Aggregate the data by Order id for table order_items

Load the Orders table

✓ validate Data flow debug Debug Settings

ordersstream
Columns: 4 total

Continuously distributing the data in order_amount groups. Export data to: dl_orders_order_pos

anonymous
Conditionally distributing the data in order_amount groups. Export data to: dl_anonymous_pos

link3

Source settings Source options Projection Optimize Inspect **Data preview**

Number of rows: INSERT: 100 UPDATE: 0 DELETE: 0 UPSERT: 0 LOOKUP: 0 ERROR: 0 TOTAL: 104

Refresh Spread Apply Map to Flat Statistics Remove Export to CSV

order_id	order_date	order_customer_id	order_status
1	2013-07-25 00:00:00.0	11586	CLOSED
2	2013-07-25 00:00:00.0	256	PENDING PAYMENT
3	2013-07-25 00:00:00.0	12111	COMPLETE
4	2013-07-25 00:00:00.0	8827	CLOSED
5	2013-07-25 00:00:00.0	11318	COMPLETE
6	2013-07-25 00:00:00.0	7130	COMPLETE
7	2013-07-25 00:00:00.0	4530	COMPLETE
8	2013-07-25 00:00:00.0	2911	PROCESSING
9	2013-07-25 00:00:00.0	5637	PENDING PAYMENT
10	2013-07-25 00:00:00.0	5648	PENDING PAYMENT

Join Orders and Order item table

The diagram illustrates a data pipeline for joining 'orders' and 'order_items' tables. It starts with an 'orderitemstream' (input data from ds_order_items_sdt) and an 'ordersstream' (input data from ds_orders_sdt). These streams feed into an 'orderitemaggrega...' (Aggregating data by order_item_order_id producing columns) and a 'join1' (Joining join1 to selectfromjoin with columns 'order_id'). The output of 'join1' is 'selectfromjoin' (Renaming join1 to selectfromjoin with columns 'order_id'). This stream then branches into 'highvalue' (Conditionally distributing the data in order amount groups) and 'lowvalue'. Both 'highvalue' and 'lowvalue' feed into 'sink1' (Export data to ds_highvalue_order_sdt) and 'sink2' (Export data to ds_lowvalue_order_sdt).

Join settings: Select settings, Optimize, Inspect, Data preview.

Number of rows: INSERT: 100, UPDATE: 0, DELETE: 0, UPSERT: 0, LOOKUP: 0, ERROR: 0, TOTAL: 104.

order_item_order_id	order_item_subtotal	order_id	order_date	order_customer_id	order_status
1	299.9900109964281	1	2013-07-25 00:00:00.0	11599	CLOSED
10	651.620015335068	10	2013-07-25 00:00:00.0	5648	PENDING PAYMENT
100	548.9400100708008	100	2013-07-25 00:00:00.0	12151	PROCESSING
101	895.9400253295898	101	2013-07-25 00:00:00.0	5116	CLOSED
NULL	NULL	102	2013-07-25 00:00:00.0	8027	COMPLETE
103	829.9200266865234	103	2013-07-25 00:00:00.0	12256	PROCESSING
104	540.9500198364258	104	2013-07-25 00:00:00.0	7790	PENDING PAYMENT
11	919.7999932891328	11	2013-07-25 00:00:00.0	916	PAYMENT REVIEW
12	1299.6700256347638	12	2013-07-25 00:00:00.0	1837	CLOSED
13	127.90999906447266	13	2013-07-25 00:00:00.0	9149	PENDING PAYMENT
14	548.9400100708008	14	2013-07-25 00:00:00.0	9642	PROCESSING
15	925.9100189206984	15	2013-07-25 00:00:00.0	2568	COMPLETE

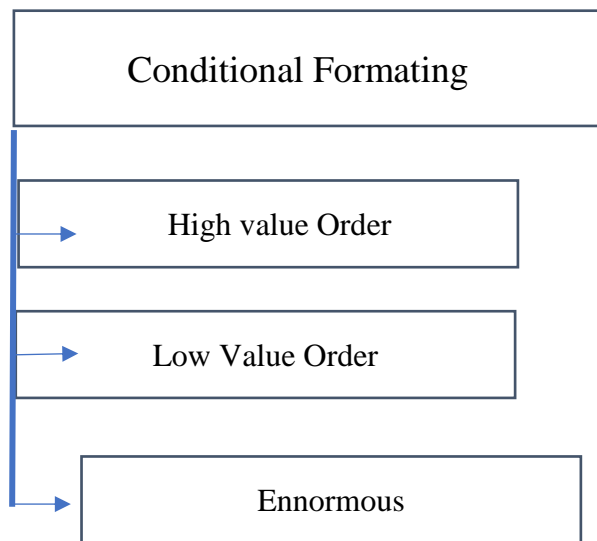
Select the desired attributes and data from the join

The screenshot shows the 'selectfromjoin' transformation settings in the Data Factory interface. The 'Output stream name' is 'selectfromjoin'. The 'Description' is 'Renaming join1 to selectfromjoin with columns 'order_id', 'order_date', 'customer_id', 'order_status', 'order_amount''. The 'Joining stream' is 'join1'. The 'Options' section has 'Skip duplicate input columns' and 'Skip duplicate output columns' checked. The 'Input columns' section shows a mapping of columns from 'join1' to 'selectfromjoin'.

Joining stream	Input column	Output column
join1	order_id	order_id
join1	order_date	order_date
join1	order_customer_id	customer_id
join1	order_status	order_status
join1	order_item_subtotal	order_amount

5 mappings: 1 column(s) from the inputs left unmapped.

Conditional Formating



Screen shot of the Conditions

The screenshot displays the Apache NiFi web console. The top section shows a data flow with components: `orderitemstream`, `orderitemaggrega...`, `join`, `selectfromjoin`, `highvalue`, and `sink`. The `highvalue` component is highlighted, and its configuration panel is open.

Conditional split settings

Output stream name: `split1`

Description: Conditionally distributing the data in order_amount groups, based on columns '11'

Incoming stream: `selectfromjoin`

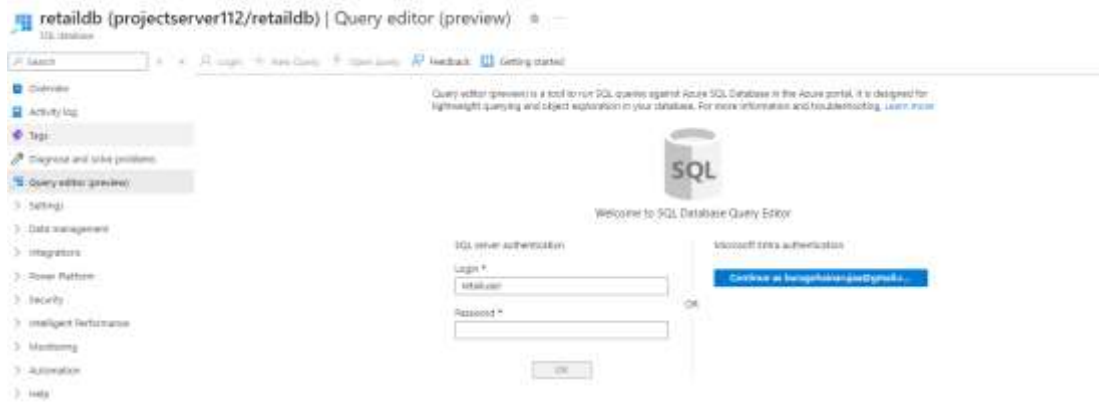
Split on: ☒ First matching condition ☐ All matching conditions

Stream names	Condition
<code>highvalue</code>	<code>order_amount > 500</code>
<code>lowvalue</code>	<code>order_amount <= 500</code>
<code>ennormous</code>	Records that do not meet any condition will use this output stream

High value orders

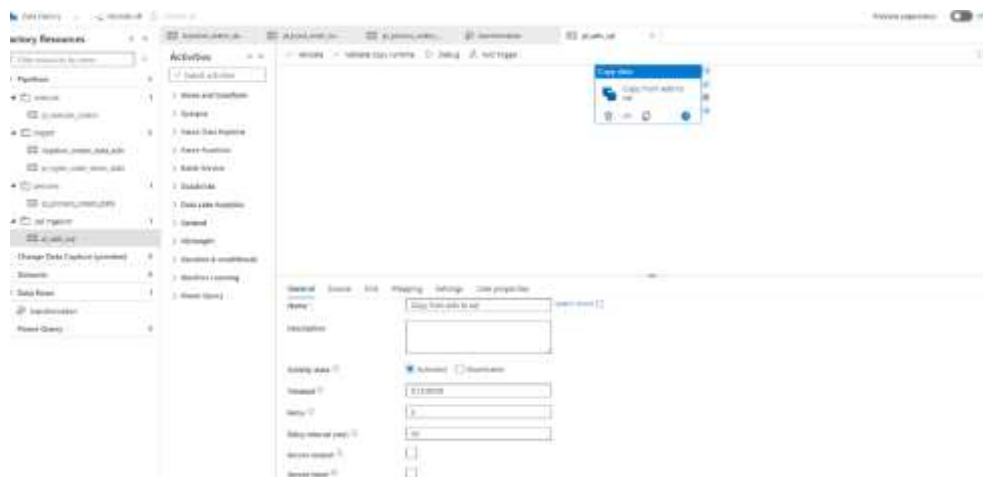
Conditional split settings Optimize Inspect Data preview 									
Number of rows: INSERT 47 UPDATE 0 DELETES 0 UPSERT 0 LOOKUP 0 ERROR 0 TOTAL									
Refresh Statistics Export to CSV									
Output stream <input type="text" value="highvalue"/>									

5.4 OPEN SQL SERVER

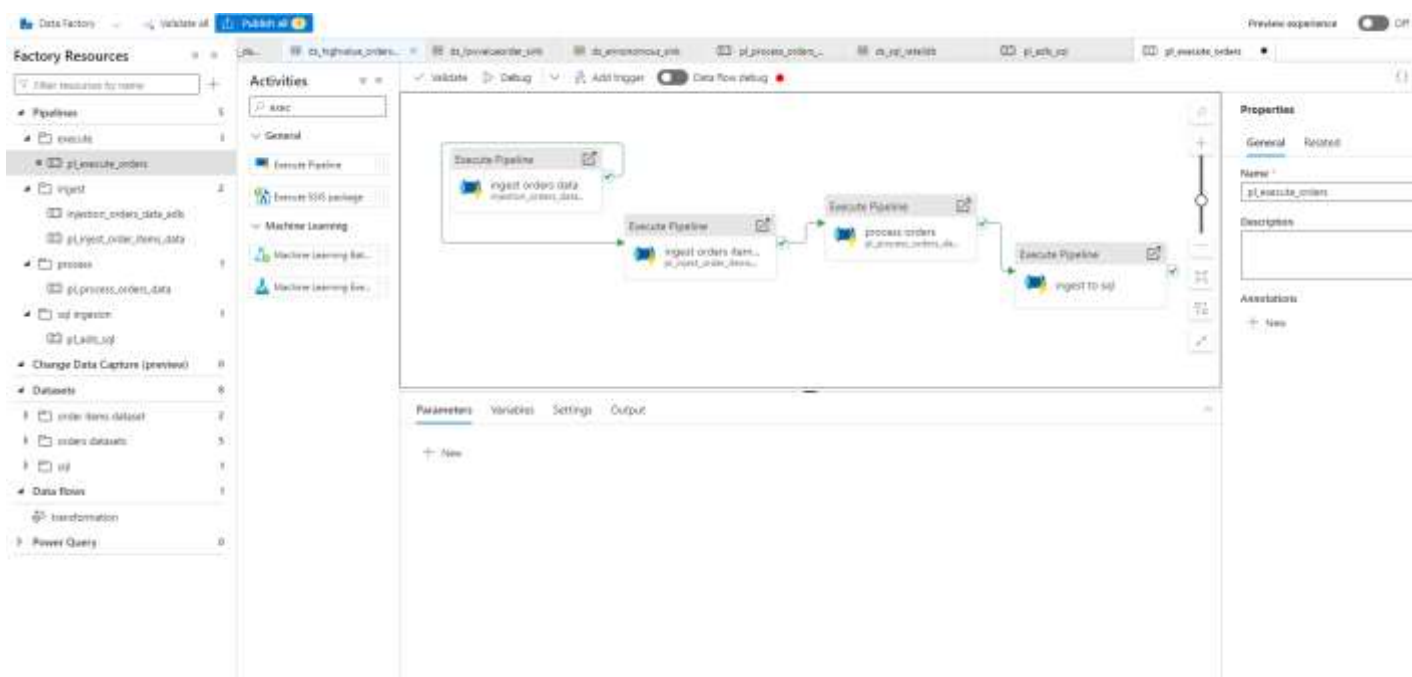


Creation of Premium table in Sql

Creating the Pipeline for Execution



Creating the Whole pipeline to execute the data



Add Triggers for the Pipeline

New trigger

Description

Type *

Account selection method * ⓘ

☒ From Azure subscription ☐ Enter manually

Azure subscription ⓘ

Storage account name * ⓘ

Container name * ⓘ

Blob path begins with ⓘ

Blob path ends with ⓘ

Event * ⓘ

☒ Blob created ☐ Blob deleted

Ignore empty blobs * ⓘ

☒ Yes ☐ No

Annotations

+ New

Start trigger ⓘ

☒ Start trigger on creation

Continue

Cancel

ENG

15:02

