

RETAIL – CASE - STUDY

NAME: Govind Polireddi

Employee Id: AS1552

Problem Statement

Retail companies struggle with integrating **in-store POS transactions, online orders, and inventory systems** into a single analytics-ready platform.

They need:

- **Batch processing** for daily sales reconciliation.
- **Real-time ingestion** for fraud detection & stock alerts.
- **Cloud-native scalability** to support peak season sales.
- **Unified warehouse (Snowflake / Synapse)** for advanced reporting
- **Monitoring & error handling** for operational resilience.
- **Data security** for customer PII.
- **Cost transparency** for infrastructure & query optimization.

I am Using **LAMBDA ARCHITECTURE** as it supports both BATCH AND STREAMING PROCESS.

APPROACH:

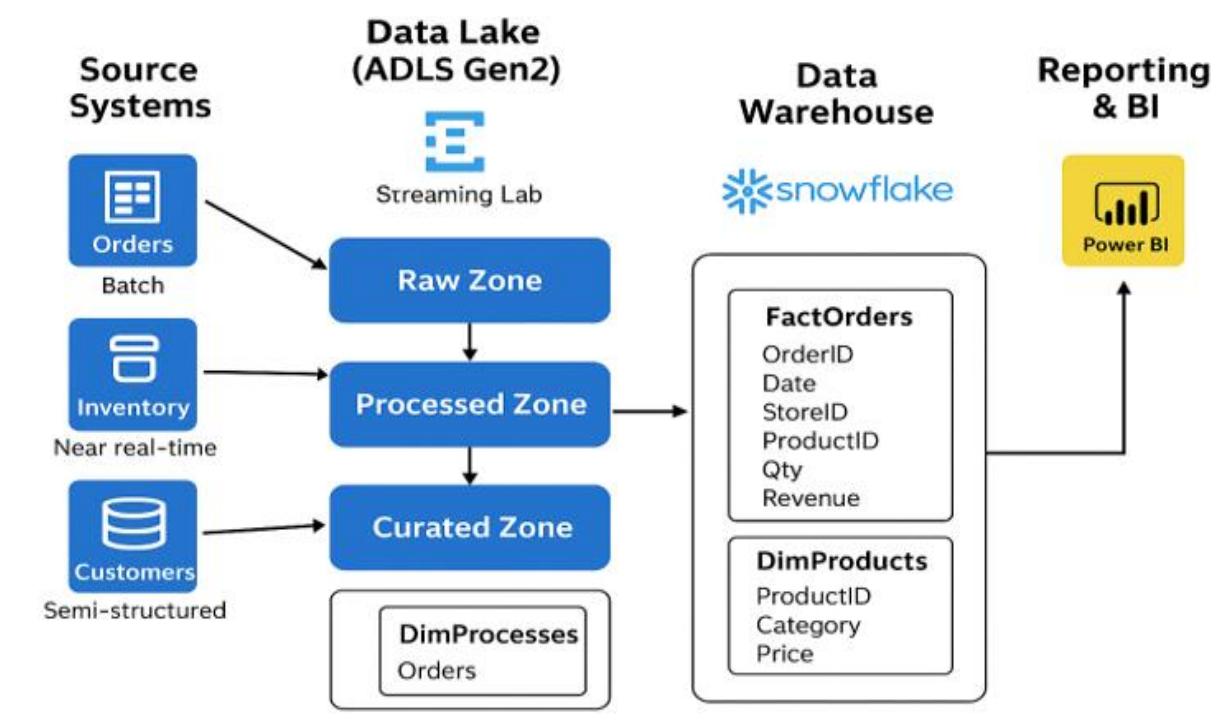
- **Step-1:** Creating a ADLS storage account to store the raw data and also the cleaned and validated data in medallion structure (Bronze, Silver, Gold).
- **Step-2:** For **BATCH-PROCESSING** create 2 ADF(Azure Data Factory) Pipelines one for each of the Batch Orders CSV and Payments CSV files.
- **Step-3:** Use Event- Grid to create trigger the pipelines when ever there is new file uploaded.
- **Step-4:** Copy the data to the bronze by using COPY activity, after clean and validate the data and store it in the Silver and Gold layer and finally load it into the Snowflake table.
- **Step-4:** Create Alerts from ADF monitor and LOG the DATABRICKS logs into **LOG ANALYTICS WORKSPACE**.
- **Step-5:** For **STREAM-PROCESSING** create a Event-hub to collect the events from the Kafka. Create Databricks Notebook and connect to the Event-hub. Collect the events from event hub and process the event and store the data in delta

format in bronze, silver and gold. Finally connect the snowflake to load the data into the fact table.

- **Step-6:** Load the customer, product and store data from ADLS to the snowflake by ADLS SAS token and creating stage in the snowflake to copy the data. Use copy into the Snowflake table.
- **Step-7:** For the customer data do the PII masking for the Email and phone number for security purpose.
- **Step-8:** Connect snowflake with Power BI for the real time analytics.

ARCHITECTURE:

I am using the below architecture



AZURE STORAGE ACCOUNT CREATION:

- ✓ Created an storage account “**azurestorageaccount52**”.

Storage center | Storage accounts (Blobs)

Name	Type	Kind	Resource group	Location	Subscription
azurestorageaccount52	Storage account	StorageV2	Azure-resource-group	Central India	Azure subscription 1

- ✓ Create a container and folders in it to store the data in medallion Architecture.

Name	Last modified	Access tier	Blob type	Size	Lease state
_Sazuretmpfolders	9/20/2025, 7:46:22 PM				
bronze	9/21/2025, 12:39:40 AM				
checkpoints	9/21/2025, 2:50:30 PM				
gold	9/20/2025, 7:39:26 PM				
raw	9/20/2025, 7:39:30 PM				
raworders	9/21/2025, 6:34:47 PM				
rawpayment	9/21/2025, 6:34:47 PM				
silver	9/20/2025, 7:39:23 PM				

- ✓ Store the customer, products and store data in **raw** directory

Name	Last modified	Access tier	Blob type	Size	Lease state
[-]	9/20/2025, 9:48:32 PM	Hot (Inferred)	Block blob	578.01 KiB	Available
dim_customers.csv	9/20/2025, 9:48:31 PM	Hot (Inferred)	Block blob	19.2 KiB	Available
dim_products.csv	9/20/2025, 9:48:31 PM	Hot (Inferred)	Block blob	593 B	Available
dim_stores.csv	9/20/2025, 9:48:32 PM	Hot (Inferred)	Block blob	3.02 MiB	Available
inventory_snapshot_daily.csv	9/20/2025, 9:48:32 PM	Hot (Inferred)	Block blob	3.02 MiB	Available

- ✓ The raworders and rawpayment folders are for the storage of Batch process files automatically when there is a new file uploaded.

Name	Last modified	Access tier	Blob type	Size	Lease state
fact_orders_daily_batch.csv	9/21/2025, 9:19:42 PM	Hot (Inferred)	Block blob	2.53 MiB	Available

data > rawpayment

Authentication method: Access key (Switch to Microsoft Entra user account)

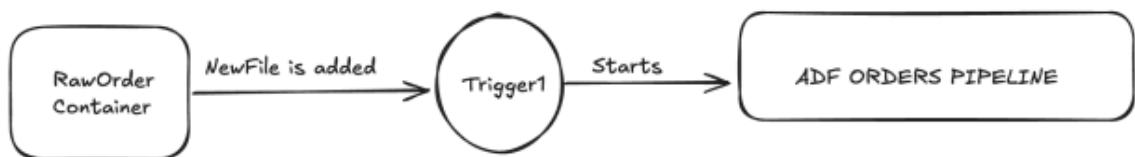
Search blobs by prefix (case-sensitive) Only show active objects

Show all 1 items

Name	Last modified	Access tier	Blob type	Size	Lease state
[...]	9/21/2025, 10:22:46 PM	Hot (Inferred)	Block blob	1.83 MiB	Available
payments.csv					

BATCH – PROCESSING:

For Batch orders pipeline.



- ✓ Create a pipeline for batch orders to run when a new file is added in raworders.

Microsoft Azure | Data Factory > azuredatad-factory123

Activities: Copy data, Notebook

Ordersdata

General

Name: Ordersdata

Description:

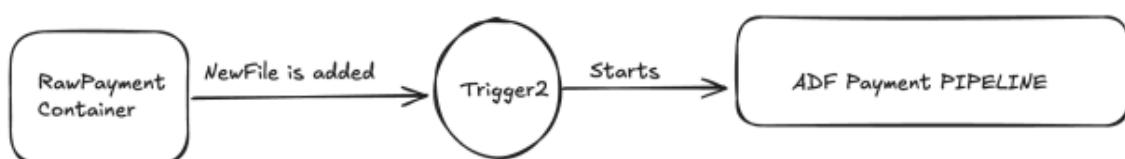
Activity state: Activated

Timeout: 0:12:00:00

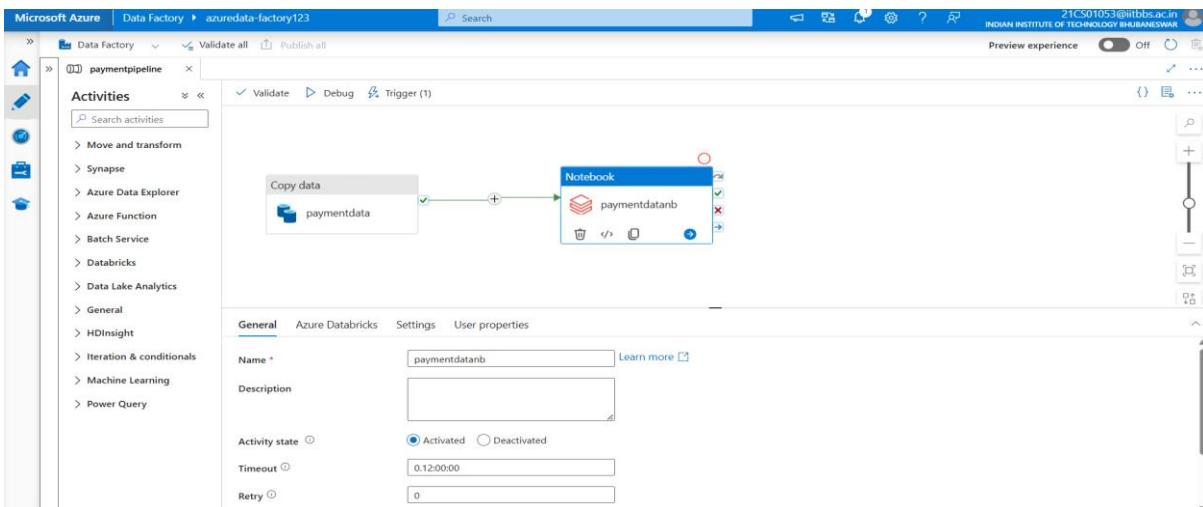
Retry: 0

ADF ORDERS PIPELINE

For Batch payment pipeline



- ✓ Create a pipeline for batch payment to run when a new file is added in rawpayments.



- ✓ Create 2 triggers to run the pipelines for new file upload.

The screenshot shows the Microsoft Azure Data Factory triggers page. It lists two triggers: "trigger1" and "trigger2". Both triggers are of type "Storage events" and are configured to trigger on "blob events" from "Azure subscription 1" on container "data" with blob path "raworders". The "trigger1" configuration includes "From Azure subscription" selected.

The screenshot shows the Microsoft Azure Data Factory triggers page. It lists two triggers: "trigger1" and "trigger2". Both triggers are of type "Storage events" and are configured to trigger on "blob events" from "Azure subscription 1" on container "data" with blob path "rawpayment". The "trigger2" configuration includes "From Azure subscription" selected.

- ✓ Create Alerts for the pipeline failures and activity failures for both the pipelines.

The screenshot shows the 'Alerts & metrics' section of the Azure Data Factory interface. On the left, there's a navigation sidebar with options like Dashboards, Runs, Pipeline runs, Trigger runs, Change Data Capture, Runtimes & sessions, Integration runtimes, Data flow debug, Notifications, and Alerts & metrics (which is currently selected). The main area displays a table of alerts:

ALERT	ENABLED	RESOURCE TYPE	RESOURCES	ACTIONS
pipeline alert	<input checked="" type="checkbox"/> On	Pipeline	3	Edit Delete
copy Activity Alert	<input checked="" type="checkbox"/> On	Activity	6	Edit Delete
Notebook Activity Alert	<input checked="" type="checkbox"/> On	Activity	6	Edit Delete
Trigger fail alert	<input checked="" type="checkbox"/> On	Trigger	2	Edit Delete
pipeline failure alert	<input checked="" type="checkbox"/> On	Pipeline	3	Edit Delete

CREATE ACTION GROUP:

- ✓ Created action group for notification alerts through email.

This screenshot shows the 'Add notification' dialog box overlaid on the 'Alerts & metrics' page. The dialog has fields for 'Action name' (set to 'email notification'), 'Select which notifications you'd like to receive' (with 'Email' checked and an example address '21cs01053@iitbbs.ac.in'), and other optional sections for SMS, Azure app push notifications, and voice. At the bottom are 'Add notification' and 'Cancel' buttons.

This screenshot shows the 'Configure notification' dialog box. It includes fields for 'Action group name' (set to 'Azurereactiongroup'), 'Short name' (set to 'azureproject'), and a table for 'Notifications', 'Action type', and 'Actions'. The table has one row: 'email notification' under 'Notifications', 'Email/SMS/Push/Voice' under 'Action type', and edit/delete icons under 'Actions'. At the bottom are 'Add action group' and 'Cancel' buttons.

Orders Batch processing Notebook:

```
1 dbutils.widgets.text("filename", "")
2 filename = dbutils.widgets.get("filename")
3 filepath = "/mnt/bronze/batch/orders/" + filename
```

```
1 batch_df = spark.read.format("parquet").load(filepath)
2 batch_df = batch_df.dropDuplicates(["OrderID"])
3 batch_df = batch_df.withColumn(
4     "OrderDateTime",
5     to_timestamp(col("OrderDateTime"), "yyyy-MM-dd'T'HH:mm:ss")
6 )
7 silver_path = "/mnt/silver/batch/orders/"
8 if DeltaTable.isDeltaTable(spark, silver_path):
9     silver_table = DeltaTable.forPath(spark, silver_path)
10    silver_table.alias("silver").merge(
11        batch_df.alias("new"),
12        "silver.OrderID = new.OrderID"
13    ).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()
14 else:
15     batch_df.write.format("delta").mode("append").save(silver_path)
```

silver to gold

```
1 gold_path = "/mnt/gold/batch/orders/"
2 if DeltaTable.isDeltaTable(spark, gold_path):
3     gold_table = DeltaTable.forPath(spark, gold_path)
4     gold_table.alias("gold").merge(
5         batch_df.alias("new"),
6         "gold.StoreID = new.StoreID AND gold.OrderDate = new.OrderDate"
7     ).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()
8 else:
9     batch_df.write.format("delta").mode("append").save(gold_path)
```

Gold to Snowflake table

```
1 pip install snowflake-snowpark-python
```

```
1 12 hours ago(<1s)
1 from snowflake.snowpark import Session
```

```
1 # Snowflake options
2 batch_df.write \
3     .format("snowflake") \
4     .options(
5         sfURL="SVXVXL-FR13613.snowflakecomputing.com",
6         sfUser="BHUBANESWAR",
7         sfPassword="0ck5n73Nm50ct8",
8         sfDatabase="TRAINING",
9         sfSchema="PUBLIC",
10        sfWarehouse="SIGMOID",
11        sfRole="ACCOUNTADMIN"
12    ) \
13    .option("dbtable", "FACTORDERS") \
14    .mode("append") \
15    .save()
```

[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette
[Esc H] to see all keyboard shortcuts

OUTPUT TABLE (FACT ORDER):

#	ORDERID	ORDERDATETIME	STOREID	CUSTOMERID	PRODUCTID	QUANTITY	UNITPRICE	DISCOUNTPCT	PAYMENTID	PAYMENTMETHOD	CHANNEL	STATUS	ORDERTOTAL	CURRENCY
1	296	2025-05-07 14:27:00.000	9	4612	90	1	59.66	0.00	PMT00000296	CARD	Online	Completed	59.66	INR
2	467	2025-07-18 05:57:00.000	6	3198	179	1	76.01	0.00	PMT00000467	UPI	Online	Completed	76.01	INR
3	675	2025-04-06 06:38:00.000	2	3965	10	1	22.57	5.00	PMT00000675	CASH	POS	Completed	21.44	INR
4	691	2025-08-14 05:17:00.000	8	4384	164	1	14.05	0.00	PMT00000691	UPI	POS	Cancelled	14.05	INR
5	829	2025-04-30 09:48:00.000	2	5610	129	1	18.64	5.00	PMT00000829	WALLET	Online	Completed	17.71	INR
6	1090	2025-05-02 05:08:00.000	9	3574	114	1	57.26	0.00	PMT00001090	CARD	Online	Completed	57.26	INR
7	1159	2025-05-17 00:52:00.000	7	5319	129	1	18.64	5.00	PMT00001159	WALLET	Online	Completed	17.71	INR
8	1436	2025-05-23 09:32:00.000	8	50	74	1	58.51	0.00	PMT00001436	CARD	POS	Completed	58.51	INR
9	1512	2025-07-14 11:27:00.000	12	2359	199	2	36.63	0.00	PMT00001512	CARD	POS	Pending	73.26	INR
10	1572	2025-07-11 17:28:00.000	1	5924	161	1	77.31	10.00	PMT00001572	WALLET	Online	Completed	69.58	INR
11	2069	2025-06-23 02:01:00.000	3	40	172	1	28.15	10.00	PMT00002069	CARD	Online	Completed	25.34	INR
12	2088	2025-07-05 14:43:00.000	5	2849	145	1	65.29	20.00	PMT00002088	UPI	Online	Completed	52.23	INR
13	2136	2025-04-12 07:37:00.000	1	126	225	1	39.69	0.00	PMT00002136	UPI	POS	Completed	39.69	INR
14	2162	2025-06-27 08:38:00.000	8	4156	161	1	77.31	10.00	PMT00002162	CARD	POS	Completed	69.58	INR
15	2294	2025-05-18 22:40:00.000	5	1355	233	3	13.78	5.00	PMT00002294	UPI	POS	Cancelled	39.27	INR
16	2904	2025-04-11 20:15:00.000	7	4403	80	1	99.68	0.00	PMT00002904	CASH	POS	Completed	99.68	INR
17	3210	2025-06-18 09:13:00.000	1	4239	249	1	72.05	0.00	PMT00003210	UPI	POS	Completed	72.05	INR
18	3414	2025-05-01 08:02:00.000	6	697	192	1	47.36	0.00	PMT00003414	CARD	POS	Pending	47.36	INR
19	3606	2025-06-09 18:48:00.000	2	4058	123	1	52.93	10.00	PMT00003606	WALLET	POS	Completed	47.64	INR
20	3959	2025-08-09 01:43:00.000	10	1769	87	3	54.84	0.00	PMT00003959	CARD	POS	Completed	164.52	INR
21	4032	2025-04-01 13:29:00.000	2	4671	176	1	26.16	0.00	PMT00004032	CARD	Online	Completed	26.16	INR
22	4821	2025-07-06 12:31:00.000	11	389	163	1	95.91	5.00	PMT00004821	UPI	POS	Pending	91.11	INR
23	4937	2025-06-04 16:31:00.000	8	5201	56	1	65.47	0.00	PMT00004937	CASH	POS	Completed	65.47	INR
24	5325	2025-07-20 12:41:00.000	11	5431	4	1	19.23	10.00	PMT00005325	CARD	POS	Completed	17.31	INR
25	5645	2025-06-02 01:14:00.000	8	2074	31	1	33.53	0.00	PMT00005645	CARD	POS	Completed	33.53	INR
26	5925	2025-08-02 03:49:00.000	7	1767	122	1	22.69	15.00	PMT00005925	CARD	POS	Completed	19.29	INR
27	6194	2025-08-07 10:18:00.000	11	2373	52	1	108.43	0.00	PMT00006194	UPI	POS	Completed	108.43	INR

Payments Batch processing Notebook:

silver to gold

```

1 Last execution failed
2
3 gold_path = "/mnt/gold/batch/payments/"
4
5 if DeltaTable.isDeltaTable(spark, gold_path):
6     gold_table = DeltaTable.forName(spark, gold_path)
7     gold_table.alias("gold").merge(
8         batch_df.alias("new"),
9         "gold.StoreID = new.StoreID AND gold.OrderDate = new.OrderDate"
10    ).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()
11 else:
12     batch_df.write.format("delta").mode("append").save(gold_path)

```

gold to Snowflake table

```

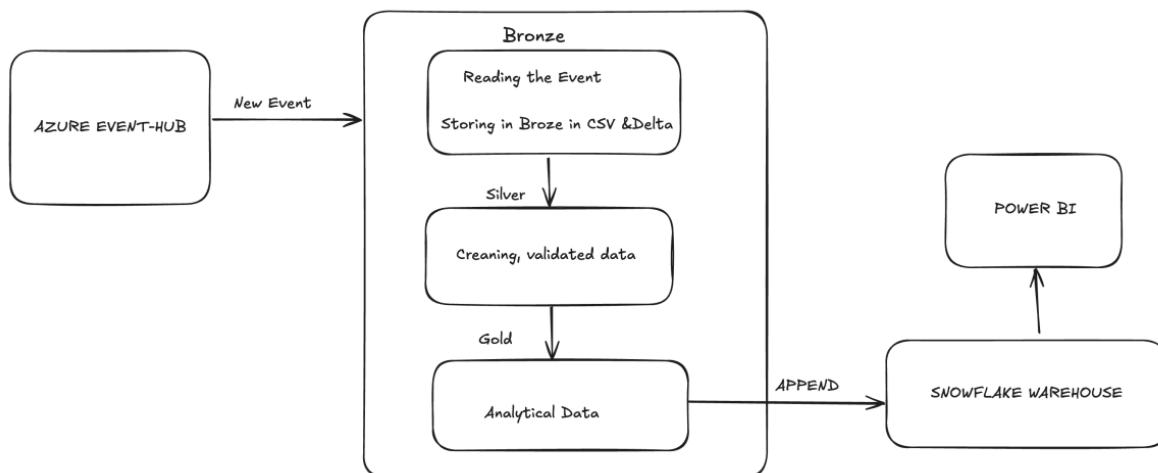
1 Last execution failed
2
3 # SnowFlake options
4 batch_df.write \
5     .format("snowflake") \
6     .options(
7         sfURL="SVVXKL-FR13613.snowflakecomputing.com",
8         sfUser="BHUBANESWAR",
9         sfPassword="Ock5n7J3m5Dct8",
10        sfDatabase="TRAINING",
11        sfSchema="PUBLIC",
12        sfWarehouse="SIGNOID",
13        sfRole="ACCOUNTADMIN"
14    ) \
15     .option("dbtable", "PAYMENTS") \
16     .mode("append") \
17     .save()

```

OUTPUT TABLE (PAYMENTS):

	PAYMENTID	ORDERID	AMOUNT	CURRENCY	METHOD	GATEWAY	STATUS	AUTHCODE	FRAUDSCORE
1	PMT00000122	122	76.65	INR	UPI	Razorpay	Pending	F8BV5ZHP	0.041
2	PMT00000638	638	344.82	INR	UPI	PayU	Captured	MRVNZQER	0.032
3	PMT00000960	960	89.16	INR	UPI	Stripe	Captured	MPEOS01F	0.167
4	PMT00001057	1057	94.66	INR	CARD	Cashdesk	Captured	YNLAIMA5	0.105
5	PMT00001141	1141	61.40	INR	CARD	Razorpay	Captured	19FG4UBV	0.100
6	PMT00001648	1648	63.62	INR	CASH	PayU	Captured	E2K3RUUW	0.051
7	PMT00001911	1911	312.84	INR	CASH	PayU	Captured	TYXEP6C1	0.352
8	PMT00002258	2258	130.96	INR	CARD	Stripe	Captured	J9PJPICR	0.033
9	PMT00002695	2695	25.95	INR	CARD	Cashdesk	Captured	ZEX85CDI	0.152
10	PMT00002949	2949	41.89	INR	UPI	PayU	Captured	YVDPT6PK	0.024
11	PMT00002980	2980	77.78	INR	CARD	Cashdesk	Pending	2W7YRB21	0.167
12	PMT00003908	3908	52.41	INR	UPI	Cashdesk	Captured	X7WMQAY4	0.104
13	PMT00003912	3912	13.32	INR	CARD	Cashdesk	Captured	8A90UUFX	0.171
14	PMT00004493	4493	25.42	INR	UPI	Stripe	Captured	FNEJWGYE	0.132
15	PMT00004494	4494	58.51	INR	CARD	Razorpay	Captured	3W5QUJJQ9	0.180
16	PMT00004928	4928	823.41	INR	WALLET	Cashdesk	Captured	FF2XZDT9	0.097

STREAM– PROCESSING:



- ✓ Create an Event hub to collect events and send to the databricks notebook

This screenshot shows the Azure Event Hubs Overview page for the namespace 'govind-event-hub'. The page includes a search bar, navigation links, and a summary section. On the left, there's a sidebar with options like 'Create', 'Manage view', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Data Explorer', 'Resource visualizer', 'Events', 'Settings', 'Entities', 'Monitoring', 'Automation', and 'Help'. The main content area displays the namespace details, including the resource group ('govind-resource-group'), status ('Succeeded'), location ('East Asia'), subscription ('Azure subscription 1'), subscription ID ('b41ac736-1671-4724-a6f2-6080db61224b'), host name ('govind-event-hub.servicebus.windows.net'), and various configuration settings like 'Pricing tier Standard', 'Throughput Units 1 unit', and 'Auto-inflate throughput units Disabled'. At the bottom, there are sections for 'NAMESPACE CONTENTS', 'KAFKA SURFACE', and 'ZONE REDUNDANCY', all set to 'ENABLED'. A footer at the bottom indicates the data is shown for the last 1 hour.

- ✓ Create a new event in the event hub namespace “events”

Microsoft Azure | Upgrade | Search resources, services, and docs (G+) | Copilot | Home > Event Hubs > govind-event-hub | events (govind-event-hub/events) | Overview | Consumer group | Delete | Refresh | Give feedback | Summarize properties for this Event Hubs Instance | How do I troubleshoot this Event Hubs Instance? | List consumers connected to this Event Hubs Instance | X

Overview

Access control (IAM)

Diagnose and solve problems

Data Explorer

Resource visualizer

Settings

Entities

Features

Automation

Help

Essentials

Resource group (move) : Azure resource group	Status : Active
Location : East Asia	Namespace : govind-event-hub
Subscription (move) : Azure subscription 1	Created : Sunday, September 21, 2025 at 10:23:34 GMT+5:30
Subscription ID : b41ac736-1671-4724-a6f2-6080db61224b	Updated : Sunday, September 21, 2025 at 10:23:34 GMT+5:30
Partition count : 1	Cleanup policy : Delete

Capture events
Use Capture to save your events to persistent storage.

Process data
Process data instantly with Azure Stream Analytics.

Connect
Authenticate with connection strings and SAS policies.

Checkpoint
Create consumer groups to checkpoint your events.

Data Explorer
Transmit or inspect data in your Event Hub.

Analyze data (preview)
Ingest to Data Explorer for near real-time analysis.

- ✓ Create a notebook and make connection with storage account.
- ✓ USE THE AZURE KEY – VAULT to STORE THE SECRETS.

```

1 storageKey = dbutils.secrets.get(scope="govindscope", key="azure-storagekey")
2
3 spark.conf.set(
4     "fs.azure.account.key.azurestorageaccount52dfs.core.windows.net",
5     storageKey
6 )
7

```

```

1 storage_account_name = "azurstorageaccount52"
2 container_name = "data"
3 account_key = storageKey
4 mount_point = "/mnt/"

```

```

1 dbutils.fs.mount(
2     source = "wasbs://{}@{}/blob.core.windows.net/",
3     mount_point = mount_point,
4     extra_configs = [
5         f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net: {account_key}"
6     ]
7 )
8
9 print(f"Container '{container_name}' mounted at '{mount_point}'")

```

- ✓ Connect to the event hub

```

1 connectionstring = dbutils.secrets.get(scope="govindscope", key="azure-eventhub-connectionstring")
2 event_hub_config = []
3 "kafka.bootstrap.servers": "govind-event-hub.servicebus.windows.net:9093",
4 "subscribe": "events",
5 "startingOffsets": "latest",
6 "kafka.security.protocol": "SASL_SSL",
7 "kafka.sasl.mechanism": "PLAIN",
8 "kafka.sasl.jaas.config": f"[connectionstring]"
9
10 df_kafka = spark.readStream.format("kafka").options(**event_hub_config).load()

```

- ✓ Schema define for the data

```
22 hours ago (x10) 9 Python ⚡ ⏺ ⏹ ⏹
```

```
1 order_schema = StructType([
2     StructField("event_time", TimestampType()),
3     StructField("event_type", StringType()),
4     StructField("payload", StructType([
5         StructField("OrderID", StringType()),
6         StructField("StoreID", IntegerType()),
7         StructField("CustomerID", IntegerType()),
8         StructField("ProductID", IntegerType()),
9         StructField("Quantity", IntegerType()),
10        StructField("UnitPrice", DoubleType()),
11        StructField("DiscountPct", DoubleType()),
12        StructField("Channel", StringType())
13    ]))
14])
15
16 inventory_schema = StructType([
17     StructField("event_time", TimestampType()),
18     StructField("event_type", StringType()),
19     StructField("payload", StructType([
20         StructField("StoreID", IntegerType()),
21         StructField("ProductID", IntegerType()),
22         StructField("DeltaQty", IntegerType())
23    ])))
24])
```

- ✓ Loading data to the bronze layer which are received from the event hub events.
 - ✓ Separate the events as per the orders and Inventory data

```
▶ 18 hours ago
1 df_kafka = spark.readStream.format("kafka").options(**event_hub_config).load()
2
3 df_kafka_str = df_kafka.selectExpr("CAST(value AS STRING) as json_str")
4
5 df_kafka_str.display()
6
7 # Orders stream
8 orders_df = df_kafka_str.select(from_json(col("json_str"), order_schema).alias("data")) \
9     .filter(col("data.event_type") == "order_created").select(
10         col("data.event_time").alias("event_time"),
11         col("data.payload.OrderID").alias("OrderID"),
12         col("data.payload.StoreID").alias("StoreID"),
13         col("data.payload.CustomerID").alias("CustomerID"),
14         col("data.payload.ProductID").alias("ProductID"),
15         col("data.payload.Quantity").alias("Quantity"),
16         col("data.payload.UnitsPrice").alias("UnitsPrice"),
17         col("data.payload.DiscountPct").alias("DiscountPct"),
18         col("data.payload.Channel").alias("Channel"),
19         to_date(col("data.event_time")).alias("event_date")
20     )
21
22 # Inventory stream
23 inventory_df = df_kafka_str \
24     .select(from_json(col("json_str"), inventory_schema).alias("data")) \
25     .filter(col("data.event_type") == "inventory_update") \
26     .select(
27         col("data.event_time").alias("event_time"),
28         col("data.payload.StoreID").alias("StoreID"),
29         col("data.payload.ProductID").alias("ProductID"),
30         col("data.payload.DeltaQty").alias("DeltaQty"),
31         to_date(col("data.event_time")).alias("event_date")
32     )
```

The screenshot shows a Stream Processing application interface. The top navigation bar includes 'File', 'Edit', 'View', 'Run', 'Help', 'Python', 'Tabs ON', and 'Last edit was 2 minutes ago'. On the right, there are buttons for 'Run all', 'Starting', 'Schedule', and 'Share'. The main area displays a code editor with Python code for stream processing:

```
35
36 # For orders CSV
37 orders_of.writeStream.format("csv").option("path", "/mnt/bronze/stream/orders") \
38 .option("checkpointLocation", "/mnt/checkpoints/orders_csv").partitionBy("event_date").outputMode("append").start()
39
40 # For inventory CSV
41 inventory_of.writeStream.format("csv").option("path", "/mnt/bronze/stream/inventory_update") \
42 .option("checkpointLocation", "/mnt/checkpoints/inventory_csv").partitionBy("event_date").outputMode("append").start()
43
44 # Orders Delta
45 orders_of.writeStream.format("delta").option("path", "/mnt/bronze/stream/orders_delta") \
46 .option("checkpointLocation", "/mnt/checkpoints/orders_delta").partitionBy("event_date").outputMode("append").start()
47
48 # Inventory Delta
49 inventory_of.writeStream.format("delta").option("path", "/mnt/bronze/stream/inventory_delta") \
50 .option("checkpointLocation", "/mnt/checkpoints/inventory_delta").partitionBy("event_date").outputMode("append").start()
51
52 (5) Spark Jobs
53
54 ⚡ 0d6081e-fc2a-428b-b27c-168b65c44c0 Last updated: 18 hours ago
55 ⚡ e662a2797-9f9e-44d7-adca-8a49c496b0e7 Last updated: 18 hours ago
56 ⚡ 8fadec0d-e5c5-4bb8-9d99-de29a2d14233 Last updated: 18 hours ago
57 ⚡ e0f2ca06-f280-49b0-bba0-f0c067978b16 Last updated: 18 hours ago
58 ⚡ display_query_6 (id: 75cbf513-6b65-4a79-add1-b9c8e7359409) Last updated: 18 hours ago
```

Below the code editor is a table section with a 'Table' dropdown and a '+' button. It contains one row:

	json_str
1	[{"event_time": "2025-08-15T00:00:30", "event_type": "order_created", "payload": {"OrderID": 5175521633011, "StoreID": 3, "CustomerID": 2054, "ProductID": 199, "Quantity": 1, "UnitPrice": 12.99, "Tax": 0.0, "Total": 12.99}, "source": "order_created"}, {"event_time": "2025-08-15T00:00:30", "event_type": "order_created", "payload": {"OrderID": 5175521633012, "StoreID": 3, "CustomerID": 2054, "ProductID": 200, "Quantity": 1, "UnitPrice": 14.99, "Tax": 0.0, "Total": 14.99}, "source": "order_created"}]

On the right side of the table are icons for search, filter, sort, and refresh.

- ✓ Broze to silver after cleaning and validating the data

```
BatchOrderProcessing > Batchpaymentprocessing > Stream processing >
File Edit View Run Help Python Tabs: ON Last edit was 2 minutes ago Run all POIREDDI GOVIND's... Schedule
BRONZE TO SILVER

18 hours ago
1 bronze_orders = spark.readStream.format("delta").load("/mnt/bronze/stream/orders_delta")
2
3 # Clean and transform
4 orders_clean = bronze_orders \
5     .withColumn("Quantity", col("Quantity").cast("int")) \
6     .withColumn("UnitPrice", col("UnitPrice").cast("double")) \
7     .withColumn("DiscountPct", col("DiscountPct").cast("double")) \
8     .filter(col("OrderID").isNotNull()) \
9     .withColumn(
10         "totalAmount",
11         (col("Quantity") * col("UnitPrice")) * (1 - col("DiscountPct") / 100).cast(DecimalType(10, 2))
12     )
13
14 # Function to write unique records to Silver using MERGE
15 def write_unique_to_silver(batch_df, batch_id):
16     # Drop duplicates by OrderID
17     batch_df = batch_df.dropDuplicates(["OrderID"])
18
19     if DeltaTable.isDeltaTable(spark, "/mnt/silver/stream/orders"):
20         silver_table = DeltaTable.forPath(spark, "/mnt/silver/stream/orders")
21         silver_table.alias("silver").merge(
22             batch_df.alias("batch"),
23             "silver.OrderID = batch.OrderID"
24         ).whenNotMatchedInsertAll().execute()
25     else:
26         # First batch: create Silver Delta table
27         batch_df.write.format("delta").mode("overwrite").save("/mnt/silver/stream/orders")
28
29 # Start streaming with forecast
30 query = orders_clean.writeStream \
31     .foreachBatch(write_unique_to_silver) \
32     .option("checkpointLocation", "/mnt/checkpoints/orders_silver") \
33     .start()
34
```

```
bronze_inventory = spark.readStream.format("delta").load("/mnt/bronze/stream/inventory_delta")
inventory_clean = bronze_inventory \
    .withColumn("DeltaQty", col("DeltaQty").cast("int")) \
    .filter(col("StoreID").isNotNull() & col("ProductID").isNotNull())
    
def write_unique_to_silver(batch_df, batch_id):
    # Drop exact duplicates within the batch
    batch_df = batch_df.dropDuplicates()
    
    if DeltaTable.isDeltaTable(spark, "/mnt/silver/stream/inventory"):
        silver_table = DeltaTable.forPath(spark, "/mnt/silver/stream/inventory")
        # Merge: insert only new rows that don't exist yet (exact match on all columns)
        join_condition = " AND ".join(["silver.{c} = batch.{c}" for c in batch_df.columns])
        silver_table.alias("silver").merge(
            batch_df.alias("batch"),
            join_condition
        ).whenNotMatchedInsertAll().execute()
    else:
        # First batch: create Silver Delta table
        batch_df.write.format("delta").mode("overwrite").save("/mnt/silver/stream/inventory")

# Start streaming with foreachBatch
query = inventory_clean.writeStream \
    .foreachBatch(write_unique_to_silver) \
    .option("checkpointlocation", "/mnt/checkpoints/inventory_silver") \
    .start()

# (1) Spark Jobs
```

The screenshot shows a Stream processing tab in a development environment with two code snippets:

SILVER TO GOLD

```
▶ 18 hours ago 18
1 orders_silver_df = spark.readStream.format("delta").load("/mnt/silver/stream/orders")
2
3 # Transform to FactOrders format
4 # orders_gold_df = orders_silver_df.withColumn("PaymentID", lit("Unknown")) \
5 # .withColumn("PaymentMethod", lit("Unknown")).withColumn("Status", lit("Unknown")).withColumn("OrderTotal", col("TotalAmount")) \
6 # .withColumn("Currency", lit("INR"))
7
8 # Write to Gold Delta
9 orders_silver_df.writeStream.format("delta").option("path", "/mnt/gold/stream/orders") \
10 .option("checkpointLocation", "/mnt/checkpoints/gold_orders") \
11 .outputMode("append") \
12 .start()
▶ (1) Spark Jobs
> ⚡ d3dc61dc-e4f9-457f-8afb-eb5499d21b22 Last updated: 18 hours ago
▶ 📄 orders_silver_df: pyspark.sql.dataframe.DataFrame = [event_time:timestamp, OrderID: string ... 9 more fields]
<pyspark.sql.streaming.query.StreamingQuery at 0x7f4a616fffc0>
```



```
▶ 18 hours ago 19
1 inventory_silver_df = spark.readStream.format("delta").load("/mnt/silver/stream/inventory")
2
3 # inventory_silver_df.show()
4
5 inventory_silver_df.writeStream.format("delta").option("path", "/mnt/gold/stream/inventory") \
6 .option("checkpointLocation", "/mnt/checkpoints/gold_inventory").outputMode("append").start()
▶ (1) Spark Jobs
> ⚡ b3fe9837-afc2-427e-b98b-abef9646b9e1 Last updated: 18 hours ago
▶ 📄 inventory_silver_df: pyspark.sql.dataframe.DataFrame = [event_time:timestamp, StoreID: integer ... 3 more fields]
<pyspark.sql.streaming.query.StreamingQuery at 0x7f4a61562bf0>
```

✓ Gold to snowflake table

The screenshot shows a Jupyter Notebook interface with three tabs: 'BatchOrderprocessing', 'Batchpaymentprocessing', and 'Stream processing'. The 'Stream processing' tab is active and contains the following PySpark code:

```
gold to snowflake

18 hours ago 22
1 from snowflake.snowpark import Session

18 hours ago 23
1 from pyspark.sql.functions import col
2
3 # Prepare DataFrame exactly as Snowflake table
4 fact_orders_df = orders_gold_df.select(
5     col("OrderID").cast("string"),
6     col("event_time").alias("OrderDateTime"),
7     col("StoreID").cast("int"),
8     col("CustomerID").cast("int"),
9     col("ProductID").cast("int"),
10    col("Quantity").cast("int"),
11    col("UnitPrice").cast("double"),
12    col("DiscountPct").cast("double"),
13    col("Channel").cast("string"),
14    col("TotalAmount").cast("double")
15 )
16

18 hours ago 23
16
17 def write_to_snowflake(batch_df, batch_id):
18     if batch_df.count() == 0:
19         return
20
21     batch_df = batch_df.dropDuplicates()
22
23     batch_df.write \
24         .format("snowflake") \
25         .options(
26             sfURL="SVVXXKL-FR13613.snowflakecomputing.com",
27             sfUser="BHUBANESWAR",
28             sfPassword="Dck5n73Nm5Dt8",
29             sfDatabase="TRAINING",
30             sfSchema="PUBLIC",
31             sfWarehouse="SIGMOID",
32             sfRole="ACCOUNTADMIN"
33         ) \
34         .option("dbtable", "Stream_FactOrders") \
35         .mode("append") \
36         .save()
37
38 query = fact_orders_df.writeStream \
39     .foreachBatch(write_to_snowflake) \
40     .option("checkpointLocation", "/mnt/checkpoints/Stream_FactOrders_snowflake") \
41     .outputMode("append") \
42     .start()
43

(I) Spark Jobs
@fc5a9d4-376e-4fec-bf5f-258d1ed9855 Last updated: 18 hours ago
fact_orders_df: pyspark.sql.dataframe.DataFrame = [OrderID: string, OrderDateTime: timestamp ... 8 more fields]
```

✓ INVENTORY UPDATE TO THE SNOWFLAKE INVENTORY TABLE

The screenshot shows a Jupyter Notebook interface with three tabs: 'BatchOrderprocessing', 'Batchpaymentprocessing', and 'Stream processing'. The 'Stream processing' tab is active and contains the following PySpark code:

```
26
1 from pyspark.sql.functions import col
2 import snowflake.connector
3
4 inventory_gold = spark.readStream.format("delta").load("/mnt/gold/stream/inventory")
5
6 # Upsert function
7 def upsert_inventory_snowflake(batch_df, batch_id):
8     if batch_df.count() == 0:
9         return
10    batch_df = batch_df.dropDuplicates()
11    conn_params = {
12        "user": "BHUBANESWAR",
13        "password": "Dck5n73Nm5Dt8",
14        "account": "SVVXXKL-FR13613",
15        "warehouse": "SIGMOID",
16        "database": "TRAINING",
17        "schema": "PUBLIC",
18        "role": "ACCOUNTADMIN"
19    }
20    conn = snowflake.connector.connect(**conn_params)
21    cs = conn.cursor()
22    # Generate and execute MERGE for each row
```

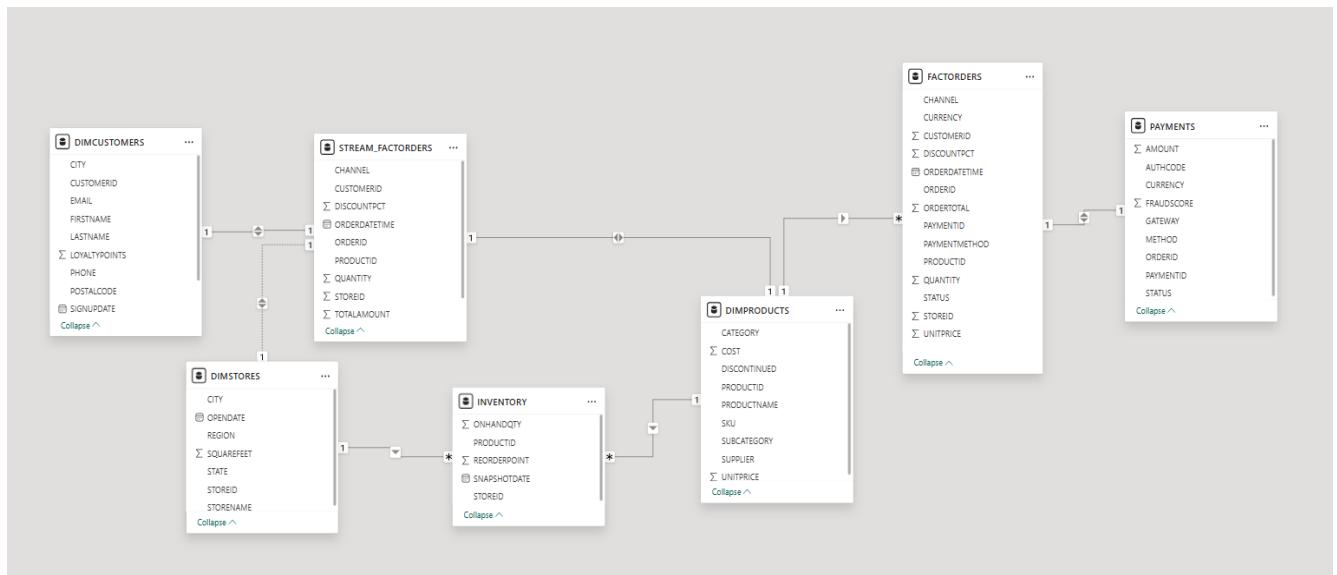
```

23     for row in batch_df.collect():
24         merge_sql = f"""
25             MERGE INTO INVENTORY target
26             USING (SELECT {row.StoreID} AS StoreID, {row.ProductID} AS ProductID, {row.DeltaQty} AS OnHandQty) source
27             ON target.StoreID = source.StoreID AND target.ProductID = source.ProductID
28             WHEN MATCHED THEN UPDATE SET target.OnhandQty = target.OnhandQty + source.OnhandQty
29             WHEN NOT MATCHED THEN INSERT (StoreID, ProductID, OnHandQty)
30             VALUES (source.StoreID, source.ProductID, source.OnhandQty)
31         """
32         cs.execute(merge_sql)
33
34     cs.close()
35     conn.close()
36
37 # Start the streaming job
38 query = inventory_gold.writeStream \
39     .foreachBatch(upsert_inventory_snowflake) \
40     .option("checkpointLocation", "/mnt/checkpoints/inventory_snowflake") \
41     .start()
42

```

SNOWFLAKE WAREHOUSE DATA TABLE:

✓ E - R DIAGRAM:



✓ Snowflake tables

The screenshot shows the Snowflake interface with the following components:

- File Browser:** On the left, it lists files in the "My W..." workspace, including "Advanced snowflake...", "case-study.sql", "Untitled 1.sql" (which is selected), "Untitled.sql", and "Week2_Day_5.sql".
- Worksheets:** Below the file browser, it shows "Owned", "Shared with you", and "Scratchpad".
- Database Explorer:** At the bottom, it displays "Objects" and "Data Products" under the "SNOWFLAKE" connection.
- SQL Editor:** The main right pane contains the following SQL code:

```

1  create warehouse Sigmoid;
2  Create database training;
3
4  use warehouse Sigmoid;
5  use database training;
6
7  CREATE TABLE DimCustomers (
8      CustomerID INT PRIMARY KEY,
9      FirstName VARCHAR(50),
10     LastName VARCHAR(50),
11     Email VARCHAR(100),
12     Phone VARCHAR(20),
13     LoyaltyPoints INT,
14     Tier VARCHAR(20),
15     SignupDate DATE,
16     City VARCHAR(50),
17     State VARCHAR(10),
18     PostalCode VARCHAR(10)
19 );
20
21  CREATE TABLE DimProducts (
22      ProductID INT PRIMARY KEY,
23      SKU VARCHAR(50),
24      ProductName VARCHAR(100),
25      Category VARCHAR(50),
26      Subcategory VARCHAR(50),
27      UnitPrice NUMBER(10,2),
28      Cost NUMBER(10,2),
29      Supplier VARCHAR(50),
30      Discontinued BOOLEAN
31 );

```

```
32      CREATE TABLE DimStores (
33          StoreID INT PRIMARY KEY,
34          StoreName VARCHAR(50),
35          Region VARCHAR(20),
36          City VARCHAR(50),
37          State VARCHAR(10),
38          OpenDate DATE,
39          SquareFeet INT
40      );
41
42      CREATE TABLE FactOrders (
43          OrderID INT PRIMARY KEY,
44          OrderDateTime TIMESTAMP,
45          StoreID INT,
46          CustomerID INT,
47          ProductID INT,
48          Quantity INT,
49          UnitPrice NUMBER(10,2),
50          DiscountPct NUMBER(5,2),
51          PaymentID VARCHAR(50),
52          PaymentMethod VARCHAR(20),
53          Channel VARCHAR(20),
54          Status VARCHAR(20),
55          OrderTotal NUMBER(12,2),
56          Currency VARCHAR(5)
57      );
58
59
```

```
60      CREATE TABLE Payments (
61          PaymentID VARCHAR(50) PRIMARY KEY,
62          OrderID INT,
63          Amount NUMBER(12,2),
64          Currency VARCHAR(5),
65          Method VARCHAR(20),
66          Gateway VARCHAR(20),
67          Status VARCHAR(20),
68          AuthCode VARCHAR(20),
69          FraudScore NUMBER(5,3)
70      );
71
72      CREATE TABLE Stream_FactOrders (
73          OrderID VARCHAR PRIMARY KEY,
74          OrderDateTime TIMESTAMP,
75          StoreID INT,
76          CustomerID INT,
77          ProductID INT,
78          Quantity INT,
79          UnitPrice NUMBER(10,2),
80          DiscountPct NUMBER(5,2),
81          Channel VARCHAR(20),
82          TotalAmount NUMBER(12,2)
83      );
84
85      CREATE TABLE INVENTORY (
86          SnapshotDate DATE,
87          StoreID INT,
88          ProductID INT,
89          OnHandQty INT,
90          ReorderPoint INT
91      );
92
```

- ✓ Loading data from ADLS to a stage in Snowflake and copy into the tables

```

92    -- Or directly use account key in stage
93    CREATE OR REPLACE STAGE my_adls_stage
94        URL='azure://azurerestorageaccount52.blob.core.windows.net/data/raw'
95        CREDENTIALS=(AZURE_SAS_TOKEN='sv=2024-11-04&ss=bfqt&rt=sco&sp=rwdlacupyx&se=2025-09-25T15:06:13Z&st=2025-09-
21T06:51:13Z&sp=https&sig=zmaJXehCq%2BfZALZVxuH14gnhSqy8SrVMuZif5IpPw%3D');
96
97    LIST @my_adls_stage;
98
99    CREATE OR REPLACE FILE FORMAT my_csv_format
100        TYPE = 'CSV'
101        FIELD_OPTIONALLY_ENCLOSED_BY = ''
102        SKIP_HEADER = 1
103        FIELD_DELIMITER = ',';
104
105    COPY INTO DIMCUSTOMERS FROM @my_adls_stage/dim_customers.csv FILE_FORMAT = (FORMAT_NAME = my_csv_format);
106    COPY INTO DIMPRODUCTS FROM @my_adls_stage/dim_products.csv FILE_FORMAT = (FORMAT_NAME = my_csv_format);
107    COPY INTO DIMSTORES FROM @my_adls_stage/dim_stores.csv FILE_FORMAT = (FORMAT_NAME = my_csv_format);
108    COPY INTO INVENTORY FROM @my_adls_stage/inventory_snapshot_daily.csv FILE_FORMAT = (FORMAT_NAME = my_csv_format);
109

```

CUSTOMER DATA SECURITY:

PII MASKING:

- ✓ For the customer table mask the email and phone number

```

124
125    -- PII masking
126
127    -- Masking policy for PII columns
128    CREATE OR REPLACE MASKING POLICY pii_mask AS (val STRING)
129        RETURNS STRING ->
130        CASE
131            WHEN CURRENT_ROLE() IN ('FULL_ACCESS_ROLE') THEN val
132            ELSE
133                CASE
134                    WHEN POSITION('@', val) > 1 THEN
135                        CONCAT(
136                            LEFT(val, 1),
137                            '*****',
138                            SUBSTR(val, POSITION('@', val)))
139                    ELSE '****REDACTED****'
140                END
141        END;
142
143

```

```

143
144
145    CREATE OR REPLACE MASKING POLICY phone_partial AS (val STRING)
146        RETURNS STRING ->
147        CASE
148            WHEN CURRENT_ROLE() IN ('FULL_ACCESS_ROLE') THEN val
149            ELSE CONCAT('*****', RIGHT(val,2))
150        END;
151
152
153    ALTER TABLE DimCustomers
154        MODIFY COLUMN Email
155        SET MASKING POLICY pii_mask;
156
157    ALTER TABLE DimCustomers
158        MODIFY COLUMN Phone
159        SET MASKING POLICY phone_partial;
160

```

Customer Table:

ID	CUSTOMERID	FIRSTNAME	LASTNAME	EMAIL	PHONE	LOYALTYPOINTS	TIER	SIGNUPDATE	CITY	STATE	POSTALCODE
1	1	Vivaan	Singh	v*****@example.com	*****23	547	Silver	2023-12-29	Nagpur	IN	432043
2	2	Pooja	Verma	p*****@example.com	*****25	559	Bronze	2021-05-21	Mumbai	IN	761820
3	3	Adrika	Patel	a*****@example.com	*****56	456	Bronze	2021-07-05	Ahmedabad	IN	203225
4	4	Ananya	Chettri	a*****@example.com	*****25	490	Silver	2022-09-06	Jaipur	IN	130353
5	5	Vihaan	Singh	v*****@example.com	*****19	336	Bronze	2021-04-25	Bhopal	IN	475876
6	6	Riya	Bose	r*****@example.com	*****78	423	Silver	2022-05-16	Jaipur	IN	532067
7	7	Kiran	Menon	k*****@example.com	*****73	951	Bronze	2021-12-25	Jaipur	IN	739098
8	8	Vikash	Patel	v*****@example.com	*****38	873	Bronze	2024-04-09	Bhubaneswar	IN	343467
9	9	Sneha	Das	s*****@example.com	*****81	0	Bronze	2022-06-07	Bhubaneswar	IN	109853
10	10	Sneha	Nair	s*****@example.com	*****47	77	Silver	2024-01-25	Pune	IN	177449
11	11	Sneha	Yadav	s*****@example.com	*****25	266	Bronze	2022-09-05	Mumbai	IN	544920
12	12	Aarohi	Chettri	a*****@example.com	*****84	780	Bronze	2022-09-11	Nagpur	IN	503380
13	13	Pooja	Iyer	p*****@example.com	*****72	881	Bronze	2023-02-21	Bengaluru	IN	476395
14	14	Kiran	Nair	k*****@example.com	*****53	716	Bronze	2022-08-28	Ahmedabad	IN	970357
15	15	Vikash	Das	v*****@example.com	*****84	161	Bronze	2024-03-05	Delhi	IN	819112
16	16	Vihaan	Mishra	v*****@example.com	*****18	342	Silver	2022-08-10	Nagpur	IN	937665
17	17	Pavanch	Yadav	p*****@example.com	*****62	646	Bronze	2023-01-25	Regaonpur	IN	262801

AZURE KEY VAULT:

✓ Created a Azure key vault

The screenshot shows the Azure Key Vault overview page for 'akv-gov-01'. Key details include:

- Resource group: Azure-resource-group
- Location: West US 2
- Subscription: Azure subscription_1
- Sku (Pricing tier): Standard
- Directory ID: 477314c4-d0b3-42bd-9855-e5d52bf6a1aa
- Directory Name: Indian Institute of Technology Bhubaneswar
- Soft-delete: Enabled
- Purge protection: Disabled

The 'Manage keys and secrets used by apps and services' section provides a recommendation to use a vault per application per environment (Development, Pre-Production and Production). It also notes that this helps to not share secrets across environments and reduces the threat in case of a breach.

✓ Created secrets to use it in the databricks by using Scope

The screenshot shows the 'Create a secret' form in the Azure Key Vault. The fields are as follows:

- Name: Azure-Account-key
- Secret value: (Redacted)
- Content type (optional): (Empty)
- Set activation date: (Empty)
- Set expiration date: (Empty)
- Enabled: Yes
- Tags: 0 tags

At the bottom are 'Create' and 'Cancel' buttons.

✓ Created a scope inside the Databricks storage

The screenshot shows the 'Create Secret Scope' dialog in Databricks. The fields are:

- Scope Name: RetailScope
- Manage Principal: Creator
- Azure Key Vault: Selected
- DNS Name: https://akv-gov-01.vault.azure.net/
- Resource ID: /subscriptions/b41ac736-1671-4724-a6f2-6080db61224b/resourceGroups/Azure-re...

A 'Create' button is at the bottom.

MONITORING & ERROR HANDLING:

MONITORING :

- ✓ Create a log analytics workspace

The screenshot shows the 'Log Analytics workspace' settings page for 'azure-log-analytics'. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Logs, Resource visualizer, Settings, Classic, Monitoring, Automation, and Help. The right panel displays workspace details such as Name (azure-log-analytics), Status (Active), Location (West US 2), Subscription (Azure subscription 1), and Tags (None). A 'Get Started' button and a 'Recommendations' section are also present.

- ✓ Open LOGS and then use KQL mode

The screenshot shows the 'Logs' page for the 'azure-log-analytics' workspace. The left sidebar lists categories such as Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Logs, Resource visualizer, Settings, Classic, Legacy agents management, Legacy activity log connector, Legacy storage account logs, Legacy computer groups, Legacy solutions, and System center. The main area features a query editor with a 'New Query 1' tab, a 'Run' button, a time range selector ('Last 24 hours'), and a results count ('Show : 1000 results'). Below the editor is a 'Query history' section listing previous queries with their results and run buttons.

- ✓ Query the logs

The screenshot shows the 'New Query 1' editor. The top bar includes a save and share button, a 'Queries hub' link, and a 'KQL mode' dropdown. The main area contains a query editor with the following KQL code:
1
2 AzureDiagnostics | where ResourceType == "DATAPROCS" | project TimeGenerated, Resource, OperationName, RunOn_s | order by TimeGenerated desc

ALERTS:

- ✓ Create alerts for pipeline failures and activity failures

The screenshot shows the 'Alerts & metrics' section of the Microsoft Azure Data Factory interface. On the left, there's a sidebar with icons for Dashboards, Runs, Pipeline runs, Trigger runs, Change Data Capture, Runtimes & sessions, Integration runtimes, Data flow debug, and Notifications. The 'Alerts & metrics' option is selected. The main area displays a table of alerts with columns: ALERT, ENABLED, RESOURCE TYPE, RESOURCES, and ACTIONS. The alerts listed are:

ALERT	ENABLED	RESOURCE TYPE	RESOURCES	ACTIONS
pipeline alert	On	Pipeline	3	
copy Activity Alert	On	Activity	6	
Notebook Activity Alert	On	Activity	6	
Trigger fail alert	On	Trigger	2	
pipeline failure alert	On	Pipeline	3	

- ✓ I have received an alert when the pipeline was failed

Azure: Deactivated Severity: 0 pipeline alert External Inbox x

The screenshot shows an email from Microsoft Azure. The subject is 'Azure: Deactivated Severity: 0 pipeline alert'. The email body starts with 'Microsoft Azure <azure-noreply@microsoft.com>' and 'Unsubscribe'. It includes a timestamp '1:22PM (3 hours ago)' and icons for reply, forward, and more. Below the subject, it says 'to me ▾'. The Microsoft Azure logo is at the top. The main content is a green checkmark icon followed by the text 'Your Azure Monitor alert was resolved'.

Azure monitor alert rule pipeline alert was resolved for azuredataproxy123 at September 22, 2025 7:52 UTC.

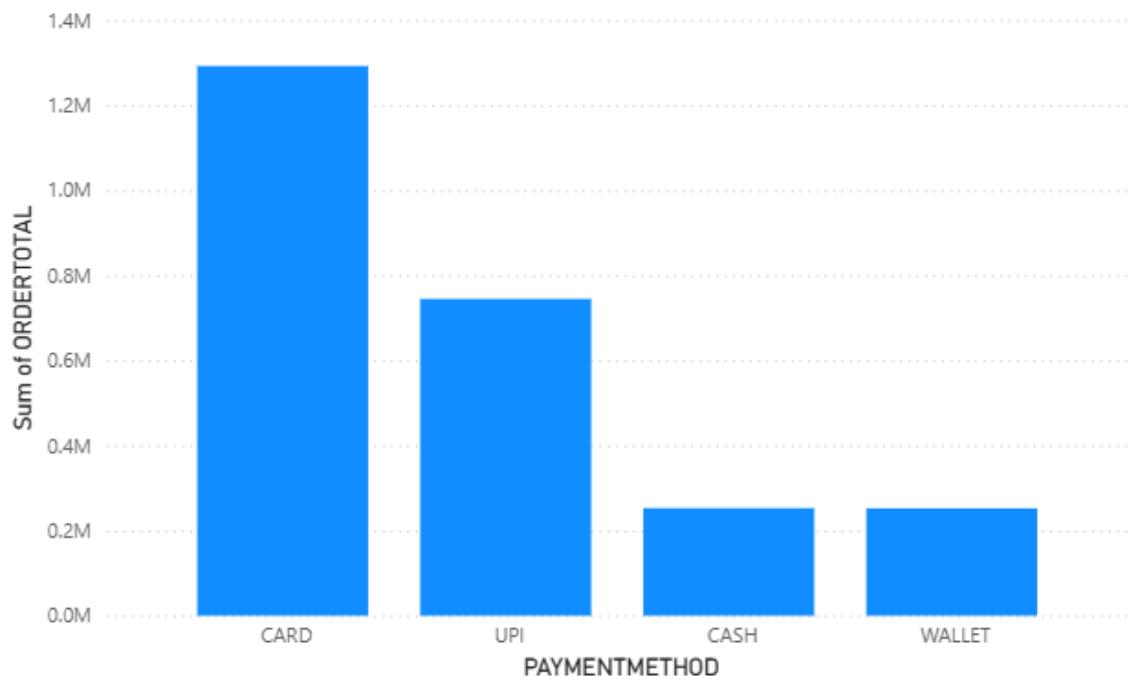
Rule ID	/subscriptions/b41ac736-1671-4724-a6f2-6080db61224b/resourcegroups/Azure-resource-group/providers/Microsoft.Insights/metricalerts/pipeline alert View Rule >
Resource ID	/subscriptions/b41ac736-1671-4724-a6f2-6080db61224b/resourcegroups/Azure-resource-group/providers/Microsoft.DataFactory/factories/azuredataproxy123 View Resource >

Alert deactivated because one of the following conditions is no longer true

POWER BI:

- ✓ Received more payments through CARD

Sum of ORDERTOTAL by PAYMENTMETHOD



- ✓ TOP 10 HIGHEST PAYMENT ORDERS:

	Sum of ORDERTOTAL	Sum of QUANTITY	PAYMENTID
1	4,134.00	78.00	PMT00005815
2	2,183.04	48.00	PMT00008562
3	6,695.04	88.00	PMT00009265
4	4,978.05	105.00	PMT00010147
5	2,329.32	42.00	PMT00020180
6	3,092.40	40.00	PMT00022914
7	2,511.88	28.00	PMT00024537
8	4,909.96	44.00	PMT00025521
9	5,337.16	43.00	PMT00027202
10	2,471.04	99.00	PMT00028704
	38,641.89	615.00	

GIT CONFIGURATION:

- ✓ Connected to my personal git account

The screenshot shows the 'Git repository' configuration page for a data factory named 'azuredadata-factory123'. The left sidebar lists various configuration sections: General, Connector upgrade advis..., Factory settings, Connections, Linked services, Integration runtimes, Microsoft Purview, ADF in Microsoft Fabric, Source control (Git configuration is selected), ARM template, Author, Triggers, Global parameters, Data flow libraries, Security, Credentials, and Customer managed key. The main pane displays the following details:

Setting	Value
Repository type	GitHub
GitHub account	polireddigovind
Repository name	AZURE-RETAIL-CAPSTONE-PROJECT
Collaboration branch	main
Publish branch	adf_publish
Root folder	/
Last published commit	455fc...f2
Publish (from ADF Studio)	Enabled
Custom comment	Enabled

- ✓ Published it into my repository

The screenshot shows the 'orderspipeline' pipeline editor in the Azure Data Factory interface. The left sidebar shows 'Factory Resources' with 'Pipelines' expanded, showing 'orderspipeline' and 'paymentpipeline'. The main workspace displays a pipeline diagram with two activities: 'Copy data' and 'Notebook'. The 'Copy data' activity is configured to copy data from a dataset named 'paymentdata' to a notebook named 'paymentdatab'. Below the diagram, the 'Parameters' tab is selected, showing a single parameter named 'filename' of type String with a default value of 'Value'. The top right corner shows the user's email '21CS01053@iitbbs.ac.in' and the message 'INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR'.

✓ Git Repository Updation

AZURE-RETAIL-CAPSTONE-PROJECT

Commits

- polireddigovind Update publish_config.json (4 commits)
- dataset
- factory
- linkedService
- pipeline
- trigger
- Create README.md
- Update publish_config.json

README

About
No description, website, or topics provided.

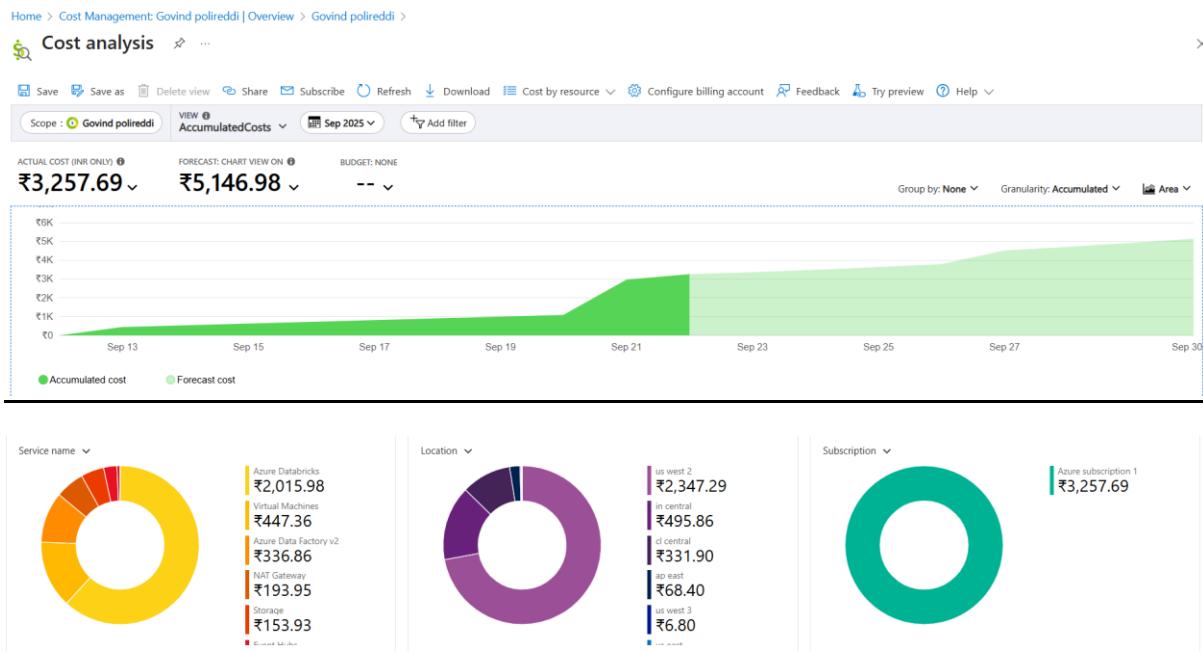
Activity
0 stars
0 watching
0 forks

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

COST ESTIMATION:

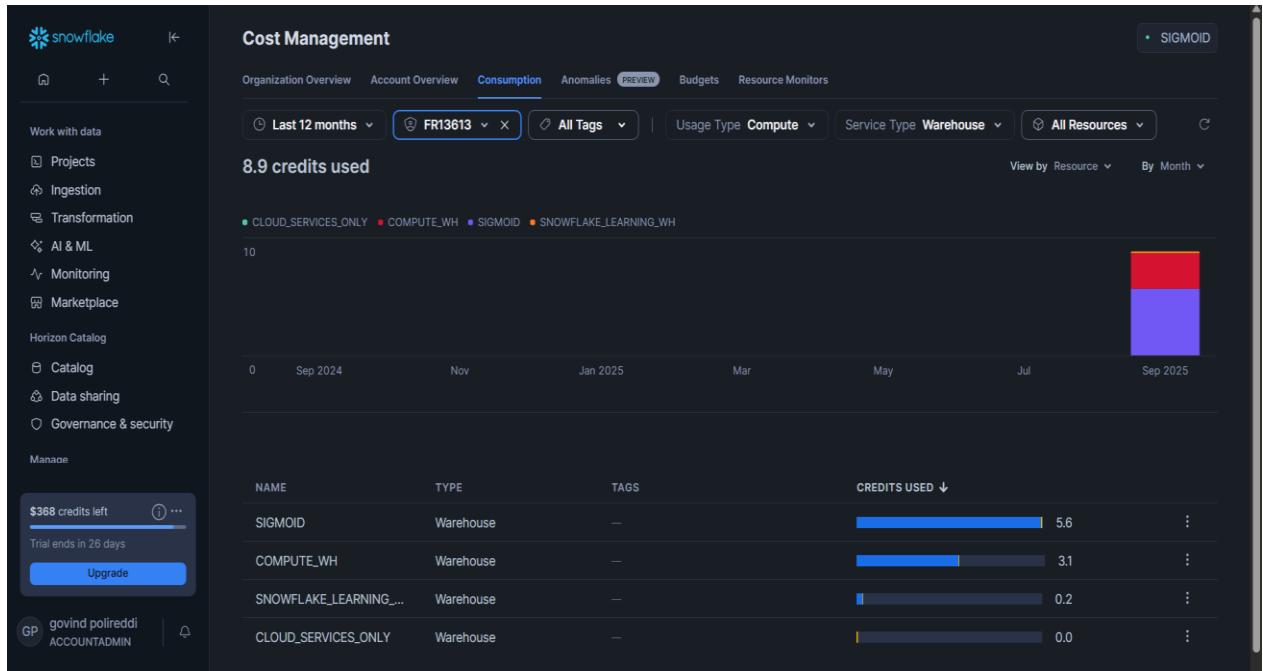
AZURE COST:



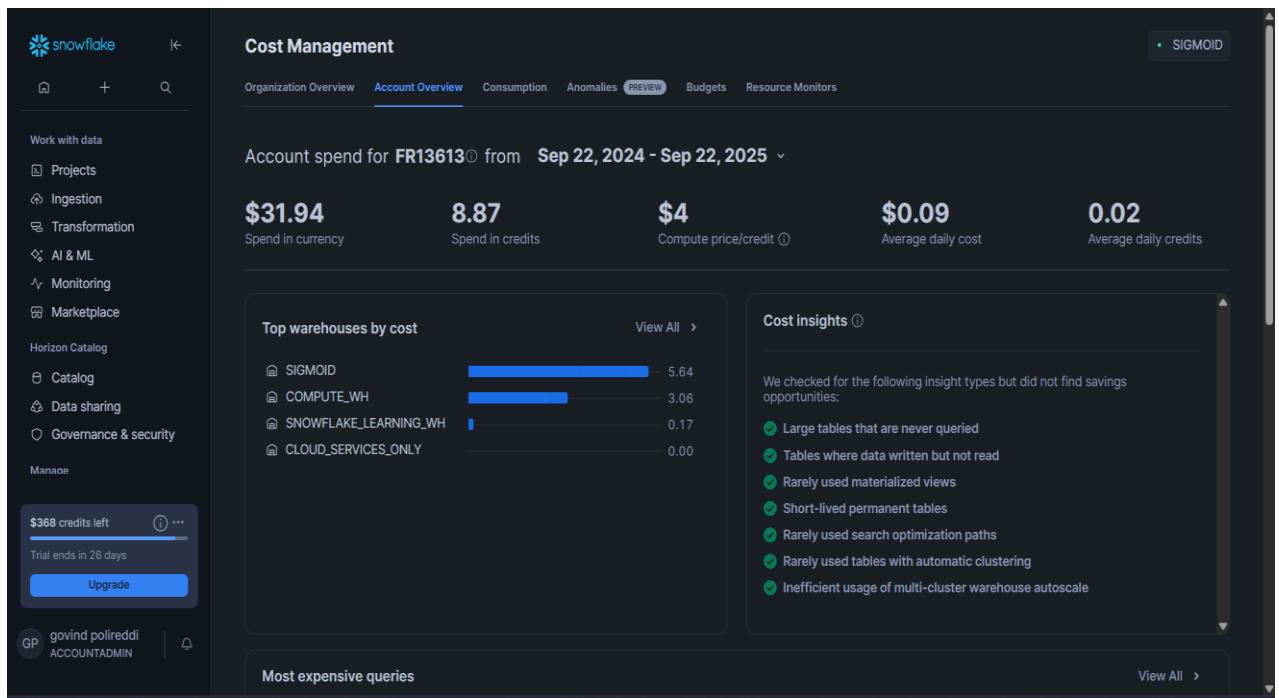
- ✓ The DataBricks Has Comsumed 2015.98 in INR.
- ✓ DataBricks has consumed more than 60% of the total cost.

SNOWFLAKE COST:

- ✓ Compute cost occurred and costed about 8.9 credits



- ✓ Total cost occurred is about 32 dollars(2700 INR)



COMPARISION BETWEEN AZURE AND SNOWFLAKE:

- ✓ For this project the Azure costed 3200 INR and SNOWFLAKE costed 2700 INR
- ✓ I have used more AZURE SERVICES when compared to SNOWFLAKE
- ✓ SNOWFLAKE is Costlier than the AZURE.

STEPS OR MEASURES FOR COST OPTIMIZATION:

Azure

- Storage: Use Hot/Cool/Archive tiers + lifecycle policies.
 - ADF: Use auto-pause on Integration Runtime, avoid too many activities.
 - Databricks: Enable auto-termination, use job clusters, pick smaller/spot nodes.
 - Event Hub: Scale down throughput units, set short retention, archive to Blob.
-

Snowflake

- Warehouses: Enable auto-suspend/auto-resume, size warehouses correctly.
- Storage: Limit Time Travel, drop unused/stale data, load compressed files.
- Queries: Avoid SELECT *, use result caching & clustering keys.
- Monitoring: Set up Resource Monitors to cap credit usage.