

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.

APPROACH:

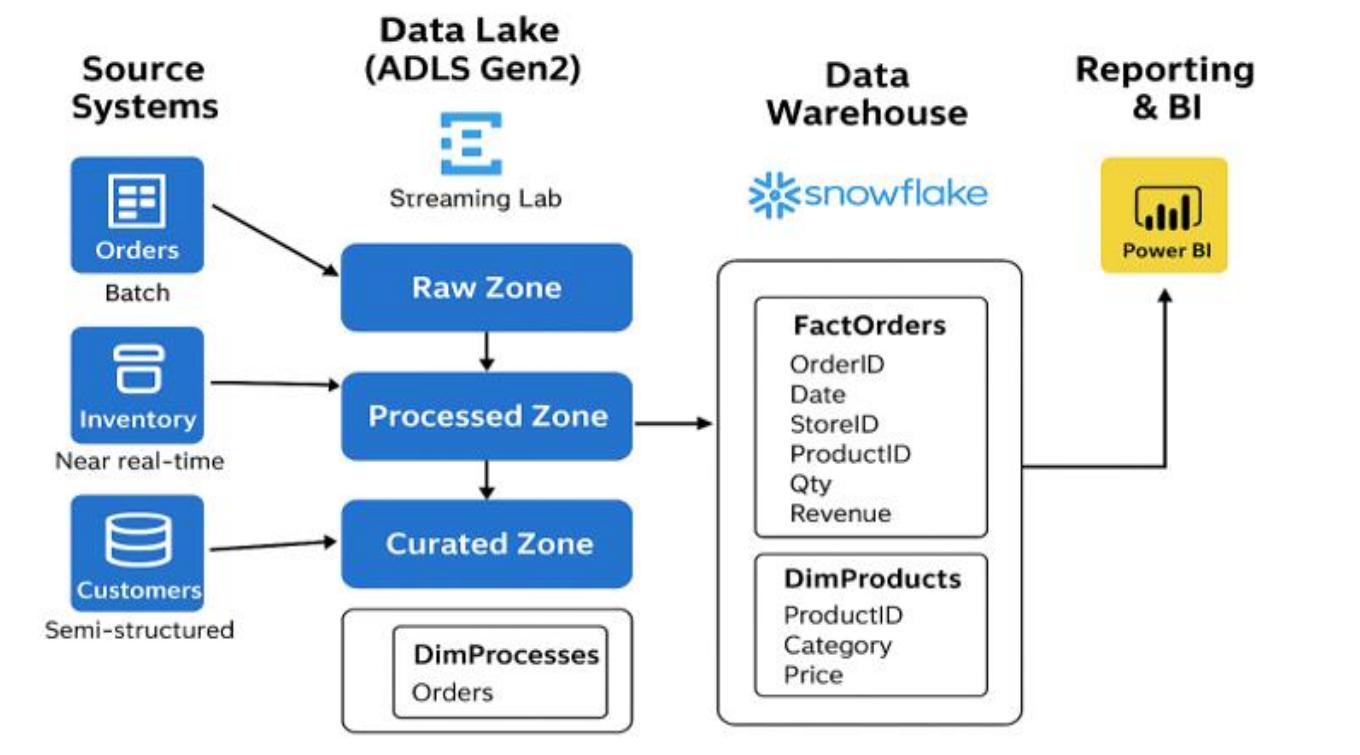
- **Step-1:** Creating a ADLS storage account to store the raw data and also the cleaned and validated data in medallion structure (Bronze, Silver, Gold).
- **Step-2:** For **BATCH-PROCESSING** create 2 ADF(Azure Data Factory) Pipelines one for each of the Batch Orders CSV and Payments CSV files.
- **Step-3:** 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
dim_customers.csv	9/20/2025, 9:48:32 PM	Hot (Inferred)	Block blob	578.01 KiB	Available
dim_products.csv	9/20/2025, 9:48:31 PM	Hot (Inferred)	Block blob	19.2 KiB	Available
dim_stores.csv	9/20/2025, 9:48:31 PM	Hot (Inferred)	Block blob	593 B	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

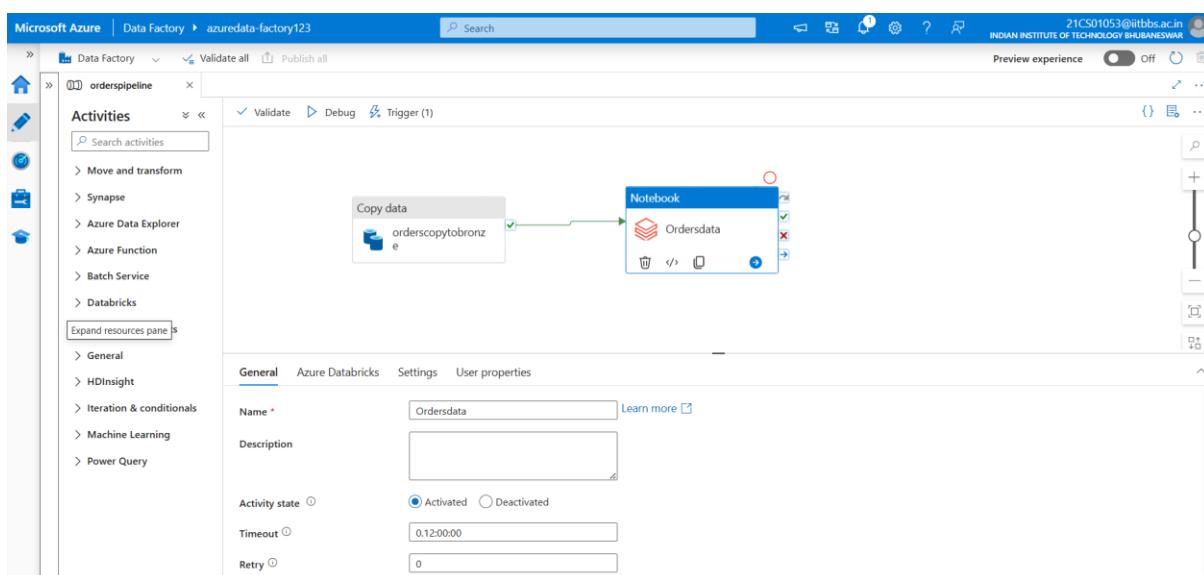
Authentication method: Access key ([Switch to Microsoft Entra user account](#))

Search blobs by prefix (case-sensitive)		Only show active objects					
		Last modified	Access tier	Blob type	Size	Lease state	
<input type="checkbox"/>	Name					...	
<input type="checkbox"/>	[-]					...	
<input type="checkbox"/>	payments.csv	9/21/2025, 10:22:46 PM	Hot (Inferred)	Block blob	1.83 MiB	Available	...

BATCH – PROCESSING:

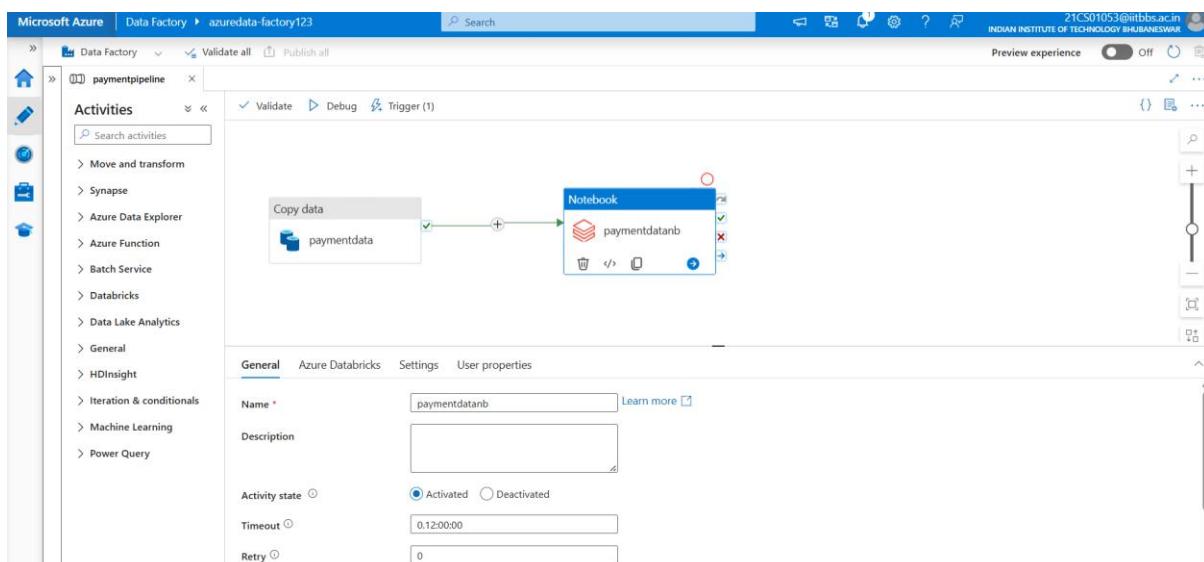
For Batch orders pipeline.

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



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 'Triggers' page in the Microsoft Azure Data Factory interface. On the left sidebar, 'Triggers' is selected under the 'Author' section. The main area displays two triggers: 'trigger1' and 'trigger2', both of type 'Storage events'. The right panel is titled 'Edit trigger' for 'trigger1'. It includes fields for 'Name' (trigger1), 'Description', 'Type' (BlobEventsTrigger), 'Account selection method' (From Azure subscription), 'Azure subscription' (selected), 'Storage account name' (azurstorageaccount52), 'Container name' (data), 'Blob path begins with' (raworders), and 'Blob path ends with'. Buttons for 'Continue' and 'Cancel' are at the bottom.

This screenshot shows the same 'Triggers' page in Microsoft Azure Data Factory. The left sidebar shows 'Triggers' selected. The main area shows triggers 'trigger1' and 'trigger2'. The right panel is titled 'Edit trigger' for 'trigger2'. It has similar configuration fields: 'Name' (trigger2), 'Description', 'Type' (BlobEventsTrigger), 'Account selection method' (From Azure subscription), 'Azure subscription' (selected), 'Storage account name' (azurstorageaccount52), 'Container name' (data), 'Blob path begins with' (rawpayment), and 'Blob path ends with'. Buttons for 'Continue' and 'Cancel' are at the bottom.

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

The screenshot shows the 'Alerts & metrics' page in Microsoft Azure Data Factory. The left sidebar has 'Alerts & metrics' selected. The main area displays a table of alert rules. The columns are 'ALERT', 'ENABLED', 'RESOURCE TYPE', 'RESOURCES', and 'ACTIONS'. The table contains five rows: 'pipeline alert' (Enabled, Pipeline, 3 resources), 'copy Activity Alert' (Enabled, Activity, 6 resources), 'Notebook Activity Alert' (Enabled, Activity, 6 resources), 'Trigger fail alert' (Enabled, Trigger, 2 resources), and 'pipeline failure alert' (Enabled, Pipeline, 3 resources). Each row has edit and delete icons in the 'ACTIONS' column.

CREATE ACTION GROUP:

- ✓ Created action group for notification alerts through email.

Alerts & metrics

ALERT	ENABLED	RESOURCE TYPE
pipeline alert	On	Pipeline
copy Activity Alert	On	Activity
Notebook Activity Alert	On	Activity
Trigger fail alert	On	Trigger
pipeline failure alert	On	Pipeline

Add notification

Action name *

Select which notifications you'd like to receive

Email

SMS
Country code

Azure app push notifications
Enter your email used to log into your Azure account. [Learn about connecting to your Azure resources using the Azure app.](#)

Voice
Country code

Add notification

Alerts & metrics

ALERT	ENABLED	RESOURCE TYPE
pipeline alert	On	Pipeline
copy Activity Alert	On	Activity
Notebook Activity Alert	On	Activity
Trigger fail alert	On	Trigger
pipeline failure alert	On	Pipeline

Configure notification

Create new Use existing

Action group name *

Short name *

Notifications

Notifications	Action type	Actions
email notification	Email/SMS/Push/Voice	

Add notification

Orders Batch processing Notebook:

```

filename to be processed

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

bronze to silver

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

```

Gold to Snowflake table

```

1 pip install snowflake-snowpark-python
2
3 12 hours ago (ts)
4
5
6
7
8
9
10
11
12
13
14
15
16

```

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 08:38:00.000	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 01:44:30:000	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	3806	2025-06-09 18:48:00.000	2	4058	123	1	52.93	10.00	PMT00003806	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 04:13:00.000	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 if DeltaTable.isDeltaTable(spark, gold_path):
4     gold_table = DeltaTable.forPath(spark, gold_path)
5     gold_table.alias("gold").merge(
6         batch_df.alias("new"),
7         "gold.StoreID = new.StoreID AND gold.OrderDate = new.OrderDate"
8     ).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()
9 else:
10     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="Dck5n7J3Nm50ct8",
10        sfDatabase="TRAINING",
11        sfSchema="PUBLIC",
12        sfWarehouse="SIGMOID",
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	MP EOSO1F	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	TXYEP6C1	0.352
8	PMT00002258	2258	130.96	INR	CARD	Stripe	Captured	J9PJICR	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	3W5QUJQ9	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

The screenshot shows the Azure Event Hubs Overview page for the 'govind-event-hub' namespace. The page includes a sidebar with navigation links like 'Name', 'Create', 'Manage view', 'Search', 'Event Hub', 'Delete', 'Refresh', 'Give feedback', 'Overview', 'Access log', 'Tags', 'Diagnose and solve problems', 'Data Explorer', 'Resource visualizer', 'Events', 'Settings', 'Entities', 'Monitoring', 'Automation', and 'Help'. The main content area displays the 'Essentials' section with details such as Resource group (move), Status (Succeeded), Location (East Asia), Subscription (move), Subscription ID, Host name, and various configuration settings like Throughput Units (1 unit), Auto-inflate throughput units (Disabled), and Local Authentication (Enabled). It also shows 'NAMESPACE CONTENTS' (EVENT HUB) and 'KAFKA SURFACE' status. A 'Tags' section allows for editing and adding tags. A time range selector at the bottom shows data for the last hour.

- ✓ Create a new event in the event hub namespace "events"

The screenshot shows the Azure Event Hubs Overview page for the 'events' instance within the 'govind-event-hub' namespace. The page has a similar structure to the previous one, with a sidebar and an 'Overview' section. The 'Essentials' section provides details about the event hub, including its resource group (move), location (East Asia), subscription (move), subscription ID, and partition count. Below the essentials, there are six cards: '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.), and '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.

The screenshot shows a Databricks notebook titled 'connecting Storage account'. The notebook has three cells:

- Cell 4:** Contains Python code to set up storage account access:

```
storageKey = dbutils.secrets.get(scope="govindscope", key="azure-storagekey")  
spark.conf.set(  
    "fs.azure.account.key.azurestorageaccount52dfs.core.windows.net",  
    storageKey  
)
```
- Cell 5:** Contains Python code to mount the storage account:

```
storage_account_name = "azurerestorageaccount52"  
container_name = "data"  
account_key = storagekeys[0]  
mount_point = "/mnt/"
```
- Cell 6:** Contains Python code to verify the mount:

```
dbutils.fs.mount(  
    source=f"wasbs://(container_name)@{storage_account_name}.blob.core.windows.net/",  
    mount_point=mount_point,  
    extra_configs=[  
        f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net: account_key  
    ]  
)  
print(f"Container '{container_name}' mounted at '{mount_point}'")  
Container 'data' mounted at '/mnt/'
```

- ✓ Connect to the event hub

connecting to Azure event-hub

```

22 hours ago (±1)
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
11 df_kafka = spark.readStream.format("kafka").options(**event_hub_config).load()

```

- ✓ Schema define for the data

```

22 hours ago (±1)
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

loading events from event-hub to bronze layer

```

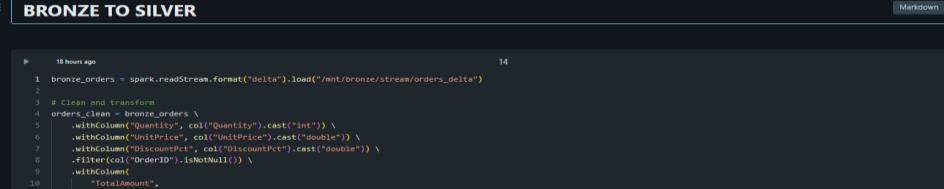
10 hours ago
1 df_kafka = spark.readStream.format("kafka").options(**event_hub_config).load()
2
3 df_kafka_str = df_kafka.selectExpr("CAST(value AS STRING) as json_str")
4
5 df_kafka_str.display()
6
7 # Orders stream
8 orders_df = df_kafka_str.select(from_json(col("json_str"), order_schema).alias("data")) \
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.UnitPrice").alias("UnitPrice"),
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 )
33

```

The screenshot shows a Jupyter Notebook interface with the following details:

- File Menu:** File, Edit, View, Run, Help, Python, Tabs: ON, Last edit was 2 minutes ago.
- Toolbar:** Run all, Starting, Schedule, Share.
- Code Cell:** Contains Scala code for stream processing. It includes logic for writing orders and inventory streams to CSV files, and for writing orders and inventory delta streams to CSV files. It also includes a display query cell.
- Table View:** A table titled "Table" with a single row labeled "json_str". The row contains a JSON object representing an event: {"event_time": "2025-08-15T00:05:30", "event_type": "order_created", "payload": {"OrderID": "S175521633011", "StoreID": 3, "CustomerID": 2054, "ProductID": 199, "Quantity": 1, "Unit...".

- ✓ 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 POLIREDDI GOVIND S... Schedule  
BRONZE TO SILVER  
18 hours ago 14  
bronze_orders = spark.readStream.format("delta").load("/mnt/bronze/stream/orders_delta")  
orders_clean = bronze_orders \  
.withColumn("Quantity", col("Quantity").cast("int")) \  
.withColumn("UnitPrice", col("UnitPrice").cast("double")) \  
.withColumn("DiscountPct", col("DiscountPct").cast("double")) \  
.filter(col("OrderID").isNotNull()) \  
.withColumn(  
    "TotalAmount",  
    (col("Quantity") * col("UnitPrice")) * (1 - col("DiscountPct") / 100).cast(DecimalType(10, 2))  
)  
  
# Function to write unique records to Silver using MERGE  
def write_unique_to_silver(batch_df, batch_id):  
    # Deduplicate batch by OrderID  
    batch_df = batch_df.dropDuplicates(["OrderID"])  
  
    if DeltaTable.isDeltaTable(spark, "/mnt/silver/stream/orders");  
        silver_table = DeltaTable.forName(spark, "/mnt/silver/stream/orders")  
        silver_table.alias("silver").merge(  
            batch_df.alias("batch"),  
            "silver.OrderID = batch.OrderID"  
        ).whenMatchedUpdateAll().execute()  
    else:  
        # First batch: create Silver Delta table  
        batch_df.write.format("delta").mode("overwrite").save("/mnt/silver/stream/orders")  
  
# Start streaming with foreachBatch  
query = orders_clean.writeStream \  
.foreachBatch(write_unique_to_silver) \  
.option("checkpointLocation", "/mnt/checkpoints/orders_silver") \  
.start()
```

```
BatchOrderprocessing Batchpaymentprocessing Stream processing +  
File Edit View Run Help Python Tabs ON Last edit was 2 minutes ago Run all POLIREDDI GOVIND's ... Schedule Sh  
1 bronze_inventory = spark.readStream.format("delta").load("/mnt/bronze/stream/inventory_delta")  
2  
3 # Clean and transform  
4 inventory_clean = bronze_inventory \  
| .withColumn("DeltaQty", col("DeltaQty").cast("int")) \  
| .filter((col("StoreID").isNotNull() & col("ProductID").isNotNull()))  
5  
6 # Function to write unique records to Silver using MERGE  
7 def write_unique_to_silver(batch_df, batch_id):  
8     # Drop exact duplicates within the batch  
9     batch_df = batch_df.dropDuplicates()  
10  
11     if DeltaTable.isDeltaTable(spark, "/mnt/silver/stream/inventory"):  
12         silver_table = DeltaTable.forName(spark, "/mnt/silver/stream/inventory")  
13         # Merge: insert only new rows that don't exist yet (exact match on all columns)  
14         join_condition = " AND ".join(["$silver.{}".format(c) == batch_.format(c) for c in batch_df.columns])  
15         silver_table.alias("silver").merge(  
16             batch_df.alias("batch"),  
17             join_condition  
18         ).whenNotMatchedInsertAll().execute()  
19     else:  
20         # First batch: create Silver Delta table  
21         batch_df.write.format("delta").mode("overwrite").save("/mnt/silver/stream/inventory")  
22  
23     # Start streaming with foreachBatch  
24     query = inventory_clean.writeStream \  
| .foreachBatch(write_unique_to_silver) \  
| .option("checkpointLocation", "/mnt/checkpoints/inventory_silver") \  
| .start()  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
203
```

```

18 hours ago
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()
13
14 (1) Spark Jobs
15 @ d3dc61dc-e4f9-457f-8afb-eb5499d2fb22 Last updated: 18 hours ago
16
17 orders_silver_df: pyspark.sql.dataframe.DataFrame = [event_time: timestamp, OrderID: string ... 9 more fields]
18 <pyspark.sql.streaming.query.StreamingQuery at 0x7fa616ff8c0>

```



```

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()
7
8 (1) Spark Jobs
9 @ b3fe9837-afc2-427e-b98b-abe9646b9e21 Last updated: 18 hours ago
10
11 inventory_silver_df: pyspark.sql.dataframe.DataFrame = [event_time: timestamp, StoreID: integer ... 3 more fields]
12 <pyspark.sql.streaming.query.StreamingQuery at 0x7fa53627bf0>

```

✓ Gold to snowflake table

```

gold to snowflake

18 hours ago (+1d)
1 from snowflake.snowpark import Session
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
916
917
918
918
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
985
986
987
987
988
989
989
990
991
992
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1001
1002
1003
1003
1004
1005
1005
1006
1007
1007
1008
1009
1009
1010
1011
1011
1012
1013
1013
1014
1015
1015
1016
1017
1017
1018
1019
1019
1020
1021
1021
1022
1023
1023
1024
1025
1025
1026
1027
1027
1028
1029
1029
1030
1031
1031
1032
1033
1033
1034
1035
1035
1036
1037
1037
1038
1039
1039
1040
1041
1041
1042
1043
1043
1044
1045
1045
1046
1047
1047
1048
1049
1049
1050
1051
1051
1052
1053
1053
1054
1055
1055
1056
1057
1057
1058
1059
1059
1060
1061
1061
1062
1063
1063
1064
1065
1065
1066
1067
1067
1068
1069
1069
1070
1071
1071
1072
1073
1073
1074
1075
1075
1076
1077
1077
1078
1079
1079
1080
1081
1081
1082
1083
1083
1084
1085
1085
1086
1087
1087
1088
1089
1089
1090
1091
1091
1092
1093
1093
1094
1095
1095
1096
1097
1097
1098
1099
1099
1100
1101
1101
1102
1103
1103
1104
1105
1105
1106
1107
1107
1108
1109
1109
1110
1111
1111
1112
1113
1113
1114
1115
1115
1116
1117
1117
1118
1119
1119
1120
1121
1121
1122
1123
1123
1124
1125
1125
1126
1127
1127
1128
1129
1129
1130
1131
1131
1132
1133
1133
1134
1135
1135
1136
1137
1137
1138
1139
1139
1140
1141
1141
1142
1143
1143
1144
1145
1145
1146
1147
1147
1148
1149
1149
1150
1151
1151
1152
1153
1153
1154
1155
1155
1156
1157
1157
1158
1159
1159
1160
1161
1161
1162
1163
1163
1164
1165
1165
1166
1167
1167
1168
1169
1169
1170
1171
1171
1172
1173
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1181
1182
1183
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1191
1192
1193
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1201
1202
1203
1203
1204
1205
1205
1206
1207
1207
1208
1209
1209
1210
1211
1211
1212
1213
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1221
1222
1223
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1231
1232
1233
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1241
1242
1243
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1251
1252
1253
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1261
1262
1263
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1271
1272
1273
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1281
1282
1283
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1291
1292
1293
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1301
1302
1303
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1311
1312
1313
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1321
1322
1323
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1331
1332
1333
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1341
1342
1343
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1351
1352
1353
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1361
1362
1363
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1371
1372
1373
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1381
1382
1383
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1401
1402
1403
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1411
1412
1413
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1421
1422
1423
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1431
1432
1433
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1441
1442
1443
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1451
1452
1453
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1461
1462
1463
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1471
1472
1473
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1481
1482
1483
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1491
1492
1493
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1501
1502
1503
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1511
1512
1513
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1521
1522
1523
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1531
1532
1533
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1541
1542
1543
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1551
1552
1553
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1561
1562
1563
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1571
1572
1573
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1581
1582
1583
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1591
1592
1593
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1601
1602
1603
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1611
1612
1613
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1621
1622
1623
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1631
1632
1633
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1641
1642
1643
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1651
1652
1653
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1661
1662
1663
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1671
1672
1673
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1681
1682
1683
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1691
1692
1693
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1701
1702
1703
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1711
1712
1713
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1721
1722
1723
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1731
1732
1733
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1741
1742
1743
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1751
1752
1753
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1761
1762
1763
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1771
1772
1773
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1781
1782
1783
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1791
1792
1793
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1801
1802
1803
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1811
1812
1813
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1821
1822
1823
1823
1824
1825
1825
1826
1827
1827

```

✓ INVENTORY UPDATE TO THE SNOWFLAKE INVENTORY TABLE

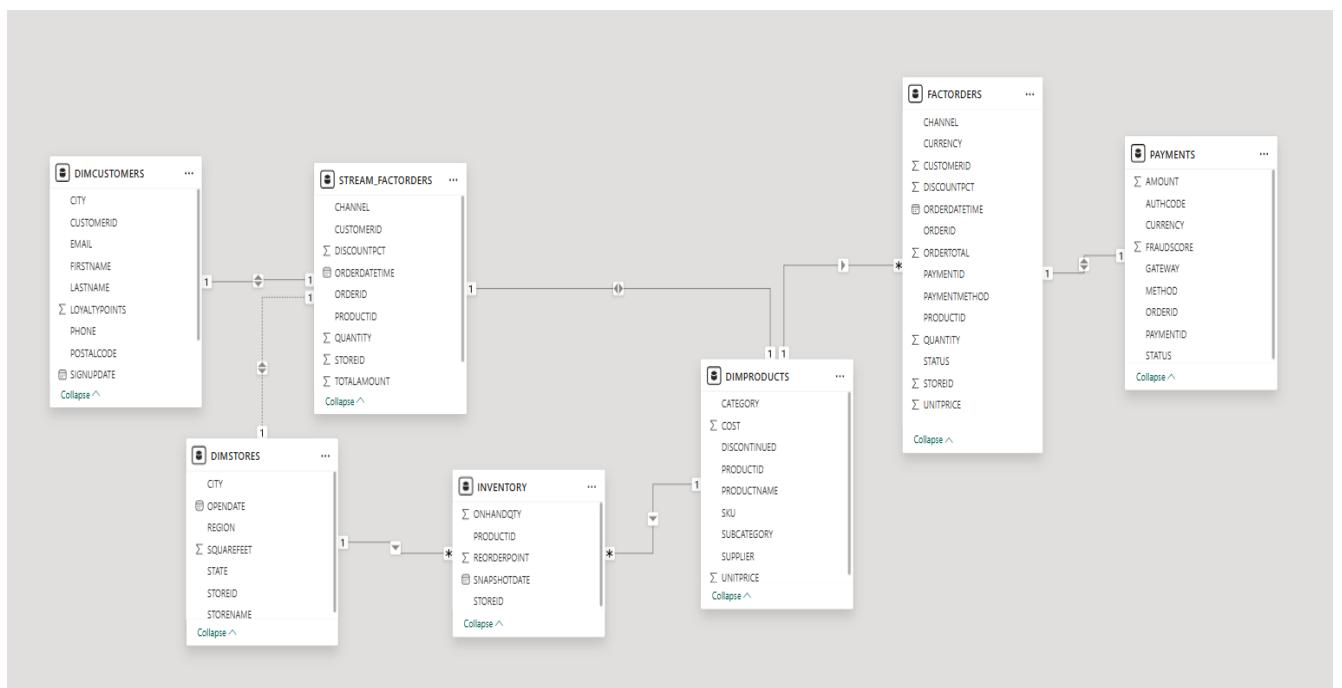
```

 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": "Dck5n73Nw5Dct8",
14          "account": "SVXVKKL-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
24      for row in batch_df.collect():
25          merge_sql = f"""
26              MERGE INTO INVENTORY target
27              USING (SELECT {row.StoreID} AS StoreID, {row.ProductID} AS ProductID, {row.DeltaQty} AS OnHandQty) source
28              ON target.StoreID = source.StoreID AND target.ProductID = source.ProductID
29              WHEN MATCHED THEN UPDATE SET target.OnHandQty = target.OnHandQty + source.OnHandQty
30              WHEN NOT MATCHED THEN INSERT (StoreID, ProductID, OnHandQty)
31              VALUES (source.StoreID, source.ProductID, source.OnHandQty)
32          """
33          cs.execute(merge_sql)
34
35      cs.close()
36      conn.close()
37
38  # Start the streaming job
39  query = inventory_gold.writeStream \
40      .foreachBatch(upsert_inventory_snowflake) \
41      .option("checkpointLocation", "/mnt/checkpoints/inventory_snowflake") \
42      .start()

```

SNOWFLAKE WAREHOUSE DATA TABLE:

✓ E - R DIAGRAM:



✓ Snowflake tables

The screenshot shows the Snowflake interface with the left sidebar expanded. The 'Worksheets' section is selected, showing a list of files including 'Advanced snowfl...', 'case-study.sql', 'Untitled 1.sql' (which is currently selected), 'Untitled.sql', and 'Week2_Day_5.sql'. The main area displays the SQL code for creating two dimension tables:

```
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 );
```

The screenshot shows the Snowflake interface with the left sidebar expanded. The 'Worksheets' section is selected, showing a list of files including 'Advanced snowfl...', 'case-study.sql', 'Untitled 1.sql' (which is currently selected), 'Untitled.sql', and 'Week2_Day_5.sql'. The main area displays the SQL code for creating two fact tables:

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

```

- ✓ 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&srt=sco&sp=rwdlacupyx&se=2025-09-25T15:06:13Z&st=2025-09-
21T06:51:15Z&spr=https&sig=zmaJXehQq%2BfZALZVxUhI4gnhSqy8tSrVMuZlF5IpPw%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

```

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                      )
140                  ELSE '***REDACTED***'
141              END
142          END;
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;

```

Customer Table:

Customer Data											
ID	CUSTOMER_ID	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	Adviya	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	Prawesh	Yadav	p*****@example.com	*****62	646	Bronze	2023-01-25	Revolvur	IN	262801

MONITORING :

- ✓ Create a log analytics workspace

The screenshot shows the Microsoft Azure Log Analytics workspace creation interface. The workspace is named "azure-log-analytics" and is located in the "West US 2" region. It has a Resource group of "move_azure-resource-group" and a Subscription ID of "b41ac736-1671-4724-a6f2-6080db61224b". The workspace is currently active and connected to an Azure Monitor Agent.

- ✓ Open LOGS and then use KQL mode

✓ Query the logs

ALERTS:

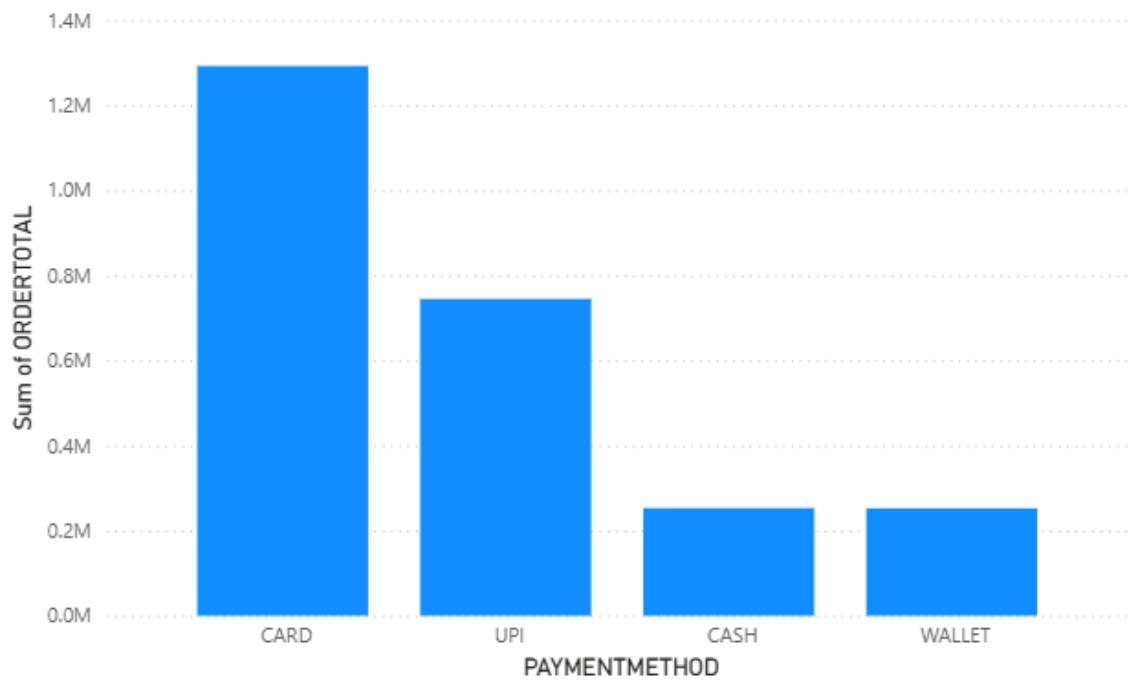
✓ Create alerts for pipeline failures and activity failures

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	

POWER BI:

- ✓ Received more payments through CARD

Sum of ORDERTOTAL by PAYMENTMETHOD



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