

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:

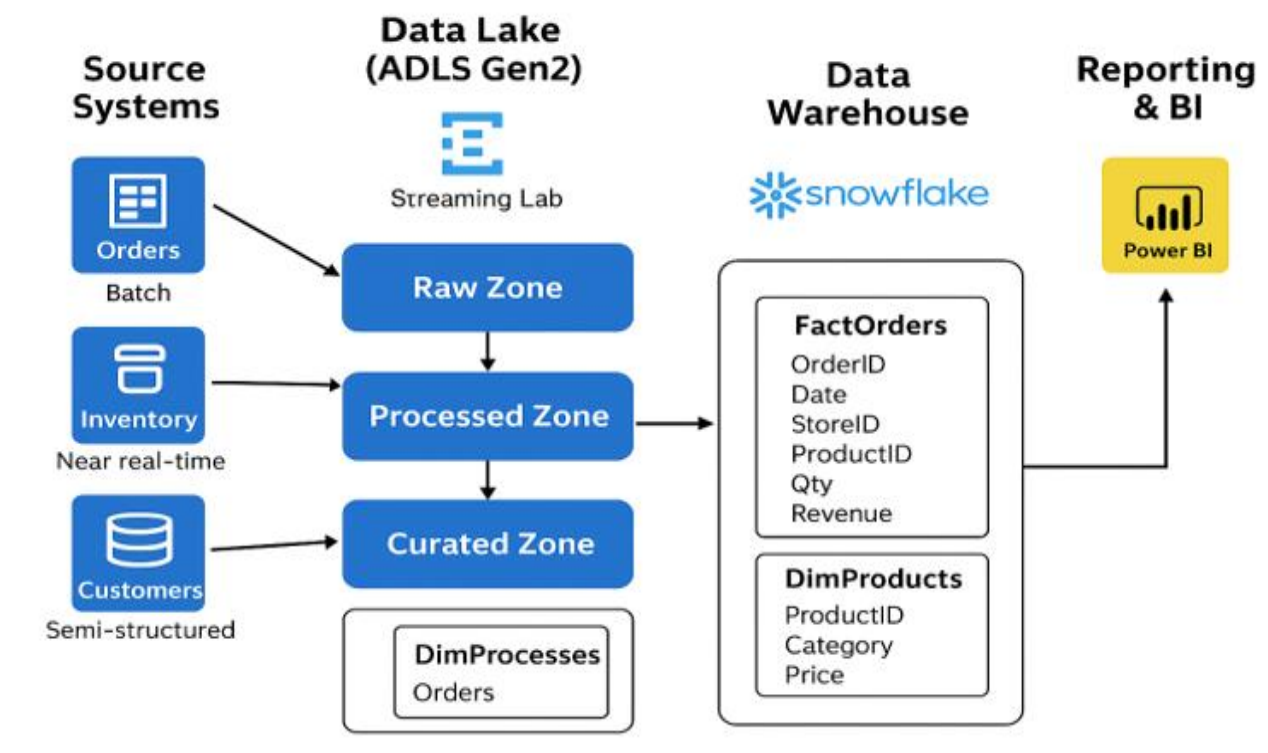
- **Step-1:** Creating a ADLS storage account to store the raw data and also the cleaned and validated data in medallion structure (Bronze, Silver, Gold).
- **Step-2:** For **BATCH-PROCESSING** create 2 ADF(Azure Data Factory) Pipelines one for each of the Batch Orders CSV and Payments CSV files.
- **Step-3:** 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**”.

Home > Storage center

Storage center | Storage accounts (Blobs) Evaluate data resiliency for storage accounts Help me identify cost-saving opportunities in storage accounts View storage accounts with critical alerts

Search x << Summary Resources

Overview All storage resources Object storage File storage Block storage Data management

+ Create Restore Manage view Refresh Export to CSV Open query Assign tags Delete

Filter for any field... Subscription equals all Resource group equals all Location equals all Add filter

Showing 1 to 2 of 2 records. No grouping List view

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.

Home > Storage center | Storage accounts (Blobs) > azurestorageaccount52 | Containers

data Container

Search << + Add Directory Upload Refresh Delete Copy Paste Rename Acquire lease Break lease Edit columns

Overview Diagnose and solve problems Access Control (IAM) Settings

data

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

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

Showing all 8 items

Name	Last modified	Access tier	Blob type	Size	Lease state
._azurtmpfolder\$	9/20/2025, 7:46:22 PM				...
bronze	9/21/2025, 12:39:49 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, 6:34:38 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**

Home > Storage center | Storage accounts (Blobs) > azurestorageaccount52 | Containers

data Container

Search << + Add Directory Upload Refresh Delete Copy Paste Rename Acquire lease Break lease Edit columns

Overview Diagnose and solve problems Access Control (IAM) Settings

data > raw

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

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

Showing all 4 items

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.

data > raworders

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

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

Showing all 1 items

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

Showing all 1 items

<input type="checkbox"/>	Name	Last modified	Access tier	Blob type	Size	Lease state
<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.

The screenshot shows the Microsoft Azure Data Factory interface for a pipeline named 'orderspipeline'. The pipeline is configured with two activities: 'Copy data' (labeled 'orderscopytobronze') and 'Notebook' (labeled 'Ordersdata'). The 'Copy data' activity is connected to the 'Notebook' activity. The 'Notebook' activity is currently selected, and its configuration is shown in the 'General' tab. The configuration includes the following details:

- Name:** Ordersdata
- Description:** (Empty field)
- Activity state:** Activated (selected), Deactivated
- Timeout:** 0.12:00:00
- Retry:** 0

For Batch payment pipeline

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

The screenshot shows the Microsoft Azure Data Factory interface for a pipeline named 'paymentpipeline'. The pipeline is configured with two activities: 'Copy data' (labeled 'paymentdata') and 'Notebook' (labeled 'paymentdatanb'). The 'Copy data' activity is connected to the 'Notebook' activity. The 'Notebook' activity is currently selected, and its configuration is shown in the 'General' tab. The configuration includes the following details:

- Name:** paymentdatanb
- Description:** (Empty field)
- Activity state:** Activated (selected), Deactivated
- Timeout:** 0.12:00:00
- Retry:** 0

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

The screenshot shows the 'Edit trigger' page for 'trigger1' in the Microsoft Azure Data Factory portal. The left sidebar contains navigation options like General, Connections, and Triggers. The main area displays the trigger configuration. The 'Name' field is 'trigger1', and the 'Type' is 'BlobEventsTrigger'. The 'Account selection method' is 'From Azure subscription'. The 'Azure subscription' is 'Azure subscription 1 (b41ac736-1671-4724-a6f2-6080db61224b)'. The 'Storage account name' is 'azurestorageaccount52', and the 'Container name' is 'data'. The 'Blob path begins with' is 'raworders'.

Name	Type
trigger1	Storage events
trigger2	Storage events

The screenshot shows the 'Edit trigger' page for 'trigger2' in the Microsoft Azure Data Factory portal. The configuration is similar to 'trigger1', but the 'Blob path begins with' is 'rawpayment'.

Name	Type
trigger1	Storage events
trigger2	Storage events

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

The screenshot shows the 'Alerts & metrics' page in the Microsoft Azure Data Factory portal. It displays a table of alerts with columns for Alert, Enabled, Resource Type, Resources, and Actions.

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

CREATE ACTION GROUP:

- ✓ Created action group for notification alerts through email.

Microsoft Azure | Data Factory | azuredata-factory123

Alerts & metrics

Refresh Metrics New alert rule

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

Learn more about Pricing and Privacy statement.

Action name *

email notification

Select which notifications you'd like to receive

☒ Email

21cs01053@iitbbs.ac.in

☐ SMS

Country code Phone number *

1 1234567890

Carrier charges may apply.

☐ 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.

email@example.com

☐ Voice

Country code Phone number *

1 1234567890

Add notification Cancel

Microsoft Azure | Data Factory | azuredata-factory123

Alerts & metrics

Refresh Metrics New alert rule

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

Notify your team via email and text messages or automate actions using webhooks, runbooks, functions logic apps or integrating with external ITSM solutions.

☒ Create new ☐ Use existing

Action group name *

Azureactiongroup

Short name *

azureproject

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

+ Add notification

Add action group Cancel

Orders Batch processing Notebook:

BatchOrderProcessing Batchpaymentprocessing Stream processing

File Edit View Run Help Python Tabs: ON Last edit was now

Run all Terminated Schedule Share

12 hours ago (4h)

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

```
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)
17
```

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)
10
```

Gold to Snowflake table

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

```
1 from snowflake.snowpark import Session
```

```
1 # Snowflake options
2 batch_df.write \
3     .format("snowflake") \
4     .options(
5         sfURL="SV0XXKL-FR13613.snowflakecomputing.com",
6         sfUser="BIBURAMESWAR",
7         sfPassword="Dck5n73Nw5Oct8",
8         sfDatabase="TRAINING",
9         sfSchema="PUBLIC",
10        sfWarehouse="SIGMOID",
11        sfRole="ACCOUNTADMIN"
12    ) \
13    .option("dbtable", "FACTORDERS") \
14    .mode("append") \
15    .save()
16
```

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

OUTPUT TABLE (FACT ORDER):

Table	Chart														30,000 rows	117ms	
#	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 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

Last execution failed

6

Python

```
1 gold_path = "/mnt/gold/batch/payments/"
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

Last execution failed

10

Code

Text

Assistant

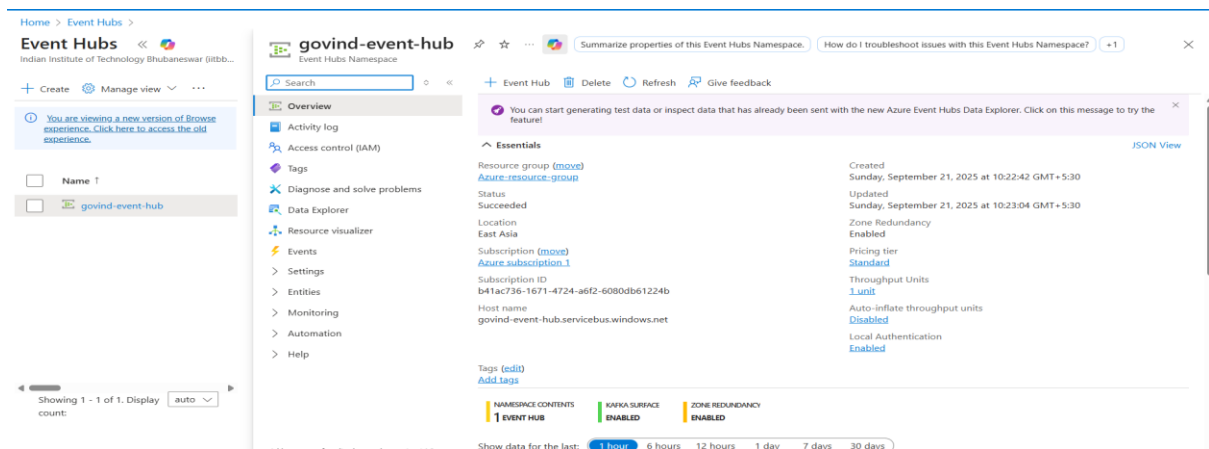
```
1 # Snowflake options
2 batch_df.write \
3     .format("snowflake") \
4     .options(
5         sfURL="SVXXKL-FR13613.snowflakecomputing.com",
6         sfUser="BHUBAHESWAR",
7         sfPassword="Dck5n73Nw5Oct8",
8         sfDatabase="TRAINING",
9         sfSchema="PUBLIC",
10        sfWarehouse="SIGMOID",
11        sfRole="ACCOUNTADMIN"
12    ) \
13    .option("dbtable", "PAYMENTS") \
14    .mode("append") \
15    .save()
16
```

OUTPUT TABLE (PAYMENTS):

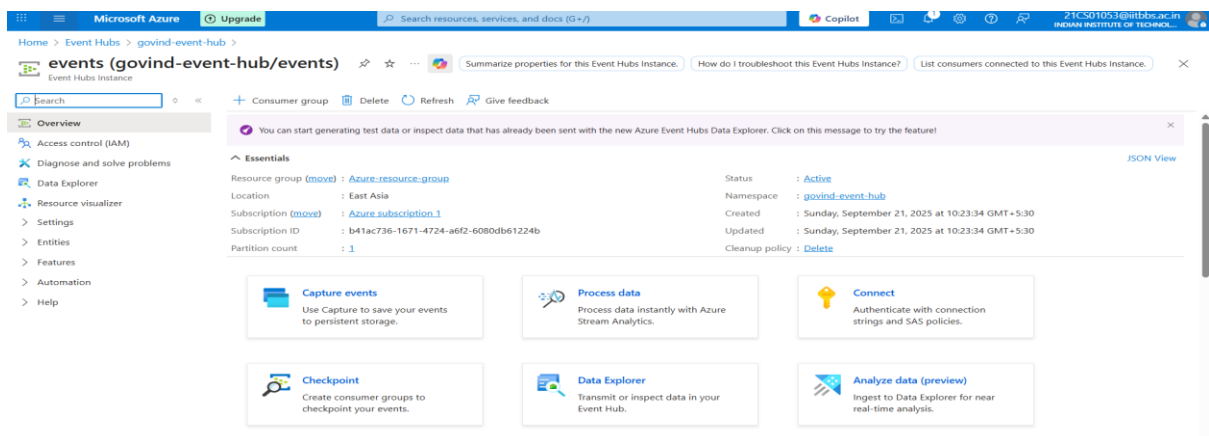
Table		Chart									30,000 rows		89ms			
	Δ	PAYMENTID	ORDERID	AMOUNT	CURRENCY	METHOD	GATEWAY	STATUS	AUTHCODE	FRAUDSCORE						
1		PMT00000122	122	76.65	INR	UPI	Razorpay	Pending	F8BV5ZHP	0.041						
2		PMT00000638	638	344.82	INR	UPI	PayU	Captured	MRVNZQER	0.032						
3		PMT00000960	960	89.16	INR	UPI	Stripe	Captured	MPEOSO1F	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	J9PJPICR	0.033						
9		PMT00002695	2695	25.95	INR	CARD	Cashdesk	Captured	ZEX85CDI	0.152						
10		PMT00002949	2949	41.89	INR	UPI	PayU	Captured	YVDP76PK	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	3WSQUJQ9	0.180						
16		PMT00004928	4928	823.41	INR	WALLET	Cashdesk	Captured	FF2XZDT9	0.097						

STREAM- PROCESSING:

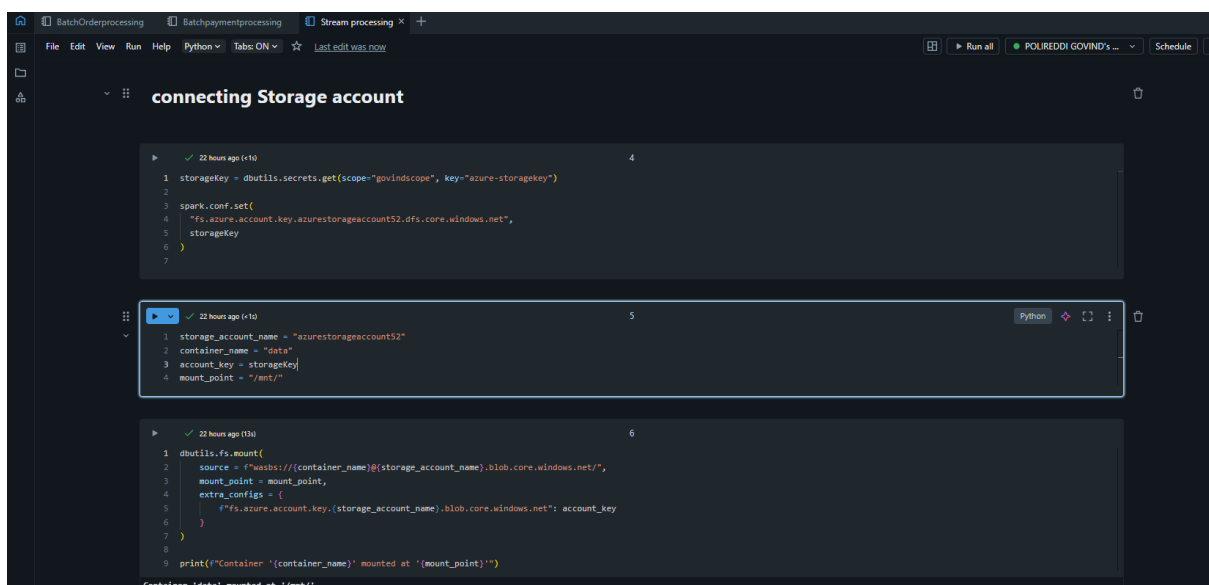
- ✓ Create an Event hub to collect events and send to the databricks notebook



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



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



- ✓ Connect to the event hub

connecting to Azure 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
11 df_kafka = spark.readStream.format("kafka").options(**event_hub_config).load()
```

- ✓ Schema define for the data

```
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

```
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") \
10     .select(
11         col("data.event_time").alias("event_time"),
12         col("data.payload.OrderID").alias("OrderID"),
13         col("data.payload.StoreID").alias("StoreID"),
14         col("data.payload.CustomerID").alias("CustomerID"),
15         col("data.payload.ProductID").alias("ProductID"),
16         col("data.payload.Quantity").alias("Quantity"),
17         col("data.payload.UnitPrice").alias("UnitPrice"),
18         col("data.payload.DiscountPct").alias("DiscountPct"),
19         col("data.payload.Channel").alias("Channel"),
20         to_date(col("data.event_time")).alias("event_date")
21     )
22
23 # Inventory stream
24 inventory_df = df_kafka_str \
25     .select(from_json(col("json_str"), inventory_schema).alias("data")) \
26     .filter(col("data.event_type") == "Inventory_update") \
27     .select(
28         col("data.event_time").alias("event_time"),
29         col("data.payload.StoreID").alias("StoreID"),
30         col("data.payload.ProductID").alias("ProductID"),
31         col("data.payload.DeltaQty").alias("DeltaQty"),
32         to_date(col("data.event_time")).alias("event_date")
33     )
```

```
35 # For orders CSV
36 orders_df.writeStream.format("csv").option("path", "/mnt/bronze/stream/orders") \
37   .option("checkpointLocation", "/mnt/checkpoints/orders_csv").partitionBy("event_date").outputMode("append").start()
38
39 # For Inventory CSV
40 inventory_df.writeStream.format("csv").option("path", "/mnt/bronze/stream/inventory_updates") \
41   .option("checkpointLocation", "/mnt/checkpoints/inventory_csv").partitionBy("event_date").outputMode("append").start()
42
43 # Orders Delta
44 orders_df.writeStream.format("delta").option("path", "/mnt/bronze/stream/orders_delta") \
45   .option("checkpointLocation", "/mnt/checkpoints/orders_delta").partitionBy("event_date").outputMode("append").start()
46
47 # Inventory Delta
48 inventory_df.writeStream.format("delta").option("path", "/mnt/bronze/stream/inventory_delta") \
49   .option("checkpointLocation", "/mnt/checkpoints/inventory_delta").partitionBy("event_date").outputMode("append").start()
50
51 # (5) Spark Jobs
52
53 > @ c69d081e-f2a-4288-b27c-1681b65c44d0 Last updated: 18 hours ago
54
55 > @ 662a2797-909e-44d7-adca-8a49c496b0e7 Last updated: 18 hours ago
56
57 > @ 8fa6cd3-e5c5-4ba8-9d99-de29a2d34233 Last updated: 18 hours ago
58
59 > @ e02ca06-f280-49b0-bba0-0c067978b16 Last updated: 18 hours ago
60
61 > @ display_query_6 (id:75cbf513-6b65-4a79-add1-b9c8e7359400) Last updated: 18 hours ago
62
63 Table +
64
65 join_str
66
67 1 > ["event_time": "2025-06-15T00:05:30", "event_type": "order_created", "payload": {"OrderID": "5175521633011", "StoreID": 3, "CustomerID": 2054, "ProductID": 199, "Quantity": 1, "Unit...
```

✓ Broze to silver after cleaning and validating the data

```
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(18, 2))
12   )
13
14 # Function to write unique records to Silver using MERGE
15 def write_unique_to_silver(batch_df, batch_id):
16     # Deduplicate batch 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 foreachBatch
30 query = orders_clean.writeStream \
31   .foreachBatch(write_unique_to_silver) \
32   .option("checkpointLocation", "/mnt/checkpoints/orders_silver") \
33   .start()
34
35 # (5) Spark Jobs
36
37 > @ 58a69341-acd4-4afa-b447-b99014464168 Last updated: 18 hours ago
38
39 > @ display_query_6 (id:75cbf513-6b65-4a79-add1-b9c8e7359400) Last updated: 18 hours ago
40
41 Table +
42
43 join_str
44
45 1 > ["event_time": "2025-06-15T00:05:30", "event_type": "order_created", "payload": {"OrderID": "5175521633011", "StoreID": 3, "CustomerID": 2054, "ProductID": 199, "Quantity": 1, "Unit...
```

```
1 bronze_inventory = spark.readStream.format("delta").load("/mnt/bronze/stream/inventory_delta")
2
3 # Clean and transform
4 inventory_clean = bronze_inventory \
5   .withColumn("DeltaQty", col("DeltaQty").cast("int")) \
6   .filter(col("StoreID").isNotNull() & col("ProductID").isNotNull())
7
8 # Function to write unique records to Silver using MERGE
9 def write_unique_to_silver(batch_df, batch_id):
10     # Drop exact duplicates within the batch
11     batch_df = batch_df.dropDuplicates()
12
13     if DeltaTable.isDeltaTable(spark, "/mnt/silver/stream/inventory"):
14         silver_table = DeltaTable.forPath(spark, "/mnt/silver/stream/inventory")
15         # Merge: Insert only new rows that don't exist yet (exact match on all columns)
16         join_condition = " AND ".join([f"silver.{c} = batch.{c}" for c in batch_df.columns])
17         silver_table.alias("silver").merge(
18             batch_df.alias("batch"),
19             join_condition
20         ).whenNotMatchedInsertAll().execute()
21     else:
22         # First batch: create Silver Delta table
23         batch_df.write.format("delta").mode("overwrite").save("/mnt/silver/stream/inventory")
24
25 # Start streaming with foreachBatch
26 query = inventory_clean.writeStream \
27   .foreachBatch(write_unique_to_silver) \
28   .option("checkpointLocation", "/mnt/checkpoints/inventory_silver") \
29   .start()
30
31 # (5) Spark Jobs
32
33 > @ 58a69341-acd4-4afa-b447-b99014464168 Last updated: 18 hours ago
34
35 > @ display_query_6 (id:75cbf513-6b65-4a79-add1-b9c8e7359400) Last updated: 18 hours ago
36
37 Table +
38
39 join_str
40
41 1 > ["event_time": "2025-06-15T00:05:30", "event_type": "order_created", "payload": {"OrderID": "5175521633011", "StoreID": 3, "CustomerID": 2054, "ProductID": 199, "Quantity": 1, "Unit...
```

The screenshot displays two Databricks notebooks. The first notebook, titled "SILVER TO GOLD", contains a PySpark script that reads a stream of orders from a Delta table, transforms it into a FactOrders format, and writes it to a Gold Delta table. The second notebook, titled "Gold to snowflake table", contains a PySpark script that reads the Gold Delta table and writes it to a Snowflake table. Both notebooks show the execution progress and the resulting data frames.

```
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-eb5499d27b22 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 0x7f4a616ff8c0 >
```

```
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-abe9646b9ec21 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 0x7f4a53627bf0 >
```

✓ Gold to snowflake table

The screenshot displays a Databricks notebook titled "gold to snowflake". It contains a PySpark script that reads a stream of orders from a Delta table and writes it to a Snowflake table. The script uses the Snowflake connector to write the data to a Snowflake table.

```
1 from snowflake.snowpark import Session

> 18 hours ago (x1) 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
```

The screenshot displays a Databricks notebook titled "gold to snowflake". It contains a PySpark script that reads a stream of orders from a Delta table and writes it to a Snowflake table. The script uses the Snowflake connector to write the data to a Snowflake table.

```
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="SCVXXDL-FR13613.snowflakecomputing.com",
27         sfUser="B4MBANESWAB",
28         sfPassword="DckSn733Mw5Oct8",
29         sfDatabase="TRAINING",
30         sfSchema="PUBLIC",
31         sfWarehouse="SIOPIDP",
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
```

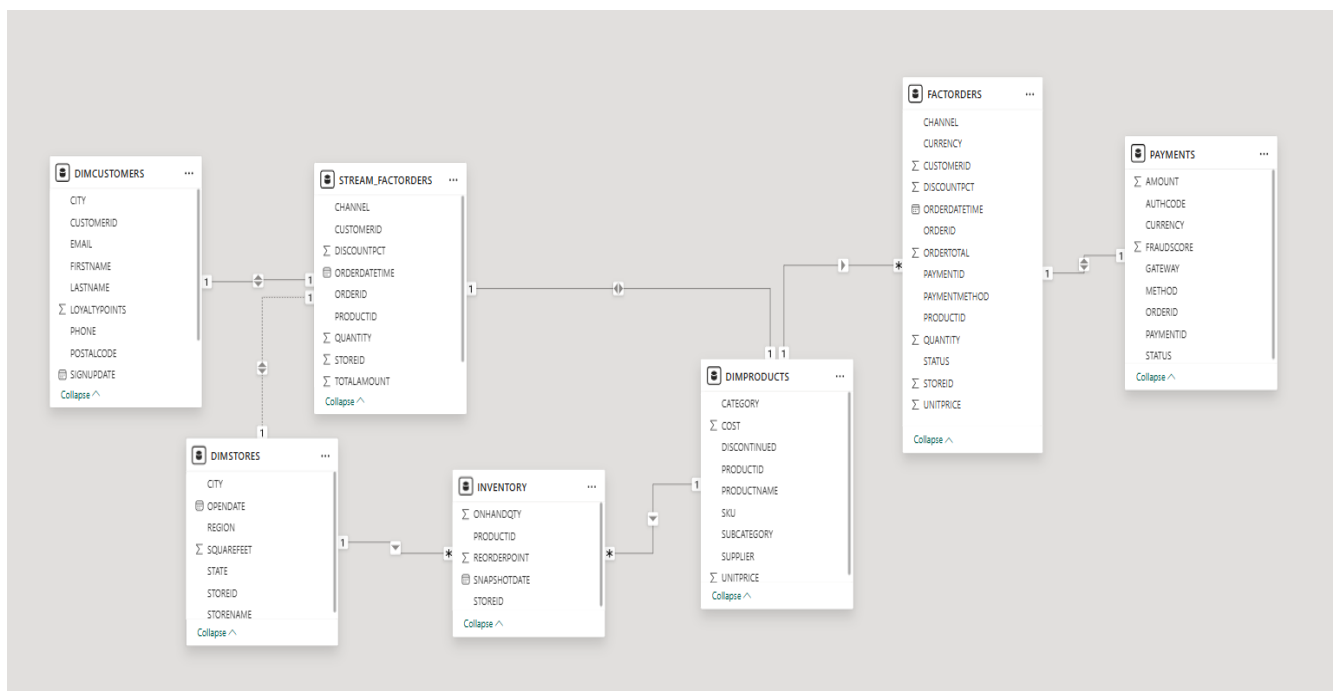
✓ INVENTORY UPDATE TO THE SNOWFLAKE INVENTORY TABLE

```
26 Python

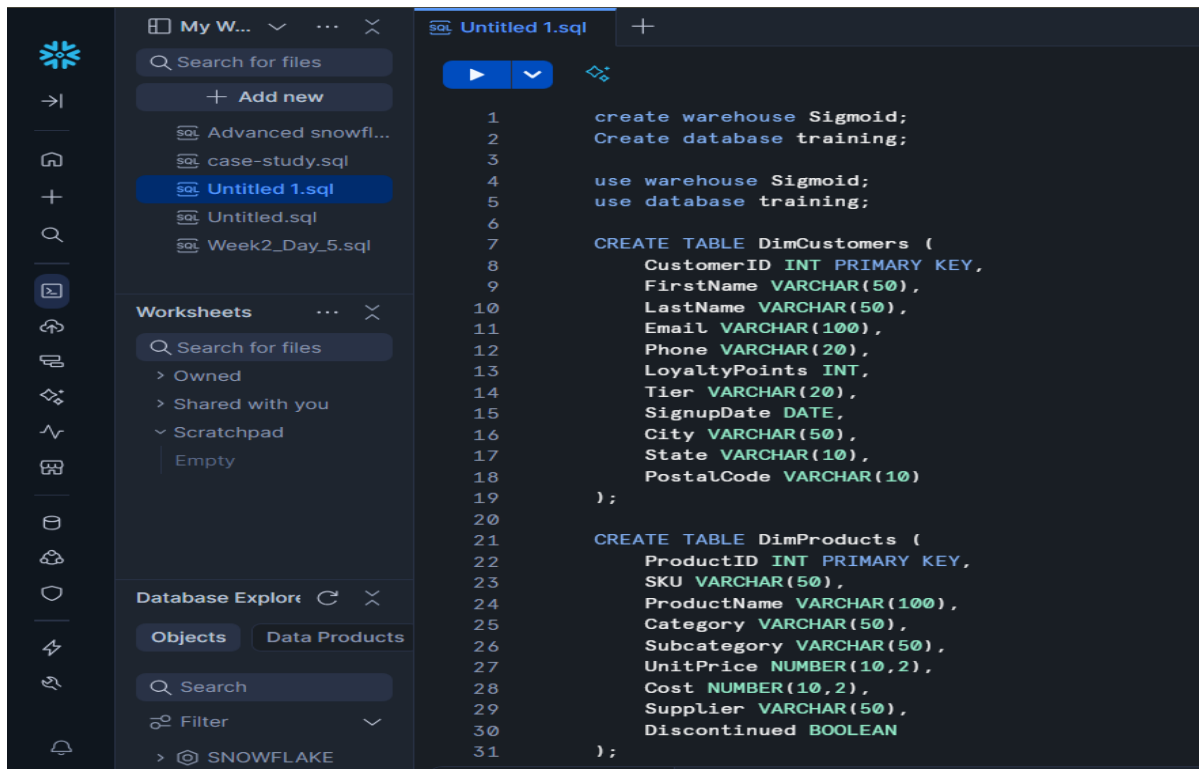
1 from pyspark.sql.functions import col
2 import snowflake.connector
3
4 inventory_gold = spark.readStream.format("delta").load("/mnt/gold/stream/inventory")
5
6 # Upsert function
7 def upsert_inventory_snowflake(batch_df, batch_id):
8     if batch_df.count() == 0:
9         return
10    batch_df = batch_df.dropDuplicates()
11    conn_params = {
12        "user": "BHUBANESWAR",
13        "password": "Dck5n7J3Nw5Dct8",
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
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    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:

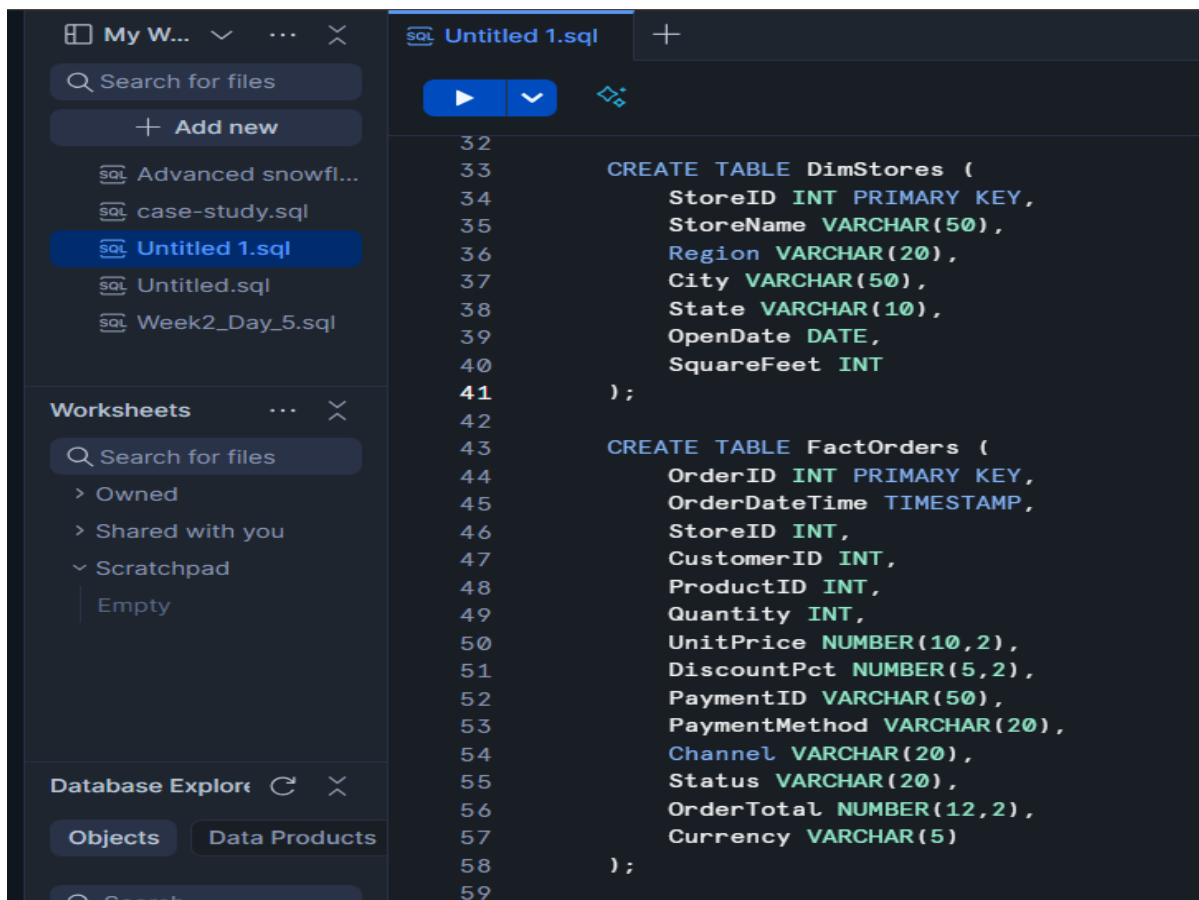
✓ E - R DIAGRAM:



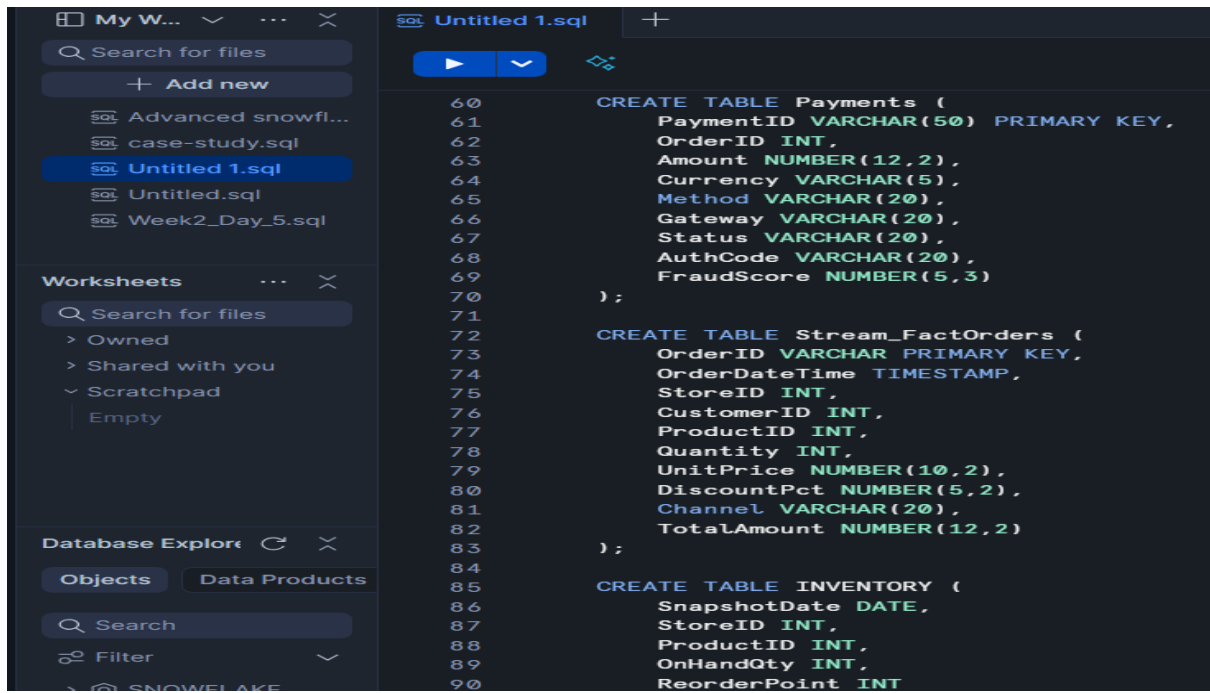
✓ Snowflake tables



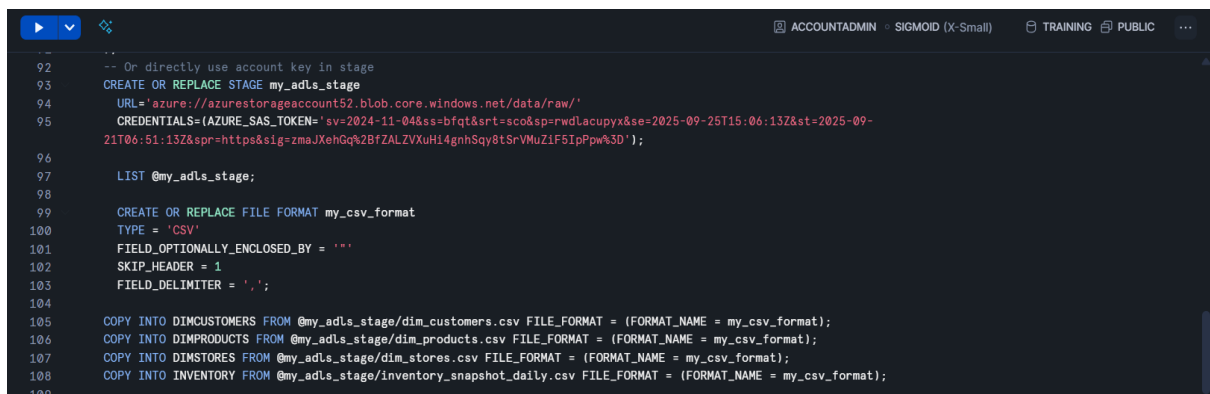
```
1 create warehouse Sigmoid;
2 Create database training;
3
4 use warehouse Sigmoid;
5 use database training;
6
7 CREATE TABLE DimCustomers (
8     CustomerID INT PRIMARY KEY,
9     FirstName VARCHAR(50),
10    LastName VARCHAR(50),
11    Email VARCHAR(100),
12    Phone VARCHAR(20),
13    LoyaltyPoints 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
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
```

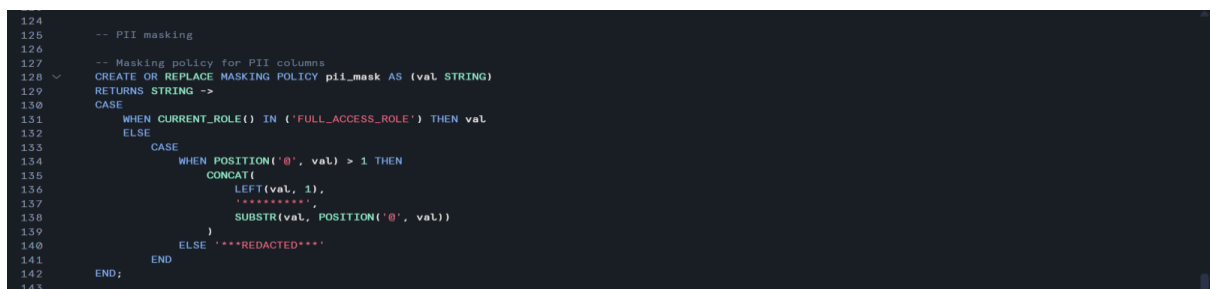


- ✓ Loading data from ADLS to a stage in Snowflake and copy into the tables



PII MASKING:

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



```

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:

Just now

Table

Chart

6,000 rows

970ms

	#	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	Advika	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	Devansh	Yadav	d*****@example.com	*****62	646	Bronze	2022-01-25	Bengaluru	IN	262801	

MONITORING :

- ✓ Create a log analytics workspace

The screenshot shows the Microsoft Azure portal interface for a Log Analytics workspace. The workspace is named 'azure-log-analytics' and is currently 'Active'. Key details include:

- Resource group:** azure-resource-group
- Status:** Active
- Location:** West US 2
- Subscription:** Azure subscription 1
- Subscription ID:** b41ac736-1671-4724-a6f2-6080db61224b
- Tags:** (None listed)
- Workspace Name:** azure-log-analytics
- Workspace ID:** 076e02fa-48b2-4b36-9939-3cf2e2faeb41
- Pricing tier:** Pay-as-you-go
- Access control mode:** Use resource or workspace permissions
- Operational issues:** None (Status: OK)

The 'Get started with Log Analytics' section at the bottom explains that Log Analytics collects data from various sources and uses a powerful query language to provide insights into application and resource operations.

- ✓ Open LOGS and then use KQL mode

Microsoft Azure | Upgrade | Search resources, services, and docs (G+J) | Copilot | 21CS01053@iitbbs.ac.in | INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

Home > Log Analytics workspaces > azure-log-analytics

Log Analytics workspace

Log Analytics Institute of Technology Bhubaneswar (iitbbs...)

+ Create Open recycle bin ...

You are viewing a new version of Browse experience. Click here to access the old experience.

☐ Name 1

☐ azure-log-analytics

Showing 1 - 1 of 1. Display count: auto

azure-log-analytics | Logs

Log Analytics workspace

Search

New Query 1

Run Time range: Last 24 hours Show: 1000 results KQL mode

1 [Type your query here or click one of the queries to start]

Query history

AzureDiagnostics take 5	0 results - 9/22/2025, 1:04 AM	Run
AzureDiagnostics take 5	0 results - 9/22/2025, 12:51 AM	Run
AzureDiagnostics where ResourceType == "DATAPROCS" project TimeGenerated, Resource, OperationName, RunOn_s order by TimeGenerated desc	0 results - 9/22/2025, 12:51 AM	Run

✓ Query the logs

New Query 1* Save Share ... Queries hub

Run Time range: Last 24 hours Show: 1000 results KQL mode

```
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

Microsoft Azure | Data Factory > azuredata-factory123 | Search | 21CS01053@iitbbs.ac.in | INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

Alerts & metrics

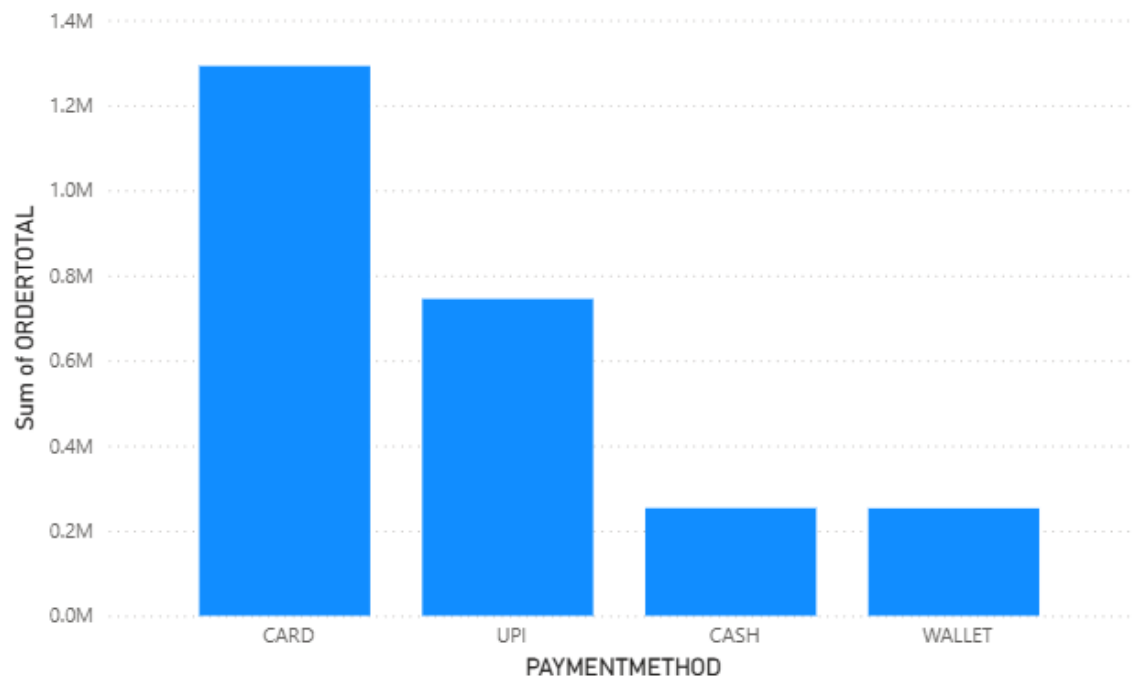
Refresh Metrics New alert rule

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	