

(https://databricks.com)

```
### Storage-cleaning ###  
# List all files within the "/FileStore" directory  
files = dbutils.fs.ls("/FileStore/")  
files = [file.path for file in files if not file.isDir()]  
# Delete each file  
for file in files:  
    dbutils.fs.rm(file)  
  
# List all directories within the "/FileStore" directory  
directories = dbutils.fs.ls("/FileStore/")  
directories = [directory.path for directory in directories if directory.isDir()]  
# Delete each directory  
for directory in directories:  
    dbutils.fs.rm(directory, recurse=True)  
  
dbutils.fs.rm('dbfs:/FileStore/streaming/AppendCheckpoint', True)  
dbutils.fs.rm('dbfs:/user/hive/warehouse/stream.db', True)  
dbutils.fs.rm('dbfs:/FileStore/streaming', True)
```

```
%sql  
---Storage-cleaning---  
DROP SCHEMA Stream CASCADE
```

OK

Data Ingestion and Setup: Set up a data stream using a simulated real-time data source. Configure Apache Spark notebook to connect and ingest streaming data.

```
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType
schema = StructType([
    StructField('InvoiceNo', IntegerType()),
    StructField('StockCode', StringType()),
    StructField('Description', StringType()),
    StructField('Quantity', IntegerType()),
    StructField('InvoiceDate', StringType()),
    StructField('UnitPrice', FloatType()),
    StructField('CustomerID', IntegerType()),
    StructField('Country', StringType())
])
```

```
source_dir = 'dbfs:/FileStore/Streaming/'
```

```
# df = spark.read.format('csv')\
#     .option('header', 'true')\
#     .schema(schema)\
#     .load(source_dir)

df = spark.readStream.format('csv')\
    .option('header', 'true')\
    .schema(schema)\
    .load(source_dir)
```

```
df.printSchema()
```

```
root
|-- InvoiceNo: integer (nullable = true)
|-- StockCode: string (nullable = true)
|-- Description: string (nullable = true)
|-- Quantity: integer (nullable = true)
|-- InvoiceDate: string (nullable = true)
|-- UnitPrice: float (nullable = true)
|-- CustomerID: integer (nullable = true)
|-- Country: string (nullable = true)
```

```
display(df)
```

Input vs. Processing Rate

0 rec/s0 rec/s

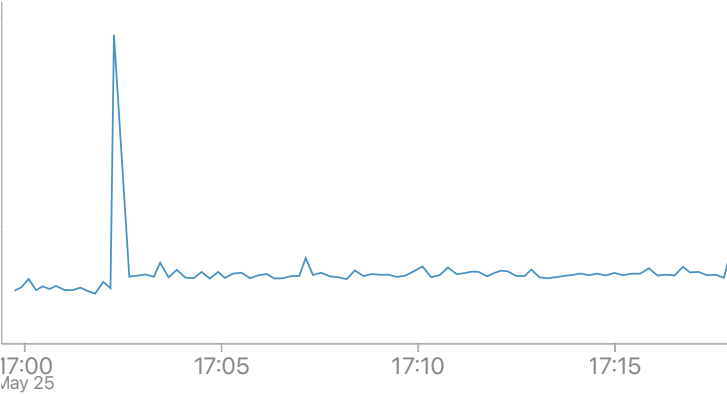
records per secondInput rateProcessing rate



Batch Duration

2.7 s2.5 s

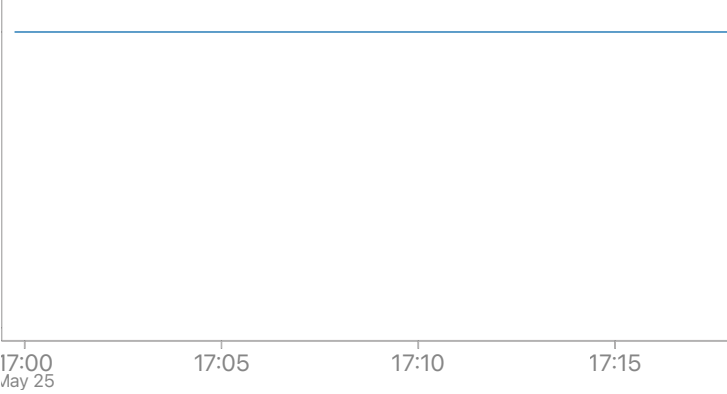
in secondsAverageLatest



Aggregation State

1

Distinct keys



Table

New result table: ON

	¹ ₃ InvoiceNo	^A _C StockCode	^A _C Description	¹ ₃ Quanti
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	
2	536365	71053	WHITE METAL LANTERN	
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	
8	536366	22633	HAND WARMER UNION JACK	
9	536366	22632	HAND WARMER RED POLKA DOT	
10	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	
11	536367	22745	POPPY'S PLAYHOUSE BEDROOM	
12	536367	22748	POPPY'S PLAYHOUSE KITCHEN	
13	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	
14	536367	22310	IVORY KNITTED MUG COSY	
15	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	

633 rows

```
%sql
CREATE DATABASE IF NOT EXISTS Stream
```


OK

```
%sql
USE Stream
```

OK

Configure output sinks to write results to external databases or files for further analysis (Appended Output). Checkpointing. Note Trigger.

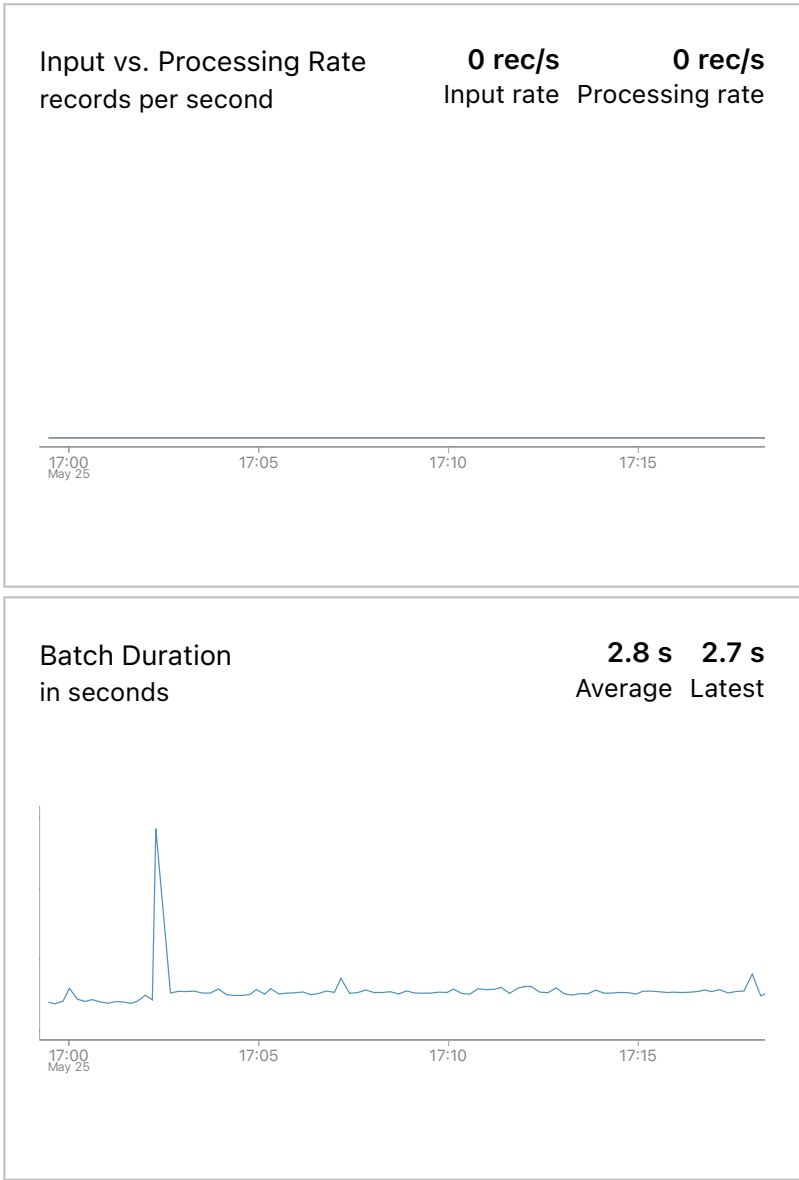
```
WriteStream = (df.writeStream
    .option('checkpointLocation',f'{source_dir}/AppendCheckpoint')
    .outputMode('append')
    .trigger(availableNow=True)
    .queryName('AppendQuery')
    .toTable('stream.AppendTable'))
```


AppendQuery (id: 846936df-eda9-40fc-8a7a-d76f5f195530)

Last updated: 19 hours ago

Dashboard

Raw Data






```
spark.sql('describe history AppendTable')
```

```
DataFrame[version: bigint, timestamp: timestamp, userId: string, userName: string, operation: string, operationParameters: map<string,string>, job: struct<jobId:string, jobName:string,jobRunId:string,runId:string,jobOwnerId:string,triggerType:string>, notebook: struct<notebookId:string>, clusterId: string, readVersion: bigint, isolationLevel: string]
```

nLevel: string, isBlindAppend: boolean, operationMetrics: map<string,string>, userMetadata: string, engineInfo: string]

```
%sql
SELECT * FROM stream.AppendTable
```

Table

New result table: ON   

	¹ ₃ InvoiceNo	^A _C StockCode	^A _C Description	¹ ₃ Quanti
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	
2	536365	71053	WHITE METAL LANTERN	
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	
8	536366	22633	HAND WARMER UNION JACK	
9	536366	22632	HAND WARMER RED POLKA DOT	
10	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	
11	536367	22745	POPPY'S PLAYHOUSE BEDROOM	
12	536367	22748	POPPY'S PLAYHOUSE KITCHEN	
13	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	
14	536367	22310	IVORY KNITTED MUG COSY	
15	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	

254 rows

```
%sql
SELECT * FROM AppendTable
--SELECT * FROM stream.AppendTable
```

Table

New result table: ON 🔍 ⚙️ 📄

	1 ² ₃ InvoiceNo	A ^B _C StockCode	A ^B _C Description	1 ² ₃ Quanti
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	
2	536365	71053	WHITE METAL LANTERN	
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	
8	536366	22633	HAND WARMER UNION JACK	
9	536366	22632	HAND WARMER RED POLKA DOT	
10	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	
11	536367	22745	POPPY'S PLAYHOUSE BEDROOM	
12	536367	22748	POPPY'S PLAYHOUSE KITCHEN	
13	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	
14	536367	22310	IVORY KNITTED MUG COSY	
15	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	

254 rows

display(df)

▼ 🔄 display_query_2 (id: 9a6dc9a2-51b2-4398-b148-92c624a16b59) Last updated: 19 hours ago

Dashboard

Raw Data

Input vs. Processing Rate

records per second

0 rec/s

Input rate

0 rec/s

Processing rate

7:00

17:05

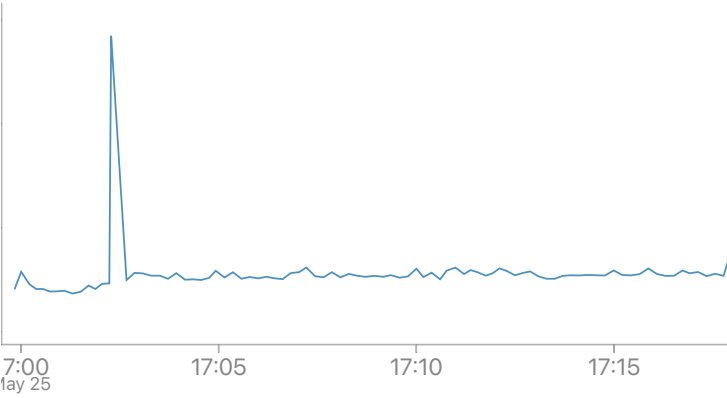
17:10

17:15

1ay 25

Batch Duration
in seconds

2.8 s 2.8 s
Average Latest



Aggregation State

1
Distinct keys

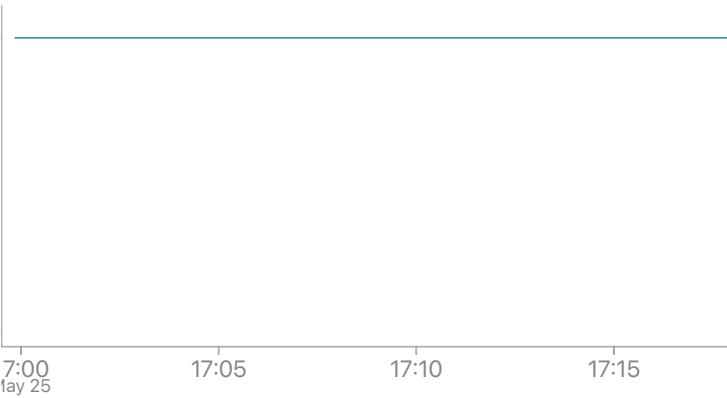



Table					New result table: ON	Q	Y	□
	i_3^2 InvoiceNo	A_C^B StockCode	A_C^B Description	i_3^2 Quanti				
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER					
2	536365	71053	WHITE METAL LANTERN					
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER					
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE					
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.					
6	536365	22752	SET 7 BABUSHKA NESTING BOXES					
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER					
8	536366	22633	HAND WARMER UNION JACK					
9	536366	22632	HAND WARMER RED POLKA DOT					
10	536367	84879	ASSORTED COLOUR BIRD ORNAMENT					
11	536367	22745	POPPY'S PLAYHOUSE BEDROOM					
12	536367	22748	POPPY'S PLAYHOUSE KITCHEN					
13	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL					
14	536367	22310	IVORY KNITTED MUG COSY					
15	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS					
633 rows								

```
%sql
describe history AppendTable
```

Table					New result table: ON	Q	Y	□
	i_3^2 version	 timestamp	A_C^B userId	A_C^B userName				
1	2	2024-05-25T17:57:50.000+00:00	7657907205508238	g39351894@gwu.ed				
2	1	2024-05-25T17:05:46.000+00:00	7657907205508238	g39351894@gwu.ed				
3	0	2024-05-25T17:05:39.000+00:00	7657907205508238	g39351894@gwu.ed				
3 rows								

Data Processing with Spark Streaming: Implement data transformation/aggregation to process the data incrementally. State

Management.

```
from pyspark.sql.functions import sum
df_complete = df.groupBy('Country').agg(sum('Quantity').alias('Total_Units'))
```

```
%sql
CREATE DATABASE if NOT EXISTS StreamDatabase
```

OK

```
%sql
USE StreamDatabase
```

OK


```
spark.sql("""
CREATE TABLE IF NOT EXISTS StreamDatabase.InvoiceTable (
    Country STRING,
    Total_Sold LONG
)
USING delta
PARTITIONED BY (Country)
""")
```

DataFrame[]

Optimization and Performance Tuning after the full data ingestion:
Optimize query performance using partitioning.

```
df_complete = (df
    .groupBy('Country')
    .agg(sum('Quantity').alias('Total_Sold')))

query = (df_complete
    .writeStream
    .format("delta")
    .outputMode("complete")
    .option("path", "/path/to/delta/table")
    .option("checkpointLocation", "/path/to/checkpoint")
    .partitionBy("Country")
    .table("StreamDatabase.InvoiceTable"))
```

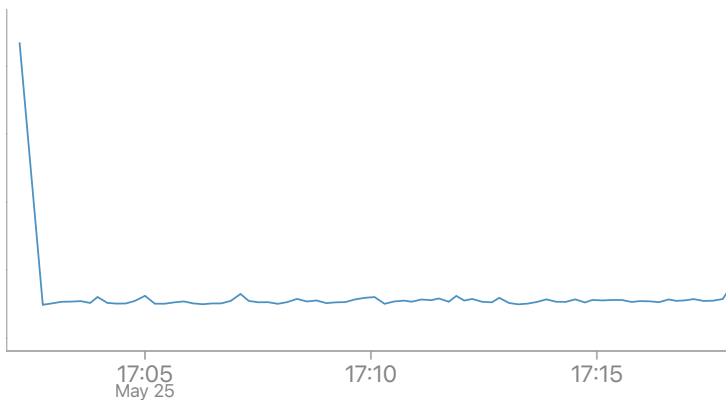
▼  54342948-373e-467b-8578-b4881833ad4e Last updated: 19 hours ago

Dashboard **Raw Data**

Input vs. Processing Rate **0 rec/s** **0 rec/s**
 records per second Input rate Processing rate



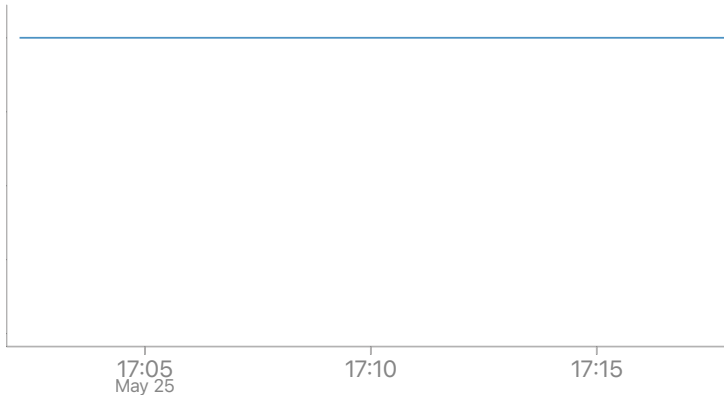
Batch Duration **2.9 s** **2.6 s**
 in seconds Average Latest



Aggregation State


4

Distinct keys



Configure output sinks to write results to external databases or files for further analysis (Completed Output). Checkpointing. Note trigger.

```
WriteStream = (df_complete.writeStream
    .option('checkpointLocation', f'{source_dir}/CompleteCheckpoint')
    .outputMode('complete')
    .trigger(processingTime='2 minutes')
    .queryName('CompleteQuery')
    .toTable('stream.CompleteTable'))
```

▼  CompleteQuery (id: 2e80bb6f-ac04-4953-8390-a87d7af69442) Last updated: 19 hours ago

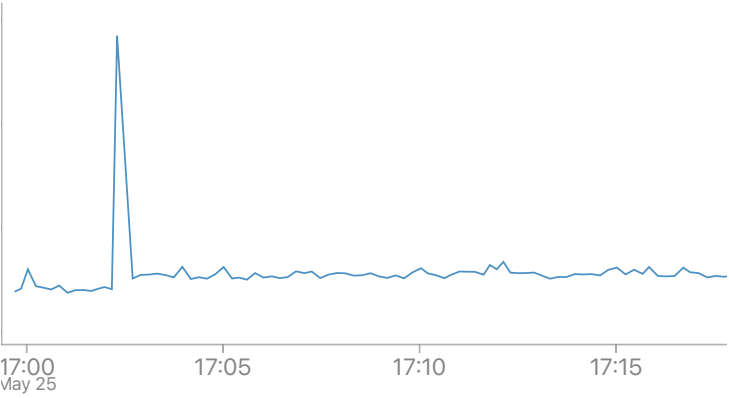
Dashboard **Raw Data**

Input vs. Processing Rate **0 rec/s** **0 rec/s**
records per second Input rate Processing rate



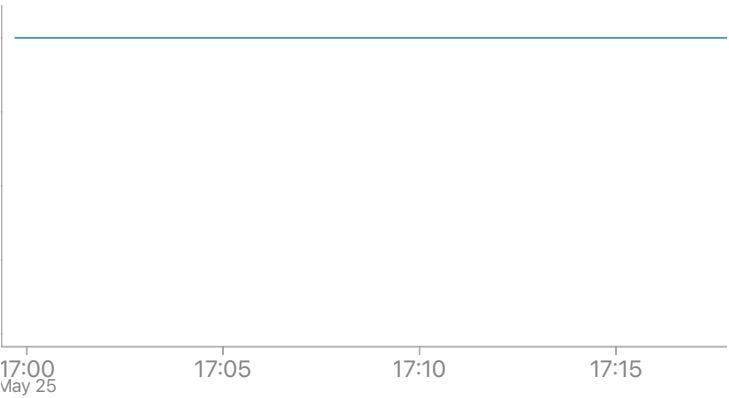
Batch Duration
in seconds

2.8 s 2.5 s
Average Latest



Aggregation State

4
Distinct keys



```
%sql
SELECT * FROM CompleteTable
--SELECT * FROM stream.CompleteTable
```

Table

New result table: ON 🔍 ⚙️ 📄


	A ^B _C Country	1 ² ₃ Total_Units
1	United Kingdom	7457
2	Australia	107
3	France	446
4	null	652

4 rows

%sql
DESCRIBE HISTORY CompleteTable

Table

New result table: ON   

	¹ ₃ version	 timestamp	^A _C userId	^A _C userName
1	2	2024-05-25T18:13:24.000+00:00	7657907205508238	g39351894@gwu.ed
2	1	2024-05-25T18:04:29.000+00:00	7657907205508238	g39351894@gwu.ed
3	0	2024-05-25T18:04:10.000+00:00	7657907205508238	g39351894@gwu.ed

3 rows

%sql
SELECT * FROM AppendTable

Table


New result table: ON   

	¹ ₃ InvoiceNo	^A _C StockCode	^A _C Description	¹ ₃ Quanti
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	
2	536365	71053	WHITE METAL LANTERN	
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	
8	536366	22633	HAND WARMER UNION JACK	
9	536366	22632	HAND WARMER RED POLKA DOT	
10	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	
11	536367	22745	POPPY'S PLAYHOUSE BEDROOM	
12	536367	22748	POPPY'S PLAYHOUSE KITCHEN	
13	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	
14	536367	22310	IVORY KNITTED MUG COSY	
15	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	

633 rows

Table

New result table: ON   

	1^2_3 version	 timestamp	A^B_c userId	A^B_c userName
1	3	2024-05-25T18:13:24.000+00:00	7657907205508238	g39351894@gwu.ed
2	2	2024-05-25T17:57:50.000+00:00	7657907205508238	g39351894@gwu.ed
3	1	2024-05-25T17:05:46.000+00:00	7657907205508238	g39351894@gwu.ed
4	0	2024-05-25T17:05:39.000+00:00	7657907205508238	g39351894@gwu.ed

4 rows