Here is the script that you will need to run, as shown in Figure 8-1.

CREATE DATABASE IF NOT EXISTS cdc;

DROP TABLE IF EXISTS cdc.OrdersSilver

DROP TABLE IF EXISTS cdc.OrdersGold

Here is the code that you will need to run to create the OrdersSilver table.

CREATE TABLE cdc.OrdersSilver (

OrderID int,

UnitPrice int,

Quantity int,

Customer string

)

USING DELTA

LOCATION "/mnt/raw/OrdersSilver"

TBLPROPERTIES (delta.enableChangeDataFeed = true);

Here is the script that you will need to run to create the OrdersGold table.

CREATE TABLE cdc.OrdersGold (

OrderID int,

OrderTotal int,

Customer string

)

USING DELTA

LOCATION "/mnt/raw/OrdersGold"

Here is the code that you will need to run to insert data into the OrdersSilver table.

INSERT INTO cdc.OrdersSilver

SELECT 1 OrderID, 96 as UnitPrice, 5 as Quantity, "A" as Customer

UNION

SELECT 2 OrderID, 450 as UnitPrice, 10 as Quantity, "B" as Customer

UNION

SELECT 3 OrderID, 134 as UnitPrice, 7 as Quantity, "C" as Customer

UNION

SELECT 4 OrderID, 847 as UnitPrice, 8 as Quantity, "D" as Customer

UNION

SELECT 5 OrderID, 189 as UnitPrice, 15 as Quantity, "E" as Customer;

SELECT \* FROM cdc.OrdersSilver

Here is the script that you will need to run to insert data into the OrdersGold table.

INSERT INTO cdc.OrdersGold

SELECT OrderID, UnitPrice \* Quantity AS OrderTotal, Customer FROM cdc.OrdersSilver;

SELECT \* FROM cdc.OrdersGold

Here is the code that you will need to run to create the latest\_version view.

CREATE OR REPLACE TEMPORARY VIEW latest\_version as

SELECT \*

FROM

(SELECT \*, rank() over (partition by OrderID order by \_commit\_version desc) as rank

FROM table\_changes('cdc.OrdersSilver', 2, 5)

WHERE \_change\_type !='update\_preimage')

WHERE rank=1

Here is the script you will need to run to update, insert, delete and finally view the committed changes made to the OrdersSilver table.

UPDATE cdc.OrdersSilver SET Quantity = 20 WHERE OrderID = 1;

DELETE FROM cdc.OrdersSilver WHERE Customer = 'D';

INSERT INTO cdc.OrdersSilver SELECT 6 OrderID, 100 as UnitPrice, 10 as Quantity, "F" as Customer;

SELECT \* FROM table\_changes('cdc.OrdersSilver', 2, 5) order by \_commit\_timestamp;

Here is the code that you will need to run to perform the insert, update and delete on the OrdersGold table.

MERGE INTO cdc.OrdersGold og USING latest\_version os ON og.OrderID = os.OrderID

WHEN MATCHED AND os.\_change\_type='update\_postimage' THEN UPDATE SET OrderTotal = os.UnitPrice \* os.Quantity

WHEN MATCHED AND os.\_change\_type='delete' THEN DELETE

WHEN NOT MATCHED THEN INSERT (OrderID, OrderTotal, Customer) VALUES (os.OrderID, os.UnitPrice \* os.Quantity, os.Customer)

You could run the following commands to read changes in batch queries:

-- version as ints or longs e.g. changes from version 0 to 10

SELECT \* FROM table\_changes('tableName', 0, 10)

-- timestamp as string formatted timestamps

SELECT \* FROM table\_changes('tableName', '2021-02-01 00:00:00', '2021-02-11 00:00:00')

-- providing only the startingVersion/timestamp

SELECT \* FROM table\_changes('tableName', 0)

-- database/schema names inside the string for table name, with backticks for escaping dots and special characters

SELECT \* FROM table\_changes('dbName.`schema.tableName`', '2022-02-01 00:00:00' , '2022-02-11 00:00:00')

-- path based tables

SELECT \* FROM table\_changes\_by\_path('\path', '2022-02-11 00:00:00')

As an example, you could read streaming data changes by running the following Python code:

# providing a starting version

spark.readStream.format("delta") \

.option("readChangeFeed", "true") \

.option("startingVersion", 0) \

.table("myDeltaTable")

# providing a starting timestamp

spark.readStream.format("delta") \

.option("readChangeFeed", "true") \

.option("startingTimestamp", "2021-02-11 00:00:00") \

.load("/pathToMyDeltaTable")

# not providing a starting version/timestamp will result in the latest snapshot being fetched first

spark.readStream.format("delta") \

.option("readChangeFeed", "true") \

.table("myDeltaTable")

Here is the code that you will need to run to drop the OrdersSilver and OrdersGold once you have completed the exercise.

DROP TABLE cdc.OrdersSilver;

DROP TABLE cdc.OrdersGold;