```
import dlt

from pyspark.sql.functions import col,expr



@dlt.table(

      name = "lookup_time_table",

      comment= "the time mapping table from the cdc schema of the application
database.",

      table_properties={

        "quality": "bronze"

      }

)

def lookup_time_table():

   return (

      spark.read

         .format("sqlserver")

         .option("host","<>.database.windows.net")

         .option("user", "<>")

         .option("password", "<>")

         .option("database", "<>")

         .option("dbtable", "cdc.lsn_time_mapping")

         .option("skipChangeCommits", "true")

         .load()

         .select("start_lsn","tran_begin_time","tran_end_time")

   )


@dlt.table(

      name = "cdctest_cdc_raw",
    comment= "the table with captured data changes of the source table dbo.cdctest
from the  application database.",

    table_properties={

      "quality": "bronze"

    }

)

def cdctest_cdc_raw():

  return (

    spark.read

      .format("sqlserver")

      .option("host","<>.database.windows.net")

      .option("user", "<>")

      .option("password", "<>")

      .option("database", "<>")

      .option("dbtable", "cdc.dbo_cdctest_CT")

      .option("skipChangeCommits", "true")

      .load()

      .withColumnRenamed("__$operation","operation")

      .withColumnRenamed("__$start_lsn","start_lsn")

  )
```

```
# another notebook that runs before the initial dlt pipeine ever runs creates a snapshot
of the data in the table for capturing data that is not present in the cdc tables any longer
or that was created before cdc was turned on in the source)

# this line returns the timestamp that the snapshot of that source table was taken, and
it is used to filter which cdc records need to be processed

max_ss_ts = spark.sql("select max(snapshot_timestamp) as max_ss_ts from
default.cdctest_hist").first()[0]
```

```
@dlt.table(
      name = "cdctest_cdc_enriched",
      comment= "the table with cdc data from the source table dbo.cdctest from the
appl. database, enriched with transaction begin time.",
      table_properties={
        "quality": "bronze"
      }
)
def cdctest_cdc_enriched():
   cdctest = dlt.read_stream("cdctest_cdc_raw")
   lookup_time = dlt.read_stream("lookup_time_table")
   return (
      cdctest.join(lookup_time,["start_lsn"],how="inner")\
         .select("ID","test1","test2","test3","operation", "tran_begin_time")
         .where("tran_begin_time>= '{}'".format(max_ss_ts))
   )


#####  THIS CELL IS REMOVED FROM PIPELINE AFTER RUNNING ONCE, USED FOR
HISTORICAL BACKFILL ONLY
 import pyspark.sql.functions as F
 from pyspark.sql.functions import *
 from pyspark.sql import *
 @dlt.append_flow(target="cdctest_cdc_enriched")
 def snapshot_inifill():
    ss = spark.readStream.table("hive_metastore.default.cdctest_hist")
    return(
       ss.withColumn('operation',F.lit(2))\
        .withColumnRenamed("snapshot_timestamp","tran_begin_time")
         .select("ID","test1","test2","test3","operation","tran_begin_time")
    )
##### THIS CELL IS REMOVED FROM PIPELINE AFTER RUNNING ONCE, USED FOR
HISTORICAL BACKFILL ONLY


# creates streaming table that is the target for apply changes into
dlt.create_streaming_table(
  name="cdctest",
  comment="the table that mirrors the dbo.cdctest table in the application database.
CDC records obtained from the source are merged into this table to keep it in synch.",
  table_properties = {
    "quality":"silver",
    "skipChangeCommits": "true"
  }
)


dlt.apply_changes(
  target = "cdctest",
  source = "cdctest_cdc_enriched",
  keys = ["ID"],
  sequence_by = col("tran_begin_time"),
  apply_as_deletes = expr("operation = 1"),
  except_column_list = ["operation","tran_begin_time"],
  stored_as_scd_type = 1
)
```