**Index**

1. **Introduction to SCD Type-2**
2. **Why Use SCD Type-2?**
3. **Real-Life Example of SCD Type-2**
4. **Creating a Pipeline for SCD Type-2**
5. **Configuring the Data Flow Activity**
6. **Generating Hash Keys Using Derived Columns**
7. **Setting the Target**
8. **Using Lookup Activity for Data Comparison**
9. **Implementing Conditional Split for Update Detection**
10. **Implementing Alter Row**
11. **Adding Historical Data Indicators**
12. **Configuring Sink for Data Mapping**
13. **Using Alter Row for Update Permissions**
14. **Finalizing Sink for Updated Data**
15. **Comparison with SCD Type-1**

**1. Introduction to SCD Type-2**

**Slowly Changing Dimension (SCD) Type-2 is a technique used in data warehousing to manage and track historical data changes in dimension tables. Unlike SCD Type-1, SCD Type-2 retains the history of changes by adding new records instead of updating existing ones.**

**2. Why Use SCD Type-2?**

**SCD Type-2 is useful when maintaining historical data is necessary. It allows for tracking changes over time, which is essential in scenarios like customer information management or employee role changes.**

**3. Real-Life Example of SCD Type-2**

**Consider a company tracking employee job titles. When an employee changes roles, SCD Type-2 creates a new record with the updated title while keeping the old record for historical purposes.**

**4. Creating a Pipeline for SCD Type-2**

* **Open Azure Data Factory.**
* **Navigate to Author -> Pipeline.**
* **Click New Pipeline and rename it appropriately.**
* **Drag and drop a Data Flow Activity into the pipeline.**
* **Enable the Cluster for the required duration.**

**5. Configuring the Data Flow Activity**

* **Open the Data Flow and define the Source Dataset.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Configure the dataset to fetch the required data from the source.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Use a Select Transformation to rename or remove unnecessary columns for clarity.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Here, select all columns and delete them, and then choose the Rule-based Mapping. on the source’s column side, put “1==1” and choose the expression and write “Concat(‘scr\_’,$$) in the output section. Save and finish**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Now go to Preview and Refresh and you will see the columns name are updated with ‘src\_’ initials.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**6. Generating Hash Keys Using Derived Columns**

* **Add a Derived Column Transformation and rename it to HashKey.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Use the Crc32 function to generate a hash key for data comparison.**
* **Convert integer data types to string using toString before concatenation.**
* crc32(concat(toString(src\_ID),src\_NAME,src\_CITY,toString(src\_PHONENUMBER)))

**A screenshot of a computer

AI-generated content may be incorrect.**

**7. We are creating a Target here**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Choose your dataset type which in my case is Azure Sql Database and your Linked service  
After wards come to source options and Choose the Query option   
and the query with the colums that you want to import  
select CUSTID,HASHKEY from CUSTOMER\_TYPE2 where ISACTIVE=1**

**Add the ISACTIVE column to get the updated records**

**A screenshot of a computer

AI-generated content may be incorrect.**

**8. Using Lookup Activity for Data Comparison**

* **Add a Lookup Activity and connect it to the target table. Here we have Renamed the lookup to join**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Select two key columns: CustID and Src\_ID.**

**A screenshot of a computer

AI-generated content may be incorrect.  
As we did not had any data in the database so the CUSTID column came out to be null.**

**9. Implementing Conditional Split for Update Detection**

**Here we add a conditional stream   
and add INSERT AND UPDATE into them as stream names and the conditions are as follows**

**isNull(CUSTID) for insert and src\_ID==CUSTID && src\_Hashkey != HASHKEY**

**A screenshot of a computer

AI-generated content may be incorrect.**

**10. Implementing Alter Row in Update Audit**

* **Choose (Update if) in the Condition Section and put “1==1”**

**A screenshot of a computer

AI-generated content may be incorrect.**

**11. Configuring Sink for Data Mapping**

* **Add a Sink Transformation to store inserted data.**
* **Use the Mapping Settings to ensure correct column mappings.**

**A screenshot of a computer

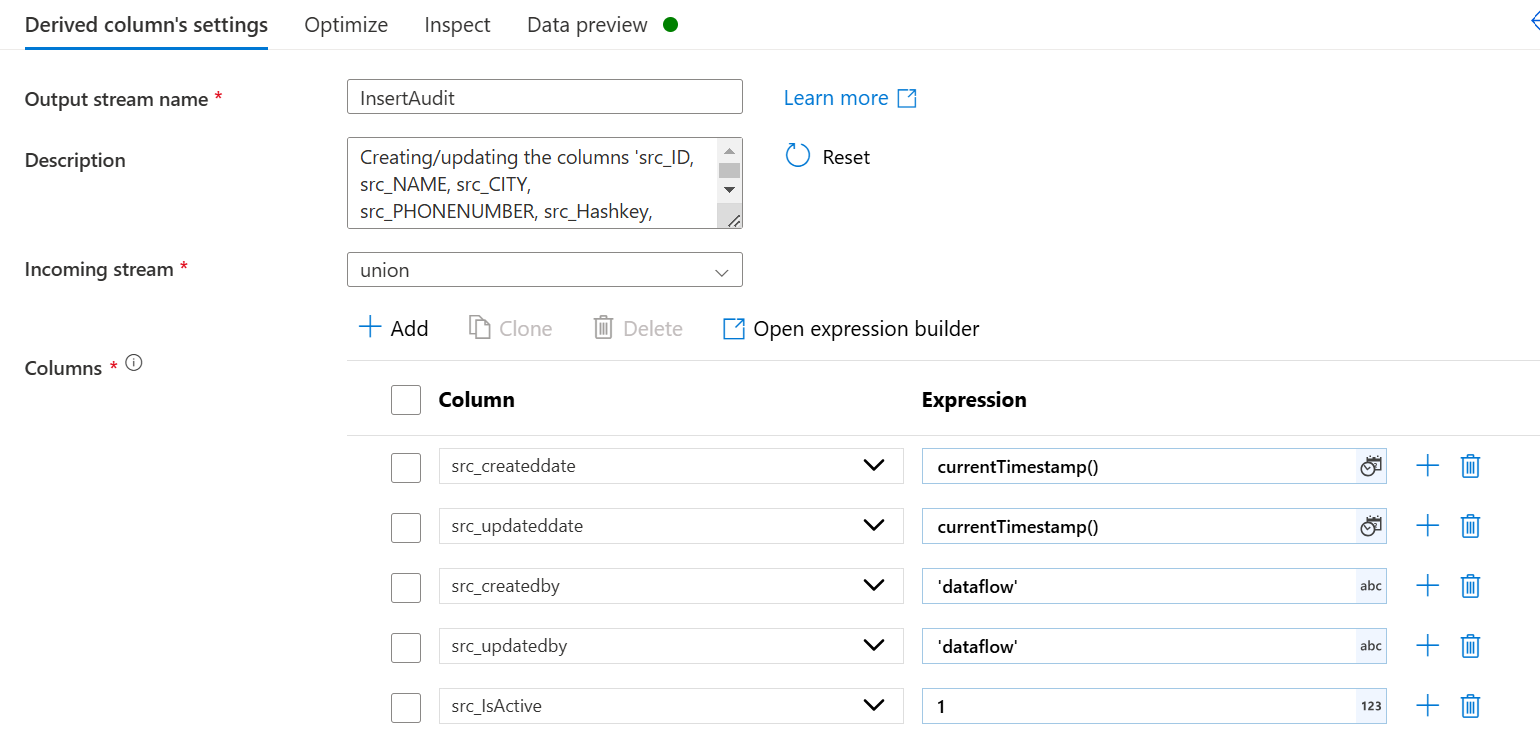
AI-generated content may be incorrect.**

**12. Adding Union to re-insert**

**A screenshot of a computer

AI-generated content may be incorrect.**

**13. Using Derived column in union and renaming it as InsertAudit  
Here I have added 5 columns**

****

**14. Finalizing Sink for Updated Data**

* **Add another Sink Transformation for updated records.**
* **Ensure correct mapping for updated records.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Validate and run the pipeline.**

**This completes the implementation of SCD Type-1 in Azure Data Factory.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**15.Discuss differences and additional steps for SCD Type-2 implementation.**

**The changes that we have done in scd type-2 are   
1:- In sink we have (Updateside) given key columns (Hash key) and (CustID)  
2:- In Mapping we are only updating 3 values that are Updateddate, updatedby and IsActive.**

**3: - Added the Union Transformation.**